IEC 61508-1

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

BASIC SAFETY PUBLICATION

PUBLICATION FONDAMENTALE DE SÉCURITÉ

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 1: Exigences générales

IEC 61508-1:2010

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.
- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.
- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.
- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:
Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm
Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,…). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.
- Just Published CEI: www.iec.ch/online_news/justpub
Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.
- Electropedia: www.electropedia.org
Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.
- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:
Email: csc@iec.ch
Tél.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61508-1

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

BASIC SAFETY PUBLICATION

PUBLICATION FONDAMENTALE DE SÉCURITÉ

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 1: Exigences générales

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE XB
CODE PRIX

ICS 13.110; 25.040; 29.020

ISBN 978-2-88910-524-3

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

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

Part 1: General 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) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61508-1 has been prepared by subcommittee 65A: System aspects, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 1998. This edition constitutes a technical revision.

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 standard is based on the following documents:

FDIS	Report on voting
65A/548/FDIS	65A/572/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2

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

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

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.

– 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.

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

Part 1: General requirements

1 Scope

1.1 This International Standard covers those aspects to be considered when electrical/electronic/programmable electronic (E/E/PE) systems are used to carry out safety functions. A major objective of this standard is to facilitate the development of product and application sector international standards by the technical committees responsible for the product or application sector. This will allow all the relevant factors, associated with the product or application, to be fully taken into account and thereby meet the specific needs of users of the product and the application sector. A second objective of this standard is to enable the development of E/E/PE safety-related systems where product or application sector international standards do not exist.

1.2 In particular, this standard

a) applies to safety-related systems when one or more of such systems incorporates electrical/electronic/programmable electronic elements;

NOTE 1 In the context of low complexity E/E/PE safety-related systems, certain requirements specified in this standard may be unnecessary, and exemption from compliance with such requirements is possible (see 4.2, and the definition of a low complexity E/E/PE safety-related system in 3.4.3 of IEC 61508-4).

NOTE 2 Although a person can form part of a safety-related system (see 3.4.1 of IEC 61508-4), human factor requirements related to the design of E/E/PE safety-related systems are not considered in detail in this standard.

b) is generically-based and applicable to all E/E/PE safety-related systems irrespective of the application;

c) covers the achievement of a tolerable risk through the application of E/E/PE safety-related systems, but does not cover hazards arising from the E/E/PE equipment itself (for example electric shock);

d) applies to all types of E/E/PE safety-related systems, including protection systems and control systems;

e) does not cover E/E/PE systems where

   – a single E/E/PE system is capable on its own of meeting the tolerable risk, and

   – the required safety integrity of the safety functions of the single E/E/PE system is less than that specified for safety integrity level 1 (the lowest safety integrity level in this standard).

f) is mainly concerned with the E/E/PE safety-related systems whose failure could have an impact on the safety of persons and/or the environment; however, it is recognized that the consequences of failure could also have serious economic implications and in such cases this standard could be used to specify any E/E/PE system used for the protection of equipment or product;

NOTE 3 See 3.1.1 of IEC 61508-4.

g) considers E/E/PE safety-related systems and other risk reduction measures, in order that the safety requirements specification for the E/E/PE safety-related systems can be determined in a systematic, risk-based manner;

h) uses an overall safety lifecycle model as the technical framework for dealing systematically with the activities necessary for ensuring the functional safety of the E/E/PE safety-related systems;

NOTE 4 Although the overall safety lifecycle is primarily concerned with E/E/PE safety-related systems, it could also provide a technical framework for considering any safety-related system irrespective of the technology of that system (for example mechanical, hydraulic or pneumatic).

i) does not specify the safety integrity levels required for sector applications (which must be based on detailed information and knowledge of the sector application). The technical committees responsible for the specific application sectors shall specify, where appropriate, the safety integrity levels in the application sector standards;

j) provides general requirements for E/E/PE safety-related systems where no product or application sector international standards exist;

k) requires malevolent and unauthorised actions to be considered during hazard and risk analysis. The scope of the analysis includes all relevant safety lifecycle phases;

NOTE 5 Other IEC/ISO standards address this subject in depth; see ISO/IEC/TR 19791 and IEC 62443 series.

l) does not cover the precautions that may be necessary to prevent unauthorized persons damaging, and/or otherwise adversely affecting, the functional safety of E/E/PE safety-related systems (see k) above);

m) does not specify the requirements for the development, implementation, maintenance and/or operation of security policies or security services needed to meet a security policy that may be required by the E/E/PE safety-related system;

n) does not apply for medical equipment in compliance with the IEC 60601 series.

1.3 This part of the IEC 61508 series of standards includes general requirements that are applicable to all parts. Other parts of the IEC 61508 series concentrate on more specific topics:

– parts 2 and 3 provide additional and specific requirements for E/E/PE safety-related systems (for hardware and software);

– part 4 gives definitions and abbreviations that are used throughout this standard;

– part 5 provides guidelines on the application of part 1 in determining safety integrity levels, by showing example methods;

– part 6 provides guidelines on the application of parts 2 and 3;

– part 7 contains an overview of techniques and measures.

1.4 IEC 61508-1, IEC 61508-2, 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 function of this international standard does not apply to medical equipment in compliance with the IEC 60601 series.

NOTE 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.

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

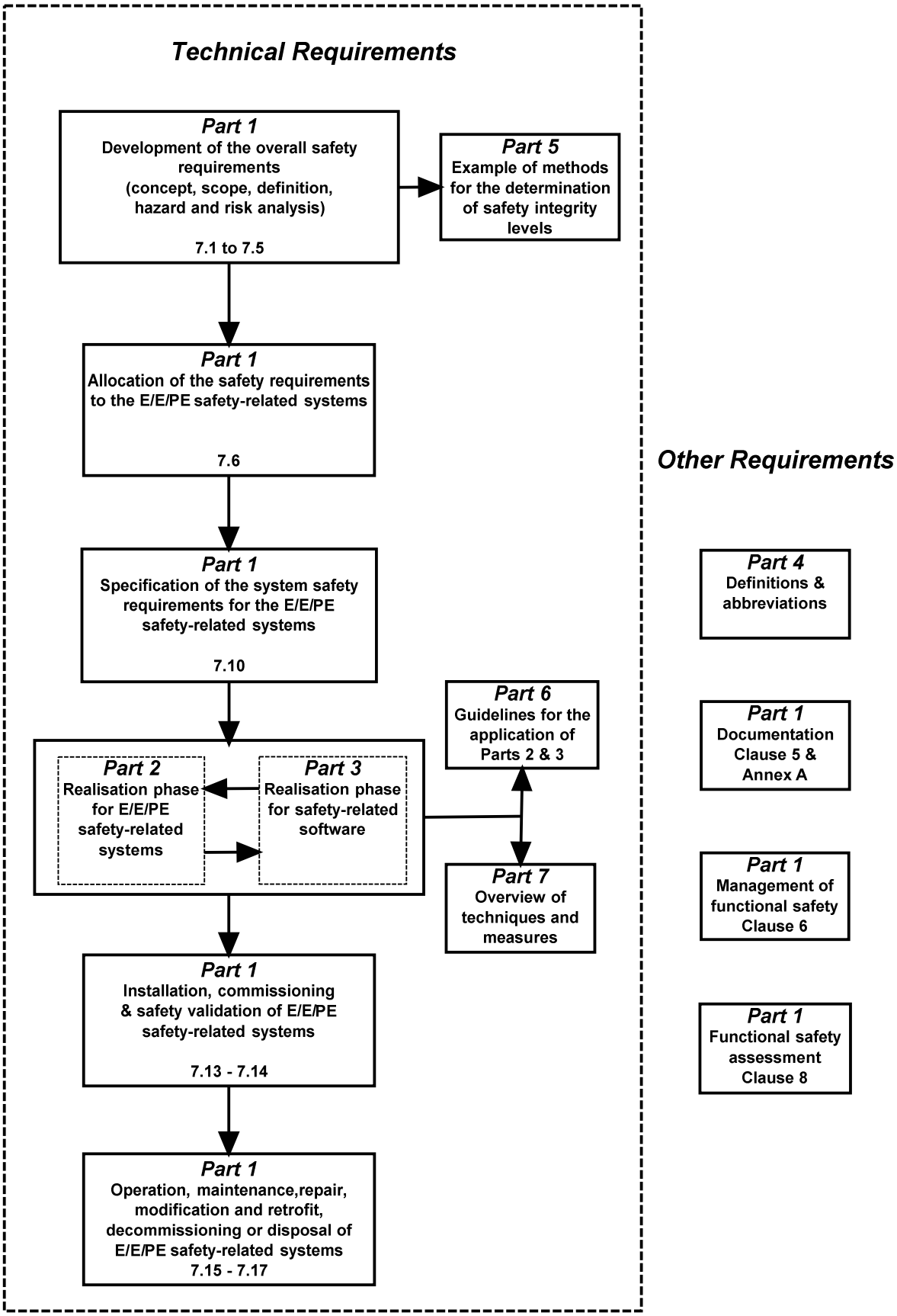


Figure 1 – Overall framework of the IEC 61508 series

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

IEC 61508-3:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements*

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*

ISO/IEC 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

4.1 To conform to this standard it shall be demonstrated that all the relevant requirements have been satisfied to the required criteria specified (for example safety integrity level) and therefore, for each clause or subclause, all the objectives have been met.

4.2 This standard specifies the requirements for E/E/PE safety-related systems and has been developed to meet the full range of complexity associated with such systems. However, for low complexity E/E/PE safety-related systems (see 3.4.3 of IEC 61508-4), where dependable field experience exists which provides the necessary confidence that the required safety integrity can be achieved, the following options are available:

– in product and application sector international standards implementing the requirements of IEC 61508-1 to IEC 61508-7, certain requirements may be unnecessary and exemption from compliance with such requirements is acceptable;

– if this standard is used directly for those situations where no product or application sector international standard exists, certain of the requirements specified in this standard may be unnecessary and exemption from compliance with such requirements is acceptable providing this is justified.

4.3 Product or application sector international standards for E/E/PE safety-related systems developed within the framework of this standard shall take into account the requirements of ISO/IEC Guide 51 and IEC Guide 104.

5 Documentation

5.1 Objectives

5.1.1 The first objective of the requirements of this clause is to specify the necessary information to be documented in order that all phases of the overall, E/E/PE system and software safety lifecycles can be effectively performed.

5.1.2 The second objective of the requirements of this clause is to specify the necessary information to be documented in order that the management of functional safety (see Clause 6), verification (see 7.18) and the functional safety assessment (see Clause 8) activities can be effectively performed.

NOTE 1 The documentation requirements in this standard are concerned, essentially, with information rather than physical documents. The information need not be contained in physical documents unless this is explicitly declared in the relevant subclause.

NOTE 2 Documentation may be available in different forms (for example on paper, film, or any data medium to be presented on screens or displays).

NOTE 3 See Annex A concerning possible documentation structures.

NOTE 4 See reference [7] in the Bibliography.

5.2 Requirements

5.2.1 The documentation shall contain sufficient information, for each phase of the overall, E/E/PE system and software safety lifecycles completed, necessary for effective performance of subsequent phases and verification activities.

NOTE What constitutes sufficient information will be dependent upon a number of factors, including the complexity and size of the E/E/PE safety-related systems and the requirements relating to the specific application.

5.2.2 The documentation shall contain sufficient information required for the management of functional safety (Clause 6).

NOTE See notes to 5.1.2.

5.2.3 The documentation shall contain sufficient information required for the implementation of a functional safety assessment, together with the information and results derived from any functional safety assessment.

NOTE See notes to 5.1.2.

5.2.4 The information to be documented shall be as stated in the various clauses of this standard unless justified or shall be as specified in the product or application sector international standard relevant to the application

5.2.5 The availability of documentation shall be sufficient for the duties to be performed in respect of the clauses of this standard.

NOTE Only the information necessary to undertake a particular activity, required by this standard, need be held by each relevant party.

5.2.6 The documentation shall:

– be accurate and concise;
– be easy to understand by those persons having to make use of it;
– suit the purpose for which it is intended;
– be accessible and maintainable.

5.2.7 The documentation or set of information shall have titles or names indicating the scope of the contents, and some form of index arrangement so as to allow ready access to the information required in this standard.

5.2.8 The documentation structure may take account of company procedures and the working practices of specific product or application sectors.

5.2.9 The documents or set of information shall have a revision index (version numbers) to make it possible to identify different versions of the document.

5.2.10 The documents or set of information shall be so structured as to make it possible to search for relevant information. It shall be possible to identify the latest revision (version) of a document or set of information.

NOTE The physical structure of the documentation will vary depending upon a number of factors such as the size of the system, its complexity and organizational requirements.

5.2.11 All relevant documents shall be revised, amended, reviewed and approved under an appropriate document control scheme.

NOTE Where automatic or semi-automatic tools are used for the production of documentation, specific procedures may be necessary to ensure effective measures are in place for the management of versions or other control aspects of the documents.

6 Management of functional safety

6.1 Objectives

6.1.1 The first objective of the requirements of this clause is to specify the responsibilities in the management of functional safety of those who have responsibility for an E/E/PE safety-related system, or for one or more phases of the overall E/E/PE system and software safety lifecycles.

6.1.2 The second objective of the requirements of this clause is to specify the activities to be carried out by those with responsibilities in the management of functional safety.

NOTE The organizational measures dealt with in this clause provide for the effective implementation of the technical requirements and are solely aimed at the achievement and maintenance of functional safety of the E/E/PE safety-related systems. The technical requirements necessary for maintaining functional safety will be specified as part of the information provided by the supplier of the E/E/PE safety-related system and its elements and components.

6.2 Requirements

6.2.1 An organisation with responsibility for an E/E/PE safety-related system, or for one or more phases of the overall, E/E/PE system or software safety lifecycle, shall appoint one or more persons to take overall responsibility for:

– the system and for its lifecycle phases;
– coordinating the safety-related activities carried out in those phases;
– the interfaces between those phases and other phases carried out by other organisations;
– carrying out the requirements of 6.2.2 to 6.2.11 and 6.2.13;
– coordinating functional safety assessments (see 6.2.12 b) and Clause 8) – particularly where those carrying out the functional safety assessment differ between phases – including communication, planning, and integrating the documentation, judgements and recommendations;
– ensuring that functional safety is achieved and demonstrated in accordance with the objectives and requirements of this standard.

NOTE Responsibility for safety-related activities, or for safety lifecycle phases, may be delegated to other persons, particularly those with relevant expertise, and different persons could be responsible for different activities and requirements. However, the responsibility for coordination, and for overall functional safety, should reside in one or a small number of persons with sufficient management authority.

6.2.2 The policy and strategy for achieving functional safety shall be specified, together with the means for evaluating their achievement, and the means by which they are communicated within the organization.

6.2.3 All persons, departments and organizations responsible for carrying out activities in the applicable overall, E/E/PE system or software safety lifecycle phases (including persons responsible for verification and functional safety assessment and, where relevant, licensing authorities or safety regulatory bodies) shall be identified, and their responsibilities shall be fully and clearly communicated to them.

6.2.4 Procedures shall be developed for defining what information is to be communicated, between relevant parties, and how that communication will take place.

NOTE See Clause 5 for documentation requirements.

6.2.5 Procedures shall be developed for ensuring prompt follow-up and satisfactory resolution of recommendations relating to E/E/PE safety-related systems, including those arising from:

a) hazard and risk analysis (see 7.4);

b) functional safety assessment (see Clause 8);

c) verification activities (see 7.18);

d) validation activities (see 7.8 and 7.14);

e) configuration management (see 6.2.10, 7.16, IEC 61508-2 and IEC 61508-3);

f) incident reporting and analysis (see 6.2.6).

6.2.6 Procedures shall be developed for ensuring that all detected hazardous events are analysed, and that recommendations are made to minimise the probability of a repeat occurrence.

6.2.7 Requirements for periodic functional safety audits shall be specified, including:

a) the frequency of the functional safety audits;

b) the level of independence of those carrying out the audits;

c) the necessary documentation and follow-up activities.

6.2.8 Procedures shall be developed for:

a) initiating modifications to the E/E/PE safety-related systems (see 7.16.2.2);

b) obtaining approval and authority for modifications.

6.2.9 Procedures shall be developed for maintaining accurate information on hazards and hazardous events, safety functions and E/E/PE safety-related systems.

6.2.10 Procedures shall be developed for configuration management of the E/E/PE safety-related systems during the overall, E/E/PE system and software safety lifecycle phases, including in particular:

a) the point, in respect of specific phases, at which formal configuration control is to be implemented;

b) the procedures to be used for uniquely identifying all constituent parts of an item (hardware and software);

c) the procedures for preventing unauthorized items from entering service.

6.2.11 Training and information for the emergency services shall be provided where appropriate.

6.2.12 Those individuals who have responsibility for one or more phases of the overall, E/E/PE system or software safety lifecycles shall, in respect of those phases for which they have responsibility and in accordance with the procedures defined in 6.2.1 to 6.2.11, specify all management and technical activities that are necessary to ensure the achievement, demonstration and maintenance of functional safety of the E/E/PE safety-related systems, including:

a) the selected measures and techniques used to meet the requirements of a specified clause or subclause (see IEC 61508-2, IEC 61508-3 and IEC 61508-6);

b) the functional safety assessment activities, and the way in which the achievement of functional safety will be demonstrated to those carrying out the functional safety assessment (see Clause 8);

NOTE Appropriate procedures for functional safety assessment should be used to define

– the selection of an appropriate organisation, person or persons, at the appropriate level of independence;

– the drawing up, and making changes to, terms of reference for functional safety assessments;

– the change of those carrying out the functional safety assessment at any point during the lifecycle of a system;

– the resolution of disputes involving those carrying out functional safety assessments.

c) the procedures for analysing operations and maintenance performance, in particular for

   – recognising systematic faults that could jeopardise functional safety, including procedures used during routine maintenance that detect recurring faults;

   – assessing whether the demand rates and failure rates during operation and maintenance are in accordance with assumptions made during the design of the system.

6.2.13 Procedures shall be developed to ensure that all persons with responsibilities defined in accordance with 6.2.1 and 6.2.3 (i.e. including all persons involved in any overall, E/E/PE system or software lifecycle activity, including activities for verification, management of functional safety and functional safety assessment), shall have the appropriate competence (i.e. training, technical knowledge, experience and qualifications) relevant to the specific duties that they have to perform. Such procedures shall include requirements for the refreshing, updating and continued assessment of competence.

6.2.14 The appropriateness of competence shall be considered in relation to the particular application, taking into account all relevant factors including:

a) the responsibilities of the person;

b) the level of supervision required;

c) the potential consequences in the event of failure of the E/E/PE safety-related systems – the greater the consequences, the more rigorous shall be the specification of competence;

d) the safety integrity levels of the E/E/PE safety-related systems – the higher the safety integrity levels, the more rigorous shall be the specification of competence;

e) the novelty of the design, design procedures or application – the newer or more untried these are, the more rigorous shall be the specification of competence;

f) previous experience and its relevance to the specific duties to be performed and the technology being employed – the greater the required competence, the closer the fit shall be between the competences developed from previous experience and those required for the specific activities to be undertaken;

g) the type of competence appropriate to the circumstances (for example qualifications, experience, relevant training and subsequent practice, and leadership and decision-making abilities);

h) engineering knowledge appropriate to the application area and to the technology;

i) safety engineering knowledge appropriate to the technology;

j) knowledge of the legal and safety regulatory framework;

k) relevance of qualifications to specific activities to be performed.

NOTE Reference [8] in the Bibliography contains an example method for managing competence for E/E/PE safety-related systems.

6.2.15 The competence of all persons with responsibilities defined in accordance with 6.2.1 and 6.2.3 shall be documented.

6.2.16 The activities specified as a result of 6.2.2 to 6.2.15 shall be implemented and monitored.

6.2.17 Suppliers providing products or services to an organization having overall responsibility for one or more phases of the overall, E/E/PE system or software safety lifecycles (see 6.2.1), shall deliver products or services as specified by that organization and shall have an appropriate quality management system.

6.2.18 Activities relating to the management of functional safety shall be applied at the relevant phases of the overall, E/E/PE system and software safety lifecycles (see 7.1.1.5).

7 Overall safety lifecycle requirements

7.1 General

7.1.1 Introduction

7.1.1.1 In order to deal in a systematic manner with all the activities necessary to achieve the required safety integrity for the safety functions carried out by the E/E/PE safety-related systems, this standard adopts an overall safety lifecycle (see Figure 2) as the technical framework.

NOTE The overall safety lifecycle should be used as a basis for claiming conformance to this standard, but a different overall safety lifecycle can be used to that given in Figure 2, providing the objectives and requirements of each clause of this standard are met.

7.1.1.2 The overall safety lifecycle encompasses the following means for meeting the tolerable risk:

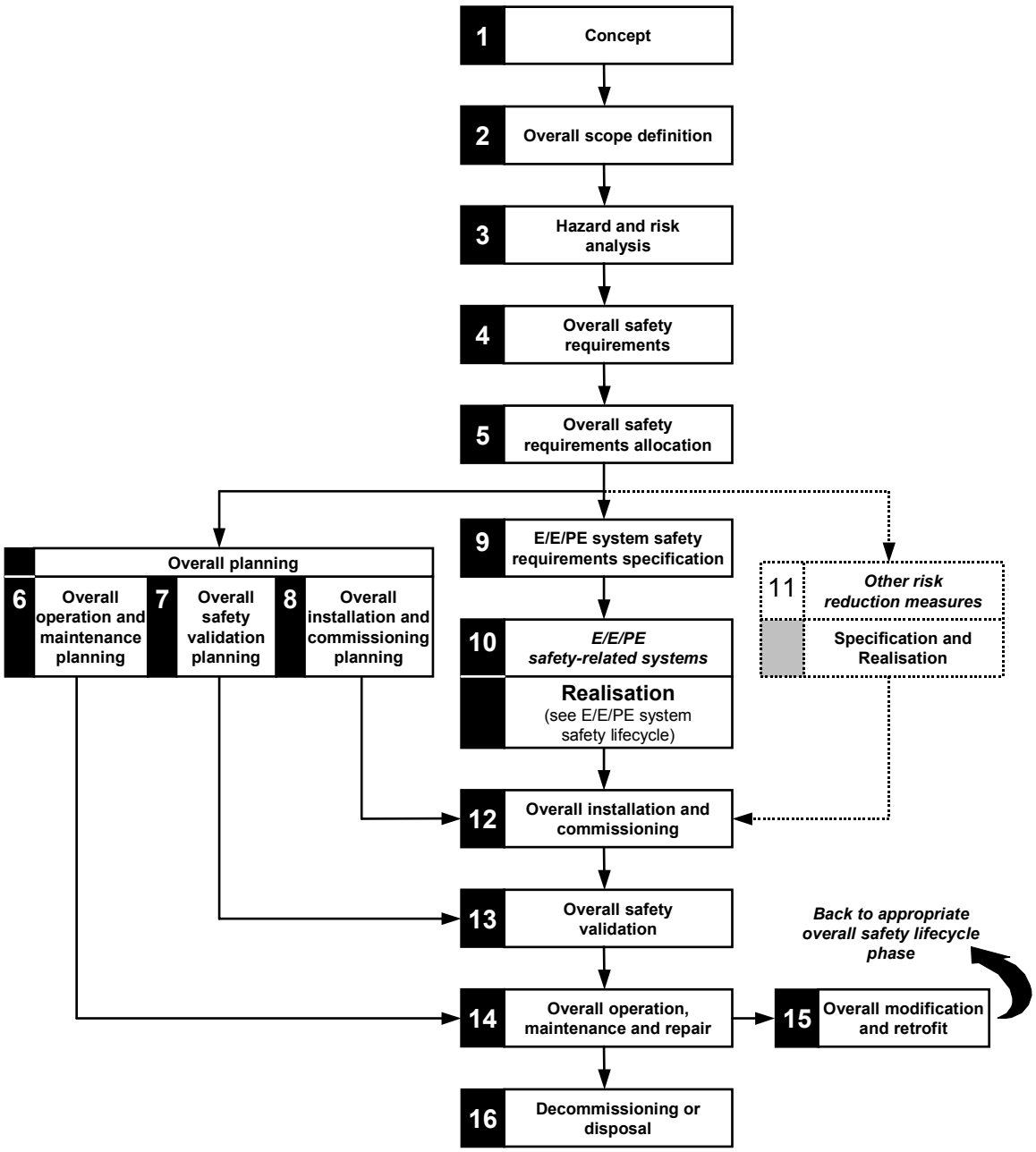
– E/E/PE safety-related systems;

– other risk reduction measures.

7.1.1.3 The E/E/PE safety-related systems realisation phase from the overall safety lifecycle is expanded and shown in Figure 3. This part of the E/E/PE system safety lifecycle forms the technical framework for IEC 61508-2. The part of the software safety lifecycle shown in Figure 4 forms the technical framework for IEC 61508-3. The relationship of the overall safety lifecycle to the E/E/PE system and software safety lifecycles for safety-related systems is shown in Figure 5.

7.1.1.4 The overall, E/E/PE system and software safety lifecycle figures (Figures 2 to 4) are simplified views of reality and as such do not show all the iterations relating to specific phases or between phases. Iteration, however, is an essential and vital part of development through the overall, E/E/PE system and software safety lifecycles.

7.1.1.5 Activities relating to the management of functional safety (Clause 6), verification (7.18) and functional safety assessment (Clause 8) are not shown on the overall, E/E/PE system or software safety lifecycles. This has been done in order to reduce the complexity of the lifecycle figures. These activities, where required, will need to be applied at the relevant phases of the overall, E/E/PE system and software safety lifecycles.



NOTE 1 Activities relating to ***verification***, ***management of functional safety*** and ***functional safety assessment*** are not shown for reasons of clarity but are relevant to all overall, E/E/PE system and software safety lifecycle phases.

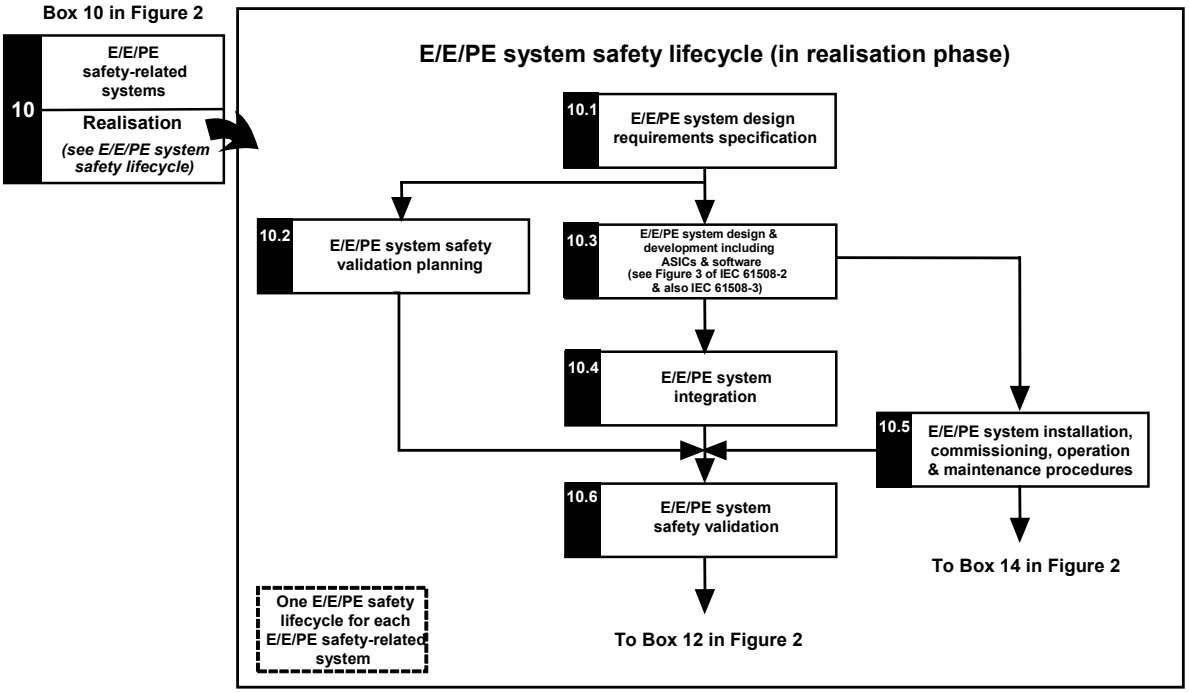
NOTE 2 The phase represented by Box 11 is outside the scope of this standard.

NOTE 3 IEC 61508-2 and IEC 61508-3 deal with Box 10 (realisation) but they also deal, where relevant, with the programmable electronic (hardware and software) aspects of Boxes 13, 14 and 15.

NOTE 4 See Table 1 for a description of the objectives and scope of the phases represented by each box.

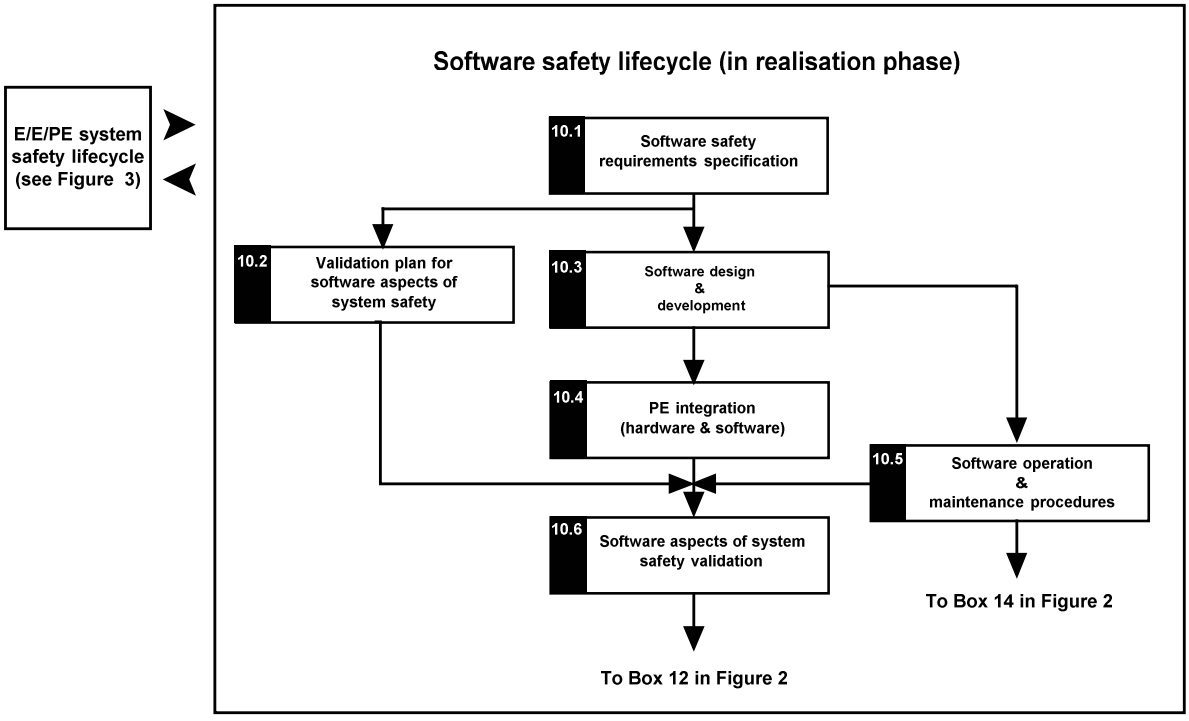
NOTE 5 The technical requirements necessary for overall operation, maintenance, repair, modification, retrofit and decommissioning or disposal will be specified as part of the information provided by the supplier of the E/E/PE safety-related system and its elements and components.

Figure 2 – Overall safety lifecycle



NOTE This figure shows only those phases of the E/E/PE system safety lifecycle that are within the realisation phase of the overall safety lifecycle. The complete E/E/PE system safety lifecycle will also contain instances, specific to the E/E/PE safety-related system, of the subsequent phases of the overall safety lifecycle (Boxes 12 to 16 in Figure 2).

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



NOTE This figure shows only those phases of the software safety lifecycle that are within the realisation phase of the overall safety lifecycle. The complete software safety lifecycle will also contain instances, specific to the software for the E/E/PE safety-related system, of the subsequent phases of the overall safety lifecycle (Boxes 12 to 16 in Figure 2).

Figure 4 – Software safety lifecycle (in realisation phase)

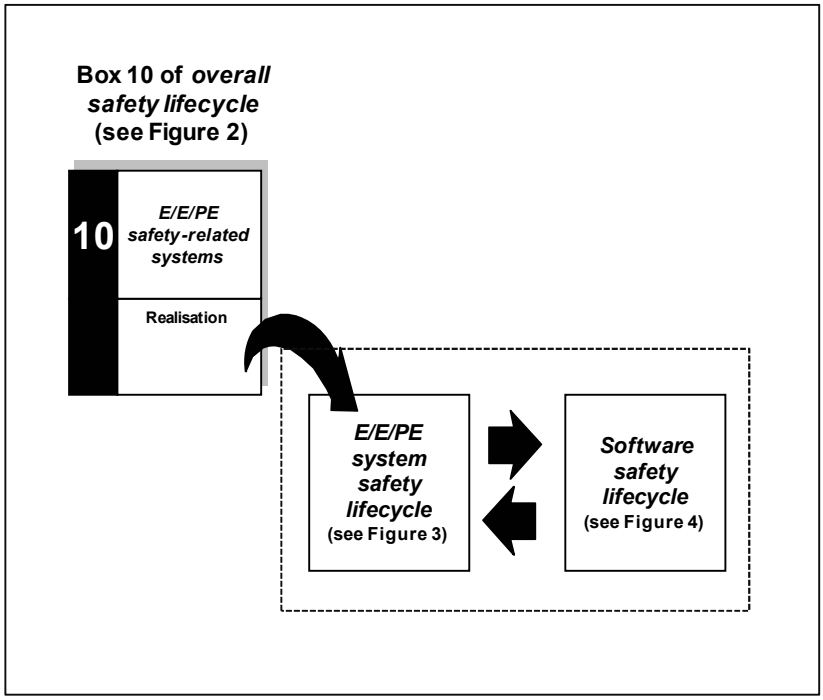


Figure 5 – Relationship of overall safety lifecycle to the E/E/PE system and software safety lifecycles

7.1.2 Objectives and requirements – general

7.1.2.1 The objectives and requirements for the overall safety lifecycle phases are contained in 7.2 to 7.17. The objectives and requirements for the E/E/PE system and software safety lifecycle phases are contained in IEC 61508-2 and IEC 61508-3 respectively.

NOTE 7.2 to 7.17 relate to specific boxes (phases) in Figure 2. The specific box is referenced in notes to these subclauses.

7.1.2.2 For all phases of the overall safety lifecycle, Table 1 indicates:

– the objectives to be achieved;
– the scope of the phase;
– the reference to the subclause containing the requirements;
– the required inputs to the phase;
– the outputs required to comply with the requirements.

Table 1 – Overall safety lifecycle – overview

Safety lifecycle phase		Objectives	Scope	Require-ments subclause	Inputs	Outputs
Figure 2 box number	Title					
1	**Concept**	7.2.1: To develop a level of understanding of the EUC and its environment (physical, legislative etc.) sufficient to enable the other safety lifecycle activities to be satisfactorily carried out.	EUC and its environment (physical, legislative etc.).	7.2.2	All relevant information necessary to meet the requirements of the subclause.	Information concerning the EUC, its environment and hazards.
2	**Overall scope definition**	7.3.1: To determine the boundary of the EUC and the EUC control system; To specify the scope of the hazard and risk analysis (for example process hazards, environmental hazards, etc.).	EUC and its environment.	7.3.2	Information concerning the EUC, its environment and hazards.	Defined scope of the hazard and risk analysis.
3	**Hazard and risk analysis**	7.4.1: To determine the hazards, hazardous events and hazardous situations relating to the EUC and the EUC control system (in all modes of operation), for all reasonably foreseeable circumstances, including fault conditions and reasonably foreseeable misuse (see 3.1.14 of IEC 61508-4); To determine the event sequences leading to the hazardous events; To determine the EUC risks associated with the hazardous events.	The scope will be dependent upon the phase reached in the overall, E/E/PE system and software safety lifecycles (since it may be necessary for more than one hazard and risk analysis to be carried out). For the preliminary hazard and risk analysis, the scope will be as defined by the output of the overall scope definition.	7.4.2	Defined scope of the hazard and risk analysis.	Description of, and information relating to, the hazard and risk analysis.
4	**Overall safety require-ments**	7.5.1: To develop the specification for the overall safety requirements, in terms of the safety functions requirements and safety integrity requirements, for the E/E/PE safety-related systems and other risk reduction measures, in order to achieve the required functional safety.	As defined by the output of the overall scope definition.	7.5.2	Description of, and information relating to, the hazard and risk analysis.	Specification of the overall safety requirements in terms of the safety functions requirements and the safety integrity requirements.

Table 1 *(continued)*

Safety lifecycle phase		Objectives	Scope	Require-ments subclause	Inputs	Outputs
Figure 2 box number	**Title**					
5	**Overall safety require-ments allocation**	7.6.1: To allocate the safety functions, contained in the specification for the overall safety requirements (both the safety functions requirements and the safety integrity requirements), to the designated E/E/PE safety-related systems and other risk reduction measures; To allocate a safety integrity level to each safety function to be carried out by an E/E/PE safety-related system.	As defined by the output of the overall scope definition.	7.6.2	Specification of the overall safety requirements in terms of the safety functions requirements and the safety integrity requirements.	Information on the allocation of the overall safety functions, their target failure measures, and associated safety integrity levels Assumptions made concerning other risk reduction measures that need to be managed throughout the life of the EUC (see 7.6.2.13).
6	**Overall operation and mainte-nance planning**	7.7.1: To develop a plan for operating and maintaining the E/E/PE safety-related systems, to ensure that the required functional safety is maintained during operation and maintenance.	EUC, the EUC control system and human factors; E/E/PE safety-related systems.	7.7.2	Information on the allocation of the overall safety functions, their target failure measures, and associated safety integrity levels Assumptions made concerning other risk reduction measures that need to be managed throughout the life of the EUC (see 7.6.2.13).	A plan for operating and maintaining the E/E/PE safety-related systems.
7	**Overall safety validation planning**	7.8.1: To develop a plan for the overall safety validation of the E/E/PE safety-related systems.	EUC, the EUC control system and human factors; E/E/PE safety-related systems.	7.8.2	Information and results of the overall safety requirements allocation.	A plan for the overall safety validation of the E/E/PE safety-related systems.
8	**Overall installa-tion and commis-sioning planning**	7.9.1: To develop a plan for the installation of the E/E/PE safety-related systems in a controlled manner, to ensure that the required functional safety is achieved; To develop a plan for the commissioning of the E/E/PE safety-related systems in a controlled manner, to ensure that the required functional safety is achieved.	EUC and the EUC control system; E/E/PE safety-related systems.	7.9.2	Information and results of the overall safety requirements allocation.	A plan for the installation of the E/E/PE safety-related systems; A plan for the commissioning of the E/E/PE safety-related systems.

Table 1 *(continued)*

Safety lifecycle phase		Objectives	Scope	Require-ments subclause	Inputs	Outputs
Figure 2 box number	Title					
9	**E/E/PE system safety require-ments specifi-cation**	7.10.1: To define the E/E/PE system safety requirements, in terms of the E/E/PE system safety functions requirements and the E/E/PE system safety integrity requirements, in order to achieve the required functional safety.	E/E/PE safety-related systems	7.10.2	Information and results of the overall safety requirements allocation.	Specification of the E/E/PE system safety requirements.
10	**E/E/PE safety-related systems: realisation**	7.11.1 and parts 2 and 3: To create E/E/PE safety-related systems conforming to the specification for the E/E/PE system safety requirements (comprising the specification for the E/E/PE system safety functions requirements and the specification for the E/E/PE system safety integrity requirements).	E/E/PE safety-related systems.	7.11.2, IEC 61508-2 and IEC 61508-3	Specification of the E/E/PE system safety requirements.	Realisation of each E/E/PE safety-related system according to the E/E/PE system safety requirements specification.
11	**Other risk reduction measures: specifi-cation and realisation**	7.12.1: To create other risk reduction measures to meet the safety functions requirements and safety integrity requirements specified for such systems (outside the scope of this standard).	Other risk reduction measures.	7.12.2	Other risk reduction measures safety requirements specification (outside the scope and not considered further in this standard).	Realisation of each other risk reduction measure according to the safety requirements for that measure.
12	**Overall installa-tion and commis-sioning**	7.13.1: To install the E/E/PE safety-related systems; To commission the E/E/PE safety-related systems.	EUC and the EUC control system; E/E/PE safety-related systems.	7.13.2	A plan for the installation of the E/E/PE safety-related systems; A plan for the commis-sioning of the E/E/PE safety-related systems.	Fully installed E/E/PE safety-related systems; Fully commissioned E/E/PE safety-related systems.
13	**Overall safety validation**	7.14.1: To validate that the E/E/PE safety-related systems meet the specification for the overall safety requirements in terms of the overall safety functions requirements and the overall safety integrity requirements, taking into account the safety requirements allocation for the E/E/PE safety-related systems developed according to 7.6.	EUC and the EUC control system; E/E/PE safety-related systems.	7.14.2	Overall safety validation plan for the E/E/PE safety-related systems; Information and results of the overall safety requirements allocation.	Confirmation that all the E/E/PE safety-related systems meet the specification for the overall safety requirements, taking into account the safety requirements allocation for the E/E/PE safety-related systems.

Table 1 *(continued)*

Safety lifecycle phase		Objectives	Scope	Require-ments subclause	Inputs	Outputs
Figure 2 box number	**Title**					
14	**Overall operation, mainte-nance and repair**	7.15.1: To ensure the functional safety of the E/E/PE safety-related systems is maintained to the specified level; To ensure that the technical requirements, necessary for the overall operation, maintenance and repair of the E/E/PE safety-related systems, are specified and provided to those responsible for the future operation and maintenance of the E/E/PE safety-related systems.	EUC and the EUC control system; E/E/PE safety-related systems.	7.15.2	Overall operation and maintenance plan for the E/E/PE safety-related systems.	Continuing achievement of the required functional safety for the E/E/PE safety-related systems; Chronological documentation of operation, repair and maintenance of the E/E/PE safety-related systems.
15	**Overall modifica-tion and retrofit**	7.16.1: To define the procedures that are necessary to ensure that the functional safety for the E/E/PE safety-related systems is appropriate, both during and after the modification and retrofit phase has taken place.	EUC and the EUC control system; E/E/PE safety-related systems.	7.16.2	Request for modification or retrofit under the procedures for the management of functional safety.	Achievement of the required functional safety for the E/E/PE safety-related systems, both during and after the modification and retrofit phase has taken place; Chronological documentation of modification and retrofit of the E/E/PE safety-related systems.
16	**Decom-mis-sioning or disposal**	7.17.1: To define the procedures that are necessary to ensure that the functional safety for the E/E/PE safety-related systems is appropriate in the circumstances during and after the activities of decommissioning or disposing of the EUC.	EUC and the EUC control system; E/E/PE safety-related systems.	7.17.2	Request for decommis-sioning or disposal under the procedures for the management of functional safety.	Achievement of the required functional safety for the E/E/PE safety-related systems both during and after the decommis-sioning or disposal activities; Chronological documentation of the decommis-sioning or disposal activities.

7.1.3 Objectives

7.1.3.1 The first objective of the requirements of this subclause is to structure, in a systematic manner, the phases in the overall safety lifecycle that shall be considered in order to achieve the required functional safety of the E/E/PE safety-related systems.

7.1.3.2 The second objective of the requirements of this subclause is to document key information relevant to the functional safety of the E/E/PE safety-related systems throughout the overall safety lifecycle.

NOTE See Clause 5 for documentation requirements and Annex A for an example documentation structure. The documentation structure may take account of company procedures, and of the working practices of specific product or application sectors.

7.1.4 Requirements

7.1.4.1 The overall safety lifecycle that shall be used as the basis for claiming conformance to this standard is that specified in Figure 2. If another overall safety lifecycle is used, it shall be specified as part of the management of functional safety activities (see Clause 6) and all the objectives and requirements in each clause or subclause in this standard shall be met.

NOTE The parts of the E/E/PE system safety lifecycle and the software safety lifecycle that form the realisation phase of the overall safety lifecycle are specified in IEC 61508-2 and IEC 61508-3 respectively.

7.1.4.2 The requirements for the management of functional safety (see Clause 6) shall run in parallel with the overall safety lifecycle phases.

7.1.4.3 Unless justified, each phase of the overall safety lifecycle shall be applied and the requirements met.

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

7.1.4.5 The scope and inputs for each overall safety lifecycle phase shall be as specified in Table 1 unless justified as part of the management of functional safety activities (see Clause 6) or specified in the product or application sector international standard.

7.1.4.6 The outputs from each phase of the overall safety lifecycle shall be those specified in Table 1 unless justified as part of the management of functional safety activities (see Clause 6) or specified in the product or application sector international standard.

7.1.4.7 The outputs from each phase of the overall safety lifecycle shall meet the objectives and requirements specified for each phase (see 7.2 to 7.17).

7.1.4.8 The verification requirements that shall be met for each overall safety lifecycle phase are specified in 7.18.

7.2 Concept

NOTE This phase is Box 1 of Figure 2.

7.2.1 Objective

The objective of the requirements of this subclause is to develop a level of understanding of the EUC and its environment (physical, legislative etc.) sufficient to enable the other safety lifecycle activities to be satisfactorily carried out.

7.2.2 Requirements

7.2.2.1 A thorough familiarity shall be acquired of the EUC, its required control functions and its physical environment.

7.2.2.2 The likely sources of hazards, hazardous situations and harmful events shall be determined.

7.2.2.3 Information about the determined hazards shall be obtained (for example, duration, intensity, toxicity, exposure limit, mechanical force, explosive conditions, reactivity, flammability etc.).

7.2.2.4 Information about the current safety regulations (national and international) shall be obtained.

7.2.2.5 Hazards, hazardous situations and harmful events due to interaction with other equipment or systems (installed or to be installed) of the EUC shall be considered together with other EUCs (installed or to be installed)

7.2.2.6 The information and results acquired in 7.2.2.1 to 7.2.2.5 shall be documented.

7.3 Overall scope definition

NOTE This phase is Box 2 of Figure 2.

7.3.1 Objectives

7.3.1.1 The first objective of the requirements of this subclause is to determine the boundary of the EUC and the EUC control system.

7.3.1.2 The second objective of the requirements of this subclause is to specify the scope of the hazard and risk analysis (for example process hazards, environmental hazards, etc.).

7.3.2 Requirements

7.3.2.1 The boundary of the EUC and the EUC control system shall be defined so as to include all equipment and systems (including humans where appropriate) that are associated with relevant hazards and hazardous events.

NOTE Several iterations between overall scope definition and hazard and risk analysis may be necessary.

7.3.2.2 The physical equipment, including the EUC and the EUC control system, to be included in the scope of the hazard and risk analysis shall be specified.

NOTE See references [9] and [10] in the Bibliography.

7.3.2.3 The external events to be taken into account in the hazard and risk analysis shall be specified.

7.3.2.4 The equipment and systems that are associated with the hazards and hazardous events shall be specified.

7.3.2.5 The type of initiating events that need to be considered (for example component failures, procedural faults, human error, dependent failure mechanisms that can cause hazardous events) shall be specified.

7.3.2.6 The information and results acquired in 7.3.2.1 to 7.3.2.5 shall be documented.

7.4 Hazard and risk analysis

NOTE This phase is Box 3 of Figure 2.

7.4.1 Objectives

7.4.1.1 The first objective of the requirements of this subclause is to determine the hazards, hazardous events and hazardous situations relating to the EUC and the EUC control system (in all modes of operation) for all reasonably foreseeable circumstances, including fault conditions and reasonably foreseeable misuse (see 3.1.14 of IEC 61508-4);

7.4.1.2 The second objective of the requirements of this subclause is to determine the event sequences leading to the hazardous events determined in 7.4.1.1.

7.4.1.3 The third objective of the requirements of this subclause is to determine the EUC risks associated with the hazardous events determined in 7.4.1.1.

NOTE 1 This subclause is necessary in order that the safety requirements for the E/E/PE safety-related systems are based on a systematic risk-based approach. This cannot be done unless the EUC and the EUC control system are considered.

NOTE 2 In application areas where valid assumptions can be made about the risks associated with the hazardous events and their consequences, the analysis required in this subclause (and 7.5) may be carried out by the developers of application sector versions of this standard, and may be embedded in simplified graphical requirements. Examples of such methods are given in IEC 61508-5, Annexes E and G.

7.4.2 Requirements

7.4.2.1 A hazard and risk analysis shall be undertaken which shall take into account information from the overall scope definition phase (see 7.3). If decisions are taken at later stages in the overall, E/E/PE system or software safety lifecycle phases that may change the basis on which the earlier decisions were taken, then a further hazard and risk analysis shall be undertaken.

NOTE 1 For guidance see references [9] and [10] in the Bibliography.

NOTE 2 As an example of the need to continue hazard and risk analysis deep into the overall safety lifecycle, consider the analysis of an EUC that incorporates a safety-related valve. A hazard and risk analysis may determine two event sequences, that include valve fails closed and valve fails open, leading to hazardous events. However, when the detailed design of the EUC control system controlling the valve is analyzed, a new failure mode, valve oscillates, may be discovered which introduces a new event sequence leading to a hazardous event.

7.4.2.2 Consideration shall be given to the elimination or reduction of the hazards.

NOTE Although not within the scope of this standard, it is of primary importance that identified hazards of the EUC are eliminated at source, for example by the application of inherent safety principles and the application of good engineering practice.

7.4.2.3 The hazards, hazardous events and hazardous situations of the EUC and the EUC control system shall be determined under all reasonably foreseeable circumstances (including fault conditions, reasonably foreseeable misuse and malevolent or unauthorised action). This shall include all relevant human factor issues, and shall give particular attention to abnormal or infrequent modes of operation of the EUC. If the hazard analysis identifies that malevolent or unauthorised action, constituting a security threat, as being reasonably foreseeable, then a security threats analysis should be carried out.

NOTE 1 For reasonably foreseeable misuse see 3.1.14 of IEC 61508-4.

NOTE 2 For guidance on hazard identification including guidance on representation and analysis of human factor issues, see reference [11] in the bibliography.

NOTE 3 For guidance on security risks analysis, see IEC 62443 series.

NOTE 4 Malevolent or unauthorised action covers security threats.

NOTE 5 The hazard and risk analysis should also consider whether the activation of a safety function due to a demand or spurious action will give rise to a new hazard. In such a situation it may be necessary to develop a new safety function in order to deal with this hazard.

7.4.2.4 The event sequences leading to the hazardous events determined in 7.4.2.3 shall be determined.

NOTE 1 The event sequences should be considered taking into account safety policy and risk management decisions.

NOTE 2 It is normally worthwhile to consider if any of the event sequences can be eliminated by modifications to the process design or equipment used.

7.4.2.5 The likelihood of the hazardous events for the conditions specified in 7.4.2.3 shall be evaluated.

7.4.2.6 The consequences associated with the hazardous events determined in 7.4.2.3 shall be determined.

7.4.2.7 The EUC risk shall be evaluated, or estimated, for each determined hazardous event.

7.4.2.8 The requirements of 7.4.2.1 to 7.4.2.7 can be met by the application of either qualitative or quantitative hazard and risk analysis techniques (see IEC 61508-5).

7.4.2.9 The appropriateness of the techniques, and the extent to which the techniques will need to be applied, will depend on a number of factors, including:

– the specific hazards and the consequences;
– the complexity of the EUC and the EUC control system;
– the application sector and its accepted good practices;
– the legal and safety regulatory requirements;
– the EUC risk;
– the availability of accurate data upon which the hazard and risk analysis is to be based.

7.4.2.10 The hazard and risk analysis shall consider the following:

– each determined hazardous event and the components that contribute to it;
– the consequences and likelihood of the event sequences with which each hazardous event is associated;
– the tolerable risk for each hazardous event;
– the measures taken to reduce or remove hazards and risks;
– the assumptions made during the analysis of the risks, including the estimated demand rates and equipment failure rates; any credit taken for operational constraints or human intervention shall be detailed.

7.4.2.11 The information and results that constitute the hazard and risk analysis shall be documented.

7.4.2.12 The information and results that constitute the hazard and risk analysis shall be maintained for the EUC and the EUC control system throughout the overall safety lifecycle, from the hazard and risk analysis phase to the decommissioning or disposal phase.

NOTE The maintenance of the information, arising from the results of the hazard and risk analysis phase, is a key means of tracking the progress on outstanding hazard and risk analysis issues.

7.5 Overall safety requirements

NOTE This phase is Box 4 of Figure 2.

7.5.1 Objective

The objective of the requirements of this subclause is to develop the specification for the overall safety requirements, in terms of the overall safety functions requirements and overall safety integrity requirements, for the E/E/PE safety-related systems and other risk reduction measures, in order to achieve the required functional safety.

NOTE In application areas where valid assumptions can be made about the risks, likely hazards, harmful events and their consequences, the analysis required in this subclause (and 7.4) may be carried out by the developers of application sector versions of this standard, and may be embedded in simplified graphical requirements. Examples of such methods are given in IEC 61508-5, Annexes E and F.

7.5.2 Requirements

7.5.2.1 A set of all necessary overall safety functions shall be developed based on the hazardous events derived from the hazard and risk analysis. This shall constitute the specification for the overall safety functions requirements.

NOTE 1 It will be necessary to create an overall safety function for each hazardous event.

NOTE 2 The overall safety functions to be performed will not, at this stage, be specified in technology-specific terms since the method and technology of implementation of the overall safety functions will not be known until later. During the allocation of overall safety requirements (see 7.6), the description of the safety functions may need to be modified to reflect the specific method of implementation.

EXAMPLE Prevent temperature in vessel X rising above 250 °C and prevent speed of drive Y exceeding 3 000 r/min are examples of overall safety functions.

7.5.2.2 If security threats have been identified, then a vulnerability analysis should be undertaken in order to specify security requirements.

NOTE Guidance is given in IEC 62443 series.

7.5.2.3 For each overall safety function, a target safety integrity requirement shall be determined that will result in the tolerable risk being met. Each requirement may be determined in a quantitative and/or qualitative manner. This shall constitute the specification for the overall safety integrity requirements.

NOTE 1 The specification of the overall safety integrity requirements is an interim stage towards the determination of the target failure measures and associated safety integrity levels for the safety functions to be implemented by the E/E/PE safety-related systems. Some of the qualitative methods used to determine the safety integrity levels (see IEC 61508-5, Annexes E and F) progress directly from the risk parameters to the safety integrity levels. In such cases, the safety integrity requirements are implicitly rather than explicitly stated because they are incorporated in the method itself.

NOTE 2 The EUC risk can be reduced either by reducing the consequences of the hazardous event (this is preferred), or by reducing the rate of hazardous events of the EUC and the EUC control system (see 7.5.2.4 below).

NOTE 3 The required reduction in frequency of the hazardous event can be achieved by additional measures comprising E/E/PE safety-related system(s) and/or other risk reduction measures including other technology safety-related systems or managed measures such as escape, occupancy or exposure time.

NOTE 4 In order to satisfy tolerable risk criteria, it may be necessary when determining the target safety integrity for each safety function to take into account that individuals may be exposed to risks from other sources.

NOTE 5 For situations where an application sector international standard exists that includes appropriate methods for directly determining the safety integrity requirements, then such standards may be used to meet the requirements of this subclause.

7.5.2.4 The overall safety integrity requirements shall be specified in terms of either

– the risk reduction required to achieve the tolerable risk, or

– the tolerable hazardous event rate so as to meet the tolerable risk.

7.5.2.5 If, in assessing the EUC risk, the average frequency of dangerous failures of a single EUC control system function is claimed as being lower than 10^{-5} dangerous failures per hour

then the EUC control system shall be considered to be a safety-related control system subject to the requirements of this standard.

NOTE For example, if a rate of dangerous failure between 10^{-6} and 10^{-5} dangerous failures per hour is claimed for the EUC control system, then the EUC control system is regarded as an E/E/PE safety-related system and the requirements appropriate to safety integrity level 1 would need to be met.

7.5.2.6 Where failures of the EUC control system place a demand on one or more E/E/PE safety-related systems and/or other risk reduction measures, and where the intention is not to designate the EUC control system as a safety-related system, the following requirements shall apply:

a) the rate of dangerous failure claimed for the EUC control system shall be supported by data acquired through one of the following:
   - actual operating experience of the EUC control system in a similar application;
   - a reliability analysis carried out to a recognised procedure;
   - an industry database of reliability of generic equipment;

b) the rate of dangerous failure that can be claimed for the EUC control system shall be no lower than 10^{-5} dangerous failures per hour;

NOTE 1 See 7.5.2.5.

c) all reasonably foreseeable dangerous failure modes of the EUC control system shall be taken into account in developing the specification for the overall safety requirements;

d) the EUC control system shall be independent from the E/E/PE safety-related systems and other risk reduction measures.

NOTE 2 Providing the safety-related systems have been designed to provide adequate safety integrity, taking into account the normal demand rate from the EUC control system, it will not be necessary to designate the EUC control system as a safety-related system (and, therefore, its functions will not be designated as safety functions within the context of this standard). In some applications, particularly where very high safety integrity is required, it may be appropriate to reduce the demand rate by designing the EUC control system to have a lower than normal failure rate. In such cases, if the failure rate claimed is less than the higher limit target safety integrity for safety integrity level 1 (see Table 3), then the control system will become safety-related and the requirements in this standard will apply.

NOTE 3 See 7.6.2.7 for meaning of independent.

7.5.2.7 If the requirements of 7.5.2.6 a) to d) inclusive cannot be met, then the EUC control system shall be designated as a safety-related system. The safety integrity level of functions of the EUC control system shall be determined by the rate of dangerous failure that is claimed for the EUC control system in accordance with Table 3 (see Note 3 of 7.6.2.9). In such cases, the requirements in this standard, relevant to the allocated safety integrity level, shall apply to the EUC control system.

NOTE See 7.5.2.5 and also 7.6.2.10.

7.6 Overall safety requirements allocation

NOTE This phase is Box 5 of Figure 2.

7.6.1 Objectives

7.6.1.1 The first objective of the requirements of this subclause is to allocate the overall safety functions, contained in the specification for the overall safety requirements (both the overall safety functions requirements and the overall safety integrity requirements), to the designated E/E/PE safety-related systems and other risk reduction measures.

NOTE Other risk reduction measures are considered of necessity, since the allocation to E/E/PE safety-related systems cannot be done unless these are taken into account.

7.6.1.2 The second objective of the requirements of this subclause is to allocate a target failure measure and an associated safety integrity level to each safety function to be carried out by an E/E/PE safety-related system.

7.6.2 Requirements

7.6.2.1 The designated safety-related systems that are to be used to achieve the required functional safety shall be specified. The tolerable risk may be met by

– E/E/PE safety-related systems; and/or

– other risk reduction measures.

NOTE This standard is applicable only if the tolerable risk is met at least in part by an E/E/PE safety-related system.

7.6.2.2 In allocating overall safety functions to the designated E/E/PE safety-related systems and other risk reduction measures, the skills and resources available during all phases of the overall safety lifecycle shall be considered.

NOTE 1 The full implications of using safety-related systems employing complex technology are often underestimated. For example, the implementation of complex technology requires a higher level of competence at all phases, from specification up to operation and maintenance. The use of other, simpler, technology solutions may be equally effective and may have several advantages because of the reduced complexity.

NOTE 2 The availability of skills and resources for operation and maintenance, and the operating environment, may be critical to achieving the required functional safety in actual operation.

7.6.2.3 Each overall safety function, with its associated overall safety integrity requirement developed according to 7.5, shall be allocated to one or more of the designated E/E/PE safety-related systems and/or other risk reduction measures, so that the tolerable risk for the safety function is achieved. This allocation is iterative, and if it is found that the tolerable risk cannot be achieved, then the specifications for the EUC control system, the designated E/E/PE safety-related systems and the other risk reduction measures shall be modified and the allocation repeated.

NOTE 1 The decision to allocate a specific overall safety function across one or more E/E/PE safety-related systems or other risk reduction measures will depend on a number of factors, but particularly on its overall safety integrity requirement. The more onerous the safety integrity requirement, the more likely the function will be shared by more than one E/E/PE safety-related system and/or other risk reduction measure.

NOTE 2 Figure 6 indicates the approach to overall safety requirements allocation.

7.6.2.4 The allocation indicated in 7.6.2.3 shall be done in such a way that all overall safety functions are allocated and target failure measures are defined for each safety function (subject to the requirements specified in 7.6.2.10).

7.6.2.5 The safety integrity requirements for each safety function shall be specified in terms of either

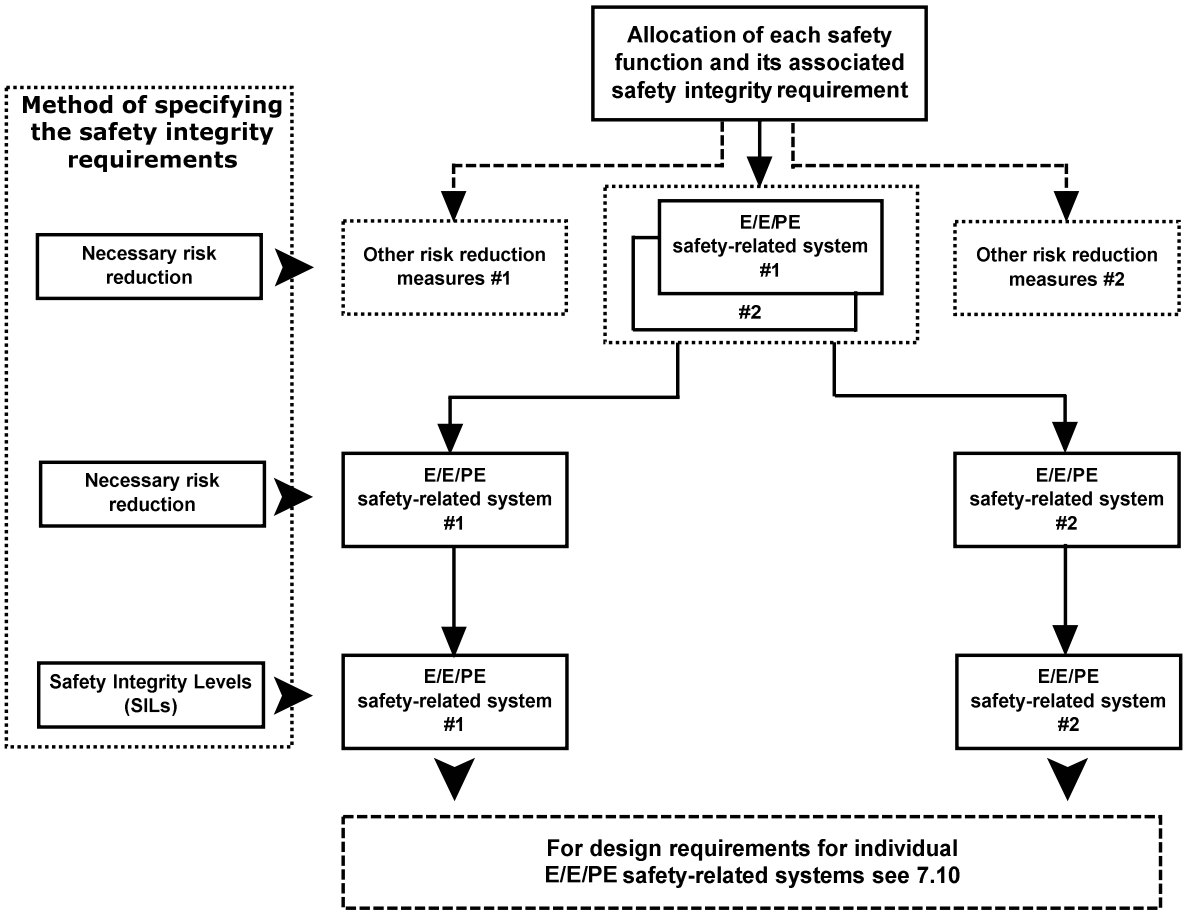
– the average probability of a dangerous failure on demand of the safety function, for a low demand mode of operation, or

– the average frequency of a dangerous failure of the safety function [h^{-1}] for a high demand or a continuous mode of operation.

7.6.2.6 The allocation of the safety integrity requirements shall be carried out using appropriate techniques for the combination of probabilities.

NOTE 1 Safety requirements allocation may be carried out in a qualitative and/or quantitative manner.

NOTE 2 Where a number of E/E/PE safety related systems and/or other risk reduction measures are necessary to achieve the tolerable risk, the actual risk achieved will depend on the systemic dependencies between the E/E/PE safety related systems and/or other risk reduction measures (see A.5.4 of IEC 61508-5 for more details of dependencies and how they can be analysed).



NOTE 1 Overall safety integrity requirements are associated with each overall safety function before allocation (see 7.5.2.3).

NOTE 2 An overall safety function may be allocated across more than one safety-related system.

Figure 6 – Allocation of overall safety requirements to E/E/PE safety-related systems and other risk reduction measures

7.6.2.7 The allocation shall proceed taking into account the possibility of common cause failures. If the EUC control system, E/E/PE safety-related systems and other risk reduction measures are to be treated as independent for the allocation, they shall:

– be independent such that the likelihood of simultaneous failures between two or more of these different systems or measures is sufficiently low in relation to the required safety integrity;

– be functionally diverse (i.e. use totally different approaches to achieve the same results);

– be based on diverse technologies (i.e. use different types of equipment to achieve the same results);

NOTE 1 It is recognised that, however diverse the technology, in the case of high safety integrity systems with particularly severe consequences in the event of failure, special precautions will have to be taken against low probability common cause events, for example aircraft crashes and earthquakes.

– not share common parts, services or support systems (for example power supplies) whose failure could result in a dangerous mode of failure of all systems;

– not share common operational, maintenance or test procedures.

NOTE 2 This standard is specifically concerned with the implementation of the safety requirements allocated to the E/E/PE safety-related systems, and requirements are specified as to how this shall be done. The implementation of safety requirements allocated to other risk reduction measures is therefore not considered in detail in this standard.

Within common cause analysis, limiting and constraint conditions for the realisation of E/E/PE safety-related systems such as the aspect of necessary separation of different channels of an E/E/PE system, subsystem or element, for example by space, shall be checked – this may not allow for example for two channels/microprocessors on one board or for on-chip redundancy (see IEC 61508-2, Annex E).

7.6.2.8 If not all of the requirements in 7.6.2.7 can be met then the E/E/PE safety-related systems and the other risk reduction measures shall not be treated as independent for the purposes of the safety allocation. Instead, the allocation shall take into account relevant common cause failures between the EUC control system, the E/E/PE safety-related systems and the other risk reduction measures.

NOTE 1 For further information on analysing dependent failures see references [13] and [14] in the Bibliography.

NOTE 2 Sufficient independence is established by showing that the probability of a dependent failure is sufficiently low for the E/E/PE safety-related systems in comparison with the overall safety integrity requirements (see 7.6.2.7).

NOTE 3 As indicated in 7.6.2.3, the allocation is iterative and, if an analysis that includes common cause failures indicates that the tolerable risk cannot be achieved based on initial assumptions, then design changes will be needed (for further guidance see A.5.4 of IEC 61508-5).

7.6.2.9 When the allocation has sufficiently progressed, the safety integrity requirements, for each safety function allocated to the E/E/PE safety-related system(s), shall be specified in terms of the safety integrity level in accordance with Table 2 or Table 3 and shall indicate whether the target failure measure is, either:

– the average probability of dangerous failure on demand of the safety function, (PFD_{avg}), for a low demand mode of operation (Table 2), or

– the average frequency of a dangerous failure of the safety function [h^{-1}], (PFH), for a high demand mode of operation (Table 3), or

– the average frequency of a dangerous failure of the safety function [h^{-1}], (PFH), for a continuous mode of operation (Table 3).

Table 2 – Safety integrity levels – target failure measures for a safety function operating in low demand mode of operation

Safety integrity level (SIL)	Average probability of a dangerous failure on demand of the safety function (PFD_{avg})
4	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-2}$ to $< 10^{-1}$

Table 3 – Safety integrity levels – target failure measures for a safety function operating in high demand mode of operation or continuous mode of operation

Safety integrity level (SIL)	Average frequency of a dangerous failure of the safety function [h^{-1}] (PFH)
4	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-6}$ to $< 10^{-5}$

NOTE 1 See 3.5.16 of IEC 61508-4 for definitions of the terms: “low demand mode of operation”, “high demand mode of operation” and “continuous mode of operation”.

NOTE 2 See IEC 61508-5 for guidance on modes of operation relating the target failure measures to the hazard and risk analysis.

NOTE 3 Tables 2 and 3 relate the target failure measures, as allocated to a safety function carried out by an E/E/PE safety-related system, to the safety integrity level. It is accepted that it will not be possible to predict quantitatively the safety integrity of all aspects of E/E/PE safety-related systems. Qualitative techniques, measures and judgements will have to be made with respect to the precautions considered necessary to ensure that the target failure measures are achieved. This is particularly true in the case of systematic safety integrity (see 3.5.6 of IEC 61508-4) where qualitative techniques and judgements have to be made with respect to the precautions considered necessary to achieve the required systematic safety integrity, for the specified safety integrity level (see IEC 61508-2, 7.4.2.2 c), 7.4.3, 7.4.6, 7.4.7 and IEC 61508-3).

NOTE 4 For hardware safety integrity it is necessary to apply quantified reliability estimation techniques in order to assess whether the target safety integrity, as determined by the risk assessment, has been achieved, taking into account random hardware failures (see IEC 61508-2, 7.4.5).

NOTE 5 When the safety integrity level has been determined using a qualitative method (for example a qualitative risk graph), either Table 2 or Table 3, as appropriate, gives the quantitative failure measures that set the limits for hardware safety integrity.

NOTE 6 The safety integrity that can be claimed when two or more E/E/PE safety-related systems are used may be better than that indicated in Table 2 providing that adequate levels of independence are achieved. For example, this would be relevant if the specified safety function was to carried out by two E/E/PE safety-related systems where adequate levels of independence between the two E/E/PE safety-related systems had been achieved.

NOTE 7 For an E/E/PE safety-related system operating in high demand or continuous mode of operation which is required to operate for a defined mission time during which no repair can take place, the required safety integrity level for a safety function can be derived as follows. Determine the required probability of failure of the safety function during the mission time and divide this by the mission time, to give a required frequency of failure per hour, then use Table 3 to derive the required safety integrity level.

7.6.2.10 For an E/E/PE safety-related system that implements safety functions of different safety integrity levels, unless it can be shown there is sufficient independence of implementation between these particular safety functions, those parts of the safety-related hardware and software where there is insufficient independence of implementation shall be treated as belonging to the safety function with the highest safety integrity level. Therefore, the requirements applicable to the highest relevant safety integrity level shall apply to all those parts.

NOTE See also IEC 61508-2, 7.4.2.4 and IEC 61508-3, 7.4.2.8.

7.6.2.11 In cases where the allocation process results in the requirement for an E/E/PE safety-related system implementing a SIL 4 safety function then the following shall apply:

a) There shall be a reconsideration of the application to determine if any of the risk parameters can be modified so that the requirement for a SIL 4 safety function is avoided. The review shall consider whether:

- additional safety-related systems or other risk reduction measures, not based on E/E/PE safety-related systems, could be introduced;
- the severity of the consequence could be reduced;
- the likelihood of the specified consequence could be reduced.

b) If after further consideration of the application, it is decided to implement the SIL 4 safety function then a further risk assessment shall be carried out using a quantitative method that takes into consideration potential common cause failures between the E/E/PE safety-related system and:
   - any other systems whose failure would place a demand on it; and,
   - any other safety-related systems.

7.6.2.12 No single safety function in an E/E/PE safety-related system shall be allocated a target safety integrity lower than specified in Tables 2 and 3. That is, for 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 the safety function 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 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.

7.6.2.13 The information and results of the overall safety requirements allocation acquired in 7.6.2.1 to 7.6.2.12, together with any assumptions and justifications made (including assumptions concerning the other risk reduction measures that need to be managed throughout the life of the EUC), shall be documented.

NOTE For each E/E/PE safety-related system, there should be sufficient information on the safety functions and their associated safety integrity levels. This information will form the basis of the safety requirements for the E/E/PE safety-related systems specified in 7.10.

7.7 Overall operation and maintenance planning

NOTE 1 This phase is Box 6 of Figure 2.

NOTE 2 An example of an operation and maintenance activities model is shown in Figure 7 hereinafter.

NOTE 3 An example of an operations and maintenance management model is shown in Figure 8 hereinafter.

NOTE 4 The requirements of 7.7.2 are specific to E/E/PE safety-related systems. They should be considered in the context of the other risk reduction measures, taking particular account of assumptions already made concerning other risk reduction measures that need to be managed throughout the life of the EUC.

NOTE 5 In order to achieve functional safety, similar requirements are necessary for all other risk reduction measures.

7.7.1 Objective

The objective of the requirements of this subclause is to develop a plan for operating and maintaining the E/E/PE safety-related systems, to ensure that the required functional safety is maintained during operation and maintenance.

7.7.2 Requirements

7.7.2.1 A plan shall be prepared that shall specify the following:

a) the routine actions that need to be carried out to maintain the required functional safety of the E/E/PE safety-related systems;
b) the actions and constraints that are necessary (for example during start-up, normal operation, routine testing, foreseeable disturbances, faults and shutdown) to prevent an

unsafe state, to reduce the demands on the E/E/PE safety-related system, or reduce the consequences of the harmful events;

NOTE 1 The following constraints, conditions and actions are relevant to E/E/PE safety-related systems:

1) constraints on the EUC operation during a fault of the E/E/PE safety-related systems;
2) constraints on the EUC operation during maintenance of the E/E/PE safety-related systems;
3) when constraints on the EUC operation may be removed;
4) the procedures for returning to normal operation;
5) the procedures for confirming that normal operation has been achieved;
6) the circumstances under which the safety functions implemented by the E/E/PE safety-related system may be by-passed for start-up, for special operation or for testing;
7) the procedures to be followed before, during and after by-passing E/E/PE safety-related systems, including permit to work procedures and authority levels.

c) the documentation that needs to be maintained showing results of functional safety audits and tests;

d) the documentation that needs to be maintained on all hazardous events and all incidents with the potential to create a hazardous event;

e) the scope of the maintenance activities (as distinct from the modification activities);

f) the actions to be taken in the event of hazardous events occurring;

g) the contents of the chronological documentation of operation and maintenance activities (see 7.15).

NOTE 2 The majority of E/E/PE safety-related systems have some failure modes that can be revealed only by testing during routine maintenance. In such cases, if testing is not carried out at sufficient frequency, the safety integrity requirements for the E/E/PE safety-related system will not be achieved.

NOTE 3 This subclause applies to a supplier of software who is required to provide information and procedures with the software product that will allow the user to ensure the required functional safety during the operation and maintenance of a safety-related system. This includes preparing procedures for any software modification that could come about as a consequence of an operational or maintenance requirement (see also 7.6 of IEC 61508-3). Implementing these procedures is covered by 7.8 of IEC 61508-3. Preparing procedures for future software changes that will come about as a consequence of a modification requirement for a safety-related system are dealt with in 7.6 of IEC 61508-3. Implementing those procedures is covered by 7.8 of IEC 61508-2.

NOTE 4 Account should be taken of the operation and maintenance procedures developed to meet the requirements in IEC 61508-2 and IEC 61508-3.

7.7.2.2 The plan shall ensure, that if any subsystem of an E/E/PE safety related system with a hardware fault tolerance of zero is taken off-line for testing, the continuing safety of the EUC shall be maintained by additional measures and constraints. The safety integrity provided by the additional measures and constraints shall be at least equal to the safety integrity provided by the E/E/PE safety-related system during normal operation. In the case of any subsystem of an E/E/PE safety related system with a hardware fault tolerance greater than zero then at least one channel of the E/E/PE safety-related system shall remain in operation during testing and the testing shall be completed within the MTTR assumed in the calculations carried out to determine compliance with the target failure measure.

NOTE For hardware fault tolerance see; 7.4.4.1 of IEC 61508-2.

7.7.2.3 The routine maintenance activities that are carried out to detect unrevealed faults shall be determined by a systematic analysis.

NOTE If unrevealed faults are not detected, they may

a) in the case of E/E/PE safety-related systems or other risk reduction measures, lead to a failure to operate on demand;

b) in the case of non-safety-related systems, lead to demands on the E/E/PE safety-related systems or other risk reduction measures.

7.7.2.4 The plan for maintaining the E/E/PE safety-related systems shall be agreed upon with those responsible for the operation and maintenance of

– the E/E/PE safety-related systems;
– the other risk reduction measures; and
– the non-safety-related systems that have the potential to place demands on the E/E/PE safety-related systems or other risk reduction measures.

7.8 Overall safety validation planning

NOTE 1 This phase is Box 7 of Figure 2.

NOTE 2 The requirements of this subclause are specific to E/E/PE safety-related systems. They should be considered in the context of the other risk reduction measures, taking particular account of assumptions already made concerning other risk reduction measures that need to be managed throughout the life of the EUC.

NOTE 3 In order to achieve functional safety, similar requirements are necessary for all other risk reduction measures.

7.8.1 Objective

The objective of the requirements of this subclause is to develop a plan for the overall safety validation of the E/E/PE safety-related systems

7.8.2 Requirements

7.8.2.1 A plan shall be developed that shall include the following:

a) details of when the validation shall take place;
b) details of those who shall carry out the validation;
c) specification of the relevant modes of the EUC operation with their relationship to the E/E/PE safety-related system, including where applicable
   – preparation for use, including setting and adjustment;
   – start up;
   – teach;
   – automatic;
   – manual;
   – semi-automatic;
   – steady state of operation;
   – re-setting;
   – shut down;
   – maintenance;
   – reasonably foreseeable abnormal conditions;
d) specification of the E/E/PE safety-related systems that need 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) the measures, techniques and procedures that shall be used for confirming that the allocation of safety functions has been carried out correctly; this shall include confirmation that each safety function conforms
   – with the specification for the overall safety functions requirements, and
   – to the specification for the overall safety integrity requirements;
g) specific reference to each element contained in the outputs from 7.5 and 7.6;
h) the required environment in which the validation activities are to take place (for example, for tests this would include calibrated tools and equipment);
i) the pass and fail criteria;

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

NOTE In planning the overall validation, account should be taken of the work planned for E/E/PE system safety validation and software safety validation as required by IEC 61508-2 and IEC 61508-3. It is important to ensure that all the interactions between two or more E/E/PE safety-related systems and/or other risk reduction measures are considered and that all safety functions (as specified in the outputs of 7.5) have been achieved.

7.8.2.2 The information from 7.8.2.1 shall be documented and shall constitute the plan for the overall safety validation of the E/E/PE safety-related systems.

7.9 Overall installation and commissioning planning

NOTE 1 This phase is Box 8 of Figure 2.

NOTE 2 The requirements of this subclause are specific to E/E/PE safety-related systems. They should be considered in the context of the other risk reduction measures, taking particular account of assumptions already made concerning other risk reduction measures that need to be managed throughout the life of the EUC.

NOTE 3 In order to achieve functional safety, similar requirements are necessary for all other risk reduction measures.

7.9.1 Objectives

7.9.1.1 The first objective of the requirements of this subclause is to develop a plan for the installation of the E/E/PE safety-related systems in a controlled manner, to ensure that the required functional safety is achieved.

7.9.1.2 The second objective of the requirements of this subclause is to develop a plan for the commissioning of the E/E/PE safety-related systems in a controlled manner, to ensure that the required functional safety is achieved.

7.9.2 Requirements

7.9.2.1 A plan for the installation of the E/E/PE safety-related systems shall be developed, specifying

a) the installation schedule;

b) those responsible for different parts of the installation;

c) the procedures for the installation;

d) the sequence in which the various elements are integrated;

e) the criteria for declaring all or parts of the E/E/PE safety-related systems ready for installation and for declaring installation activities complete;

f) procedures for the resolution of failures and incompatibilities.

7.9.2.2 A plan for the commissioning of the E/E/PE safety-related systems shall be developed, specifying:

a) the commissioning schedule;

b) those responsible for different parts of the commissioning;

c) the procedures for the commissioning;

d) the relationships to the different steps in the installation;

e) the relationships to the validation.

7.9.2.3 The overall installation and commissioning planning shall be documented.

7.10 E/E/PE system safety requirements specification

NOTE This phase is Box 9 of Figure 2.

7.10.1 Objective

The objective of the requirements of this subclause is to define the E/E/PE system safety requirements, in terms of the E/E/PE system safety functions requirements and the E/E/PE system safety integrity requirements, in order to achieve the required functional safety.

7.10.2 Requirements

7.10.2.1 The E/E/PE system safety requirements specification shall be derived from the allocation of safety requirements specified in 7.6 together with all relevant information related to the application. This information shall be made available to the E/E/PE safety-related system developer.

7.10.2.2 The E/E/PE system safety requirements specification shall contain requirements for the safety functions and their associated safety integrity levels.

NOTE The objective is to describe, in terms not specific to the equipment, the safety functions and their required functional safety performance. The specification can then be verified against the outputs of the overall safety requirements and the overall safety requirements allocation phases, and used as a basis of the realisation of the E/E/PE system (see 7.2 of IEC 61508-2). Equipment designers can use the specification as a basis for selecting the equipment and architecture.

7.10.2.3 The E/E/PE system safety requirements specification shall be made available to the developer of the E/E/PE safety-related system.

7.10.2.4 The E/E/PE system safety requirements specification shall be expressed and structured in such a way that it

a) is clear, precise, unambiguous, verifiable, testable, maintainable and feasible;
b) is written to aid comprehension by those who are likely to utilise the information at any stage of the E/E/PE system safety lifecycle;
c) is expressed in natural or formal language and/or logic, sequence or cause and effect diagrams that define the necessary safety functions with each safety function being individually defined.

7.10.2.5 The specification of the E/E/PE system safety requirements shall contain the requirements for the E/E/PE system safety functions (see 7.10.2.6) and the requirements for E/E/PE system safety integrity (see 7.10.2.7).

7.10.2.6 The E/E/PE system safety functions requirements specification shall contain:

a) a description of all the safety functions necessary to achieve the required functional safety, which shall, for each safety function,
   – provide comprehensive detailed requirements sufficient for the design and development of the E/E/PE safety-related systems,
   – include the manner in which the E/E/PE safety-related systems are intended to achieve or maintain a safe state for the EUC,
   – specify whether or not continuous control is required, and for what periods, in achieving or maintaining a safe state of the EUC, and
   – specify whether the safety function is applicable to E/E/PE safety-related systems operating in low demand, high demand or continuous modes of operation;
b) response time performance (i.e. the time within which it is necessary for the safety function to be completed);
c) E/E/PE safety-related system and operator interfaces that are necessary to achieve the required functional safety;
d) all information relevant to functional safety that may have an influence on the E/E/PE safety-related system design;

e) all interfaces, necessary for functional safety, between the E/E/PE safety-related systems and any other systems (either within, or outside, the EUC);

f) all relevant modes of operation of the EUC, including:

- preparation for use including setting and adjustment,
- start-up, teach, automatic, manual, semi-automatic, steady state of operation,
- steady state of non-operation, re-setting, shut-down, maintenance,
- reasonably foreseeable abnormal conditions;

NOTE Additional safety functions may be required for particular modes of operation (for example setting, adjustment or maintenance), to enable these operations to be carried out safely.

g) all required modes of behaviour of the E/E/PE safety-related systems shall be specified. In particular, the failure behaviour and the required response in the event of failure (for example alarms, automatic shut-down, etc.) of the E/E/PE safety-related systems.

7.10.2.7 The E/E/PE system safety integrity requirements specification shall contain:

a) the safety integrity level for each safety function and, when required, a specified value for the target failure measure;

NOTE 1 The specified value for the target failure measure can be derived using a quantitative method (see 7.5.2.3). Alternatively, when the safety integrity requirement has been developed using a qualitative method and expressed as a safety integrity level, then the target failure measure is derived from Table 2 or 3, as appropriate, according to the safety integrity level. In this case the specified target failure measure is the smallest average probability of failure or failure rate for the safety integrity level, unless a different value has been used to calibrate the method.

NOTE 2 In the case of a safety function operating in the low demand mode of operation, the target failure measure will be expressed in terms of the average probability of dangerous failure on demand, as determined by the safety integrity level of the safety function (see Table 2), unless there is a requirement in the E/E/PE system safety integrity requirements specification for the safety function to meet a specific target failure measure, rather than a specific safety integrity level. For example, when a target failure measure of $1{,}5 \times 10^{-2}$ (average probability of dangerous failure on demand) is specified in order to meet the required tolerable risk, then the average probability of dangerous failure on demand of the safety function due to random hardware failures will need to be equal to or less than $1{,}5 \times 10^{-2}$.

NOTE 3 In the case of a safety function operating in the high demand or the continuous mode of operation, the target failure measure will be expressed in terms of the average frequency of a dangerous failure [h^{-1}], as determined by the safety integrity level of the safety function (see Table 3), unless there is a requirement in the E/E/PE system safety integrity requirements specification for the safety function to meet a specific target failure measure, rather than a specific safety integrity level. For example, when a target failure measure of $1{,}5 \times 10^{-6}$ (average frequency of a dangerous failure [h^{-1}]) is specified in order to meet the required tolerable risk, then the average frequency of a dangerous failure of the safety function due to random hardware failures will need to be equal to or less than $1{,}5 \times 10^{-6}$ [h^{-1}].

b) the mode of operation (low demand, high demand or continuous) of each safety function;

c) the required duty cycle and lifetime;

d) the requirements, constraints, functions and facilities to enable the proof testing of the E/E/PE hardware to be undertaken;

NOTE 4 In developing the E/E/PE system safety requirements specification, the application in which the E/E/PE safety-related systems are to be used should be taken into consideration. This is particularly important for maintenance, where the specified proof test interval should not be less than can be reasonably expected for the particular application. For example, the time between services that can be realistically attained for mass-produced items used by the public is likely to be greater than in a more controlled application.

e) the extremes of all environmental conditions that are likely to be encountered during the E/E/PE system safety lifecycle including manufacture, storage, transport, testing, installation, commissioning, operation and maintenance;

f) the electromagnetic immunity limits that are required to achieve functional safety. These limits should be derived taking into account both the electromagnetic environment and the required safety integrity levels (see IEC/TS 61000-1-2);

NOTE 5 Due to the nature and physics of electromagnetic phenomena no simple, evident and provable correlation can be established between the required immunity level and safety integrity level for nearly all cases of electromagnetic phenomena. Specifying effective immunity levels solely according to the required SIL is therefore not possible and reasonable in those cases. Alternative approaches may be used which, to some degree, specify

the required immunity level according to the required SIL but also involve special test arrangements or test performance criteria. See IEC/TS 61000-1-2.

NOTE 6 See also reference [15] in the Bibliography.

g) limiting and constraint conditions for the realisation of E/E/PE safety-related systems due to the possibility of common cause failures (see 7.6.2.7).

7.11 E/E/PE safety-related systems – realisation

NOTE This phase is Box 10 of Figure 2 and Boxes 10.1 to 10.6 of Figures 3 and 4.

7.11.1 Objective

The objective of the requirements of this subclause is to create E/E/PE safety-related systems conforming to the specification for the E/E/PE system safety requirements (comprising the specification for the E/E/PE system safety functions requirements and the specification for the E/E/PE system safety integrity requirements). (See IEC 61508-2 and IEC 61508-3).

7.11.2 Requirements

The requirements that shall be met are contained in IEC 61508-2 and IEC 61508-3.

7.12 Other risk reduction measures – specification and realisation

NOTE This phase is Box 11 of Figure 2.

7.12.1 Objective

The objective of the requirements of this subclause is to create other risk reduction measures to meet the safety functions requirements and safety integrity requirements specified for such systems.

7.12.2 Requirements

The specification to meet the safety functions requirements and safety integrity requirements for other risk reduction measures is not covered in this standard.

NOTE Other risk reduction measures are based on a technology other than electrical/electronic/programmable electronic (for example hydraulic, pneumatic etc.) or may be physical structures (for example a drain system, a fire wall or a bund). They have been included in the overall safety lifecycle to ensure that the risk reduction from E/E/PE safety related systems is determined in the context of the risk reduction from other risk reduction measures.

7.13 Overall installation and commissioning

NOTE 1 This phase is Box 12 of Figure 2.

NOTE 2 The requirements of this subclause are specific to E/E/PE safety-related systems. They should be considered in the context of the other risk reduction measures, taking particular account of assumptions already made concerning other risk reduction measures that need to be managed throughout the life of the EUC.

NOTE 3 In order to achieve functional safety, similar requirements are necessary for all other risk reduction measures.

7.13.1 Objectives

7.13.1.1 The first objective of the requirements of this subclause is to install the E/E/PE safety-related systems.

7.13.1.2 The second objective of the requirements of this subclause is to commission the E/E/PE safety-related systems.

7.13.2 Requirements

7.13.2.1 Installation activities shall be carried out in accordance with the plan for the installation of the E/E/PE safety-related systems (see 7.9).

7.13.2.2 The information documented during installation shall include

– documentation of installation activities;
– resolution of failures and incompatibilities.

7.13.2.3 Commissioning activities shall be carried out in accordance with the plan for the commissioning of the E/E/PE safety-related systems.

7.13.2.4 The information documented during commissioning shall include

– documentation of commissioning activities;
– references to failure reports;
– resolution of failures and incompatibilities.

7.14 Overall safety validation

NOTE 1 This phase is Box 13 of Figure 2.

NOTE 2 The requirements of this subclause are specific to E/E/PE safety-related systems. They should be considered in the context of the other risk reduction measures, taking particular account of assumptions already made concerning other risk reduction measures that need to be managed throughout the life of the EUC.

NOTE 3 In order to achieve functional safety, similar requirements are necessary for all other risk reduction measures.

7.14.1 Objective

The objective of the requirements of this subclause is to validate that the E/E/PE safety-related systems meet the specification for the overall safety requirements in terms of the overall safety functions requirements and overall safety integrity requirements, taking into account the safety requirements allocation for the E/E/PE safety-related systems developed according to 7.6.

7.14.2 Requirements

7.14.2.1 Validation activities shall be carried out in accordance with the overall safety validation plan for the E/E/PE safety-related systems (see 7.8).

7.14.2.2 All equipment used for quantitative measurements as part of the validation activities shall be calibrated against a specification traceable to a national standard or to the vendor specification.

7.14.2.3 The information documented during validation shall include

– documentation in chronological form of the validation activities;
– the version of the specification for the overall safety requirements being used;
– the safety function being validated (by test or by analysis);
– tools and equipment used, along with calibration data;
– the results of the validation activities;
– configuration identification of the item under test, the procedures applied and the test environment;
– discrepancies between expected and actual results.

7.14.2.4 When discrepancies occur between expected and actual results, the analysis made, and the decisions taken on whether to continue the validation or issue a change request and return to an earlier part of the validation, shall be documented.

7.15 Overall operation, maintenance and repair

NOTE 1 This phase is Box 14 of Figure 2.

NOTE 2 The organizational measures dealt with in this subclause provide for the effective implementation of the technical requirements and are solely aimed at the achievement and maintenance of functional safety of the E/E/PE safety-related systems. The technical requirements necessary for maintaining functional safety will be specified as part of the information provided by the supplier of the E/E/PE safety-related system and its elements and components.

NOTE 3 The functional safety requirements during the maintenance and repair activities may be different from those required during operation.

NOTE 4 It should not be assumed that test procedures developed for initial installation and commissioning can be used without checking their validity and practicability in the context of on-line EUC operations.

NOTE 5 The requirements of this subclause are specific to E/E/PE safety-related systems. They should be considered in the context of the other risk reduction measures, taking particular account of assumptions already made concerning other risk reduction measures that need to be managed throughout the life of the EUC.

NOTE 6 In order to achieve functional safety, similar requirements are necessary for all other risk reduction measures.

7.15.1 Objective

7.15.1.1 The first objective of the requirements of this subclause is to ensure the functional safety of the E/E/PE safety-related systems is maintained to the specified level.

7.15.1.2 The second objective of the requirements of this subclause is to ensure that the technical requirements, necessary for the overall operation, maintenance and repair of the E/E/PE safety-related systems, are specified and provided to those responsible for the future operation and maintenance of the E/E/PE safety-related systems.

7.15.2 Requirements

7.15.2.1 The following shall be implemented:

– the plan for operating and maintaining the E/E/PE safety-related systems (see 7.7);

– the operation, maintenance and repair procedures for the E/E/PE safety-related systems.

7.15.2.2 Implementation of the items specified in 7.15.2.1 shall include initiation of the following actions:

– the implementation of procedures;

– the following of maintenance schedules;

– the maintaining of documentation;

– the carrying out, periodically, of functional safety audits (see 6.2.7);

– the documenting of modifications that have been made to the E/E/PE safety-related systems.

NOTE 1 An example of an operation and maintenance activities model is shown in Figure 7.

NOTE 2 An example of an operations and maintenance management model is shown in Figure 8.

7.15.2.3 Chronological documentation of operation, repair and maintenance of the E/E/PE safety-related systems shall be maintained which shall contain the following information:

– the results of functional safety audits and tests;

– documentation of the time and cause of demands on the E/E/PE safety-related systems (in actual operation), together with the performance of the E/E/PE safety-related systems when subject to those demands, and the faults found during routine maintenance;

– documentation of modifications that have been made to the EUC, to the EUC control system and to the E/E/PE safety-related systems.

7.15.2.4 The exact requirements for chronological documentation will be dependent on the specific product or application and shall, where relevant, be detailed in product and application sector international standards.

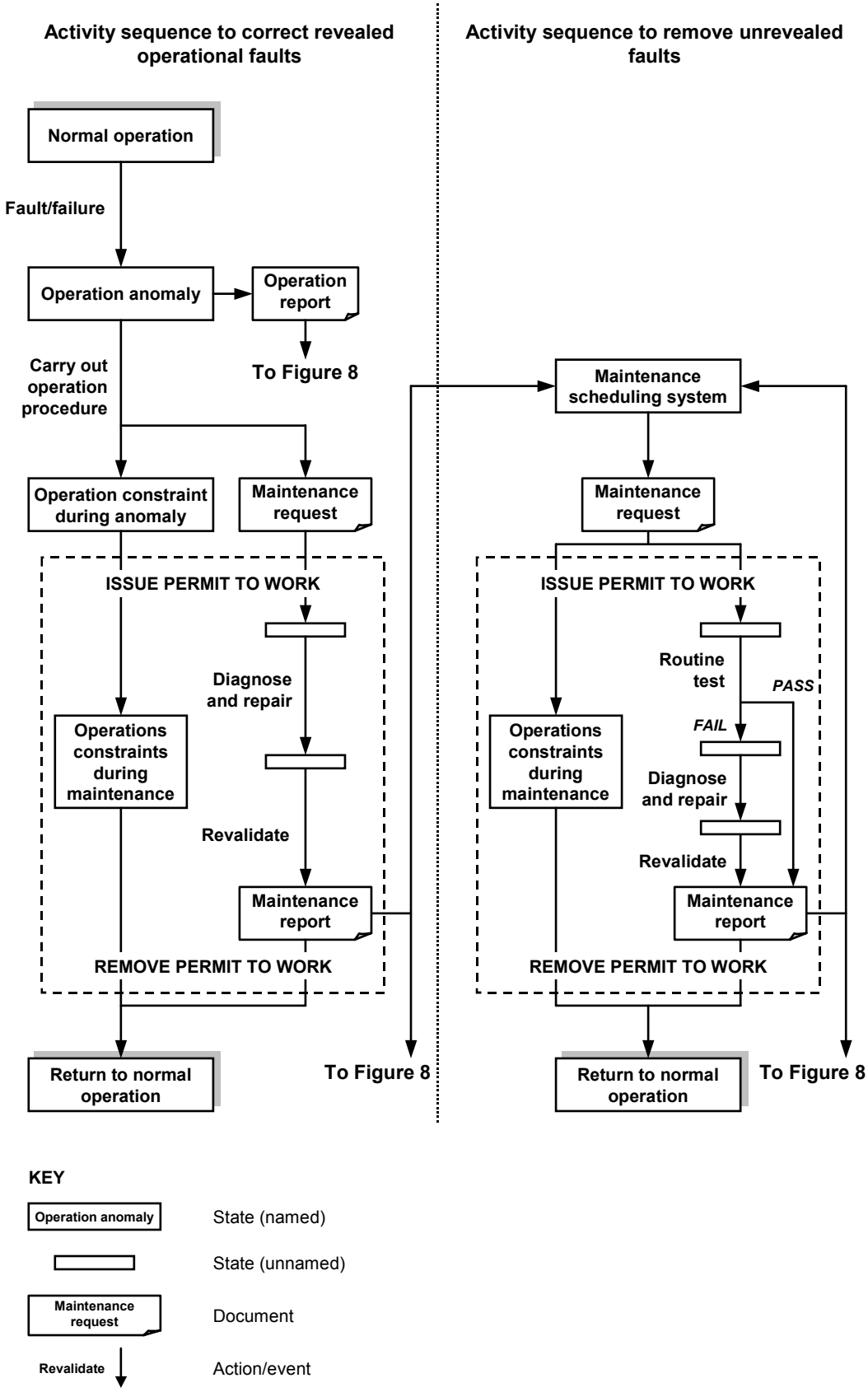


Figure 7 – Example of operations and maintenance activities model

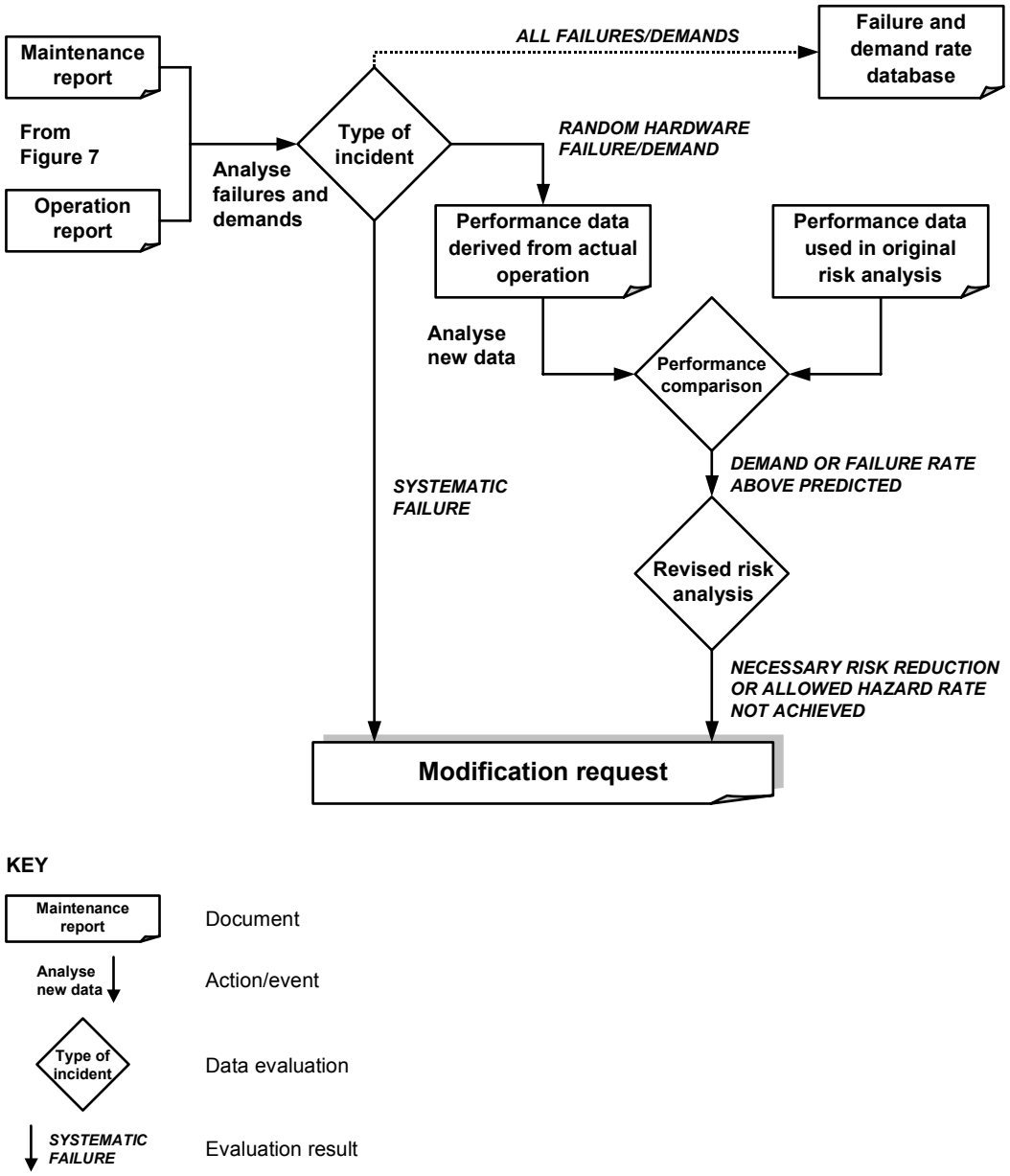


Figure 8 – Example of operation and maintenance management model

7.16 Overall modification and retrofit

NOTE 1 This phase is Box 15 of Figure 2.

NOTE 2 The organizational measures dealt with in this subclause provide for the effective implementation of the technical requirements, and are solely aimed at the achievement and maintenance of functional safety of the E/E/PE safety-related systems. The technical requirements necessary for maintaining functional safety will be specified as part of the information provided by the supplier of the E/E/PE safety-related system and its elements and components.

NOTE 3 The requirements of this subclause are specific to E/E/PE safety-related systems. They should be considered in the context of the other risk reduction measures, taking particular account of assumptions already made concerning other risk reduction measures that need to be managed throughout the life of the EUC.

NOTE 4 In order to achieve functional safety, similar requirements are necessary for all other risk reduction measures.

7.16.1 Objective

The objective of the requirements of this subclause is to define the procedures that are necessary to ensure that the functional safety for the E/E/PE safety-related systems is appropriate, both during and after the modification and retrofit phase has taken place.

7.16.2 Requirements

7.16.2.1 Prior to carrying out any modification or retrofit activity, procedures shall be planned (see 6.2.8).

NOTE An example of a modification procedure model is shown in Figure 9.

7.16.2.2 The modification and retrofit phase shall be initiated only by the issue of an authorized request under the procedures for the management of functional safety (see 6.2.8). The request shall detail the following:

– the determined hazards that may be affected;

– the proposed change (both hardware and software);

– the reasons for the change.

NOTE The reason for the request for the modification could arise from, for example:

a) functional safety below that specified;

b) systematic fault experience;

c) new or amended safety legislation;

d) modifications to the EUC or its use;

e) modification to the overall safety requirements;

f) analysis of operations and maintenance performance, indicating that the performance is below target;

g) routine functional safety audits.

7.16.2.3 An impact analysis shall be carried out that shall include an assessment of the impact of the proposed modification or retrofit activity on the functional safety of any E/E/PE safety-related system. The assessment shall include a hazard and risk analysis sufficient to determine the breadth and depth to which subsequent overall, E/E/PE system or software safety lifecycle phases will need to be undertaken. The assessment shall also consider the impact of other concurrent modification or retrofit activities, and shall also consider the functional safety both during and after the modification and retrofit activities have taken place.

7.16.2.4 The results described in 7.16.2.3 shall be documented.

7.16.2.5 Authorization to carry out the required modification or retrofit activity shall be dependent on the results of the impact analysis.

7.16.2.6 All modifications that have an impact on the functional safety of any E/E/PE safety-related system shall initiate a return to an appropriate phase of the overall, E/E/PE system or software safety lifecycles. All subsequent phases shall then be carried out in accordance with the procedures specified for the specific phases in accordance with the requirements in this standard.

NOTE 1 It may be necessary to implement a full hazard and risk analysis which may generate a need for safety integrity levels that are different to those currently specified for the safety functions implemented by the E/E/PE safety-related systems.

NOTE 2 It should not be assumed that test procedures developed for initial installation and commissioning can be used without checking their validity and practicability in the context of on-line EUC operations.

7.16.2.7 Chronological documentation shall be established and maintained that shall document details of all modifications and retrofits, and shall include references to:

– the modification or retrofit request;

– the impact analysis;

– reverification and revalidation of data and results;

– all documents affected by the modification and retrofit activity.

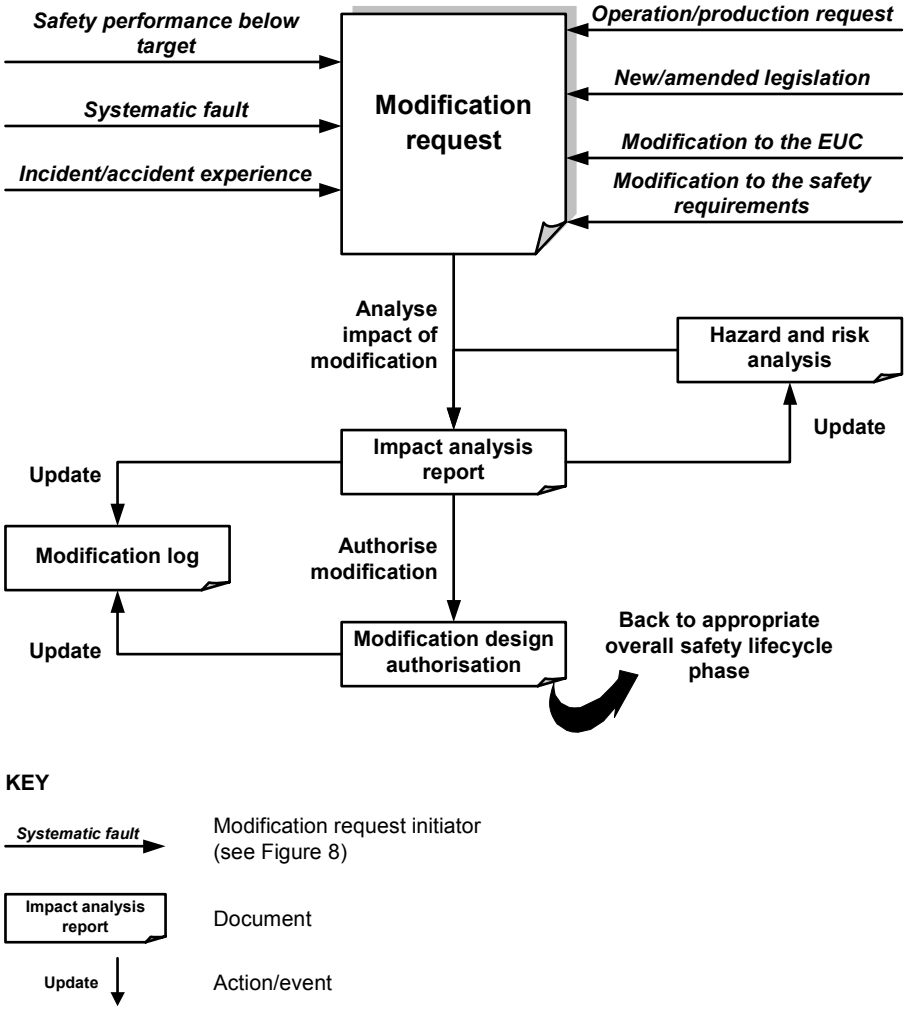


Figure 9 – Example of modification procedure model

7.17 Decommissioning or disposal

NOTE 1 This phase is Box 16 of Figure 2.

NOTE 2 The requirements of this subclause are specific to E/E/PE safety-related systems. They should be considered in the context of the other risk reduction measures, taking particular account of assumptions already made concerning other risk reduction measures that need to be managed throughout the life of the EUC.

NOTE 3 In order to achieve functional safety, similar requirements are necessary for all other risk reduction measures.

7.17.1 Objective

The objective of the requirements of this subclause is to define the procedures that are necessary to ensure that the functional safety for the E/E/PE safety-related systems is appropriate for the circumstances during and after the activities of decommissioning or disposing of the EUC.

7.17.2 Requirements

7.17.2.1 Prior to any decommissioning or disposal activity, an impact analysis shall be carried out that shall include an assessment of the impact of the proposed decommissioning or disposal activity on the functional safety of any E/E/PE safety-related system associated with the EUC. The impact analysis shall also consider adjacent EUCs and the impact on their E/E/PE safety-related systems. The assessment shall include a hazard and risk analysis sufficient to determine the necessary breadth and depth of subsequent overall, E/E/PE system or software safety lifecycle phases.

7.17.2.2 The results described in 7.17.2.1 shall be documented.

7.17.2.3 The decommissioning or disposal phase shall only be initiated by the issue of an authorized request under the procedures for the management of functional safety (see Clause 6).

7.17.2.4 Authorization to carry out the required decommissioning or disposal shall be dependent on the results of the impact analysis.

7.17.2.5 Prior to decommissioning or disposal taking place a plan shall be prepared that shall include procedures for:

– the closing down of the E/E/PE safety-related systems;

– dismantling the E/E/PE safety-related systems.

7.17.2.6 If any decommissioning or disposal activity has an impact on the functional safety of any E/E/PE safety-related system, this shall initiate a return to the appropriate phase of the overall, E/E/PE system or software safety lifecycles. All subsequent phases shall then be carried out in accordance with the procedures specified in this standard for the safety integrity levels of the safety functions implemented by the E/E/PE safety-related systems.

NOTE 1 It may be necessary to implement a full hazard and risk analysis which may generate a need for different safety integrity levels for the safety functions implemented by the E/E/PE safety-related systems.

NOTE 2 The functional safety requirements during the decommissioning or disposal phase may be different from those required during the operational phase.

7.17.2.7 Chronological documentation shall be established and maintained that shall document details of the decommissioning or disposal activities and shall include references to:

– the plan used for the decommissioning or disposal activities;

– the impact analysis.

7.18 Verification

7.18.1 Objective

The objective of the requirements of this subclause is to demonstrate, for each phase of the overall, E/E/PE system and software safety lifecycles (by review, analysis and/or tests), that the outputs meet in all respects the objectives and requirements specified for the phase.

7.18.2 Requirements

7.18.2.1 For each phase of the overall, E/E/PE system and software safety lifecycles, a plan for the verification shall be established concurrently with the development for the phase.

7.18.2.2 The verification plan shall document or refer to the criteria, techniques, tools to be used in the verification activities.

7.18.2.3 The verification shall be carried out according to the verification plan.

NOTE Selection of techniques and measures for verification, and the degree of independence for the verification activities, will depend upon a number of factors and may be specified in product and application sector international standards.

The factors could include, for example

– size of project;

– degree of complexity;

– degree of novelty of design;

– degree of novelty of technology.

7.18.2.4 Information on the verification activities shall be collected and documented as evidence that the phase being verified has, in all respects, been satisfactorily completed.

8 Functional safety assessment

8.1 Objective

The objective of the requirements of this clause is to specify the activities necessary to investigate and arrive at a judgement on the adequacy of the functional safety achieved by the E/E/PE safety-related system(s) or compliant items (e.g. elements/subsystems)based on compliance with the relevant clauses of this standard.

8.2 Requirements

8.2.1 One or more persons shall be appointed to carry out one or more functional safety assessments in order to arrive at a judgement on the adequacy of:

– the functional safety achieved by the E/E/PE safety-related systems, within their particular environment, in respect to the relevant clauses of this standard;

– the compliance to the relevant clauses of this standard, achieved in the case of elements/subsystems.

8.2.2 Those carrying out a functional safety assessment shall have access to all persons involved in any overall, E/E/PE system or software safety lifecycle activity and all relevant information and equipment (both hardware and software).

NOTE It is recognised that access to those persons who where previously involved in a safety lifecycle phase may not be achievable and in such a case reliance has necessarily to be placed on those persons currently having relevant responsibilities.

8.2.3 A functional safety assessment shall be applied to all phases throughout the overall, E/E/PE system and software safety lifecycles, including documentation, verification and management of functional safety.

8.2.4 Those carrying out a functional safety assessment shall consider the activities carried out and the outputs obtained during each phase of the overall, E/E/PE system and software safety lifecycles and judge whether adequate functional safety has been achieved based on the objectives and requirements in this standard.

8.2.5 All relevant claims of compliance made by suppliers and other parties responsible for achieving functional safety shall be included in the functional safety assessment.

NOTE Such claims may be made for an operational system or for the contribution to functional safety of activities and/or equipment in each phase of the overall, E/E/PE system and software safety lifecycles.

8.2.6 A functional safety assessment may be carried out after each phase of the overall, E/E/PE system and software safety lifecycles, or after a number of safety lifecycle phases, subject to the overriding requirement that a functional safety assessment shall be undertaken prior to the determined hazards being present.

8.2.7 A functional safety assessment shall include assessment of the evidence that functional safety audit(s) have been carried out (either full or partial) relevant to its scope.

8.2.8 Each functional safety assessment shall consider at least the following:

– the work done since the previous functional safety assessment;

– the plans or strategy for implementing further functional safety assessments of the overall, E/E/PE system and software safety lifecycles;

– the recommendations of the previous functional safety assessments and the extent to which changes have been made to meet them.

8.2.9 Each functional safety assessment shall be planned. The plan shall specify all information necessary to facilitate an effective assessment, including:

– the scope of the functional safety assessment;
– the organisations involved;
– the resources required;
– those to undertake the functional safety assessment;
– the level of independence of those undertaking the functional safety assessment;
– the competence of each person involved in the functional safety assessment;
– the outputs from the functional safety assessment;
– how the functional safety assessment relates to, and shall be integrated with, other functional safety assessments where appropriate (see 6.2.1).

NOTE 1 In establishing the scope of each functional safety assessment, it will be necessary to specify the documents, and their revision status, that are to be used as inputs for each assessment activity.

NOTE 2 The plan can be made by either those responsible for functional safety assessment or those responsible for management of functional safety, or can be shared between them.

8.2.10 Prior to a functional safety assessment taking place, its plan shall be approved by those carrying it out and by those responsible for the management of functional safety.

8.2.11 At the conclusion of a functional safety assessment, those carrying out the assessment shall document, in accordance with the assessment's plans and terms of reference:

– the activities conducted;
– the findings made;
– the conclusions arrived at;
– a judgement on the adequacy of functional safety in accordance with the requirements of this standard;
– recommendations that arise from the assessment, including recommendations for acceptance, qualified acceptance or rejection.

8.2.12 The relevant outputs of the functional safety assessment of a compliant item shall be made available to those having responsibilities for any overall, E/E/PE system or software safety lifecycle activity including the designers and assessors of the E/E/PE safety-related system. The output of the assessment of the E/E/PE safety-related system shall be made available to the E/E/PE system integrator.

NOTE A compliant item is any item (e.g. an element) on which a claim is being made with respect the clauses of IEC 61508 series.

8.2.13 The output of the functional safety assessment of a compliant item shall include the following information to facilitate the re-use of the assessment results in the context of a larger system (see Annex D of IEC 61508-2; Annex D of IEC 61508-3 and 3.8.17 of IEC 61508-4).

a) the precise identification of the compliant item including the version of its hardware and software;

NOTE If the compliant item was assessed as a part of a larger system or equipment family, the precise identification of that system or equipment family should also be documented.

b) the conditions assumed during the assessment (e.g. the conditions of use of the E/E/PE safety-related system);

c) reference to the documentation evidence on which the assessment conclusion was based;

d) the procedures, methods and tools used for assessing the systematic capability along with the justification of its effectiveness;

e) the procedures, methods and tools used for assessing the hardware safety integrity together with the justification of the approach adopted and the quality of the data (e.g. the failure rate/distribution data sources);

f) the assessment results obtained in relation to the requirements of this standard and to the specification of the safety characteristics of the compliant item in its safety manual;

g) the accepted deviations to IEC 61508 requirements, with corresponding explanation and / or reference to evidence contained in documentation.

8.2.14 Those carrying out a functional safety assessment shall be competent for the activities to be undertaken, according to the requirements of 6.2.13 to 6.2.15.

8.2.15 The minimum level of independence of those carrying out a functional safety assessment shall be as specified in Tables 4 and 5. Product and application sector international standards may specify, with respect to compliance to their standards, different levels of independence to those specified in Tables 4 and 5. The tables shall be interpreted as follows:

– X: the level of independence specified is the minimum for the specified consequence (Table 4) or safety integrity level/systematic capability (Table 5). If a lower level of independence is adopted, then the rationale for using it shall be detailed.

– X1 and X2: see 8.2.16.

– Y: the level of independence specified is considered insufficient for the specified consequence (Table 4) or safety integrity level/ systematic capability (Table 5).

8.2.16 In the context of Tables 4 and 5, only cells marked X, X1, X2 or Y shall be used as a basis for determining the level of independence. For cells marked X1 or X2, either X1 or X2 is applicable (not both), depending on a number of factors specific to the application. The rationale for choosing X1 or X2 should be detailed. Factors that will make X2 more appropriate than X1 are:

– lack of previous experience with a similar design;

– greater degree of complexity;

– greater degree of novelty of design;

– greater degree of novelty of technology.

NOTE 1 Depending upon the company organization and expertise within the company, the requirement for independent persons and departments may have to be met by using an external organization. Conversely, companies that have internal organizations skilled in risk assessment and the application of safety-related systems, that are independent of and separate (by ways of management and other resources) from those responsible for the main development, may be able to use their own resources to meet the requirements for an independent organization.

NOTE 2 See 3.8.11, 3.8.12 and 3.8.13 of IEC 61508-4 for definitions of independent person, independent department, and independent organization respectively.

NOTE 3 Those carrying out a functional safety assessment should be careful in offering advice on anything within the scope of the assessment, since this could compromise their independence. It is often appropriate to give advice on aspects that could incur a judgement of inadequate safety, such as a shortfall in evidence, but it is usually inappropriate to offer advice or give recommendations for specific remedies for these or other problems.

8.2.17 In the context of Table 4, the consequence values for the specified level of independence are:

– Consequence A: minor injury (for example temporary loss of function);

– Consequence B: serious permanent injury to one or more persons, death to one person;

– Consequence C: death to several people;

– Consequence D: very many people killed.

The consequences specified in Table 4 are those that would arise in the event of failure of all the risk reduction measures including the E/E/PE safety-related systems.

8.2.18 In the context of Table 5, the minimum levels of independence shall be based on the safety function, carried out by the E/E/PE safety-related system, that has the highest safety integrity level or for elements/subsystems, the highest systematic capability, specified in terms of the safety integrity level.

Table 4 – Minimum levels of independence of those carrying out functional safety assessment (overall safety lifecycle phases 1 to 8 and 12 to 16 inclusive (see Figure 2))

Minimum level of independence	Consequence (see 8.2.17)			
	A	B	C	D
Independent person	X	X1	Y	Y
Independent department		X2	X1	Y
Independent organization			X2	X
NOTE See 8.2.15, 8.2.16 and 8.2.17 for details on interpreting this table.				

Table 5 – Minimum levels of independence of those carrying out functional safety assessment (overall safety lifecycle phases 9 and 10, including all phases of E/E/PE system and software safety lifecycles (see Figures 2, 3 and 4))

Minimum level of independence	Safety integrity level/Systematic capability			
	1	2	3	4
Independent person	X	X1	Y	Y
Independent department		X2	X1	Y
Independent organization			X2	X
NOTE See 8.2.15, 8.2.16 and 8.2.18 for details on interpreting this table.				

Annex A
(informative)

Example of a documentation structure

A.1 General

This annex provides an example documentation structure and method for specifying the documents for structuring the information in order to meet the requirements in Clause 5. The documentation has to contain sufficient information necessary to effectively perform

– each phase of the overall, E/E/PE system and software safety lifecycles;

– the management of functional safety (Clause 6);

– functional safety assessments (Clause 8).

What constitutes sufficient information will be dependent upon a number of factors, including the complexity and size of the E/E/PE safety-related systems and the requirements relating to the specific application. The necessary documentation may be specified in product and application specific international standards.

The amount of information in each document may vary from a few lines to many pages, and the complete set of information may be divided and presented in many physical documents or one physical document. The physical documentation structure will again depend upon the size and complexity of the E/E/PE safety-related systems, and will take into account company procedures and the working practices of the specific product or application sector.

The example documentation structure indicated in this annex has been provided to illustrate one particular way in which the information could be structured and the way the documents could be titled. See reference [7] in the Bibliography for more details.

A document is a structured amount of information intended for human perception, that may be interchanged as a unit between users and/or systems (see reference [16] in the Bibliography). The term applies therefore not only to documents in the traditional sense, but also to concepts such as data files and database information.

In this standard, the term document is understood normally to mean information rather than physical documents, unless this is explicitly declared or understood in the context of the clause or subclause in which it is stated. Documents may be available in different forms for human presentation (for example on paper, film or any data medium to be presented on screens or displays).

The example documentation structure in this annex specifies documents in two parts:

– **document kind**;

– **activity or object**.

The document kind is defined in Bibliography reference [16] and characterizes the content of the document, for example function description or circuit diagram. The activity or object describes the scope of the content, for example pump control system.

The basic document kinds specified in this annex are

– **specification** – specifies a required function, performance or activity (for example requirements specification);

– **description** – specifies a planned or actual function, design, performance or activity (for example function description);

– **instruction** – specifies in detail the instructions as to when and how to perform certain jobs (for example operator instruction);
– **plan** – specifies the plan as to when, how and by whom specific activities shall be performed (for example maintenance plan);
– **diagram** – specifies the function by means of a diagram (symbols and lines representing signals between the symbols);
– **list** – provides information in a list form (for example code list, signal list);
– **log** – provides information on events in a chronological log form;
– **report** – describes the results of activities such as investigations, assessments, tests etc. (for example test report);
– **request** – provides a description of requested actions that have to be approved and further specified (for example maintenance request).

The basic document kind may have a prefix, such as **requirements** specification or **test** specification, which further characterizes the content.

A.2 Safety lifecycle document structure

Tables A.1, A.2 and A.3 provide an example documentation structure for structuring the information in order to meet the requirements specified in Clause 5. The tables indicate the safety lifecycle phase that is mainly associated with the documents (usually the phase in which they are developed). The names given to the documents in the tables are in accordance with the scheme outlined in A.1.

In addition to the documents listed in Tables A.1, A.2 and A.3, there may be supplementary documents giving detailed additional information or information structured for a specific purpose, for example parts lists, signal lists, cable lists, wiring tables, loop diagrams, list of variables.

NOTE Examples of such variables are values for regulators, alarm values for variables, priorities in the execution of tasks in the computer. Some of the values of the variables could be given before the delivery of the system, others could be given during commissioning or maintenance.

Table A.1 – Example of a documentation structure for information related to the overall safety lifecycle

Overall safety lifecycle phase	Information
Concept	Description (overall concept)
Overall scope definition	Description (overall scope definition)
Hazard and risk analysis	Description (hazard and risk analysis)
Overall safety requirements	Specification (overall safety requirements, comprising: overall safety functions requirements and overall safety integrity requirements)
Overall safety requirements allocation	Description (overall safety requirements allocation)
Overall operation and maintenance planning	Plan (overall operation and maintenance)
Overall safety validation planning	Plan (overall safety validation)
Overall installation and commissioning planning	Plan (overall installation); Plan (overall commissioning)
E/E/PE system safety requirements	Specification (E/E/PE system safety requirements, comprising: E/E/PE system safety functions requirements and E/E/PE system safety integrity requirements)
E/E/PE safety related system realisation	See Table A.2 and Table A.3
Overall installation and commissioning	Report (overall installation); Report (overall commissioning)
Overall safety validation	Report (overall safety validation)
Overall operation and maintenance	Log (overall operation and maintenance)
Overall modification and retrofit	Request (overall modification); Report (overall modification and retrofit impact analysis); Log (overall modification and retrofit)
Decommissioning or disposal	Report (overall decommissioning or disposal impact analysis); Plan (overall decommissioning or disposal); Log (overall decommissioning or disposal)
Concerning all phases	Plan (safety); Plan (verification); Report (verification); Plan (functional safety assessment); Report (functional safety assessment)

Table A.2 – Example of a documentation structure for information related to the E/E/PE system safety lifecycle

E/E/PE system safety lifecycle phase	Information
E/E/PE system validation planning	Plan (E/E/PE system safety validation)
E/E/PE system design and development	
E/E/PE system architecture	Description (E/E/PE system architecture design, comprising: hardware architecture and software architecture); Specification (programmable electronic integration tests); Specification (integration tests of programmable electronic and non programmable electronic hardware)
Hardware architecture	Description (hardware architecture design); Specification (hardware architecture integration tests)
Hardware module design	Specification (hardware module design); Specifications (hardware module tests)
Component construction and/or procurement	Hardware modules; Report (hardware modules tests)
Programmable electronic integration	Report (programmable electronic hardware and software integration tests) (see Table A.3)
E/E/PE system integration	Report (programmable electronic and other hardware integration tests)
E/E/PE system operation and maintenance procedures	Instruction (user); Instruction (operation and maintenance)
E/E/PE system safety validation	Report (E/E/PE system safety validation)
E/E/PE system modification	Instruction (E/E/PE system modification procedures); Request (E/E/PE system modification); Report (E/E/PE system modification impact analysis); Log (E/E/PE system modification)
Concerning all phases	Plan (E/E/PE system safety); Plan (E/E/PE system verification); Report (E/E/PE system verification); Plan (E/E/PE system functional safety assessment); Report (E/E/PE system functional safety assessment)
Concerning all relevant phases	Safety manual for compliant items

Table A.3 – Example of a documentation structure for information related to the software safety lifecycle

Software safety lifecycle phase	Information
Software safety requirements	Specification (software safety requirements, comprising: software safety functions requirements and software safety integrity requirements)
Software validation planning	Plan (software safety validation)
Software design and development	
Software architecture	Description (software architecture design) (see Table A.2 for hardware architecture design description); Specification (software architecture integration tests); Specification (programmable electronic hardware and software integration tests); Instruction (development tools and coding manual)
Software system design	Description (software system design); Specification (software system integration tests)
Software module design	Specification (software module design); Specification (software module tests)
Coding	List (source code); Report (software module tests); Report (code review)
Software module testing	Report (software module tests)
Software integration	Report (software module integration tests); Report (software system integration tests); Report (software architecture integration tests)
Programmable electronic integration	Report (programmable electronic hardware and software integration tests)
Software operation and maintenance procedures	Instruction (user); Instruction (operation and maintenance)
Software safety validation	Report (software safety validation)
Software modification	Instruction (software modification procedures); Request (software modification); Report (software modification impact analysis); Log (software modification)
Concerning all phases	Plan (software safety); Plan (software verification); Report (software verification); Plan (software functional safety assessment); Report (software functional safety assessment)
Concerning all relevant phases	Safety manual for compliant items

A.3 Physical document structure

The physical structure of the documentation is the way that the different documents are combined into documents, document sets, binders and groups of binders. The same document may occur in different sets.

For a large and complex system, the many physical documents are likely to be split into several binders. For a small, low complexity system with a limited number of physical documents, they may be combined into one binder with different tabs for the different sets of documents. Figure A.1 shows examples of combining documents into binders according to user groups.

The physical structure provides a means of selecting the documentation needed for the specific activities by the person or group of persons performing the activities. Consequently, some of the physical documents may occur in several binder sets or other media (for example computer disks).

NOTE The information required by the documents in Table A.1 may be contained within the different sets of documents shown in Figure A.1. For example, the engineering set may contain the hazard and risk analysis description and the overall safety requirements specification.

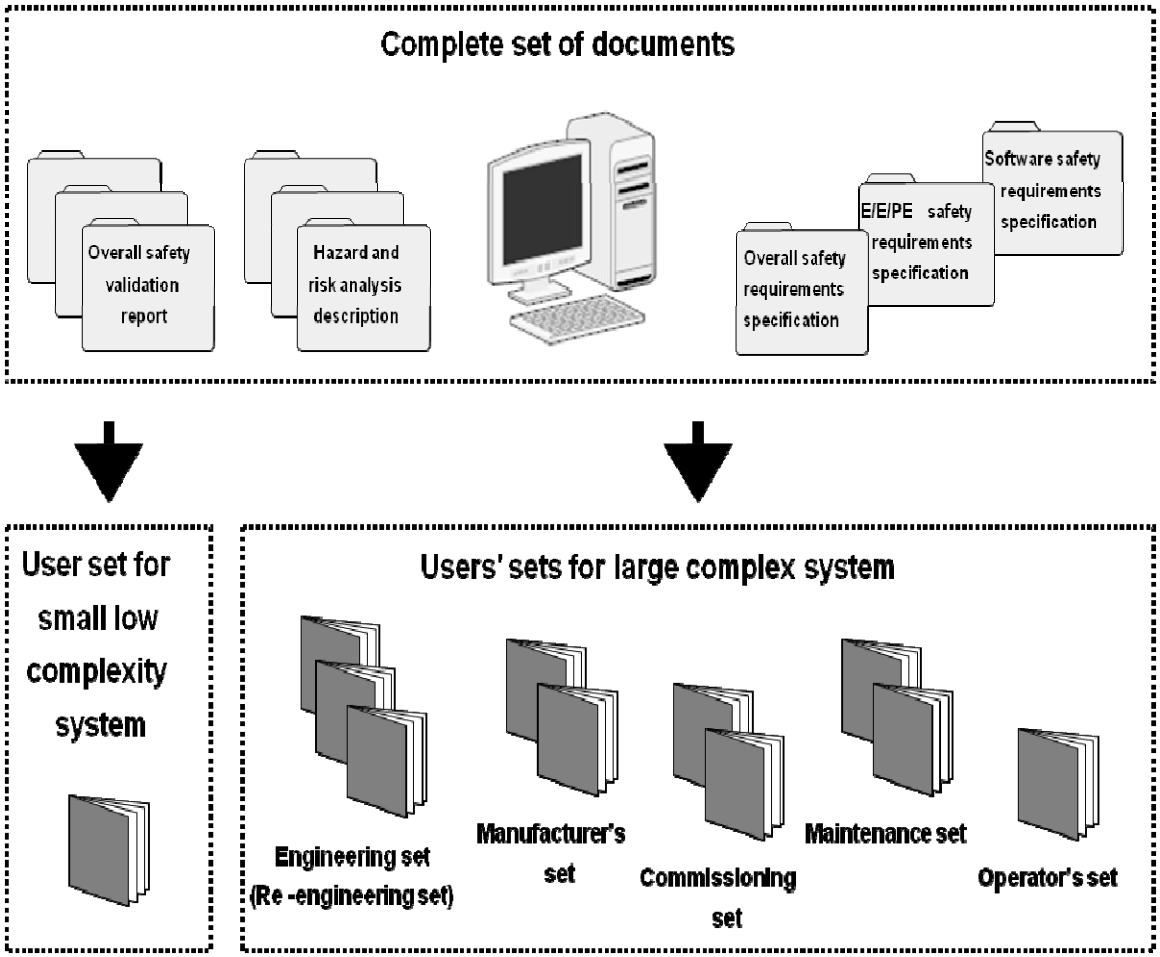


Figure A.1 – Structuring information into document sets for user groups

A.4 List of documents

The list of documents will typically include the following information:

– drawing or document number;
– revision index;
– document designation code;
– title;
– date of revision;
– data carrier.

This list may appear in different forms, for example in a database capable of being sorted according to drawing, document number or document designation code. The document designation code may contain the reference designation for the function, location or product described in the document, making it a powerful tool in searching for information.

Bibliography

[1] IEC 61511 (all parts), *Functional safety – Safety instrumented systems for the process industry sector*

[2] IEC 62061, *Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems*

[3] IEC 61800-5-2, *Adjustable speed electrical power drive systems – Part 5-2: Safety requirements – Functional*

[4] IEC/TR 61508-0:2005, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 0: Functional safety and IEC 61508*

[5] IEC 61508-6:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3*

[6] IEC 61508-7:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures*

[7] IEC 61506:1997, *Industrial-process measurement and control – Documentation of application software*

[8] *Managing Competence for Safety-Related Systems*, IET/BCS/HSE, 2007; (Part 1: Key guidance; Part 2 Supplementary material). HSE. 2007

[9] *Competence criteria for safety-related system practitioners*. IET. 2006

[10] IEC 60300-3-1:2003, *Dependability management – Part 3-1: Application guide – Analysis techniques for dependability – Guide on methodology*

[11] IEC 60300-3-9:1995, *Dependability management – Part 3: Application guide – Section 9: Risk analysis of technological systems*

[12] IEC 61882:2001, *Hazard and operability studies (HAZOP studies) – Application guide*

[13] NUREG/CR-4780, Volume 1, January 1988, *Procedures for treating common cause failures in safety and reliability studies – Procedural framework and examples*

[14] NUREG/CR-4780, Volume 2, January 1989, *Procedures for treating common cause failures in safety and reliability studies – Analytical background and techniques*

[15] IEC 61326-3-1, *Electrical equipment for measurement, control and laboratory use – EMC requirements – Part 3-1: Immunity requirements for safety-related systems and for equipment intended to perform safety-related functions (functional safety) – General industrial applications*

[16] ISO 8613-1:1994, *Information technology – Open Document Architecture (ODA) and Interchange Format: Introduction and general principles*

[17] IEC 61355 (all parts), *Classification and designation of documents for plants, systems and equipment*

[18] IEC 60601 (all parts), *Medical electrical equipment*

[19] IEC/TS 61000-1-2, *Electromagnetic compatibility (EMC) – Part 1-2: General – Methodology for the achievement of functional safety of electrical and electronic systems including equipment with regard to electromagnetic phenomena*

[20] IEC 61508-5:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 5: Examples of methods for the determination of safety integrity levels*

[21] IEC 62443(all parts), *Industrial communication networks – Network and system security*

[22] ISO/IEC/TR 19791, *Information technology – Security techniques – Security assessment of operational systems*

_____

IEC 61508-2

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

BASIC SAFETY PUBLICATION

PUBLICATION FONDAMENTALE DE SÉCURITÉ

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 2: Exigences pour les systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité

IEC 61508-2:2010

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.
- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.
- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.
- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:
Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm
Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,…). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.
- Just Published CEI: www.iec.ch/online_news/justpub
Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.
- Electropedia: www.electropedia.org
Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.
- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:
Email: csc@iec.ch
Tél.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61508-2

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

BASIC SAFETY PUBLICATION

PUBLICATION FONDAMENTALE DE SÉCURITÉ

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 2: Exigences pour les systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE XD
CODE PRIX

ICS 25.040.40

ISBN 978-2-88910-525-0

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

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

Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems

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) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61508-2 has been prepared by subcommittee 65A: System aspects, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 2000. This edition constitutes a technical revision.

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 standard is based on the following documents:

FDIS	Report on voting
65A/549/FDIS	65A/573/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2

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

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

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, though 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.

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

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

Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems

1 Scope

1.1 This part of the IEC 61508 series

a) is intended to be used only after a thorough understanding of IEC 61508-1, which provides the overall framework for the achievement of functional safety;

b) applies to any safety-related system, as defined by IEC 61508-1, that contains at least one electrical, electronic or programmable electronic element;

c) applies to all elements within an E/E/PE safety-related system (including sensors, actuators and the operator interface);

d) specifies how to refine the E/E/PE system safety requirements specification, developed in accordance with IEC 61508-1 (comprising the E/E/PE system safety functions requirements specification and the E/E/PE system safety integrity requirements specification), into the E/E/PE system design requirements specification;

e) specifies the requirements for activities that are to be applied during the design and manufacture of the E/E/PE safety-related systems (i.e. establishes the E/E/PE system safety lifecycle model) except software, which is dealt with in IEC 61508-3 (see Figures 2 to 4). These requirements include the application of techniques and measures that are graded against the safety integrity level, for the avoidance of, and control of, faults and failures;

f) specifies the information necessary for carrying out the installation, commissioning and final safety validation of the E/E/PE safety-related systems;

g) does not apply to the operation and maintenance phase of the E/E/PE safety-related systems – this is dealt with in IEC 61508-1 – however, IEC 61508-2 does provide requirements for the preparation of information and procedures needed by the user for the operation and maintenance of the E/E/PE safety-related systems;

h) specifies requirements to be met by the organisation carrying out any modification of the E/E/PE safety-related systems;

NOTE 1 This part of IEC 61508 is mainly directed at suppliers and/or in-company engineering departments, hence the inclusion of requirements for modification.

NOTE 2 The relationship between IEC 61508-2 and IEC 61508-3 is illustrated in Figure 4.

i) does not apply for medical equipment in compliance with the IEC 60601 series.

1.2 IEC 61508-1, IEC 61508-2, 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 standards. The horizontal safety 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.

NOTE The functional safety of an E/E/PE safety-related system can only be achieved when all related requirements are met. Therefore, it is important that all related requirements are carefully considered and adequately referenced.

1.4 Figure 1 shows the overall framework of the IEC 61508 series and indicates the role that IEC 61508-2 plays in the achievement of functional safety for E/E/PE safety-related systems. Annex A of IEC 61508-6 describes the application of IEC 61508-2 and IEC 61508-3.

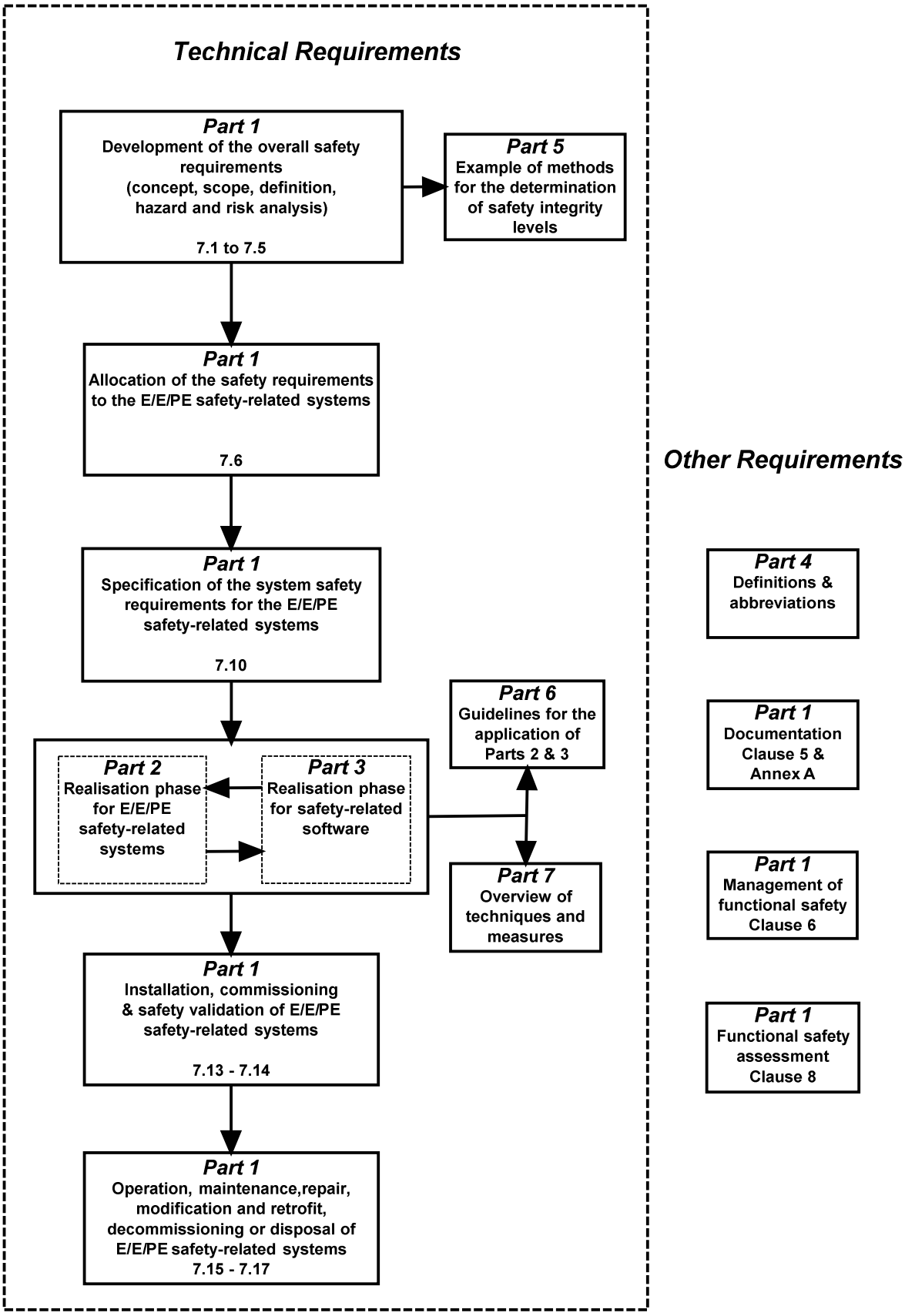


Figure 1 – Overall framework of the IEC 61508 series

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 60947-5-1, *Low-voltage switchgear and controlgear – Part 5-1: Control circuit devices and switching elements – Electromechanical control circuit devices*

IEC/TS 61000-1-2, *Electromagnetic compatibility (EMC) – Part 1-2: General – Methodology for the achievement of functional safety of electrical and electronic systems including equipment with regard to electromagnetic phenomena*

IEC 61326-3-1, *Electrical equipment for measurement, control and laboratory use – EMC requirements – Part 3-1: Immunity requirements for safety-related systems and for equipment intended to perform safety-related functions (functional safety) – General industrial applications*

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

IEC 61508-3: 2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements*

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

IEC 61508-7: 2010, *Functional safety of electrical/electronic/programmable electronic safety related systems – Part 7: Overview of techniques and measures*

IEC 61784-3, *Industrial communication networks – Profiles – Part 3: Functional safety fieldbuses – General rules and profile definitions*

IEC 62280-1, *Railway applications – Communication, signalling and processing systems – Part 1: Safety-related communication in closed transmission systems*

IEC 62280-2, *Railway applications – Communication, signalling and processing systems – Part 2: Safety-related communication in open transmission systems*

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*

EN 50205, *Relays with forcibly guided (mechanically linked) contacts*

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 as detailed in Clause 4 of IEC 61508-1.

5 Documentation

The requirements for documentation are as detailed in Clause 5 of IEC 61508-1.

6 Management of functional safety

The requirements for management of functional safety are as detailed in Clause 6 of IEC 61508-1.

7 E/E/PE system safety lifecycle requirements

7.1 General

7.1.1 Objectives and requirements – general

7.1.1.1 This subclause sets out the objectives and requirements for the E/E/PE system safety lifecycle phases.

NOTE The objectives and requirements for the overall safety lifecycle, together with a general introduction to the structure of the standard, are given in IEC 61508-1.

7.1.1.2 For all phases of the E/E/PE system safety lifecycle, Table 1 indicates

- the objectives to be achieved;
- the scope of the phase;
- a reference to the subclause containing the requirements;
- the required inputs to the phase;
- the outputs required to comply with the subclause.

7.1.2 Objectives

7.1.2.1 The first objective of the requirements of this subclause is to structure, in a systematic manner, the phases in the E/E/PE system safety lifecycle that shall be considered in order to achieve the required functional safety of the E/E/PE safety-related systems.

7.1.2.2 The second objective of the requirements of this subclause is to document all information relevant to the functional safety of the E/E/PE safety-related systems throughout the E/E/PE system safety lifecycle.

7.1.3 Requirements

7.1.3.1 The E/E/PE system safety lifecycle that shall be used in claiming conformance with this standard is that specified in Figure 2. A detailed V-model of the ASIC development lifecycle for the design of ASICs (see IEC 61508-4, 3.2.15) is shown in Figure 3. If another E/E/PE system safety lifecycle or ASIC development lifecycle is used, it shall be specified as part of the management of functional safety activities (see Clause 6 of IEC 61508-1), and all the objectives and requirements of each subclause of IEC 61508-2 shall be met.

NOTE 1 The relationship between and scope for IEC 61508-2 and IEC 61508-3 are shown in Figure 4.

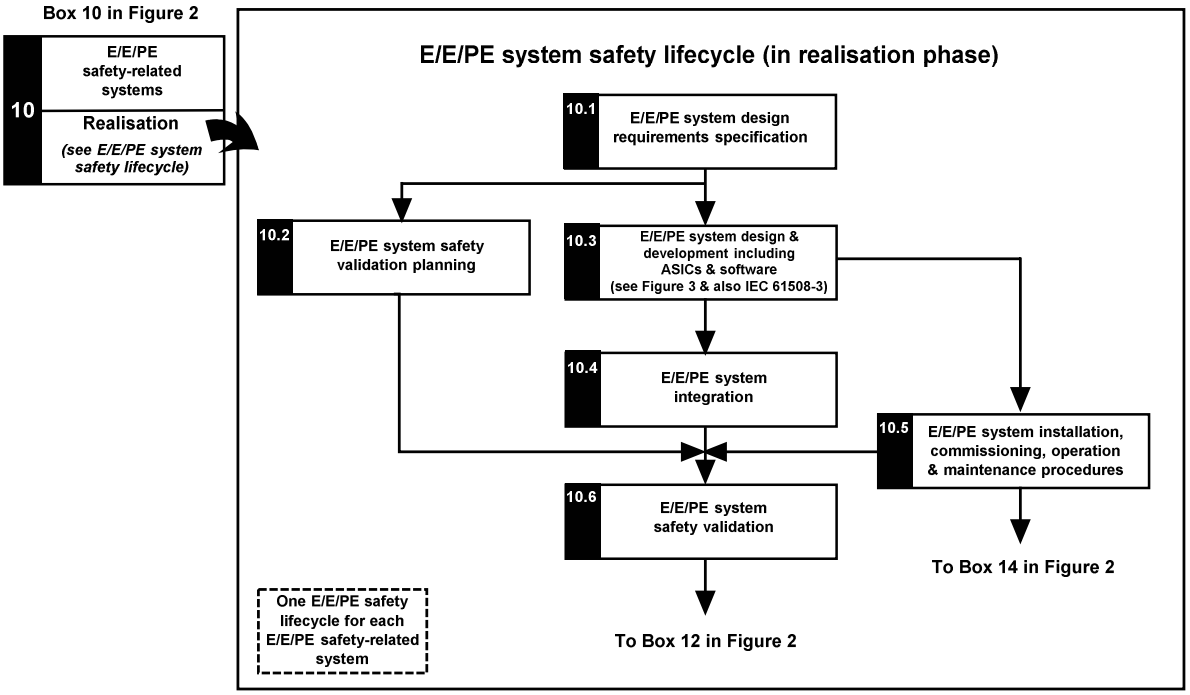
NOTE 2 There are significant similarities between the ASIC and the software design processes. IEC 61508-3 recommends the V-model for designing safety-related software. The V-model requires a clearly structured design process and a modular software structure for avoiding and controlling systematic faults. The ASIC development lifecycle for the design of ASICs in Figure 3 follows this model. At first the requirements for the ASIC specification are derived from the system requirements. ASIC architecture, ASIC design and module design follow. The results of each step on the left-hand side of the V become the input to the next step, and are also fed back to the preceding step for iteration where appropriate, until the final code is created. This code is verified against the corresponding design through post-layout simulation, module testing, module integration testing and verification of the complete ASIC. The results of any step may necessitate a revision to any of the preceding steps. Finally, the ASIC is validated after its integration into the E/E/PE safety-related system.

7.1.3.2 The procedures for management of functional safety (see Clause 6 of IEC 61508-1) shall run in parallel with the E/E/PE system safety lifecycle phases.

7.1.3.3 Each phase of the E/E/PE system safety lifecycle shall be divided into elementary activities, with the scope, inputs and outputs specified for each phase (see Table 1).

7.1.3.4 Unless justified as part of the management of functional safety activities (see Clause 6 of IEC 61508-1), the outputs of each phase of the E/E/PE system safety lifecycle shall be documented (see Clause 5 of IEC 61508-1).

7.1.3.5 The outputs for each E/E/PE system safety lifecycle phase shall meet the objectives and requirements specified for each phase (see 7.2 to 7.9).



NOTE 1 See also IEC 61508-6, A.2 b).

NOTE 2 This figure shows only those phases of the E/E/PE system safety lifecycle that are within the realisation phase of the overall safety lifecycle. The complete E/E/PE system safety lifecycle will also contain instances, specific to the E/E/PE safety-related system, of the subsequent phases of the overall safety lifecycle (Boxes 12 to 16 in Figure 2 of IEC 61508-1).

Figure 2 – E/E/PE system safety lifecycle (in realisation phase)

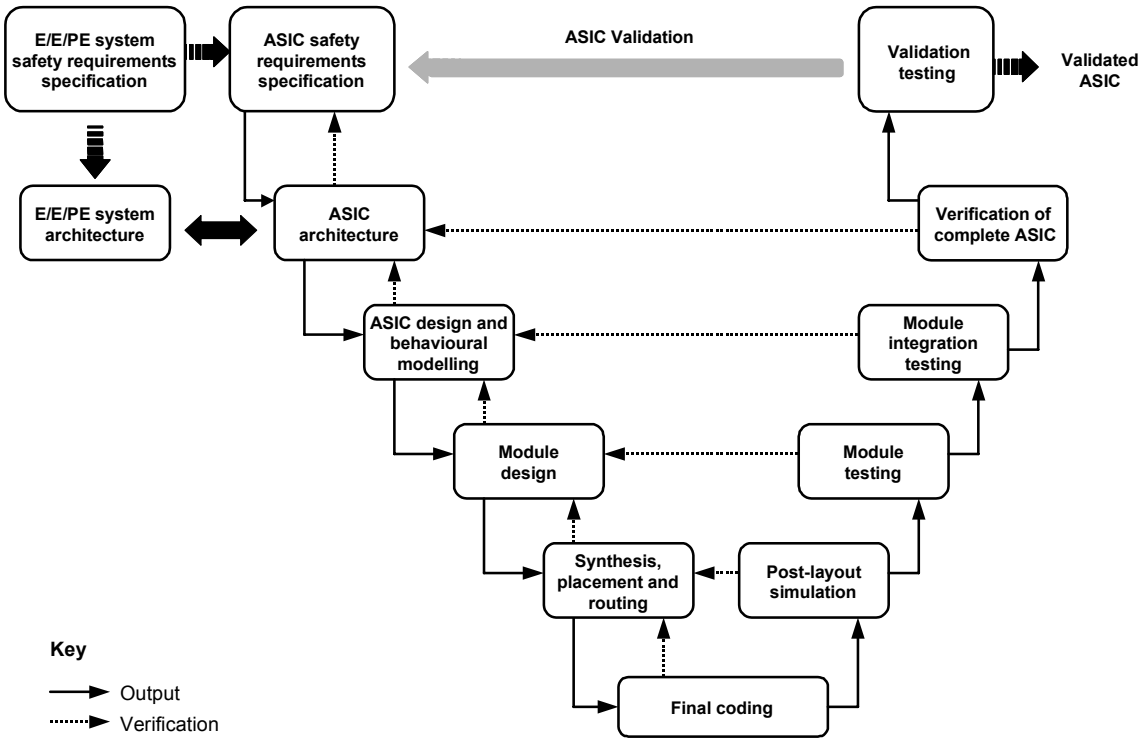


Figure 3 – ASIC development lifecycle (the V-Model)

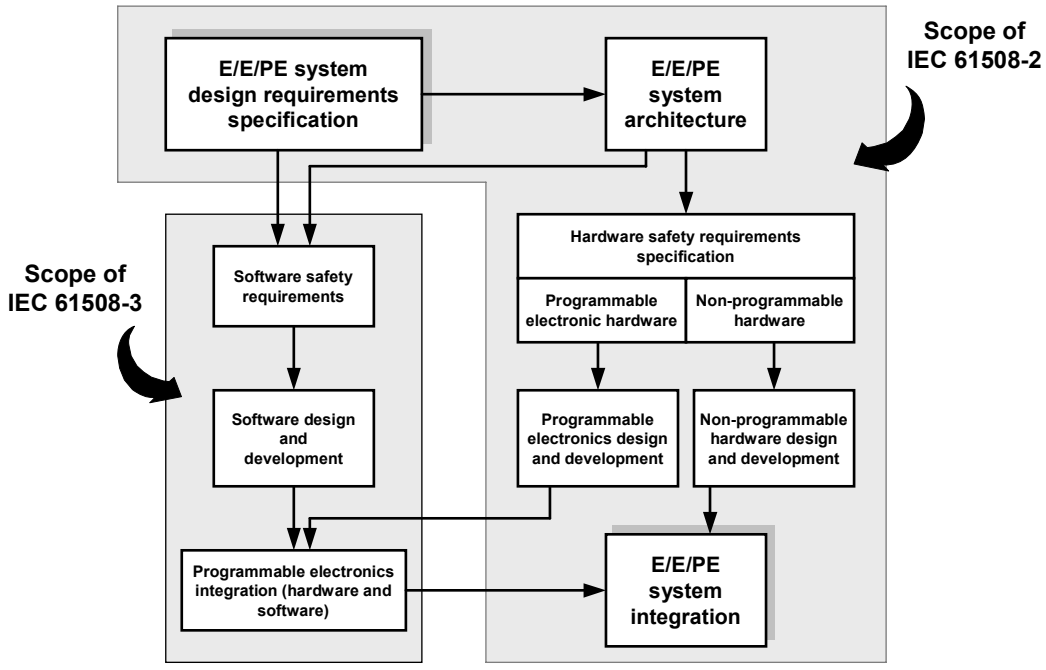


Figure 4 – Relationship between and scope of IEC 61508-2 and IEC 61508-3

Table 1 – Overview – realisation phase of the E/E/PE system safety lifecycle

Safety lifecycle phase or activity		Objectives	Scope	Require-ments sub-clause	Inputs	Outputs
Figure 2 box number	Title					
10.1	E/E/PE system design requirements specification	To specify the design requirements for each E/E/PE safety-related system, in terms of the subsystems and elements (see 7.10.2 of IEC 61508-1)	E/E/PE safety-related system	7.2.2	E/E/PE system safety requirements specification (see IEC 61508-1, 7.10)	E/E/PE system design requirements specification, describing the equipment and architectures for the E/E/PE system
10.2	E/E/PE system safety validation planning	To plan the validation of the safety of the E/E/PE safety-related system	E/E/PE safety-related system	7.3.2	E/E/PE system safety requirements specification and E/E/PE system design requirements specification	Plan for the safety validation of the E/E/PE safety-related systems
10.3	E/E/PE system design & development including ASICs & software (see Figure 3 & also IEC 61508-3)	To design and develop the E/E/PE safety-related system (including ASICs if appropriate) to meet the E/E/PE system design requirements specification (with respect to the safety functions requirements and the safety integrity requirements (see 7.2))	E/E/PE safety-related system	7.4.2 to 7.4.11	E/E/PE system design requirements specification	Design of the E/E/PE safety related systems in conformance with the E/E/PE system design requirements specification Plan for the E/E/PE system integration test PE system architectural information as an input to the software requirements specification
10.4	E/E/PE system integration	To integrate and test the E/E/PE safety-related system	E/E/PE safety-related system	7.5.2	E/E/PE system design E/E/PE system integration test plan Programmable electronics hardware and software	Fully functioning E/E/PE safety-related systems in conformance with the E/E/PE system design Results of E/E/PE system integration tests
10.5	E/E/PE system installation, commissioning, operation and maintenance procedures	To develop procedures to ensure that the required functional safety of the E/E/PE safety-related system is maintained during operation and maintenance	E/E/PE safety-related system EUC	7.6.2	E/E/PE system design requirements specification E/E/PE system design	E/E/PE system installation, commissioning, operation and maintenance procedures for each individual E/E/PE system
10.6	E/E/PE system safety validation	To validate that the E/E/PE safety-related system meets, in all respects, the requirements for safety in terms of the required safety functions and safety integrity	E/E/PE safety-related system	7.7.2	E/E/PE system safety requirements specification and E/E/PE system design requirements specification Plan for the safety validation of the E/E/PE safety-related systems	Fully safety validated E/E/PE safety-related systems Results of E/E/PE system safety validation

Table 1 *(continued)*

Safety lifecycle phase or activity		Objectives	Scope	Require-ments sub-clause	Inputs	Outputs
Figure 2 box number	Title					
–	E/E/PE system modification	To make corrections, enhancements or adaptations to the E/E/PE safety-related system, ensuring that the required safety integrity is achieved and maintained	E/E/PE safety-related system	7.8.2	E/E/PE system design requirements specification	Results of E/E/PE system modification
–	E/E/PE system verification	To test and evaluate the outputs of a given phase to ensure correctness and consistency with respect to the products and standards provided as input to that phase	E/E/PE safety-related system	7.9.2	As above – depends on the phase Plan for the verification of the E/E/PE safety-related systems for each phase	As above – depends on the phase Results of the verification of the E/E/PE safety-related systems for each phase
–	E/E/PE system functional safety assessment	To investigate and arrive at a judgement on the functional safety achieved by the E/E/PE safety-related system	E/E/PE safety-related system	8	Plan for E/E/PE system functional safety assessment	Results of E/E/PE system functional safety assessment

7.2 E/E/PE system design requirements specification

NOTE This phase is Box 10.1 of Figure 2.

7.2.1 Objective

The objective of the requirements of this subclause is to specify the design requirements for each E/E/PE safety-related system, in terms of the subsystems and elements.

NOTE The E/E/PE system design requirements specification is normally derived from the E/E/PE system safety requirements specification by decomposing the safety functions and allocating parts of the safety function to subsystems (for example groups of sensors, logic solvers or actuators). The requirements for the subsystems may be included in the E/E/PE system design requirements specification or may be separate and referenced from the E/E/PE system design requirements specification. Subsystems may be further decomposed into elements and architectures to satisfy the design and development requirements of 7.4. The requirements for these elements may be included in the requirements for the subsystems or may be separate and referenced from the subsystem requirements.

7.2.2 General

7.2.2.1 The specification of the E/E/PE system design requirements shall be derived from the E/E/PE system safety requirements, specified in 7.10 of IEC 61508-1.

NOTE Caution should be exercised if non-safety functions and safety functions are implemented in the same E/E/PE safety-related system. While this is allowed in the standard, it may lead to greater complexity and increase the difficulty in carrying out E/E/PE safety lifecycle activities (for example design, validation, functional safety assessment and maintenance). See also 7.4.2.3.

7.2.2.2 The specification of the E/E/PE system design requirements shall be expressed and structured in such a way that they are:

a) clear, precise, unambiguous, verifiable, testable, maintainable and feasible;

b) written to aid comprehension by those who are likely to utilise the information at any phase of the E/E/PE safety lifecycle; and

c) traceable to the E/E/PE system safety requirements specification.

7.2.3 E/E/PE system design requirements specification

7.2.3.1 The specification of the E/E/PE system design requirements shall contain design requirements relating to safety functions (see 7.2.3.2) and design requirements relating to safety integrity (see 7.2.3.3).

7.2.3.2 The specification of the E/E/PE system design requirements shall contain details of all the hardware and software necessary to implement the required safety functions, as specified by the E/E/PE system safety functions requirements specification (see 7.10.2.6 of IEC 61508-1). The specification shall include, for each safety function:

a) requirements for the subsystems and requirements for their hardware and software elements as appropriate;

b) requirements for the integration of the subsystems and their hardware and software elements to meet the E/E/PE system safety functions requirements specification;

c) throughput performance that enables response time requirements to be met;

d) accuracy and stability requirements for measurements and controls;

e) E/E/PE safety-related system and operator interfaces;

f) interfaces between the E/E/PE safety-related systems and any other systems (either within, or outside, the EUC);

g) all modes of behaviour of the E/E/PE safety-related systems, in particular, failure behaviour and the required response (for example alarms, automatic shut-down) of the E/E/PE safety-related systems;

h) the significance of all hardware/software interactions and, where relevant, any required constraints between the hardware and the software;

NOTE Where these interactions are not known before finishing the design, only general constraints can be stated.

i) any limiting and constraint conditions for the E/E/PE safety-related systems and their associated elements, for example timing constraints or constraints due to the possibility of common cause failures;

j) any specific requirements related to the procedures for starting-up and restarting the E/E/PE safety-related systems.

7.2.3.3 The specification of the E/E/PE system design requirements shall contain details, relevant to the design, to achieve the safety integrity level and the required target failure measure for the safety function, as specified by the E/E/PE system safety integrity requirements specification (see 7.10.2.7 of IEC 61508-1), including:

a) the architecture of each subsystem required to meet the architectural constraints on the hardware safety integrity (see 7.4.4);

b) all relevant reliability modelling parameters such as the required proof testing frequency of all hardware elements necessary to achieve the target failure measure;

NOTE 1 Information on the specific application cannot be understated (see 7.10.2.1 of IEC 61508-1). This is particularly important for maintenance, where the specified proof test interval should not be less than can be reasonably expected for the particular application. For example, the time between services that can be realistically attained for mass-produced items used by the public is likely to be greater than in a more controlled application.

c) the actions taken in the event of a dangerous failure being detected by diagnostics;

d) the requirements, constraints, functions and facilities to enable the proof testing of the E/E/PE hardware to be undertaken;

e) the capabilities of equipment used to meet the extremes of all environmental conditions (e.g. temperature, humidity, mechanical, electrical) that are specified as required during the E/E/PE system safety lifecycle including manufacture, storage, transport, testing, installation, commissioning, operation and maintenance;

f) the electromagnetic immunity levels that are required (see IEC/TS 61000-1-2: 2008);

NOTE 2 The required immunity levels may vary for different elements of the safety-related system, depending on physical location and power supply arrangements.

NOTE 3 Guidance may be found in EMC product standards, but it is important to recognise that higher immunity levels, or additional immunity requirements, than those specified in such standards may be necessary for particular locations or when the equipment is intended for use in harsher, or different, electromagnetic environments.

g) the quality assurance/quality control measures necessary to safety management (see 6.2.5 of IEC 61508-1);

7.2.3.4 The E/E/PE system design requirements specification shall be completed in detail as the design progresses and updated as necessary after modification.

7.2.3.5 For the avoidance of mistakes during the development of the specification for the E/E/PE system design requirements, an appropriate group of techniques and measures according to Table B.1 shall be used.

7.2.3.6 The implications imposed on the architecture by the E/E/PE system design requirements shall be considered.

NOTE This should include the consideration of the simplicity of the implementation to achieve the required safety integrity level (including architectural considerations and apportionment of functionality to configuration data or to the embedded system).

7.3 E/E/PE system safety validation planning

NOTE This phase is Box 10.2 of Figure 2. It will normally be carried out in parallel with E/E/PE system design and development (see 7.4).

7.3.1 Objective

The objective of the requirements of this subclause is to plan the validation of the safety of the E/E/PE safety-related system.

7.3.2 Requirements

7.3.2.1 Planning shall be carried out to specify the steps (both procedural and technical) that are to be used to demonstrate that the E/E/PE safety-related system satisfies the E/E/PE system safety requirements specification (see 7.10 of IEC 61508-1) and the E/E/PE system design requirements specification (see 7.2).

7.3.2.2 Planning for the validation of the E/E/PE safety-related system shall consider the following:

a) all of the requirements defined in the E/E/PE system safety requirements specification and the E/E/PE system design requirements specification;

b) the procedures to be applied to validate that each safety function is correctly implemented, and the pass/fail criteria for accomplishing the tests;

c) the procedures to be applied to validate that each safety function is of the required safety integrity, and the pass/fail criteria for accomplishing the tests;

d) the required environment in which the testing is to take place including all necessary tools and equipment (also plan which tools and equipment should be calibrated);

e) test evaluation procedures (with justifications);

f) the test procedures and performance criteria to be applied to validate the specified electromagnetic immunity limits;

NOTE Guidance on the specification of electromagnetic immunity tests for elements of safety-related systems is given in IEC/TS 61000-1-2.

g) policies for resolving validation failure.

7.4 E/E/PE system design and development

NOTE This phase is Box 10.3 of Figure 2. It will normally be carried out in parallel with E/E/PE system safety validation planning (see 7.3).

7.4.1 Objective

The objective of the requirements of this subclause is to design and develop the E/E/PE safety-related system (including ASICs if appropriate, see IEC 61508-4, 3.2.15) to meet the E/E/PE system design requirements specification (with respect to the safety functions requirements and the safety integrity requirements (see 7.2).

7.4.2 General requirements

7.4.2.1 The design of the E/E/PE safety-related system shall be created in accordance with the E/E/PE system design requirements specification (see 7.2.3), taking into account all the requirements of 7.2.3.

7.4.2.2 The design of the E/E/PE safety-related system (including the overall hardware and software architecture, sensors, actuators, programmable electronics, ASICs, embedded software, application software, data etc.), shall meet all of the requirements a) to e) as follows:

a) the requirements for hardware safety integrity comprising;
   - the architectural constraints on hardware safety integrity (see 7.4.4), and
   - the requirements for quantifying the effect of random failures (see 7.4.5);

b) the special architecture requirements for ICs with on-chip redundancy (see Annex E), where relevant, unless justification can be given that the same level of independence between different channels is achieved by applying a different set of measures;

c) the requirements for systematic safety integrity (systematic capability), which can be met by achieving one of the following compliance routes:
   - Route 1_S: compliance with the requirements for the avoidance of systematic faults (see 7.4.6 and IEC 61508-3) and the requirements for the control of systematic faults (see 7.4.7 and IEC 61508-3), or
   - Route 2_S: compliance with the requirements for evidence that the equipment is proven in use (see 7.4.10), or
   - Route 3_S (pre-existing software elements only): compliance with the requirements of IEC 61508-3, 7.4.2.12;

     NOTE The “S” subscript in the above routes designates systematic safety integrity to distinguish it from Route 1_H, and Route 2_H for hardware safety integrity.

d) the requirements for system behaviour on detection of a fault (see 7.4.8);

e) the requirements for data communication processes (see 7.4.11).

7.4.2.3 Where an E/E/PE safety-related system is to implement both safety and non-safety functions, then all the hardware and software shall be treated as safety-related unless it can be shown that the implementation of the safety and non-safety functions is sufficiently independent (i.e. that the failure of any non-safety-related functions does not cause a dangerous failure of the safety-related functions).

NOTE 1 Sufficient independence of implementation is established by showing that the probability of a dependent failure between the non-safety and safety-related parts is sufficiently low in comparison with the highest safety integrity level associated with the safety functions involved.

NOTE 2 Caution should be exercised if non-safety functions and safety functions are implemented in the same E/E/PE safety-related system. While this is allowed in the standard, it may lead to greater complexity and increase the difficulty in carrying out E/E/PE system safety lifecycle activities (for example design, validation, functional safety assessment and maintenance).

7.4.2.4 The requirements for hardware and software shall be determined by the safety integrity level of the safety function having the highest safety integrity level unless it can be shown that the implementation of the safety functions of the different safety integrity levels is sufficiently independent.

NOTE 1 Sufficient independence of implementation is established by showing that the probability of a dependent failure between the parts implementing safety functions of different integrity levels is sufficiently low in comparison with the highest safety integrity level associated with the safety functions involved.

NOTE 2 Where several safety functions are implemented in an E/E/PE safety-related system then it will be necessary to consider the possibility that a single fault could cause a failure of several safety functions. In such a situation, it may be appropriate to determine the requirements for hardware and software on the basis of a higher safety integrity level than is associated with any one of the safety functions, depending on the risk associated with such a failure.

7.4.2.5 When independence between safety functions is required (see 7.4.2.3 and 7.4.2.4) then the following shall be documented during the design:

a) the method of achieving independence;

b) the justification of the method.

EXAMPLE Addressing foreseeable failure modes, that may undermine independence, and their failure rates, use of FMECA or dependant failure analysis.

7.4.2.6 The requirements for safety-related software (see IEC 61508-3) shall be made available to the developer of the E/E/PE safety-related system.

7.4.2.7 The developer of the E/E/PE safety-related system shall review the requirements for safety-related software and hardware to ensure that they are adequately specified. In particular, the E/E/PE system developer shall consider the following:

a) safety functions;

b) E/E/PE safety-related system safety integrity requirements;

c) equipment and operator interfaces.

7.4.2.8 The E/E/PE safety-related system design documentation shall specify those techniques and measures necessary during the E/E/PE system safety lifecycle phases to achieve the safety integrity level.

7.4.2.9 The E/E/PE safety-related system design documentation shall justify the techniques and measures chosen to form an integrated set that satisfies the required safety integrity level.

NOTE The adoption of an overall approach employing independent type approval of the E/E/PE safety-related systems (including sensors, actuators, etc) for hardware and software, diagnostic tests and programming tools, and using appropriate languages for software wherever possible, has the potential to reduce the complexity of E/E/PE system application engineering.

7.4.2.10 During the design and development activities, the significance (where relevant) of all hardware and software interactions shall be identified, evaluated and documented.

7.4.2.11 The design shall be based on a decomposition into subsystems with each subsystem having a specified design and set of integration tests (see 7.5.2).

NOTE 1 A subsystem may be considered to comprise a single element or any group of elements. See IEC 61508-4 for definitions. A complete E/E/PE safety-related system is made up from a number of identifiable and separate subsystems, which when put together implement the safety function under consideration. A subsystem can have more than one channel (see 7.4.9.3 and 7.4.9.4).

NOTE 2 Wherever practicable, existing verified subsystems should be used in the implementation. This statement is generally valid only if there is almost 100 % mapping of the existing subsystem or element functionality, capacity and performance on to the new requirement or the verified subsystem or element is structured in such a way that the user is able to select only the functions, capacity or performance required for the specific application. Excessive functionality, capacity or performance can be detrimental to system safety if the existing subsystem or element is overly complicated or has unused features and if protection against unintended functions cannot be obtained.

7.4.2.12 When the initial design of the E/E/PE safety-related system has been completed, an analysis shall be undertaken to determine whether any reasonably foreseeable failure of the E/E/PE safety-related system could cause a hazardous situation or place a demand on any

other risk control measure. If any reasonably foreseeable failure could have either of these effects, then the first priority shall be to change the design of the E/E/PE safety-related system to avoid such failure modes. If this cannot be done, then measures shall be taken to reduce the likelihood of such failure modes to a level commensurate with the target failure measure. These measures shall be subject to the requirements of this standard.

NOTE The intention of this clause is to identify failure modes of the E/E/PE safety-related system that place a demand on other risk control measures. There may be cases where the failure rate of the specified failure modes cannot be reduced and either a new safety function will be required or the SIL of the other safety functions reconsidered taking into account the failure rate.

7.4.2.13 De-rating (see IEC 61508-7) should be considered for all hardware components. Justification for operating any hardware elements at their limits shall be documented (see IEC 61508-1, Clause 5).

NOTE Where de-rating is appropriate, a de-rating factor of approximately two-thirds is typical.

7.4.2.14 Where the design of an E/E/PE safety-related system includes one or more ASICs to implement a safety function, an ASIC development lifecycle (see 7.1.3.1) shall be used.

7.4.3 Synthesis of elements to achieve the required systematic capability

7.4.3.1 To meet the requirements for systematic safety integrity, the designated safety-related E/E/PE system may, in the circumstances described in this section, be partitioned into elements of different systematic capability.

NOTE 1 The systematic capability of an element determines the potential for systematic faults of that element to lead to a failure of the safety function. The concept of systematic capability of an element is applicable to both hardware and software elements.

NOTE 2 Subclause 7.6.2.7 of IEC 61508-1 recognises the value of independence and diversity at the level of a safety function and the E/E/PE safety related systems to which it could be allocated. These concepts can also be applied at the detailed design level where an assembly of elements implementing a safety function can potentially achieve a better systematic performance than the individual elements.

7.4.3.2 For an element of systematic capability SC N (N=1, 2, 3), where a systematic fault of that element does not cause a failure of the specified safety function but does so only in combination with a second systematic fault of another element of systematic capability SC N, the systematic capability of the combination of the two elements can be treated as having a systematic capability of SC (N + 1) providing that sufficient independence exists between the two elements (see 7.4.3.4).

NOTE The independence of elements can be assessed only when the specific application of the elements is known in relation to the defined safety functions.

7.4.3.3 The systematic capability that can be claimed for a combination of elements each of systematic capability SC N can at most be SC (N+1). A SC N element may be used in this way only once. It is not permitted to achieve SC (N+2) and higher by successively building assemblies of SC N elements.

7.4.3.4 Sufficient independence, in the design between elements and in the application of elements, shall be justified by common cause failure analysis to show that the likelihood of interference between elements and between the elements and the environment is sufficiently low in comparison with the safety integrity level of the safety function under consideration.

NOTE 1 For systematic capability, with respect to hardware design, realisation, operation and maintenance, possible approaches to the achievement of sufficient independence include:

– functional diversity: use of different approaches to achieve the same results;

– diverse technologies: use of different types of equipment to achieve the same results);

– common parts/services: ensuring that there are no common parts or services or support systems (for example power supplies) whose failure could result in a dangerous mode of failure of all systems;

– common procedures: ensuring that there are no common operational, maintenance or test procedures.

NOTE 2 Independence of application means that elements will not adversely interfere with each other's execution behaviour such that a dangerous failure would occur.

NOTE 3 For independence of software elements see 7.4.2.8 and 7.4.2.9 of IEC 61508-3.

7.4.4 Hardware safety integrity architectural constraints

NOTE 1 The equation, relating to the hardware safety integrity constraints, are specified in Annex C and the safety integrity constraints are summarized in Table 2 and Table 3

NOTE 2 Clause A.2 of IEC 61508-6 gives an overview of the necessary steps in achieving required hardware safety integrity, and shows how this subclause relates to other requirements of this standard.

In the context of hardware safety integrity, the highest safety integrity level that can be claimed for a safety function is limited by the hardware safety integrity constraints which shall be achieved by implementing one of two possible routes (to be implemented at system or subsystem level):

– Route 1_H based on hardware fault tolerance and safe failure fraction concepts; or,
– Route 2_H based on component reliability data from feedback from end users, increased confidence levels and hardware fault tolerance for specified safety integrity levels.

Application standards based on the IEC 61508 series may indicate the preferred Route (i.e. Route 1_H or Route 2_H).

NOTE 3 The "H" subscript in the above routes designates hardware safety integrity to distinguish it from Route 1_S , Route 2_S and Route 3_S for systematic safety integrity.

7.4.4.1 General requirements

7.4.4.1.1 With respect to the hardware fault tolerance requirements

a) a hardware fault tolerance of N means that N+1 is the minimum number of faults that could cause a loss of the safety function (for further clarification see Note 1 and Table 2 and Table 3). In determining the hardware fault tolerance no account shall be taken of other measures that may control the effects of faults such as diagnostics; and
b) where one fault directly leads to the occurrence of one or more subsequent faults, these are considered as a single fault;
c) when determining the hardware fault tolerance achieved, certain faults may be excluded, provided that the likelihood of them occurring is very low in relation to the safety integrity requirements of the subsystem. Any such fault exclusions shall be justified and documented (see Note 2).

NOTE 1 The constraints on hardware safety integrity have been included in order to achieve a sufficiently robust architecture, taking into account the level of element and subsystem complexity (see 7.4.4.1.1 and 7.4.4.1.2). The highest allowable safety integrity level for the safety function implemented by the E/E/PE safety-related system, derived through applying these requirements, is the maximum that is permitted to be claimed for the safety function even though, in some cases reliability calculations show that a higher safety integrity level could be achieved. It should also be noted that even if the hardware fault tolerance is achieved for all subsystems, a reliability calculation will still be necessary to demonstrate that the specified target failure measure has been achieved and this may require that the hardware fault tolerance be increased to meet design requirements.

NOTE 2 The hardware fault tolerance requirements apply to the subsystem architecture that is used under normal operating conditions. The hardware fault tolerance requirements may be relaxed while the E/E/PE safety-related system is being repaired on-line. However, the key parameters relating to any relaxation should have been previously evaluated (for example MTTR compared to the probability of a demand).

NOTE 3 Certain faults may be excluded because if an element clearly has a very low probability of failure by virtue of properties inherent to its design and construction (for example, a mechanical actuator linkage), then it would not normally be considered necessary to constrain (on the basis of hardware fault tolerance) the safety integrity of any safety function that uses the element.

NOTE 4 The choice of the route is application and sector dependent and the following should be considered when selecting the Route:

– a safe failure of one function may create a new hazard or be an additional cause for an existing hazard;
– redundancy may not be practicable for all functions;

– repair is not always possible or rapid (e.g. not feasible within a time that is negligible compared to the proof test interval).

NOTE 5 Special architecture requirements for ICs with on-chip redundancy are given in Annex E.

7.4.4.1.2 An element can be regarded as type A if, for the components required to achieve the safety function

a) the failure modes of all constituent components are well defined; and

b) the behaviour of the element under fault conditions can be completely determined; and

c) there is sufficient dependable failure data to show that the claimed rates of failure for detected and undetected dangerous failures are met (see 7.4.9.3 to 7.4.9.5).

7.4.4.1.3 An element shall be regarded as type B if, for the components required to achieve the safety function,

a) the failure mode of at least one constituent component is not well defined; or

b) the behaviour of the element under fault conditions cannot be completely determined; or

c) there is insufficient dependable failure data to support claims for rates of failure for detected and undetected dangerous failures (see 7.4.9.3 to 7.4.9.5).

NOTE This means that if at least one of the components of an element itself satisfies the conditions for a type B element then that element will be regarded as type B rather than type A.

7.4.4.1.4 When estimating the safe failure fraction of an element, intended to be used in a subsystem having a hardware fault tolerance of 0, and which is implementing a safety function, or part of a safety function, operating in high demand mode or continuous mode of operation, credit shall only be taken for the diagnostics if:

– the sum of the diagnostic test interval and the time to perform the specified action to achieve or maintain a safe state is less than the process safety time; or,

– when operating in high demand mode of operation, the ratio of the diagnostic test rate to the demand rate equals or exceeds 100.

7.4.4.1.5 When estimating the safe failure fraction of an element which,

– has a hardware fault tolerance greater than 0, and which is implementing a safety function, or part of a safety function, operating in high demand mode or continuous mode of operation; or,

– is implementing a safety function, or part of a safety function, operating in low demand mode of operation,

credit shall only be taken for the diagnostics if the sum of the diagnostic test interval and the time to perform the repair of a detected failure is less than the MTTR used in the calculation to determine the achieved safety integrity for that safety function.

7.4.4.2 Route 1_H

7.4.4.2.1 To determine the maximum safety integrity level that can be claimed, with respect to a specified safety function, the following procedure shall be followed:

1) Define the subsystems making up the E/E/PE safety-related system.

2) For each subsystem determine the safe failure fraction for all elements in the subsystem separately (i.e. on an individual element basis with each element having a hardware fault tolerance of 0). In the case of redundant element configurations, the SFF may be calculated by taking into consideration the additional diagnostics that may be available (e.g. by comparison of redundant elements).

3) For each element, use the achieved safe failure fraction and hardware fault tolerance of 0 to determine the maximum safety integrity level that can be claimed from column 2 of Table 2 (for Type A elements) and column 2 of Table 3 (for Type B elements).

4) Use the method in 7.4.4.2.3 and 7.4.4.2.4 for determining the maximum safety integrity level that can be claimed for the subsystem.

5) The maximum safety integrity level that can be claimed for an E/E/PE safety-related system shall be determined by the subsystem that has achieved the lowest safety integrity level.

7.4.4.2.2 For application to subsystems comprising elements that meet the specific requirements detailed below, as an alternative to applying the requirements of 7.4.4.2.1 2) to 7.4.4.2.1 4), the following is applicable:

1) the subsystem is comprised of more than one element; and

2) the elements are of the same type; and

3) all the elements have achieved safe failure fractions that are in the same range (see Note 1 below) specified in Tables 2 or 3;then the following procedure may be followed,

   a) determine the safe failure fraction of all individual elements. In the case of redundant element configurations, the SFF may be calculated by taking into consideration the additional diagnostics that may be available (e.g. by comparison of redundant elements);

   b) determine the hardware fault tolerance of the subsystem;

   c) determine the maximum safety integrity level that can be claimed for the subsystem if the elements are type A from Table 2;

   d) determine the maximum safety integrity level that can be claimed for the subsystem if the elements are type B from Table 3.

NOTE 1 The range indicated in 3) above refers to Tables 2 and 3 where the safe failure fraction is classified into one of four ranges (i.e. (<60 %); (60 % to <90 %); (90% to <99 %) and (≥99 %)). All SFFs would need to be in the same range (e.g. all in the range (90 % to <99 %)).

EXAMPLE 1 To determine the maximum allowable safety integrity level that has been achieved, for the specified safety function, by a subsystem having a hardware fault tolerance of 1, where an element safety function is implemented through parallel elements, the following approach may be adopted providing the subsystem meets the requirements of 7.4.4.2.2. In this example, all the elements are type B and the safe failure fractions of the elements are in the (90 % to < 99 %) range.

From Table 3, it can be seen by inspection, that for a hardware fault tolerance equal to 1, with safe failure fractions of both elements in the (90 % to <99 %) range, the maximum allowable safety integrity level for the specified safety function is SIL 3.

EXAMPLE 2 To determine the required hardware fault tolerance for a subsystem, for the specified safety function, where an element safety function is implemented through parallel elements, the following approach may be adopted providing the subsystem meets the requirements of 7.4.4.2.2. In this example, all the elements are type A and the safe failure fractions of the elements are in the (60 % to <90 % range). The safety integrity level of the safety function is SIL 3.

From Table 2, it can be seen by inspection, that to meet the requirement of SIL 3, the required hardware fault tolerance needs to equal 1. This means that two elements in parallel are necessary.

Table 2 – Maximum allowable safety integrity level for a safety function carried out by a type A safety-related element or subsystem

Safe failure fraction of an element	Hardware fault tolerance		
	0	1	2
< 60 %	SIL 1	SIL 2	SIL 3
60 % – < 90 %	SIL 2	SIL 3	SIL 4
90 % – < 99 %	SIL 3	SIL 4	SIL 4
≥ 99 %	SIL 3	SIL 4	SIL 4

NOTE 1 This table, in association with 7.4.4.2.1 and 7.4.4.2.2, is used for the determination of the maximum SIL that can be claimed for a subsystem: given the fault tolerance of the subsystem and the SFF to the elements used.

i. For general application to any subsystem see 7.4.4.2.1.

ii. For application to subsystems comprising elements that meet the specific requirements of 7.4.4.2.2. To claim that a subsystem meets a specified SIL directly from this table it will be necessary to meet all the requirements in 7.4.4.2.2.

NOTE 2 The table, in association with 7.4.4.2.1 and 7.4.4.2.2,can also be used:

i. For the determination of the hardware fault tolerance requirements for a subsystem given the required SIL of the safety function and the SFFs of the elements to be used.

ii. For the determination of the SFF requirements for elements given the required SIL of the safety function and the hardware fault tolerance of the subsystem.

NOTE 3 The requirements in 7.4.4.2.3 and 7.4.4.2.4 are based on the data specified in this table and Table 3.

NOTE 4 See Annex C for details of how to calculate safe failure fraction.

Table 3 – Maximum allowable safety integrity level for a safety function carried out by a type B safety-related element or subsystem

Safe failure fraction of an element	Hardware fault tolerance		
	0	1	2
<60 %	Not Allowed	SIL 1	SIL 2
60 % – <90 %	SIL 1	SIL 2	SIL 3
90 % – <99 %	SIL 2	SIL 3	SIL 4
≥ 99 %	SIL 3	SIL 4	SIL 4

NOTE 1 This table, in association with 7.4.4.2.1 and 7.4.4.2.2, is used for the determination of the maximum SIL that can be claimed for a subsystem given the fault tolerance of the subsystem and the SFF to the elements used.

i. For general application to any subsystem see 7.4.4.2.1.

ii. For application to subsystems comprising elements that meet the specific requirements of 7.4.4.2.2. To claim that a subsystem meets a specified SIL directly from this table it will be necessary to meet all the requirements in 7.4.4.2.2.

NOTE 2 The table, in association with 7.4.4.2.1 and 7.4.4.2.2,can also be used:

i. For the determination of the hardware fault tolerance requirements for a subsystem given the required SIL of the safety function and the SFFs of the elements to be used.

ii. For the determination of the SFF requirements for elements given the required SIL of the safety function and the hardware fault tolerance of the subsystem.

NOTE 3 The requirements in 7.4.4.2.3 and 7.4.4.2.4 are based on the data specified in this table and Table 2.

NOTE 4 See Annex C for details of how to calculate safe failure fraction.

NOTE 5 When using 7.4.4.2.1 for the combination of type B elements, with a hardware fault tolerance of 1, in which both elements have a safe failure fraction of less than 60 %, the maximum allowable safety integrity level for a safety function carried out by the combination is SIL 1.

7.4.4.2.3 In an E/E/PE safety-related subsystem where a number of element safety functions are implemented through a serial combination of elements (such as in Figure 5), the maximum safety integrity level that can be claimed for the safety function under consideration shall be determined by the element that has achieved the lowest safety integrity level for the achieved safe failure fraction for a hardware fault tolerance of 0. To illustrate the method, assume an architecture as indicated in Figure 5 and see example below.

EXAMPLE (see Figure 5): Assume an architecture where a number of element safety functions are performed by a subsystem comprising a single channel of elements 1, 2 and 3 and the elements meet the requirements of Tables 2 and 3 as follows:

– Element 1 achieves the requirements, for a hardware fault tolerance of 0 and, for a specific safe failure fraction, for SIL 1;

– Element 2 achieves the requirements, for a hardware fault tolerance of 0 and, for a specific safe failure fraction, for SIL 2;

– Element 3 achieves the requirements, for a hardware fault tolerance of 0 and, for a specific safe failure fraction, for SIL 1;

– Both element 1 and element 3 restrict the maximum SIL that can be claimed, for the achieved hardware fault tolerance and safe failure fraction to just SIL 1.

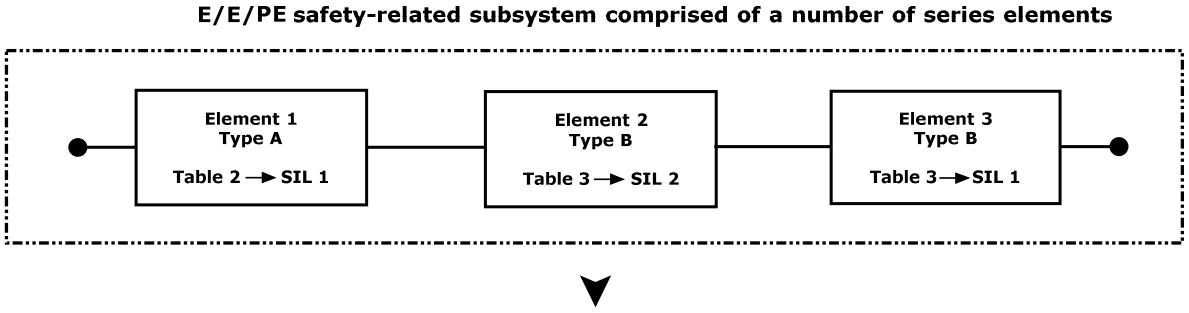


Figure 5 – Determination of the maximum SIL for specified architecture (E/E/PE safety-related subsystem comprising a number of series elements, see 7.4.4.2.3)

7.4.4.2.4 In an E/E/PE safety-related subsystem where an element safety function is implemented through a number of channels (combination of parallel elements) having a hardware fault tolerance of N, the maximum safety integrity level that can be claimed for the safety function under consideration shall be determined by:

a) grouping the serial combination of elements for each channel and then determining the maximum safety integrity level that can be claimed for the safety function under consideration for each channel (see 7.4.4.2.3); and

b) selecting the channel with the highest safety integrity level that has been achieved for the safety function under consideration and then adding N safety integrity levels to determine the maximum safety integrity level for the overall combination of the subsystem.

To illustrate the method, assume architecture as indicated in Figure 6 and see example below.

NOTE 1 N is the hardware fault tolerance of the combination of parallel elements (see 7.4.4.1.1).

NOTE 2 See example below regarding the application of this subclause.

EXAMPLE The grouping and analysis of these combinations may be carried out in various ways. To illustrate one possible method, assume an architecture in which a particular safety function is performed by two subsystems, X and Y, where subsystem X consists of elements 1, 2, 3 and 4, and subsystem Y consists of a single element 5, as shown in Figure 6. The use of parallel channels in subsystem X ensures that elements 1 and 2 implement the part of the safety function required of subsystem X independently from elements 3 and 4, and vice-versa. The safety function will be performed:

– in the event of a fault in either element 1 or element 2 (because the combination of elements 3 and 4 is able to perform the required part of the safety function); or

– in the event of a fault in either element 3 or element 4 (because the combination of elements 1 and 2 is able to perform the required part of the safety function).

The determination of the maximum safety integrity level that can be claimed, for the safety function under consideration, is detailed in the following steps.

For subsystem X, in respect of the specified safety function under consideration, each element meets the requirements of Tables 2 and 3 as follows:

– Element 1 achieves the requirements, for a hardware fault tolerance of 0 and, for a specific safe failure fraction, for SIL 3;

– Element 2 achieves the requirements, for a hardware fault tolerance of 0 and, for a specific safe failure fraction, for SIL 2;

– Element 3 achieves the requirements, for a hardware fault tolerance of 0 and, for a specific safe failure fraction, for SIL 2;

– Element 4 achieves the requirements, for a hardware fault tolerance of 0 and, for a specific safe failure fraction, for SIL 1.

Elements are combined to give a maximum hardware safety integrity level for the safety function under consideration, for subsystem X as follows:

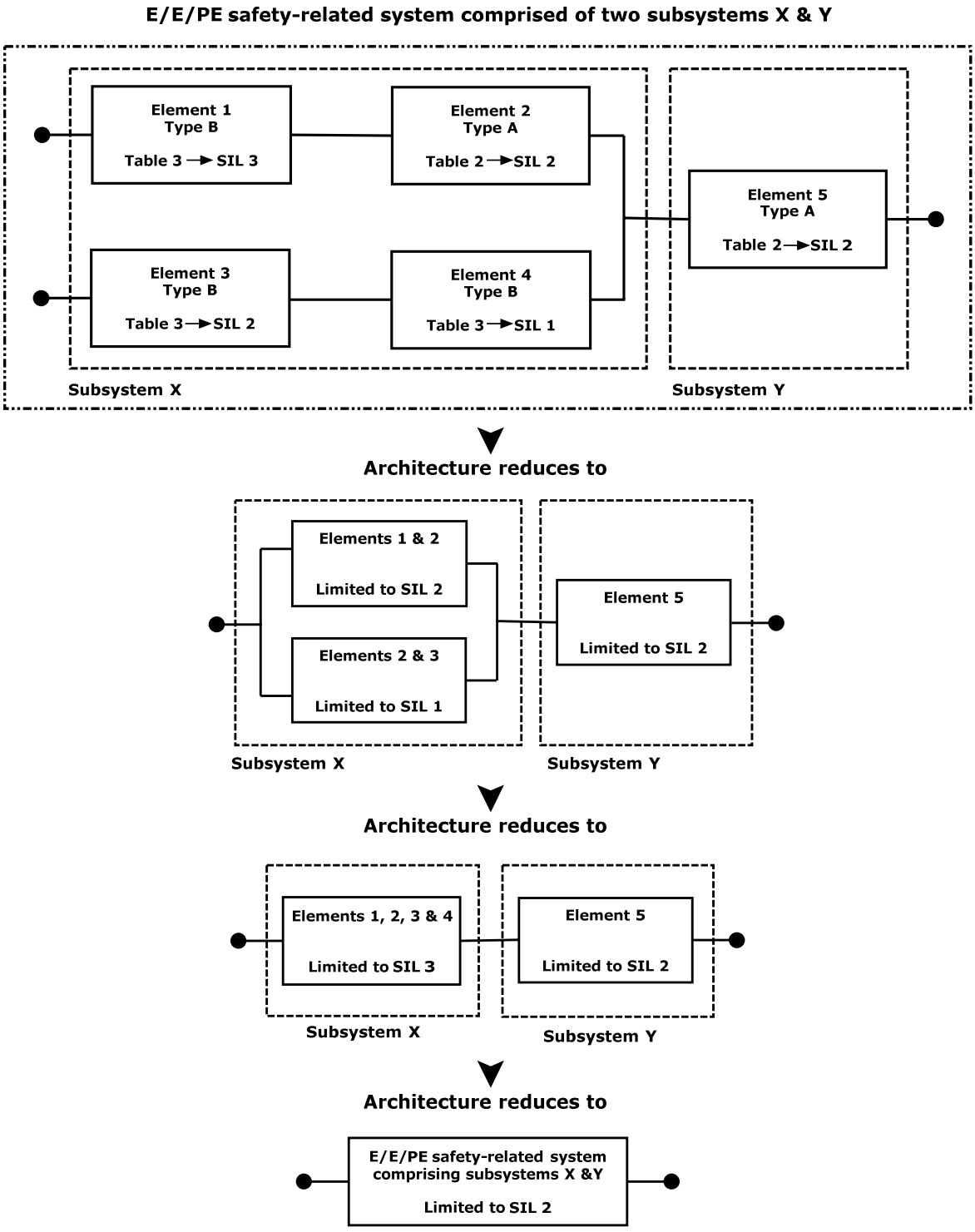
a) Combining elements 1 and 2: The hardware fault tolerance and safe failure fraction achieved by the combination of elements 1 and 2 (each separately meeting the requirements for SIL 3 and SIL 2 respectively) meets the requirements of SIL 2 (determined by element 2; see 7.4.4.2.3);

b) Combining elements 3 and 4: The hardware fault tolerance and safe failure fraction achieved by the combination of elements 3 and 4 (each separately meeting the requirements for SIL 2 and SIL 1 respectively) meets the requirements of SIL 1 (determined by element 4 see 7.4.4.2.3);

c) Further combining the combination of elements 1 and 2 with the combination of elements 3 and 4: the maximum safety integrity level that can be claimed for the safety function under consideration is determined by selecting the channel with the highest safety integrity level that has been achieved and then adding N safety integrity levels to determine the maximum safety integrity level for the overall combination of elements. In this case the subsystem comprises two parallel channels with a hardware fault tolerance of 1. The channel with the highest safety integrity level, for the safety function under consideration was that comprising elements 1 and 2 which achieved the requirements for SIL 2. Therefore, the maximum safety integrity level for the subsystem for a hardware fault tolerance of 1 is (SIL 2 + 1) = SIL 3 (see 7.4.4.2.4).

For subsystem Y, element 5 achieves the requirements, for a hardware fault tolerance of 0 and, for a specific safe failure fraction, for SIL 2.

For the complete E/E/PE safety-related system (comprising two subsystems X and Y that have achieved the requirements, for the safety function under consideration, of SIL 3 and SIL 2 respectively), the maximum safety integrity level that can be claimed for an E/E/PE safety-related system is determined by the subsystem that has achieved the lowest safety integrity level (7.4.4.2.1 5)). Therefore, for this example, the maximum safety integrity level, that can be claimed for the E/E/PE safety-related system, for the safety function under consideration, is SIL 2.



NOTE 1 Elements 1 and 2 implement the part of the safety function required of subsystem X independently from elements 3 and 4, and vice versa.

NOTE 2 The subsystems implementing the safety function will be across the entire E/E/PE safety-related system in terms of ranging from the sensors to the actuators.

Figure 6 – Determination of the maximum SIL for specified architecture (E/E/PE safety-related subsystem comprised of two subsystems X & Y, see 7.4.4.2.4)

7.4.4.3 Route 2_H

7.4.4.3.1 The minimum hardware fault tolerance for each subsystem of an E/E/PE safety-related system implementing a safety function of a specified safety integrity level shall be as follows:

NOTE In the following clauses, unless otherwise specified, the safety function may be operating in either a low demand mode of operation or a high demand or continuous mode of operation.

a) a hardware fault tolerance of 2 for a specified safety function of SIL 4 unless the conditions in 7.4.4.3.2 apply.

b) a hardware fault tolerance of 1 for a specified safety function of SIL 3 unless the conditions in 7.4.4.3.2 apply.

c) a hardware fault tolerance of 1 for a specified safety function of SIL 2, operating in a high demand or continuous mode of operation, unless the conditions in 7.4.4.3.2 apply.

d) a hardware fault tolerance of 0 for a specified safety function of SIL 2 operating in a low demand mode of operation.

e) a hardware fault tolerance of 0 for a specified safety function of SIL 1.

7.4.4.3.2 For type A elements only, if it is determined that by following the HFT requirements specified in 7.4.4.3.1, for the situation where an HFT greater than 0 is required, it would introduce additional failures and lead to a decrease in the overall safety of the EUC, then a safer alternative architecture with reduced HFT may be implemented. In such a case this shall be justified and documented. The justification shall provide evidence that:

a) compliance with the HFT requirements specified in 7.4.4.3.1 would introduce additional failures and lead to a decrease in the overall safety of the EUC; and

b) if the HFT is reduced to zero, the failure modes, identified in the element performing the safety function, can be excluded because the dangerous failure rate(s) of the identified failure mode(s) are very low compared to the target failure measure for the safety function under consideration (see 7.4.4.1.1 c)). That is, the sum of the dangerous failure frequencies of all serial elements, on which fault exclusion is being claimed, should not exceed 1 % of the target failure measure. Furthermore the applicability of fault exclusions shall be justified considering the potential for systematic faults

NOTE Fault tolerance is the preferred solution to achieve the required confidence that a robust architecture has been achieved. When 7.4.4.3.2 applies, the purpose of the justification is to demonstrate that the proposed alternative architecture provides an equivalent or better solution. This may depend on the technical field and/or the application. Examples include: back-up arrangements (e.g., analytical redundancy, replacing a failed sensor output by physical calculation results from other sensors outputs); using more reliable items of the same technology (if available); changing for a more reliable technology; decreasing common cause failure impact by using diversified technology; increasing the design margins; constraining the environmental conditions (e.g. for electronic components); decreasing the reliability uncertainty by gathering more field feedback or expert judgement.

7.4.4.3.3 If Route 2_H is selected, then the reliability data used when quantifying the effect of random hardware failures (see 7.4.5) shall be:

a) based on field feedback for elements in use in a similar application and environment; and,

b) based on data collected in accordance with international standards (e.g., IEC 60300-3-2 or ISO 14224:); and,

c) evaluated according to:

   i) the amount of field feedback; and,

   ii) the exercise of expert judgement; and where needed,

   iii) the undertaking of specific tests;

in order to estimate the average and the uncertainty level (e.g., the 90 % confidence interval or the probability distribution (see Note 2)) of each reliability parameter (e.g., failure rate) used in the calculations.

NOTE 1 End-users are encouraged to organize relevant component reliability data collections as described in published standards.

NOTE 2 The 90 % confidence interval of a failure rate λ is the interval [$\lambda_{5\ \%}$, $\lambda_{95\ \%}$] in which its actual value has a probability of 90 % to belong to. λ has a probability of 5 % to be better than $\lambda_{5\ \%}$ and worse than $\lambda_{95\ \%}$. On a pure statistical basis, the average of the failure rate may be estimated by using the "maximum likelihood estimate" and the confidence bounds ($\lambda_{5\ \%}$, $\lambda_{95\ \%}$) may be calculated by using the χ^2 function. The accuracy depends on the cumulated observation time and the number of failures observed. The Bayesian approach may be used to handle statistical observations, expert judgement and specific test results. This can be used to fit relevant probabilistic distribution functions for further use in Monte Carlo simulation.

If route 2_H is selected, then the reliability data uncertainties shall be taken into account when calculating the target failure measure (i.e. PFD_{avg} or PFH) and the system shall be improved until there is a confidence greater than 90 % that the target failure measure is achieved.

7.4.4.3.4 All type B elements used in Route 2_H shall have, as a minimum, a diagnostic coverage of not less than 60 %.

7.4.5 Requirements for quantifying the effect of random hardware failures

NOTE Clause A.2 of IEC 61508-6, gives an overview of the necessary steps in achieving required hardware safety integrity, and shows how this subclause relates to other requirements of this standard.

7.4.5.1 For each safety function, the achieved safety integrity of the E/E/PE safety-related system due to random hardware failures (including soft-errors) and random failures of data communication processes shall be estimated in accordance with 7.4.5.2 and 7.4.11, and shall be equal to or less than the target failure measure as specified in the E/E/PE system safety requirements specification (see IEC 61508-1, 7.10).

NOTE In order to demonstrate that this has been achieved, it is necessary to carry out a reliability prediction for the relevant safety function using an appropriate technique (see 7.4.5.2) and compare the result to the target failure measure of the relevant safety function (see IEC 61508-1).

7.4.5.2 The estimate of the achieved failure measure for each safety function, as required by 7.4.5.1, shall take into account:

a) the architecture of the E/E/PE safety-related system, in terms of its subsystems, as it relates to each safety function under consideration;

NOTE 1 This involves deciding which failure modes of the elements of the subsystems are in a series configuration (i.e. any failure causes failure of the relevant safety function to be carried out) and which are in a parallel configuration (i.e. coincident failures are necessary for the relevant safety function to fail).

b) the architecture of each subsystem of the E/E/PE safety-related system, in terms of its elements, as it relates to each safety function under consideration;

c) the estimated failure rate of each subsystem and its elements in any modes that would cause a dangerous failure of the E/E/PE safety-related system but are detected by diagnostic tests (see 7.4.9.4 to 7.4.9.5). Justification for the failure rates should be given considering the source of the data and its accuracy or tolerance. This may include consideration and the comparison of data from a number of sources and the selection of failure rates from systems most closely resembling that under consideration. Failure rates used for quantifying the effect of random hardware failures and calculating safe failure fraction or diagnostic coverage shall take into account the specified operating conditions.

NOTE 2 To take into account the operating conditions it will normally be necessary to adjust failure rates from data bases for example due to contact load or temperature.

d) the susceptibility of the E/E/PE safety-related system and its subsystems to common cause failures (see Notes 3 and 4). There shall be a justification of the assumptions made;

NOTE 3 Failures due to common cause effects may result from effects other than actual failures of hardware elements (e.g. electromagnetic interference, decoding errors, etc). However, such failures are considered, for the purposes of this standard, in the quantification of the effect of random hardware failures. Staggering the testing of elements decreases the likelihood of common cause failure.

NOTE 4 In the case of common cause failures being identified between the E/E/PE safety–related systems and demand causes or other protection layers there will need to be confirmation that this has been taken into account when the safety integrity level and target failure measure requirements have been determined. For methods of determining common cause factors see IEC 61508-6, Annex D.

e) the diagnostic coverage of the diagnostic tests (determined according to Annex C), the associated diagnostic test interval and the rate of dangerous unrevealed failure of the diagnostics due to random hardware failures of each subsystem. Where relevant, only those diagnostic tests that meet the requirements of 7.4.5.3 shall be considered. The MTTR and MRT (see 3.6.21 and 3.6.22 of IEC 61508-4), shall be considered in the reliability model.

NOTE 5 When establishing the diagnostic test interval, the intervals between all of the tests that contribute to the diagnostic coverage will need to be considered.

f) the intervals at which proof tests are undertaken to reveal dangerous faults;

g) whether the proof test is likely to be 100 % effective;

NOTE 6 An imperfect proof test will result in a safety function that is not restored to 'as good as new' and therefore the probability of failure will increase. Justification should be given for the assumptions made, in particular, the renewable period of the elements or the effect on the risk reduction over the life of the safety function should be included. It will be necessary to consider the test duration if the item is tested off-line whilst testing is being undertaken.

h) the repair times for detected failures;

NOTE 7 The mean repair time (MRT) is one part of the mean time to restoration (MTTR), (see 3.6.22 and 3.6.21of IEC 61508-4), which will also include the time taken to detect a failure and any time period during which repair is not possible (see Annex B of IEC 61508-6, for an example of how the MTTR and the MRT can be used to calculate the probability of failure). The repair can be considered to be instantaneous only when the EUC is shut-down or in a safe state during repair. For situations where the repair cannot be carried out whilst the EUC is shut down and in a safe state, it is particularly important that full account is taken of the time period when no repair can be carried out, especially when this is relatively large. All relevant factors relating to repairs should be taken into account.

i) the effect of random human error if a person is required to take action to achieve the safety function.

NOTE 8 The random nature of human error should be considered in cases where a person is alerted to an unsafe condition and is required to take action and the probability of human error should be included in the overall calculation.

j) the fact that a number of modelling methods are available and that the most appropriate method is a matter for the analyst and will depend on the circumstances. Available methods include cause consequence analysis (B.6.6.2 of IEC 61508-7;), fault tree analysis (B.6.6.5 of IEC 61508-7;), Markov models (Annex B of IEC 61508-6 and B.6.6.6 of IEC 61508-7), reliability block diagrams (Annex B of IEC 61508-6 and B.6.6.7 of IEC 61508-7;) and Petri nets (Annex B of IEC 61508-6 and B.2.3.3 of IEC 61508-7).

NOTE 9 Annex B of IEC 61508-6 describes a simplified approach that may be used to estimate the average probability of a dangerous failure on demand of a safety function due to random hardware failures in order to determine that an architecture meets the required target failure measure.

NOTE 10 Clause A.2 of IEC 61508-6 gives an overview of the necessary steps in achieving required hardware safety integrity, and shows how this subclause relates to other requirements of this standard.

NOTE 11 It is necessary to quantify separately for each safety function the reliability of the E/E/PE safety-related systems because different element failure modes will apply and the architecture of the E/E/PE safety-related systems (in terms of redundancy) may also vary.

7.4.5.3 When quantifying the effect of random hardware failures of a subsystem, having a hardware fault tolerance of 0, and which is implementing a safety function, or part of a safety function, operating in high demand mode or continuous mode of operation, credit shall only be taken for the diagnostics if:

– the sum of the diagnostic test interval and the time to perform the specified action to achieve or maintain a safe state is less than the process safety time; or

– in high demand mode of operation the ratio of the diagnostic test rate to the demand rate equals or exceeds 100.

7.4.5.4 The diagnostic test interval of any subsystem:

– having a hardware fault tolerance greater than 0, and which is implementing a safety function, or part of a safety function, operating in high demand mode or continuous mode of operation; or

– which is implementing a safety function, or part of a safety function, operating in low demand mode of operation,

shall be such that the sum of the diagnostic test interval and the time to perform the repair of a detected failure is less than the MTTR used in the calculation to determine the achieved safety integrity for that safety function.

7.4.5.5 If, for a particular design, the safety integrity requirement for the relevant safety function is not achieved then:

a) determine the elements, subsystems and/or parameters contributing most to the function's calculated failure rate;

b) evaluate the effect of possible improvement measures on the identified critical elements, subsystems or parameters (for example, more reliable components, additional defences against common mode failures, increased diagnostic coverage, increased redundancy, reduced proof test interval, staggering tests, etc);

c) select and implement the applicable improvements;

d) repeat the necessary steps to establish the new probability of a random hardware failure.

7.4.6 Requirements for the avoidance of systematic faults

NOTE See 7.4.2.2 c) for details, when the requirements of this subclause apply.

7.4.6.1 An appropriate group of techniques and measures shall be used that are designed to prevent the introduction of faults during the design and development of the hardware and software of the E/E/PE safety-related system (see Table B.2 and IEC 61508-3).

NOTE This standard does not contain specific requirements relating to the avoidance of systematic faults during the design of mass-produced electronic integrated circuits such as standard microprocessors. This is because the likelihood of faults in such devices is minimised by stringent development procedures, rigorous testing and extensive experience of use with significant feedback from users. For electronic integrated circuits that cannot be justified on such a basis (for example, new devices or ASICs), the requirements for ASICs (see 7.4.6.7 and informative Annex F) will apply if they are to be used in an E/E/PE safety-related system. In case of doubt (about extensive experience of use with significant feedback from users) the requirements for “field experience” from Table B.6 should be taken into account with an effectiveness of “low” for SIL 1 and SIL 2, an effectiveness of “medium” for SIL 3 and an effectiveness of “high” for SIL 4.

7.4.6.2 In accordance with the required safety integrity level the design method chosen shall possess features that facilitate

a) transparency, modularity and other features that control complexity;

b) clear and precise expression of
   - functionality;
   - subsystem and element interfaces;
   - sequencing and time-related information;
   - concurrency and synchronisation;

c) clear and precise documentation and communication of information;

d) verification and validation.

7.4.6.3 Maintenance requirements, to ensure that the safety integrity requirements of the E/E/PE safety-related systems continue to be met, shall be formalised at the design stage.

7.4.6.4 Where applicable, automatic testing tools and integrated development tools shall be used.

7.4.6.5 During the design, E/E/PE system integration tests shall be planned. Documentation of the test planning shall include

a) the types of tests to be performed and procedures to be followed;

b) the test environment, tools, configuration and programs;

c) the pass/fail criteria.

7.4.6.6 During the design, those activities that can be carried out on the developer's premises shall be distinguished from those that require access to the user's site.

7.4.6.7 An appropriate group of techniques and measures shall be used that are essential to prevent the introduction of faults during the design and development of ASICs.

NOTE Techniques and measures that support the achievement of relevant properties are given in informative Annex F. The related ASIC development lifecycle is shown in Figure 3.

7.4.7 Requirements for the control of systematic faults

NOTE See 7.4.2.2 c) for details, when the requirements of this subclause apply.

7.4.7.1 For controlling systematic faults, the E/E/PE system design shall possess design features that make the E/E/PE safety-related systems tolerant against:

a) any residual design faults in the hardware, unless the possibility of hardware design faults can be excluded (see Table A.15);

b) environmental stresses, including electromagnetic disturbances (see Table A.16);

c) mistakes made by the operator of the EUC (see Table A.17);

d) any residual design faults in the software (see 7.4.3 of IEC 61508-3 and associated table);

e) errors and other effects arising from any data communication process (see 7.4.11).

7.4.7.2 Maintainability and testability shall be considered during the design and development activities in order to facilitate implementation of these properties in the final E/E/PE safety-related systems.

7.4.7.3 The design of the E/E/PE safety-related systems shall take into account human capabilities and limitations and be suitable for the actions assigned to operators and maintenance staff. Such design requirements shall follow good human-factor practice and shall accommodate the likely level of training or awareness of operators, for example in mass-produced E/E/PE safety-related systems where the operator is a member of the public.

NOTE 1 The design goal should be that foreseeable critical mistakes made by operators or maintenance staff are prevented or eliminated by design wherever possible, or that the action requires secondary confirmation before completion.

NOTE 2 Some mistakes made by operators or maintenance staff may not be recoverable by E/E/PE safety-related systems, for example if they are not detectable or realistically recoverable except by direct inspection, such as some mechanical failures in the EUC.

7.4.8 Requirements for system behaviour on detection of a fault

NOTE The requirements of this subclause apply to specified safety functions implemented by a single E/E/PE safety-related system where the overall safety function has not been allocated to other risk reduction measures.

7.4.8.1 The detection of a dangerous fault (by diagnostic tests, proof tests or by any other means) in any subsystem that has a hardware fault tolerance of more than 0 shall result in either:

a) a specified action to achieve or maintain a safe state (see Note); or

b) the isolation of the faulty part of the subsystem to allow continued safe operation of the EUC whilst the faulty part is repaired. If the repair is not completed within the mean repair time (MRT), see 3.6.22 of IEC 61508-4, assumed in the calculation of the probability of random hardware failure (see 7.4.5.2), then a specified action shall take place to achieve or maintain a safe state (see Note).

NOTE The specified action required to achieve or maintain a safe state will be specified in the E/E/PE system safety requirements (see IEC 61508-1, 7.10). It may consist, for example, of the safe shut-down of the EUC, or that part of the EUC that relies, for functional safety, on the faulty subsystem.

7.4.8.2 The detection of a dangerous fault (by diagnostic tests, proof tests or by any other means) in any subsystem having a hardware fault tolerance of 0 shall, in the case that the subsystem is used only by safety function(s) operating in the low demand mode, result in either:

a) a specified action to achieve or maintain a safe state; or

b) the repair of the faulty subsystem within the mean repair time (MRT), see 3.6.22 of IEC 61508-4,assumed in the calculation of the probability of random hardware failure (see 7.4.5.2). During this time the continuing safety of the EUC shall be ensured by additional measures and constraints. The safety integrity provided by these measures and constraints shall be at least equal to the safety integrity provided by the E/E/PE safety-related system in the absence of any faults. The additional measures and constraints shall be specified in the E/E/PE system operation and maintenance procedures (see 7.6).

NOTE The specified action required to achieve or maintain a safe state will be specified in the E/E/PE system safety requirements specification (see 7.10 of IEC 61508-1). It may consist, for example, of the safe shut-down of the EUC, or that part of the EUC that relies, for functional safety, on the faulty subsystem.

7.4.8.3 The detection of a dangerous fault (by diagnostic tests, proof tests or by any other means) in any subsystem having a hardware fault tolerance of 0 shall, in the case of a subsystem that is implementing any safety function(s) operating in the high demand or the continuous mode, result in a specified action to achieve or maintain a safe state (see Note).

NOTE The specified action required to achieve or maintain a safe state will be specified in the E/E/PE system safety requirements (see IEC 61508-1, 7.10). It may consist, for example, of the safe shut-down of the EUC, or that part of the EUC that relies, for functional safety, on the faulty subsystem.

7.4.9 Requirements for E/E/PE system implementation

7.4.9.1 The E/E/PE safety-related system shall be implemented according to the E/E/PE system design requirements specification (7.2.3).

7.4.9.2 All subsystems and their elements that are used by one or more safety functions shall be identified and documented as safety-related subsystems and elements.

7.4.9.3 The following information shall be available for each safety-related subsystem and each element as appropriate (see also 7.4.9.4):

NOTE It will be necessary for a supplier of a subsystem or element, claimed as being compliant with IEC 61508, to make this information available to the designer of a safety-related system (or another subsystem or element) in the safety manual for compliant items, see Annex D.

a) a functional specification of the subsystem and its elements as appropriate;

b) any instructions or constraints relating to the application of the subsystem and its elements, that should be observed in order to prevent systematic failures of the subsystem;

c) the systematic capability of each element (see 7.4.2.2 c));

d) identification of the hardware and/or software configuration of the element to enable configuration management of the E/E/PE safety-related system in accordance with 6.2.1 of IEC 61508-1;

e) documentary evidence that the subsystem and its elements have been verified as meeting their specified functional requirements and systematic capabilities in accordance with the E/E/PE design requirements specification (see 7.2.3).

7.4.9.4 The following information shall be available for each safety-related element that is liable to random hardware failure (see also 7.4.9.3 and 7.4.9.5):

NOTE 1 It will be necessary for a supplier of an element, claimed as being compliant with IEC 61508 series, to make this information available to the designer of a safety-related system in the element safety manual, see Annex D.

a) the failure modes of the element (in terms of the behaviour of its outputs), due to random hardware failures, that result in a failure of the safety function and that are not detected by diagnostic tests internal to the element or are not detectable by diagnostics external to the element (see 7.4.9.5);

b) for every failure mode in a), an estimated failure rate with respect to specified operating conditions;

c) the failure modes of the element (in terms of the behaviour of its outputs), due to random hardware failures, that result in a failure of the safety function and that are detected by diagnostic tests internal to the element or are detectable by diagnostics external to the element (see 7.4.9.5);

d) for every failure mode in c), an estimated failure rate with respect to specified operating conditions;

e) any limits on the environment of the element that should be observed in order to maintain the validity of the estimated rates of failure due to random hardware failures;

f) any limit on the lifetime of the element that should not be exceeded in order to maintain the validity of the estimated rates of failure due to random hardware failures;

g) any periodic proof test and/or maintenance requirements;

h) for every failure mode in c) that is detected by diagnostics internal to the element, the diagnostic coverage derived according to Annex C (see Note 2);

i) for every failure mode in c) that is detected by diagnostics internal to the element, the diagnostic test interval (see Note 2);

NOTE 2 The diagnostic coverage and diagnostic test interval is required to allow credit to be claimed for the action of the diagnostic tests performed in the element in the hardware safety integrity model of the E/E/PE safety-related system (see 7.4.5.2, 7.4.5.3 and 7.4.5.4).

j) the failure rate of the diagnostics, due to random hardware failures;

k) any additional information (for example repair times) that is necessary to allow the derivation of the mean repair time (MRT), see 3.6.22 of IEC 61508-4,following detection of a fault by the diagnostics;

l) all information that is necessary to enable the derivation of the safe failure fraction (SFF) of the element as applied in the E/E/PE safety-related system, determined according to Annex C, including the classification as type A or type B according to 7.4.4;

m) the hardware fault tolerance of the element.

7.4.9.5 The estimated failure rates, due to random hardware failures, for elements (see 7.4.9.4 a) and c)) can be determined either

a) by a failure modes and effects analysis of the design using element failure data from a recognised industry source; or

b) from experience of the previous use of the element in a similar environment (see 7.4.10).

NOTE 1 Any failure rate data used should have a confidence level of at least 70 %. The statistical determination of confidence level is defined in reference [9] of the Bibliography. For an equivalent term: “significance level”, see reference [10].

NOTE 2 If site-specific failure data are available then this is preferred. If this is not the case then generic data may have to be used.

NOTE 3 Although a constant failure rate is assumed by most probabilistic estimation methods this only applies provided that the useful lifetime of elements is not exceeded. Beyond their useful lifetime (i.e. as the probability of failure significantly increases with time) the results of most probabilistic calculation methods are therefore meaningless. Thus any probabilistic estimation should include a specification of the elements’ useful lifetimes. The useful lifetime is highly dependent on the element itself and its operating conditions – temperature in particular (for example, electrolyte capacitors can be very sensitive). Experience has shown that the useful lifetime often lies within a range of 8 to 12 years. It can, however, be significantly less if elements are operated near to their specification limits.

7.4.9.6 Suppliers shall provide a safety manual for compliant items, in accordance with Annex D, for each compliant item that they supply and for which they claim compliance with IEC 61508 series.

7.4.9.7 The supplier shall document a justification for all the information that is provided in each safety manual for compliant items.

NOTE 1 It is essential that the claimed safety performance of an element is supported by sufficient evidence. Unsupported claims do not help establish the correctness and integrity of the safety function to which the element contributes.

NOTE 2 There may be commercial or legal restrictions on the availability of the evidence. These restrictions are outside the scope of this standard. If such restrictions deny the functional safety assessment adequate access to the evidence, then the element is not suitable for use in E/E/PE safety-related systems.

7.4.10 Requirements for proven in use elements

NOTE See 7.4.2.2 c) for details, when the requirements of this subclause apply.

7.4.10.1 An element shall only be regarded as proven in use when it has a clearly restricted and specified functionality and when there is adequate documentary evidence to demonstrate that the likelihood of any dangerous systematic faults is low enough that the required safety integrity levels of the safety functions that use the element is achieved. Evidence shall be based on analysis of operational experience of a specific configuration of the element together with suitability analysis and testing.

NOTE Suitability analysis and testing focuses on the demonstration of the element's performance within the intended application. The results of existing analysis and testing should 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.

7.4.10.2 The documentary evidence required by 7.4.10.1 shall demonstrate that:

a) the previous conditions of use (see Note 1) of the specific element are the same as, or sufficiently close to, those that will be experienced by the element in the E/E/PE safety-related system;

NOTE 1 The conditions of use (operational profile) include all the factors that may trigger systematic faults in the hardware and software of the element. For example environment, modes of use, functions performed, configuration, interfaces to other systems, operating system, translator, human factors. Rigorous conditions for similarity of operational profile may be found in IEC 61784-3.

b) the dangerous failure rate has not been exceeded in previous use.

NOTE 2 See IEC 61508-7, Annex D, for guidelines on the use of a probabilistic approach to determining software safety integrity for pre-developed software based on operational experience

NOTE 3 The collection of evidence for proven in use elements requires an effective system for reporting failures.

7.4.10.3 When there is any difference between the previous conditions of use and those that will be experienced in the E/E/PE safety-related system, then an impact analysis on the differences shall be carried out using a combination of appropriate analytical methods and testing, in order to demonstrate that the likelihood of any dangerous systematic faults is low enough that the required safety integrity level(s) of the safety function(s) that use the element is achieved.

7.4.10.4 A proven in use safety justification shall be documented, using the information available from 7.4.10.2, that the element supports the required safety function with the required systematic safety integrity. This shall include:

a) the suitability analysis and testing of the element for the intended application;

b) the demonstration of equivalence between the intended operation and the previous operation experience, including the impact analysis on the differences;

c) the statistical evidence.

7.4.10.5 The following factors shall be taken into account when determining whether or not the above requirements (7.4.10.1 to 7.4.10.4) have been met, in terms of both the coverage and degree of detail of the available information (see also 4.1 of IEC 61508-1):

a) the complexity of the element;

b) the systematic capability required for the element;

c) the novelty of design.

7.4.10.6 There shall be satisfactory evidence that, the existing element's functions that are not covered by the proven in use demonstration, cannot adversely affect the safety integrity of the element functions that are used.

NOTE This requirement can be achieved by ensuring that the functions are physically or electrically disabled or that software to implement these functions is excluded from the operational configuration, or by other forms of arguments and evidence.

7.4.10.7 Any future modification of a proven in use element shall comply with the requirements of 7.8, and IEC 61508-3.

7.4.11 Additional requirements for data communications

7.4.11.1 When data communication is used in the implementation of a safety function then the failure measure (such as the residual error rate) of the communication process shall be estimated taking into account transmission errors, repetitions, deletion, insertion, re-sequencing, corruption, delay and masquerade. This failure measure shall be taken into account when estimating the failure measure of the safety function due to random failures (see 7.4.5).

NOTE The term: "masquerade" means that the true source of a message is not correctly identified. For example, a message from a non-safety element is incorrectly identified as a message from a safety element.

7.4.11.2 The techniques and measures necessary to ensure the required failure measure (such as the residual error rate) of the communication process (see 7.4.11.1) shall be implemented according to the requirements of this standard and IEC 61508-3. This allows two possible approaches:

– the entire communication channel shall be designed, implemented and validated according to the IEC 61508 series and IEC 61784-3 or IEC 62280 series. This a so-called 'white channel' (see Figure 7 a); or

– parts of the communication channel are not designed or validated according to the IEC 61508 series. This is a so-called 'black channel' (see Figure 7 b). In this case, the measures necessary to ensure the failure performance of the communication process shall be implemented in the E/E/PE safety-related subsystems or elements that interface with the communication channel in accordance with the IEC 61784-3 or IEC 62280 series as appropriate.

Figure 7 (a) White channel

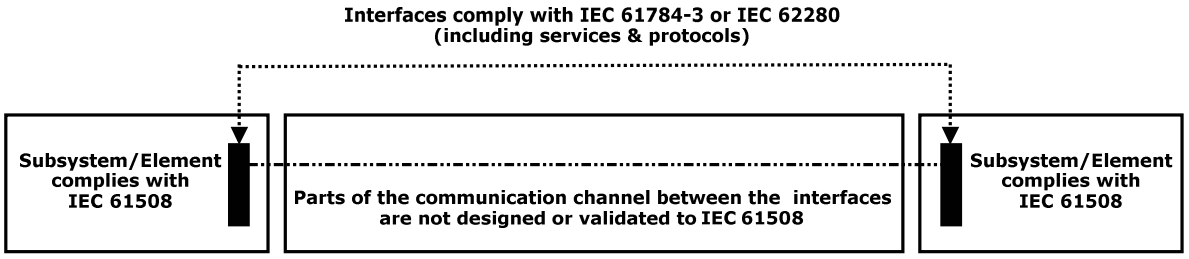


Figure 7 (b) Black channel

Figure 7 – Architectures for data communication

7.5 E/E/PE system integration

NOTE This phase is Box 10.4 of Figure 2.

7.5.1 Objective

The objective of the requirements of this subclause is to integrate and test the E/E/PE safety-related system.

7.5.2 Requirements

7.5.2.1 The E/E/PE safety-related system shall be integrated according to the specified E/E/PE system design and shall be tested according to the specified E/E/PE system integration tests (see 7.4.2.11).

7.5.2.2 As part of the integration of all modules into the E/E/PE safety-related system, the E/E/PE safety-related system shall be tested as specified (see 7.4). These tests shall show that all modules interact correctly to perform their intended function and are designed not to perform unintended functions.

NOTE 1 This does not imply testing of all input combinations. Testing all equivalence classes (see B.5.2 of IEC 61508-7) may suffice. Static analysis (see B.6.4 of IEC 61508-7), dynamic analysis (see B.6.5 of IEC 61508-7) or failure analysis (see B.6.6 of IEC 61508-7) may reduce the number of test cases to an acceptable level. The requirements are easier to fulfil if the E/E/PE safety-related system is developed using structured design (see B.3.2 of 61508-7) or semi-formal methods (see B.2.3 of 61508-7).

NOTE 2 Where the development uses formal methods (see B.2.2 of IEC 61508-7) or formal proofs or assertions (see C.5.12 and C.3.3 of 61508-7), such tests may be reduced in scope.

NOTE 3 Statistical evidence may be used as well (see B.5.3 of IEC 61508-7).

7.5.2.3 The integration of safety-related software into the E/E/PE safety-related system shall be carried out according to 7.5 of IEC 61508-3.

7.5.2.4 Appropriate documentation of the integration testing of the E/E/PE safety-related system shall be produced, stating the test results and whether the objectives and criteria specified during the design and development phase have been met. If there is a failure, the reasons for the failure and its correction shall be documented.

7.5.2.5 During the integration and testing, any modifications or change to the E/E/PE safety-related system shall be subject to an impact analysis which shall identify all subsystems and elements affected and the necessary re-verification activities.

7.5.2.6 The E/E/PE system integration testing shall document the following information:

a) the version of the test specification used;

b) the criteria for acceptance of the integration tests;

c) the version of the E/E/PE safety-related system being tested;

d) the tools and equipment used along with calibration data;

e) the results of each test;

f) any discrepancy between expected and actual results;

g) the analysis made and the decisions taken on whether to continue the test or issue a change request, in the case when discrepancies occur.

7.5.2.7 For the avoidance of faults during the E/E/PE system integration, an appropriate group of techniques and measures according to Table B.3 shall be used.

7.6 E/E/PE system operation and maintenance procedures

NOTE This phase is Box 10.5 of Figure 2.

7.6.1 Objective

The objective of the requirements of this subclause is to develop procedures to ensure that the required functional safety of the E/E/PE safety-related system is maintained during operation and maintenance.

7.6.2 Requirements

7.6.2.1 E/E/PE system operation and maintenance procedures shall be prepared. They shall specify the following:

a) the routine actions that need to be carried out to maintain the as-designed functional safety of the E/E/PE safety-related system, including routine replacement of elements with a pre-defined life, for example cooling fans, batteries; etc.

b) the actions and constraints that are necessary (for example, during installation, start-up, normal operation, routine testing, foreseeable disturbances, faults or failures, and shut-down) to prevent an unsafe state and/or reduce the consequences of a harmful event;

c) the documentation that needs to be maintained on system failure and demand rates on the E/E/PE safety-related system;

d) the documentation that needs to be maintained showing results of audits and tests on the E/E/PE safety-related system;

e) the maintenance procedures to be followed when faults or failures occur in the E/E/PE safety-related system, including:

   – procedures for fault diagnoses and repair;

   – procedures for revalidation;

   – maintenance reporting requirements;

   – procedures to re-validate if original equipment items are no longer available or have been superseded by new versions.

f) the procedures for reporting maintenance performance shall be specified. In particular:

   – procedures for reporting failures;

   – procedures for analysing failures;

g) the tools necessary for maintenance and revalidation and procedures for maintaining the tools and equipment.

NOTE 1 It may be beneficial, for reasons of both safety and economics, to integrate the E/E/PE system operation and maintenance procedures with the EUC overall operation and maintenance procedures.

NOTE 2 The E/E/PE system operation and maintenance procedures should include the software modification procedures (see IEC 61508-3, 7.8).

7.6.2.2 The E/E/PE safety-related system operation and maintenance procedures shall be continuously upgraded from inputs such as (1) the results of functional safety audits and (2) tests on the E/E/PE safety-related system.

7.6.2.3 The routine maintenance actions required to maintain the required functional safety (as designed) of the E/E/PE safety-related system shall be determined by a systematic method. This method shall determine unrevealed failures of all safety-related elements (from sensors through to final elements) that would cause a reduction in the safety integrity achieved. Suitable methods include:

– examination of fault trees;

– failure mode and effect analysis.

NOTE 1 A consideration of human factors is a key element in determining the actions required and the appropriate interface(s) with the E/E/PE safety-related system.

NOTE 2 Proof tests will be carried out with a frequency necessary to achieve the target failure measure.

NOTE 3 The frequency of the proof tests, the diagnostic test interval and the time for subsequent repair will be dependent upon several factors (see Annex B of IEC 61508-6), including:

– the target failure measure associated with the safety integrity level;

– the architecture;

– the diagnostic coverage of the diagnostic tests, and

– the expected demand rate.

NOTE 4 The frequency of the proof tests and the diagnostic test interval are likely to have a crucial bearing on the achievement of hardware safety integrity. One of the principal reasons for carrying out hardware reliability analysis (see 7.4.5.2) is to ensure that the frequencies of the two types of tests are appropriate for the target hardware safety integrity.

NOTE 5 Manufacturer's maintenance requirements should be followed and sole reliance should not be placed on reliability centred maintenance methods until it can be fully justified (e.g. by reliability analysis that demonstrates that the E/E/PE safety-related system's target failure measures are satisfied).

7.6.2.4 The E/E/PE system operation and maintenance procedures shall be assessed for the impact they may have on the EUC.

7.6.2.5 For the avoidance of faults and failures during the E/E/PE system operation and maintenance procedures, an appropriate group of techniques and measures according to Table B.4 shall be used.

7.7 E/E/PE system safety validation

NOTE This phase is Box 10.6 of Figure 2.

7.7.1 Objective

The objective of the requirements of this subclause is to validate that the E/E/PE safety-related system meets in all respects the requirements for safety in terms of the required safety functions and safety integrity (see 7.2 above and 7.10 of IEC 61508-1).

7.7.2 Requirements

7.7.2.1 The validation of the E/E/PE system safety shall be carried out in accordance with a prepared plan (see also 7.7 of IEC 61508-3).

NOTE 1 The E/E/PE system safety validation is shown on the E/E/PE system safety lifecycle as being carried out prior to installation but, in some cases, the E/E/PE system safety validation cannot be carried out until after installation (for example, when the application software development is not finalised until after installation).

NOTE 2 Validation of a programmable electronic safety-related system comprises validation of both hardware and software. The requirements for validation of software are contained in IEC 61508-3.

7.7.2.2 All test measurement equipment used for validation shall be calibrated against a standard traceable to a national standard, if available, or to a well-recognised procedure. All test equipment shall be verified for correct operation.

7.7.2.3 The adequate implementation of each safety function specified in the E/E/PE system safety requirements (see 7.10 of IEC 61508-1), the E/E/PE system design requirements (see 7.2), and all the E/E/PE system operation and maintenance procedures shall be validated by test and/or analysis. If adequate independence or decoupling between individual elements or subsystems cannot be demonstrated analytically, the related combinations of functional behaviour shall be tested.

NOTE As the number of necessary test combinations can get very large, a restructuring of the system may be required at this occasion.

7.7.2.4 Appropriate documentation of the E/E/PE system safety validation testing shall be produced which shall state for each safety function:

a) the version of the E/E/PE system safety validation plan being used;

b) the safety function under test (or analysis), along with the specific reference to the requirement specified during E/E/PE system safety validation planning;

c) tools and equipment used, along with calibration data;

d) the results of each test;

e) discrepancies between expected and actual results.

NOTE Separate documentation is not needed for each safety function, but the information in a) to e) must apply to every safety function and where it differs by safety function the relationship must be stated.

7.7.2.5 When discrepancies occur (i.e. the actual results deviate from the expected results by more than the stated tolerances), the results of the E/E/PE system safety validation testing shall be documented, including:

a) the analysis made; and

b) the decision taken on whether to continue the test or issue a change request and return to an earlier part of the validation test.

7.7.2.6 The supplier or developer shall make available results of the E/E/PE system safety validation testing to the developer of the EUC and the EUC control system so as to enable them to meet the requirements for overall safety validation in IEC 61508-1.

7.7.2.7 For the avoidance of faults during the E/E/PE system safety validation an appropriate group of techniques and measures according to Table B.5 shall be used.

7.8 E/E/PE system modification

7.8.1 Objective

The objective of the requirements of this subclause is to make corrections, enhancements or adaptations to the E/E/PE safety-related system, ensuring that the required safety integrity is achieved and maintained.

7.8.2 Requirements

7.8.2.1 Appropriate documentation shall be established and maintained for each E/E/PE system modification activity. The documentation shall include:

a) the detailed specification of the modification or change;

b) an analysis of the impact of the modification activity on the overall system, including hardware, software (see IEC 61508-3), human interaction and the environment and possible interactions;

c) all approvals for changes;

d) progress of changes;

e) test cases for subsystems and elements including revalidation data;

f) E/E/PE system configuration management history;

g) deviation from normal operations and conditions;

h) necessary changes to system procedures;

i) necessary changes to documentation.

7.8.2.2 Manufacturers or system suppliers that claim compliance with all or part of this standard shall maintain a system to initiate changes as a result of defects being detected in hardware or software and to inform users of the need for modification in the event of the defect affecting safety.

7.8.2.3 Modifications shall be performed with at least the same level of expertise, automated tools (see 7.4.4.2 of IEC 61508-3), and planning and management as the initial development of the E/E/PE safety-related systems.

7.8.2.4 After modification, the E/E/PE safety-related systems shall be reverified and revalidated.

NOTE See also 7.16.2.6 of IEC 61508-1.

7.9 E/E/PE system verification

7.9.1 Objective

The objective of the requirements of this subclause is to test and evaluate the outputs of a given phase to ensure correctness and consistency with respect to the products and standards provided as input to that phase.

NOTE For convenience all verification activities have been drawn together under 7.9, but they are actually performed for each relevant phase.

7.9.2 Requirements

7.9.2.1 The verification of the E/E/PE safety-related systems shall be planned concurrently with the development (see 7.4), for each phase of the E/E/PE system safety lifecycle, and shall be documented.

7.9.2.2 The E/E/PE system verification planning shall refer to all the criteria, techniques and tools to be utilised in the verification for that phase.

7.9.2.3 The E/E/PE system verification planning shall specify the activities to be performed to ensure correctness and consistency with respect to the products and standards provided as input to that phase.

7.9.2.4 The E/E/PE system verification planning shall consider the following:

a) the selection of verification strategies and techniques;

b) the selection and utilisation of the test equipment;

c) the selection and documentation of verification activities;

d) the evaluation of verification results gained from verification equipment direct and from tests.

7.9.2.5 In each design and development phase it shall be shown that the functional and safety integrity requirements are met.

7.9.2.6 The result of each verification activity shall be documented, stating either that the E/E/PE safety-related systems have passed the verification, or the reasons for the failures. The following shall be considered:

a) items that do not conform to one or more relevant requirements of the E/E/PE system safety lifecycle (see 7.2);

b) items that do not conform to one or more relevant design standards (see 7.4);

c) items that do not conform to one or more relevant safety management requirements (see Clause 6).

7.9.2.7 For E/E/PE system design requirements verification, after E/E/PE system design requirements have been established (see 7.2), and before the next phase (design and development) begins, verification shall:

a) determine whether the E/E/PE system design requirements are adequate to satisfy the E/E/PE system safety requirements specification (see 7.10 of IEC 61508-1) for safety, functionality, and other requirements specified during safety planning; and

b) check for incompatibilities between:
   - the E/E/PE system safety requirements (see 7.10 of IEC 61508-1);
   - the E/E/PE system design requirements (see 7.2);
   - the E/E/PE system tests (see 7.4); and
   - the user documentation and all other system documentation.

7.9.2.8 For E/E/PE system design and development verification, after E/E/PE system design and development (see 7.4) has been completed and before the next phase (integration) begins, verification shall:

a) determine whether the E/E/PE system tests are adequate for the E/E/PE system design and development;

b) determine the consistency and completeness (down to and including module level) of the E/E/PE system design and development with respect to the E/E/PE system safety requirements (see 7.10 of IEC 61508-1); and

c) check for incompatibilities between:
   - the E/E/PE system safety requirements (see 7.10 of IEC 61508-1);
   - the E/E/PE system design requirements (see 7.2);
   - the E/E/PE system design and development (see 7.4); and
   - the E/E/PE system tests (see 7.4).

NOTE 1 Table B.5 recommends safety validation, failure analysis and testing techniques that are also applicable to verification.

NOTE 2 Verification that the diagnostic coverage has been achieved will take into account Table A.1, which gives the faults and failures that must be detected.

7.9.2.9 For E/E/PE system integration verification, the integration of the E/E/PE safety-related system shall be verified to establish that the requirements of 7.5 have been achieved.

7.9.2.10 Test cases and their results shall be documented.

8 Functional safety assessment

The requirements for functional safety assessment are as detailed in Clause 8 of IEC 61508-1.

Annex A
(normative)

Techniques and measures for E/E/PE safety-related systems – control of failures during operation

A.1 General

This annex shall be used in conjunction with 7.4. It limits the maximum diagnostic coverage that may be claimed for relevant techniques and measures. For each safety integrity level, the annex recommends techniques and measures for controlling random hardware, systematic, environmental and operational failures. More information about architectures and measures can be found in Annex B of IEC 61508-6 and Annex A of IEC 61508-7.

It is not possible to list every individual physical cause of a failure in complex hardware for two main reasons:

– the cause/effect relationship between faults and failures is often difficult to determine;
– the emphasis on failures changes from random to systematic when complex hardware and software is used.

Failures in E/E/PE safety-related systems may be categorised, according to the time of their origin, into:

– failures caused by faults originating **before or during system installation** (for example, software faults include specification and program faults, hardware faults include manufacturing faults and incorrect selection of elements); and
– failures caused by faults or human errors originating **after system installation** (for example random hardware failures, or failures caused by incorrect use).

In order to avoid or control such failures when they occur, a large number of measures are normally necessary. The structure of the requirements in Annexes A and B results from dividing the measures into those used to **avoid failures** during the different phases of the E/E/PE system safety lifecycle (Annex B), and those used to **control failures** during operation (this Annex). The measures to control failures are built-in features of the E/E/PE safety-related systems.

Diagnostic coverage and safe failure fraction are determined on the basis of Table A.1 and according to procedures detailed in Annex C. Tables A.2 to A.14 support the requirements of Table A.1 by recommending techniques and measures for diagnostic tests and recommending maximum levels of diagnostic coverage that can be achieved using them. The tables do not replace any of the requirements of Annex C. Tables A.2 to A.14 are not exhaustive. Other measures and techniques may be used, provided evidence is produced to support the claimed diagnostic coverage. If high diagnostic coverage is being claimed then, as a minimum, at least one technique of high diagnostic coverage should be applied from each of these tables.

Similarly, Tables A.15 to A.17 recommends techniques and measures for each safety integrity level for controlling systematic failures. Table A.15 recommends overall measures to control systematic failures (see also IEC 61508-3), Table A.16 recommends measures to control environmental failures and Table A.17 recommends measures to control operational failures. Most of these control measures can be graded according to Table A.18.

All techniques and measures in these tables are described in Annex A of IEC 61508-7. Software techniques and measures required for each safety integrity level are given in IEC 61508-3. Guidelines for determining the architecture for an E/E/PE safety-related system are given in Annex B of IEC 61508-6.

Following the guidelines in this annex does not guarantee by itself the required safety integrity. It is important to consider the following:

– the consistency of the chosen techniques and measures, and how well they will complement each other; and

– which techniques and measures are most appropriate for the specific problems encountered during the development of each particular E/E/PE safety-related system.

A.2 Hardware safety integrity

Table A.1 provides the requirements for faults or failures that shall be detected by techniques and measures to control hardware failures, in order to achieve the relevant level of diagnostic coverage (see also Annex C). Tables A.2 to A.14 support the requirements of Table A.1 by recommending techniques and measures for diagnostic tests and recommending maximum levels of diagnostic coverage that can be achieved using them. These tests may operate continuously or periodically. The tables do not replace any of the requirements of 7.4. Tables A.2 to A.14 are not exhaustive. Other measures and techniques may be used, provided evidence is produced to support the claimed diagnostic coverage.

NOTE 1 The overview of techniques and measures associated with these tables is in Annex A of IEC 61508-7. The relevant subclause is referenced in the second column of Tables A.2 to A.14.

NOTE 2 The designations low, medium and high diagnostic coverage are quantified as 60 %, 90 % and 99 % respectively.

Table A.1 – Faults or failures to be assumed when quantifying the effect of random hardware failures or to be taken into account in the derivation of safe failure fraction

Component	See table(s)	Requirements for diagnostic coverage claimed		
		Low (60 %)	Medium (90 %)	High (99 %)
Electromechanical devices	A.2	Does not energize or de-energize	Does not energize or de-energize	Does not energize or de-energize
		Welded contacts	Individual contacts welded	Individual contacts welded No positive guidance of contacts (for relays this failure is not assumed if they are built and tested according to EN 50205 or equivalent)
				No positive opening (for position switches this failure is not assumed if they are built and tested according to IEC 60947-5-1, or equivalent)
Discrete hardware	A.3, A.7, A.9			
Digital I/O		Stuck-at (see Note 1)	DC fault model (see Note 2)	DC fault model drift and oscillation
Analogue I/O		Stuck-at	DC fault model drift and oscillation	DC fault model drift and oscillation
Power supply		Stuck-at	DC fault model drift and oscillation	DC fault model drift and oscillation
Bus	A.3			
General	A.7	Stuck-at of the addresses	Time out	Time out
Memory management unit (MMU)	A.8	Stuck-at of data or addresses	Wrong address decoding	Wrong address decoding
			Change of addresses caused by soft-errors in the MMU registers (see Notes 3 and 4)	Change of addresses caused by soft-errors in the MMU registers
Direct memory access (DMA)		No or continuous access	DC fault model for data and addresses	All faults that affect data in the memory
			Change of information caused by soft-errors in the DMA registers	Wrong access time
			Wrong access time	
Bus-arbitration (see Note 5)		Stuck-at of arbitration signals	No or continuous arbitration	No or continuous or wrong arbitration
Central Processing Unit (CPU)	A.4, A.10			
Register, internal RAM		Stuck-at for data and addresses	DC fault model for data and addresses	DC fault model for data and addresses
			Change of information caused by soft-errors	Dynamic cross-over for memory cells
				Change of information caused by soft-errors
				No, wrong or multiple addressing
Coding and execution including flag register		Wrong coding or no execution	Wrong coding or wrong execution	No definite failure assumption
Address calculation		Stuck-at	DC fault model	No definite failure assumption
			Change of addresses caused by soft-errors	
Program counter, stack pointer		Stuck-at	DC fault model	DC fault model
			Change of addresses caused by soft-errors	Change of addresses caused by soft-errors

Table A.1 *(continued)*

Component	See table(s)	Requirements for diagnostic coverage claimed		
		Low (60 %)	**Medium (90 %)**	**High (99 %)**
Interrupt handling	A.4			
Interrupt		No or continuous interrupts (see Note 6)	No or continuous interrupts	No or continuous interrupts
			Cross-over of interrupts	Cross-over of interrupts
Reset circuitry		Stuck-at	DC fault model	DC fault model
		Individual components do not initialize to reset state	Drift and oscillation	Drift and oscillation
			Individual components do not initialize to reset state	Individual components do not initialize to reset state
Invariable memory	A.5	Stuck-at for data and addresses	DC fault model for data and addresses	All faults that affect data in the memory
Variable memory	A.6	Stuck-at for data and addresses	DC fault model for data and addresses	DC fault model for data and addresses
			Change of information caused by soft-errors	Dynamic cross-over for memory cells
				Change of information caused by soft-errors
				No, wrong or multiple addressing
Clock (quartz, oscillator, PLL)	A.11	Sub- or super-harmonic	Incorrect frequency	Incorrect frequency
		Period jitter	Period jitter	Period jitter
Communication and mass storage	A.12	Wrong data or addresses	All faults that affect data in the memory	All faults that affect data in the memory
		No transmission	Wrong data or addresses	Wrong data or addresses
			Wrong transmission time	Wrong transmission time
			Wrong transmission sequence	Wrong transmission sequence
Sensors	A.13	Stuck-at	DC fault model	DC fault model
			Drift and oscillation	Drift and oscillation
Final elements	A.14	Stuck-at	DC fault model	DC fault model
			Drift and oscillation	Drift and oscillation

NOTE 1 "Stuck-at" is a fault category that can be described with continuous "0" or "1" or "on" at the pins of an element.

NOTE 2 "DC fault model" includes the following failure modes: stuck-at faults, stuck-open, open or high impedance outputs as well as short circuits between signal lines. For integrated circuits, short circuit between any two connections (pins) is considered.

NOTE 3 The soft-error rate (SER) for low energized semiconductors is known to be more than one order of magnitude higher (50x..500x) than the hard-error rate (permanent damage of the device).

NOTE 4 Causes of soft errors are: alpha particles from package decay, neutrons, external EMI noise and internal cross-talk. The effect of soft-errors can only be mastered by safety integrity measures at runtime. Safety integrity measures effective for random hardware failures may not be effective for soft-errors.

EXAMPLE: RAM tests, such as walk-path, galpat, etc. are not effective, whereas monitoring techniques using Parity and ECC with recurring read of the memory cells or techniques using redundancy (and comparison or voting) can be.

NOTE 5 Bus-arbitration is the mechanism for deciding which device has control of the bus.

NOTE 6 No interrupt means that no interrupt is carried out when an interrupt(s) should take place. Continuous interrupts means that continuous interrupts are carried out when they should not take place.

NOTE 7 For ASICs, this table and Tables A.2 to A.18 apply where relevant.

Table A.2 – Electrical components

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring	A.1.1	Low (low demand mode) Medium (high demand or continuous mode)	Depends on diagnostic coverage of failure detection
Monitoring of relay contacts	A.1.2	High	Relay switching rate should be taken into account when quantifying the effect of random failures
Comparator	A.1.3	High	High if failure modes are predominantly in a safe direction
Majority voter	A.1.4	High	Depends on the quality of the voting
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.3 – Electronic components

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring	A.1.1	Low (low demand mode) Medium (high demand or continuous mode)	Depends on diagnostic coverage of failure detection
Comparator	A.1.3	High	High if failure modes are predominantly in a safe direction
Majority voter	A.1.4	High	Depends on the quality of the voting
Tests by redundant hardware	A.2.1	Medium	Depends on diagnostic coverage of failure detection
Dynamic principles	A.2.2	Medium	Depends on diagnostic coverage of failure detection
Standard test access port and boundary-scan architecture	A.2.3	High	Depends on the diagnostic coverage of failure detection
Monitored redundancy	A.2.5	High	Depends on the degree of redundancy and of the monitoring
Hardware with automatic check	A.2.6	High	Depends on the diagnostic coverage of the tests
Analogue signal monitoring	A.2.7	Low	
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.4 – Processing units

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Comparator	A.1.3	High	Depends on the quality of the comparison
Majority voter	A.1.4	High	Depends on the quality of the voting
Self-test by software: limited number of patterns (one channel)	A.3.1	Low	
Self-test by software: walking bit (one-channel)	A.3.2	Medium	
Self-test supported by hardware (one-channel)	A.3.3	Medium	
Coded processing (one-channel)	A.3.4	High	
Reciprocal comparison by software	A.3.5	High	Depends on the quality of the comparison
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1. NOTE 4 As a number of processing unit faults lead to a modification of flow control, diagnostic measures and techniques listed in Table A.10 may also be taken into account for processing unit faults. These diagnostic measures and techniques cover the control flow only, not the data flow.			

Table A.5 – Invariable memory ranges

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Word-protection multi-bit redundancy	A.4.1	Medium	The effectiveness of the Word-protection multi-bit redundancy depends on the inclusion of the word address into the multiple bit redundancy, and relies on respective measure to detect multi-bit common cause faults, e.g. multiple addressing (multiple row select, multiple local to global bit line switches activated), power supply issues (e.g. charge pump flaws), production row and column replacement (production yield measure to mask production faults), etc.
Modified checksum	A.4.2	Low	
Signature of one word (8-bit)	A.4.3	Medium	The effectiveness of the signature depends on the width of the signature in relation to the block length of the information to be protected
Signature of a double word (16-bit)	A.4.4	High	The effectiveness of the signature depends on the width of the signature in relation to the block length of the information to be protected
Block replication	A.4.5	High	
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.6 – Variable memory ranges

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
RAM test checkerboard or march	A.5.1	Low	
RAM test walk-path	A.5.2	Medium	
RAM test galpat or transparent galpat	A.5.3	High	
RAM test Abraham	A.5.4	High	
Parity-bit for RAM	A.5.5	Low	
RAM monitoring with a modified Hamming code, or detection of data failures with error-detection-correction codes (EDC)	A.5.6	Medium	The effectiveness of the RAM monitoring with a modified Hamming code, or detection of data failures with error detection-correction codes (EDC) depends on the inclusion of the address into the Hamming code, and relies on respective measure to detect multi-bit common cause faults, e.g. multiple addressing (multiple row select, multiple local to global bit line switches activated), production row and column replacement (production yield measure to mask production faults), etc.
Double RAM with hardware or software comparison and read/write test	A.5.7	High	
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1. NOTE 4 For RAM that is read/written only infrequently (for example during configuration) the measures A.4.1 to A.4.4 of IEC 61508-7 are effective if they are executed after each read/write access.			

Table A.7 – I/O units and interface (external communication)

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring	A.1.1	Low (low demand mode) Medium (high demand or continuous mode)	Depends on diagnostic coverage of failure detection
Test pattern	A.6.1	High	
Code protection	A.6.2	High	
Multi-channel parallel output	A.6.3	High	Only if dataflow changes within diagnostic test interval
Monitored outputs	A.6.4	High	Only if dataflow changes within diagnostic test interval
Input comparison/voting (1oo2, 2oo3 or better redundancy)	A.6.5	High	Only if dataflow changes within diagnostic test interval
Antivalent signal transmission	A.11.4	High	For example transmission of inverted signals.
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.8 – Data paths (internal communication)

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
One-bit hardware redundancy	A.7.1	Low	In case of multiplane crossbar switch type of data path, the given effectiveness can only be assumed if the address and control lines are covered by the safety measures.
Multi-bit hardware redundancy	A.7.2	Medium	In case of multiplane crossbar switch type of data path, the given effectiveness can only be assumed if the address and control lines are covered by the safety measures.
Complete hardware redundancy	A.7.3	High	
Inspection using test patterns	A.7.4	High	
Transmission redundancy	A.7.5	High	Effective only against transient faults
Information redundancy	A.7.6	High	
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.9 – Power supply

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Overvoltage protection with safety shut-off or switch-over to second power unit	A.8.1	Low	
Voltage control (secondary) with safety shut-off or switch-over to second power unit	A.8.2	High	
Power-down with safety shut-off or switch-over to second power unit	A.8.3	High	
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.10 – Program sequence (watch-dog)

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Watch-dog with separate time base without time-window	A.9.1	Low	
Watch-dog with separate time base and time-window	A.9.2	Medium	
Logical monitoring of program sequence	A.9.3	Medium	Depends on the quality of the monitoring
Combination of temporal and logical monitoring of programme sequences	A.9.4	High	
Temporal monitoring with on-line check	A.9.5	Medium	
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.11 – Clock

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Watch-dog with separate time base without time-window	A.9.1	Low	
Watch-dog with separate time base and time-window	A.9.2	High	Depends on time restriction for the time-window
Logical monitoring of program sequence	A.9.3	Medium	Only effective against clock failures if external temporal events influence the logical program flow
Temporal and logical monitoring	A.9.4	High	
Temporal monitoring with on-line check	A.9.5	Medium	
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.12 – Communication and mass-storage

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Information exchange between E/E/PE safety-related system and process	A.6	See Table A.7	See I/O units and interface
Information exchange between E/E/PE safety-related systems	A.7	See Table A.8	See data paths/bus
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.13 – Sensors

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring	A.1.1	Low (low demand mode) Medium (high demand or continuous mode)	Depends on diagnostic coverage of failure detection
Analogue signal monitoring	A.2.7	Low	
Test pattern	A.6.1	High	
Input comparison/voting (1oo2, 2oo3 or better redundancy)	A.6.5	High	Only if dataflow changes within diagnostic test interval
Reference sensor	A.12.1	High	Depends on diagnostic coverage of failure detection
Positive-activated switch	A.12.2	High	
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

Table A.14 – Final elements (actuators)

Diagnostic technique/measure	See IEC 61508-7	Maximum diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring	A.1.1	Low (low demand mode) Medium (high demand or continuous mode)	Depends on diagnostic coverage of failure detection
Monitoring of relay contacts	A.1.2	High	Relay switching rate should be taken into account when quantifying the effect of random failures
Test pattern	A.6.1	High	
Monitoring	A.13.1	High	Depends on diagnostic coverage of failure detection
Cross-monitoring of multiple actuators	A.13.2	High	
NOTE 1 This table does not replace any of the requirements of Annex C. NOTE 2 The requirements of Annex C are relevant for the determination of diagnostic coverage. NOTE 3 For general notes concerning this table, see the text preceding Table A.1.			

A.3 Systematic safety integrity

The following tables give recommendations for techniques and measures to:

– control failures caused by hardware design (see Table A.15);

– control failures due to environmental stress or influences (see Table A.16); and

– control failures during operation (see Table A.17).

In Tables A.15 to A.17, recommendations are made and requirements are given by safety integrity level, stating firstly the importance of the technique or measure and secondly the effectiveness required if it is used. The importance is signified as follows:

– M: the technique or measure is required (mandatory) for this safety integrity level;

– HR: the technique or measure is highly recommended for this safety integrity level. If this technique or measure is not used then the rationale behind not using it shall be detailed;

– R: the technique or measure is recommended for this safety integrity level;

– -: the technique or measure has no recommendation for or against being used;

– NR: the technique or measure is positively not recommended for this safety integrity level; If this technique or measure is used then the rationale behind using it shall be detailed.

The required effectiveness is signified as follows:

– Low: if used, the technique or measure shall be used to the extent necessary to give at least low effectiveness against systematic failures;

– Medium: if used, the technique or measure shall be used to the extent necessary to give at least medium effectiveness against systematic failures;

– High: if used, the technique or measure shall be used to the extent necessary to give high effectiveness against systematic failures.

Guidance on levels of effectiveness for most techniques and measures is given in Table A.18.

If a measure is not mandatory, it is in principle replaceable by other measures (either individually or in combination); this is governed by the shading, as explained in the table.

All techniques and measures given here are built-in features of the E/E/PE safety-related systems, which may help to control failures on-line. Procedural and organisational techniques and measures are necessary throughout the E/E/PE system safety lifecycle to avoid introducing faults, and validation techniques to test the E/E/PE safety-related systems' behaviour against expected external influences are necessary to demonstrate that the built-in features are appropriate for the specific application (see Annex B).

Annex D of IEC 61508-6 gives information on common cause failures.

NOTE Most of the measures in Tables A.15 to A.17 can be used with varying effectiveness according to Table A.18, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and high effectiveness.

Table A.15 – Techniques and measures to control systematic failures caused by hardware design

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Program sequence monitoring	A.9	HR low	HR low	HR medium	HR high
	Failure detection by on-line monitoring (see Note 4)	A.1.1	R low	R low	R medium	R high
	Tests by redundant hardware	A.2.1	R low	R low	R medium	R high
	Standard test access port and boundary-scan architecture	A.2.3	R low	R low	R medium	R high
	Code protection	A.6.2	R low	R low	R medium	R high
	Diverse hardware	B.1.4	– low	– low	R medium	R high

At least one of the techniques in the light grey shaded group, or one of the techniques specified in Table A.3 of IEC 61508-3, is required.

NOTE 1 For the meaning of the entries under each safety integrity level, see the text immediately preceding this table.

NOTE 2 The measures can be used to varying effectiveness according to Table A.18, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and for high effectiveness.

NOTE 3 The overview of techniques and measures associated with this table is in Annexes A, B and C of IEC 61508-7. The relevant subclause is referenced in the second column.

NOTE 4 For E/E/PE safety-related systems operating in a low demand mode of operation (for example emergency shutdown systems), the diagnostic coverage achieved from failure detection by on-line monitoring is generally low or none.

Table A.16 – Techniques and measures to control systematic failures caused by environmental stress or influences

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Measures against voltage breakdown, voltage variations, overvoltage, low voltage and other phenomena such as a.c. power supply frequency variation that can lead to dangerous failure	A.8	M low	M medium	M medium	M high
	Separation of electrical energy lines from information lines (see Note 4)	A.11.1	M	M	M	M
	Increase of interference immunity	A.11.3	M low	M low	M medium	M high
	Measures against the physical environment (for example, temperature, humidity, water, vibration, dust, corrosive substances)	A.14	M low	M high	M high	M high
	Program sequence monitoring	A.9	HR low	HR low	HR medium	HR high
	Measures against temperature increase	A.10	HR low	HR low	HR medium	HR high
	Spatial separation of multiple lines	A.11.2	HR low	HR low	HR medium	HR high
	Idle current principle (where continuous control is not needed to achieve or maintain a safe state of the EUC)	A.1.5	R	R	R	R
	Measure to detect breaks and shorts in signal lines		R	R	R	R
	Failure detection by on-line monitoring (see Note 5)	A.1.1	R low	R low	R medium	R high
	Tests by redundant hardware	A.2.1	R low	R low	R medium	R high
	Code protection	A.6.2	R low	R low	R medium	R high
	Antivalent signal transmission	A.11.4	R low	R low	R medium	R high
	Diverse hardware (see Note 6)	B.1.4	– low	– low	– medium	R high
	Software architecture	**7.4.3 of IEC 61508-3**	See Tables A.2 and C.2 of IEC 61508-3			

This table is divided into three groups, as indicated by the sidebar shading. All techniques marked "R" in the grey and black shaded groups are replaceable by other techniques within that group, but at least one of the techniques in the grey shaded group and at least one of the techniques of the black shaded group is required.

NOTE 1 For the meaning of the entries under each safety integrity level, see the text immediately preceding Table A.15.

NOTE 2 Most of these measures in this table can be used to varying effectiveness according to Table A.18, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and for high effectiveness.

NOTE 3 The overview of techniques and measures associated with this table is in Annexes A and B of IEC 61508-7. The relevant subclause is referenced in the second column.

NOTE 4 Separation of electrical energy lines from information lines is not necessary if the information is transported optically, nor is it necessary for low power energy lines that are designed for energising elements of the E/E/PE system and carrying information from or to these elements.

NOTE 5 For E/E/PE safety-related systems operating in a low demand mode of operation (for example emergency shut-down systems), the diagnostic coverage achieved from failure detection by on-line monitoring is generally low or none.

NOTE 6 Diverse hardware is not required if it has been demonstrated, by validation and extensive operational experience, that the hardware is sufficiently free of design faults and sufficiently protected against common cause failures to fulfil the target failure measures.

Table A.17 – Techniques and measures to control systematic operational failures

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Modification protection	B.4.8	M low	M medium	M high	M high
	Failure detection by on-line monitoring (see Note 4)	A.1.1	R low	R low	R medium	R high
	Input acknowledgement	B.4.9	R low	R low	R medium	R high
	Failure assertion programming	C.3.3	See Tables A.2 and C.2 of IEC 61508-3			

At least one of the techniques in the light grey shaded group is required.

NOTE 1 For the meaning of the entries under each safety integrity level, see the text immediately preceding Table A.15.

NOTE 2 Two of these measures in this table can be used to varying effectiveness according to Table A.18, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and for high effectiveness.

NOTE 3 The overview of techniques and measures associated with this table is in Annexes A, B, and C of IEC 61508-7. The relevant subclause is referenced in the second column.

NOTE 4 For E/E/PE safety-related systems operating in a low-demand mode of operation (for example emergency shut-down systems), the diagnostic coverage achieved from failure detection by on-line monitoring is generally low or none.

Table A.18 – Effectiveness of techniques and measures to control systematic failures

Technique/measure	See IEC 61508-7	Low effectiveness	High effectiveness
Failure detection by on-line monitoring (see Note)	A.1.1	Trigger signals from the EUC and its control system are used to check the proper operation of the E/E/PE safety-related systems (only time behaviour with an upper time limit)	E/E/PE safety-related systems are retriggered by temporal and logical signals from the EUC and its control system (time window for temporal watch-dog function)
Tests by redundant hardware (see Note)	A.2.1	Additional hardware tests the trigger signals of the E/E/PE safety-related systems (only time behaviour with an upper time limit), this hardware switches a secondary final element	Additional hardware is retriggered by temporal and logical signals of the E/E/PE safety-related systems (time window for temporal watch-dog); voting between multiple channels
Standard test access port and boundary-scan architecture	A.2.3	Testing the used solid-state logic, during the proof test, through defined boundary scan tests	Diagnostic test of solid-state logic, according to the functional specification of the E/E/PE safety-related systems; all functions are checked for all integrated circuits
Code protection	A.6.2	Failure detection via time redundancy of signal transmission	Failure detection via time and information redundancy of signal transmission
Measures against voltage breakdown, voltage variations, overvoltage and low voltage	A.8	Overvoltage protection with safety shut-off or switch-over to secondary power unit	Voltage control (secondary) with safety shut-off or switch-over to secondary power unit; or power-down with safety shut-off or switch-over to secondary power unit
Program sequence monitoring	A.9	Temporal or logical monitoring of the program sequence	Temporal and logical monitoring of the program sequence at very many checking points in the program
Measures against temperature increase	A.10	Detecting over-temperature	Actuation of the safety shut-off via thermal fuse; or several levels of over-temperature sensing and alarms; or connection of forced-air cooling and status indication
Increase of interference immunity (see Note)	A.11.3	Noise filter at power supply and critical inputs and outputs; shielding, if necessary	Filter against electromagnetic injection that is normally not expected; shielding
Measures against physical environment	A.14	Generally accepted practice according to the application	Techniques referred to in standards for a particular application
Diverse hardware	B.1.4	Two or more items carrying out the same function but being different in design	Two or more items carrying out different functions
Modification protection	B.4.8	Modification requires specific tools	Modification requires use of key lock or dedicated tool with password
Input acknowledgement	B.4.9	Echoing of input actions back to the operator	Checking strict rules for the input of data by the operator, rejecting incorrect inputs
NOTE In the cases of the techniques with references A.1.1, A.2.1, A.11.3, and A.14 for high effectiveness of the technique or measure it is assumed that the low effectiveness approaches are also used.			

Annex B
(normative)

Techniques and measures for E/E/PE safety-related systems – avoidance of systematic failures during the different phases of the lifecycle

Tables B.1 to B.5 in this annex recommend, for each safety integrity level, techniques and measures to avoid failures in E/E/PE safety-related systems. More information about the techniques and measures can be found in Annex B of IEC 61508-7. Requirements for measures to control failures during operation are given in Annex A and described in Annex A of IEC 61508-7.

It is not possible to list every individual cause of systematic failures, originating throughout the safety life cycle, or every remedy, for two main reasons:

– the effect of a systematic fault depends on the lifecycle phase in which it was introduced; and

– the effectiveness of any single measure to avoid systematic failures depends on the application.

A quantitative analysis for the avoidance of systematic failures is therefore impossible.

Failures in E/E/PE safety-related systems may be categorised, according to the lifecycle phase in which a causal fault is introduced, into:

– failures caused by faults originating *before or during system installation* (for example, software faults include specification and program faults, hardware faults include manufacturing faults and incorrect selection of elements); and

– failures caused by faults originating *after system installation* (for example random hardware failures, or failures caused by incorrect use).

In order to avoid or control such failures when they occur, a large number of measures are normally necessary. The structure of the requirements in Annexes A and B results from dividing the measures into those used to *avoid failures* during the different phases of the E/E/PE system safety lifecycle (this annex), and those used to *control failures* during operation (Annex A). The measures to control failures are built-in features of the E/E/PE safety-related systems, while the measures to avoid failures are performed during the safety lifecycle.

In Tables B.1 to B.5, recommendations are made and requirements are given by safety integrity level, stating firstly the importance of the technique or measure and secondly the effectiveness required if it is used. The importance is signified as follows:

– M: the technique or measure is required (mandatory) for this safety integrity level.

– HR: the technique or measure is highly recommended for this safety integrity level. If this technique or measure is not used then the rationale behind not using it shall be detailed;

– R: the technique or measure is recommended for this safety integrity level.

– -: the technique or measure has no recommendation for or against being used;

– NR: the technique or measure is positively not recommended for this safety integrity level. If this technique or measure is used then the rationale behind using it shall be detailed;

The required effectiveness is signified as follows:

– Low: if used, the technique or measure shall be used to the extent necessary to give at least low effectiveness against systematic failures;

– Medium: if used, the technique or measure shall be used to the extent necessary to give at least medium effectiveness against systematic failures;
– High: the technique or measure shall be used to the extent necessary to give high effectiveness against systematic failures.

NOTE Most of the measures in Tables B.1 to B.5 can be used with varying effectiveness according to Table B.6, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and for high effectiveness.

If a measure is not mandatory, it is in principle replaceable by other measures (either individually or in combination); this is governed by the shading, as explained in each table.

Following the guidelines in this annex does not guarantee by itself the required safety integrity. It is important to consider the following:

– the consistency of the chosen techniques and measures, and how well they will complement each other;
– which techniques and measures are appropriate, for every phase of the development lifecycle; and
– which techniques and measures are most appropriate for the specific problems encountered during the development of each different E/E/PE safety-related system.

Table B.1 – Techniques and measures to avoid mistakes during specification of E/E/PE system design requirements (see 7.2)

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Project management	B.1.1	M low	M low	M medium	M high
	Documentation	B.1.2	M low	M low	M medium	M high
	Separation of E/E/PE system safety functions from non-safety functions	B.1.3	HR low	HR low	HR medium	HR high
	Structured specification	B.2.1	HR low	HR low	HR medium	HR high
	Inspection of the specification	B.2.6	– low	HR low	HR medium	HR high
	Semi-formal methods	B.2.3, see also Table B.7 of IEC 61508-3	R low	R low	HR medium	HR high
	Checklists	B.2.5	R low	R low	R medium	R high
	Computer aided specification tools	B.2.4	– low	R low	R medium	R high
	Formal methods	B.2.2	– low	– low	R medium	R high

All techniques marked “R” in the grey shaded group are replaceable, but at least one of these is required.

For the verification of this safety lifecycle phase, at least one of the techniques or measures shaded grey in this table or listed in Table B.5 shall be used.

NOTE 1 For the meaning of the entries under each safety integrity level, see the text preceding this table.

NOTE 2 The measures in this table can be used to varying effectiveness according to Table B.6, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and for high effectiveness.

NOTE 3 The overview of techniques and measures associated with this table is in Annex B of IEC 61508-7. Relevant subclauses are referenced in the second column.

Table B.2 – Techniques and measures to avoid introducing faults during E/E/PE system design and development (see 7.4)

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Observance of guidelines and standards	B.3.1	M high	M high	M high	M high
	Project management	B.1.1	M low	M low	M medium	M high
	Documentation	B.1.2	M low	M low	M medium	M high
	Structured design	B.3.2	HR low	HR low	HR medium	HR high
	Modularisation	B.3.4	HR low	HR low	HR medium	HR high
	Use of well-tried components	B.3.3	R low	R low	R medium	R high
	Semi-formal methods	B.2.3, see also Table B.7 of IEC 61508-3	R low	R low	HR medium	HR high
	Checklists	B.2.5	– low	R low	R medium	R high
	Computer-aided design tools	B.3.5	– low	R low	R medium	R high
	Simulation	B.3.6	– low	R low	R medium	R high
	Inspection of the hardware or walk-through of the hardware	B.3.7 B.3.8	– low	R low	R medium	R high
	Formal methods	B.2.2	– low	– low	R medium	R high

All techniques marked "R" in the grey shaded group are replaceable, but at least one of these is required.

For the verification of this safety lifecycle phase, at least one of the techniques or measures shaded grey in this table or listed in Table B.5 shall be used.

NOTE 1 For the meaning of the entries under each safety integrity level, see the text preceding Table B.1.

NOTE 2 Most of these measures in this table can be used to varying effectiveness according to Table B.6, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and for high effectiveness.

NOTE 3 The overview of techniques and measures associated with this table is in Annex B of IEC 61508-7. Relevant subclauses are referenced in the second column.

Table B.3 – Techniques and measures to avoid faults during E/E/PE system integration (see 7.5)

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Functional testing	B.5.1	M high	M high	M high	M high
	Project management	B.1.1	M low	M low	M medium	M high
	Documentation	B.1.2	M low	M low	M medium	M high
	Black-box testing	B.5.2	R low	R low	R medium	R high
	Field experience	B.5.4	R low	R low	R medium	R high
	Statistical testing	B.5.3	– low	– low	R medium	R high

All techniques marked "R" in the grey shaded group are replaceable, but at least one of these is required.

For the verification of this safety lifecycle phase, at least one of the techniques or measures shaded grey in this table or listed in Table B.5 shall be used.

NOTE 1 For the meaning of the entries under each safety integrity level, see the text preceding Table B.1.

NOTE 2 Most of these measures in this table can be used to varying effectiveness according to Table B.6, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and for high effectiveness.

NOTE 3 The overview of techniques and measures associated with this table is in Annex B of IEC 61508-7. Relevant subclauses are referenced in the second column.

Table B.4 – Techniques and measures to avoid faults and failures during E/E/PE system operation and maintenance procedures (see 7.6)

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Operation and maintenance instructions	B.4.1	HR high	HR high	HR high	HR high
	User friendliness	B.4.2	HR high	HR high	HR high	HR high
	Maintenance friendliness	B.4.3	HR high	HR high	HR high	HR high
	Project management	B.1.1	M low	M low	M medium	M high
	Documentation	B.1.2	M low	M low	M medium	M high
	Limited operation possibilities	B.4.4	– low	R low	HR medium	HR high
	Protection against operator mistakes	B.4.6	– low	R low	HR medium	HR high
	Operation only by skilled operators	B.4.5	– low	R low	R medium	HR high

All techniques marked "R" in the grey shaded group are replaceable, but at least one of these is required.

The verification of this safety lifecycle phase shall be done by checklists (see B.2.5 of IEC 61508-7) or inspection (see B.2.6 of IEC 61508-7).

NOTE 1 For the meaning of the entries under each safety integrity level, see the text preceding Table B.1.

NOTE 2 Most of these measures in this table can be used to varying effectiveness according to Table B.6, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and for high effectiveness.

NOTE 3 The overview of techniques and measures associated with this table is in Annex B of IEC 61508-7. Relevant subclauses are referenced in the second column.

Table B.5 – Techniques and measures to avoid faults during E/E/PE system safety validation (see 7.7)

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Functional testing	B.5.1	HR high	HR high	HR high	HR high
	Functional testing under environmental conditions	B.6.1	HR high	HR high	HR high	HR high
	Interference surge immunity testing	B.6.2	HR high	HR high	HR high	HR high
	Fault insertion testing (when required diagnostic coverage ≥ 90 %)	B.6.10	HR high	HR high	HR high	HR high
	Project management	B.1.1	M low	M low	M medium	M high
	Documentation	B.1.2	M low	M low	M medium	M high
	Static analysis, dynamic analysis and failure analysis	B.6.4 B.6.5 B.6.6	– low	R low	R medium	R high
	Simulation and failure analysis	B.3.6 B.6.6	– low	R low	R medium	R high
	Worst-case analysis, dynamic analysis and failure analysis	B.6.7 B.6.5 B.6.6	– low	– low	R medium	R high
	Static analysis and failure analysis (see Note 4)	B.6.4 B.6.6	R low	R low	NR	NR
	Expanded functional testing	B.6.8	– low	HR low	HR medium	HR high
	Black-box testing	B.5.2	R low	R low	R medium	R high
	Fault insertion testing (when required diagnostic coverage < 90 %)	B.6.10	R low	R low	R medium	R high
	Statistical testing	B.5.3	– low	– low	R medium	R high
	Worst-case testing	B.6.9	– low	– low	R medium	R high
	Field experience	B.5.4	R low	R low	R medium	NR

This table is divided into three groups, as indicated by the sidebar shading. All techniques marked "R" in the grey and black shaded groups are replaceable by other techniques within that group, but at least one of the techniques of the grey shaded group (analytical techniques) and at least one of the techniques of the black shaded group (testing techniques) is required.

NOTE 1 For the meaning of the entries under each safety integrity level, see the text preceding Table B.1.

NOTE 2 Most of these measures in this table can be used to varying effectiveness according to Table B.6, which gives examples for low and high effectiveness. The effort required for medium effectiveness lies somewhere between that specified for low and for high effectiveness.

NOTE 3 The overview of techniques and measures associated with this table is in Annex B of IEC 61508-7. Relevant subclauses are referenced in the second column.

NOTE 4 Static analysis and failure analysis is not recommended for SIL 3 and SIL 4, because these techniques are not sufficient unless used in combination with dynamic analysis.

Table B.6 – Effectiveness of techniques and measures to avoid systematic failures

Technique/measure	See IEC 61508-7	Low effectiveness	High effectiveness
Project management (see Note)	B.1.1	Definition of actions and responsibilities; scheduling and resource allocation; training of relevant personnel; consistency checks after modifications	Validation independent from design; project monitoring; standardised validation procedure; configuration management; failure statistics; computer aided engineering; computer-aided software engineering
Documentation (see Note)	B.1.2	Graphical and natural language descriptions, for example block-diagrams, flow-diagrams	Guidelines for consistent content and layout across organization; contents checklists; computer-aided documentation management, formal change control
Separation of E/E/PE system safety functions from non–safety functions	B.1.3	Well-defined interfaces between E/E/PE safety-related systems and non-safety-related systems	Total separation of E/E/PE safety-related systems from non-safety-related systems, i.e. no write access of non-safety-related systems to E/E/PE safety-related systems and separate physical locations to avoid common cause influences
Structured specification	B.2.1	Manual hierarchical separation into sub-requirements; description of the interfaces	Hierarchical separation described using computer-aided engineering tools; automatic consistency checks; refinement down to functional level
Formal methods	B.2.2	Used by personnel experienced in formal methods	Used by personnel experienced in formal methods in similar applications, with computer support tools
Semi-formal methods	B.2.3	Describing some critical parts with semi-formal methods	Describing total E/E/PE safety-related systems with different semi-formal methods to show different aspects; consistency check between the methods
Computer-aided specification tools	B.2.4	Tools without preference for one particular design method	Model-oriented procedures with hierarchical subdivision; description of all objects and their relationships; common data base; automatic consistency checks
Checklists	B.2.5	Prepared checklists for all safety life-cycle phases; concentration on the main safety issues	Prepared detailed checklists for all safety life-cycle phases
Inspection of the specification	B.2.6	Inspection of the safety requirements specification by an independent person	Inspection and re-inspection by an independent organisation using a formal procedure with correction of all faults found
Structured design	B.3.2	Hierarchical circuit design, produced manually	Reuse of tested circuit parts; traceability between specification, design, circuit diagram and parts lists; computer-aided; based on defined methods (see also 7.4.6)
Use of well-tried components (see Note)	B.3.3	Sufficient over-dimensioning; constructive characteristics	Proven in use (see 7.4.10)
Modularization (see Note)	B.3.4	Modules of limited size; each module functionally isolated	Re-use of well-proven modules; easily comprehensible modules; each module has a maximum of one input, one output, and one failure exit

Table B.6 *(continued)*

Technique/measure	See IEC 61508-7	Low effectiveness	High effectiveness
Computer-aided design tools	B.3.5	Computer support for complex phases of the safety lifecycle	Use of tools that are proven in use (see 7.4.10) or validated; general computer-aided development for all phases of the safety lifecycle
Simulation	B.3.6	Modelling at a module level, including boundary data of peripheral units	Modelling on a component level, including boundary data
Inspection of the hardware	B.3.7	Inspection by a person independent of the design	Inspection and re-inspection by an independent organisation using a formal procedure with correction of all faults found
Walk-through of the hardware	B.3.8	Walk-through includes a person independent of the design	Walk-through includes an independent organisation and follows a formal procedure with correction of all faults found
Limited operation possibilities (see Note)	B.4.4	Key-operated switch or password to govern change of operating mode	Defined, robust procedure for allowing operation
Operation only by skilled operators	B.4.5	Basic training in the type of safety systems being operated, plus two years' relevant on-the-job experience	Yearly training of all operators; each operator has at least five years' experience with safety-related devices at lower safety integrity levels
Protection against operator mistakes (see Note)	B.4.6	Input acknowledgement	Confirmation and consistency checks on each input command
Black-box testing (see Note)	B.5.2	Equivalence classes and input partition testing, boundary value testing, using pre-written test cases	Test case execution from cause consequence diagrams, combining critical cases at extreme operating boundaries
Statistical testing (see Note)	B.5.3	Statistical distribution of all input data	Test reports by tools; very many test cases; distribution of the input data according to real-life application conditions and assumed failure models
Field experience (see Note)	B.5.4	10 000 h operation time; at least one year's experience with at least 10 devices in different applications; statistical accuracy 95 %; no safety critical failures	10 million h operation time; at least two years' experience with at least 10 devices in different applications; statistical accuracy 99,9 %; detailed documentation of all changes (including minor) during past operation
Surge immunity testing	B.6.2		Surge immunity shall be demonstrably higher than the boundary values for real operating conditions
Static analysis	B.6.4	Based on block diagrams; highlighting weak points; specifying test cases	Based on detailed diagrams; predicting expected behaviour during test cases; using testing tools

Table B.6 *(continued)*

Technique/measure	See IEC 61508-7	Low effectiveness	High effectiveness
Dynamic analysis	B.6.5	Based on block diagrams; highlighting weak points; specifying test cases	Based on detailed diagrams; predicting expected behaviour during test cases; using testing tools
Failure analysis	B.6.6	At module level, including boundary data of the peripheral units	At component level, including boundary data
Worst-case analysis	B.6.7	Performed on safety functions; derived using boundary value combinations for real operating conditions	Performed on non-safety functions; derived using boundary value combinations for real operating conditions
Expanded functional testing	B.6.8	Test that all safety functions are maintained in the case of static input states caused by faulty process or operating conditions	Test that all safety functions are maintained in the case of static input states and/or unusual input changes, caused by faulty process or operating conditions (including those that may be very rare)
Worst-case testing	B.6.9	Test that safety functions are maintained for a combination of boundary values found in real operating conditions	Test that non-safety functions are maintained for a combination of the boundary values found in real operating conditions
Fault insertion testing	B.6.10	At subunit level including boundary data or the peripheral units	At component level including boundary data
NOTE In the cases of the techniques with references B.1.1, B.1.2, B.3.3, B.3.4, B.4.4, B.4.6, B.5.2, B.5.3, B.5.4, B.6.7 and B.6.9, for high effectiveness of the technique or measure, it is assumed that the low effectiveness approaches are also used.			

Annex C
(normative)

Diagnostic coverage and safe failure fraction

C.1 Calculation of diagnostic coverage and safe failure fraction of a hardware element

The diagnostic coverage and safe failure fraction of an element (see 3.8.6 and 3.6.15 of IEC 61508-4) shall be calculated as follows:

a) Carry out a failure mode and effect analysis to determine the effect of each failure mode of each component or group of components in the element on the behaviour of the E/E/PE safety-related systems in the absence of diagnostic tests. Sufficient information shall be available (see Notes 1 and 2) to enable the failure mode and effects analysis to be undertaken so as to enable an adequate level of confidence to be established commensurate with the safety integrity requirements.

NOTE 1 In order to undertake this analysis the following information is required:

– a detailed block diagram of the E/E/PE safety-related system describing the element together with the interconnections for that part of the E/E/PE safety-related system which will affect the safety function(s) under consideration;

– the hardware schematics of the element describing each component or group of components and the interconnections between components;

– the failure modes and rates of each component or group of components and associated percentages of the total failure probability corresponding to safe and dangerous failures.

NOTE 2 The required rigour of this analysis will depend on a number of factors (see IEC 61508-1, 4.1). In particular, the safety integrity level of the safety functions involved will need to be taken into account. For higher safety integrity levels it is expected that the failure modes and effects analysis is very specific according to particular component types and application environments. Also, a thorough and detailed analysis is very important for an element that is to be used in a hardware architecture having zero hardware fault tolerance.

b) Categorize each failure mode according to whether it leads (in the absence of diagnostic tests) to:

   – a safe failure; or

   – a dangerous failure;

c) No-effect and no-part failures shall not play any part in the calculation of the diagnostic coverage or the safe failure fraction.

d) From an estimate of the failure rate of each component or group of components, (λ), (see Note 4) and the results of the failure mode and effect analysis, for each component or group of components, calculate the safe failure rate (λ_S), and the dangerous failure rate (λ_D). When one of these failure rates is not constant, its average over the period shall be estimated and used in DC and SFF calculations.

NOTE 3 The failure rate of each component or group of components can be estimated using data from a recognised industry source, taking the application environment into account. However, application specific data is preferred, particularly in cases where the element consists of a small number of components and where any error in estimating the probability of safe and dangerous failures of a particular component could have a significant impact on the estimation of the safe failure fraction.

e) For each component or group of components, estimate the fraction of dangerous failures that will be detected by the diagnostic tests (see C.2) and therefore the dangerous failure rate that is detected by the diagnostic tests, (λ_{Dd}).

f) For the element, calculate the total dangerous failure rate, ($\Sigma\lambda_D$), the total dangerous failure rate that is detected by the diagnostic tests, ($\Sigma\lambda_{Dd}$), and the total safe failure rate, ($\Sigma\lambda_S$).

g Calculate the diagnostic coverage of the element as ($\Sigma\lambda_{Dd}/\Sigma\lambda_D$).

h) Calculate safe failure fraction of the element as:

$$SFF = (\Sigma\lambda_S + \Sigma\lambda_{Dd})/(\Sigma\lambda_S + \Sigma\lambda_{Dd} + \Sigma\lambda_{Du})$$

NOTE 4 The above equation is applicable when the failure rates are based on constant failure rates (see 3.6.15 of IEC 61508-4 for the definitive formula).

NOTE 5 The diagnostic coverage (if any) of each element in the E/E/PE safety-related system is taken into account in the estimation of the achieved failure measure for each safety function (see 7.4.5.2). The safe failure fraction is taken into account when determining the architectural constraints on hardware safety integrity (see 7.4.4).

The analysis used to determine the diagnostic coverage and safe failure fraction shall include all of the components, including electrical, electronic, electromechanical, mechanical etc, that are necessary to allow the element to process the safety function(s) as required by the E/E/PE safety-related system. All of the possible dangerous modes of failure that will lead to an unsafe state, prevent a safe response when such a response is demanded or otherwise compromise the safety integrity of the E/E/PE safety-related systems, shall be considered for each of the components.

Table A.1 sets out the faults or failures to be detected during operation or to be analysed in the derivation of the safe failure fraction.

If field data is used to support the failure modes and effects analysis it shall be sufficient to support the safety integrity requirements. As a minimum, a statistical single-sided lower confidence limit of at least 70 % is required.

NOTE 6 An example of calculation of diagnostic coverage and safe failure fraction is included in Annex C of IEC 61508-6.

NOTE 7 Alternative methods are available for calculating diagnostic coverage involving, for example, simulation of faults using a computer model containing details of both the circuitry of the E/E/PE safety-related systems and the electronic components used in its design (for example, down to the transistor level in an integrated circuit).

C.2 Determination of diagnostic coverage factors

In the calculation of diagnostic coverage for an element (see C.1) it is necessary to estimate, for each component or group of components, the fraction of dangerous failures that are detected by the diagnostic tests. The diagnostic tests that can contribute to the diagnostic coverage include, but are not limited to:

– comparison checks, for example monitoring and comparison of redundant signals;
– additional built-in test routines, for example checksums on memory;
– test by external stimuli, for example sending a pulsed signal through control paths;
– continuous monitoring of an analogue signal, for example, to detect out of range values indicative of sensor failure.

In order to calculate diagnostic coverage it is necessary to determine those failure modes that are detected by the diagnostic tests. It is possible that open-circuit or short-circuit failures for simple components (resistors, capacitors, transistors) can be detected with a coverage of 100 %. However, for more complex type B elements, see 7.4.4.1.3, account should be taken of the limitations to diagnostic coverage for the various components shown in Table A.1. This analysis shall be carried out for each component, or group of components, of each element and for each element of the E/E/PE safety-related system.

NOTE 1 Tables A.2 to A.14 recommend techniques and measures for diagnostic tests and recommend maximum diagnostic coverage that can be claimed. These tests may operate continuously or periodically (depending on the diagnostic test interval). The tables do not replace any of the requirements of this annex.

NOTE 2 Diagnostic tests can provide significant benefits in the achievement of functional safety of an E/E/PE safety-related system. However, care must be exercised not to unnecessarily increase the complexity which, for example, may lead to increased difficulties in verification, validation, functional safety assessment, and

maintenance and modification activities. Increased complexity may also make it more difficult to maintain the long-term functional safety of the E/E/PE safety-related system.

The calculations to obtain the diagnostic coverage, and the ways it is used, assume that the EUC can operate safely in the presence of an otherwise dangerous fault that is detected by the diagnostic tests. If this assumption is not correct then the E/E/PE safety-related system shall be treated as operating in a high demand or a continuous mode of operation (see 7.4.8.3, 7.4.5.3 and 7.4.5.4).

NOTE 3 The definition of diagnostic coverage is given in 3.8.6 of IEC 61508-4. It is important to note that alternative definitions of the diagnostic coverage are sometimes assumed but these are not applicable within this standard.

NOTE 4 The diagnostic tests used to detect a dangerous failure within an element may be implemented by another element within the E/E/PE safety-related system.

NOTE 5 Diagnostic tests may operate either continuously or periodically, depending on the diagnostic test interval. There may be some cases or times where a diagnostic test should not be run due to the possibility of a test affecting the system state in an adverse manner. In this case, no benefits in the calculations may be claimed from the diagnostic tests.

Annex D
(normative)

Safety manual for compliant items

D.1 General

The purpose of the safety manual for compliant items is to document all the information, relating to a compliant item, which is required to enable the integration of the compliant item into a safety-related system, or a subsystem or element, in compliance with the requirements of this standard.

D.2 Contents

D.2.1 The safety manual shall specify the functions of the compliant item. These may be used to support a safety function of a safety-related system or functions in a subsystem or element. The specification should clearly describe both the functions and the input and output interfaces.

For every compliant item, the safety manual shall contain:

a) a functional specification of the functions capable of being performed;

b) identification of the hardware and/or software configuration of the compliant item to enable configuration management of the E/E/PE safety-related system in accordance with 6.2.1 of IEC 61508-1.

c) constraints on the use of the compliant item and/or assumptions on which analysis of the behaviour or failure rates of the item are based.

D.2.2 For every function, the safety manual shall contain:

a) the failure modes of the compliant item (in terms of the behaviour of its outputs), due to random hardware failures, that result in a failure of the function and that are not detected by diagnostics internal to the compliant item;

b) for every failure mode in a), an estimated failure rate;

c) the failure modes of the compliant item (in terms of the behaviour of its outputs), due to random hardware failures, that result in a failure of the function and that are detected by diagnostics internal to the compliant item;

d) the failure modes of the diagnostics, internal to the compliant item (in terms of the behaviour of its outputs), due to random hardware failures, that result in a failure of the diagnostics to detect failures of the function;

e) for every failure mode in c) and d), the estimated failure rate;

f) for every failure mode in c) that is detected by diagnostics internal to the compliant item, the diagnostic test interval;

g) for every failure mode in c) the outputs of the compliant item initiated by the internal diagnostics;

NOTE 1 The outputs of the internal diagnostics could be used to initiate additional measures (technical/procedural) to the E/E/PE safety-related system, subsystem or element to achieve or maintain a safe state of the EUC.

h) any periodic proof test and/or maintenance requirements;

i) for those failure modes, in respect of a specified function, that are capable of being detected by external diagnostics, sufficient information shall be provided to facilitate the development of an external diagnostics capability. The information shall include details of failure modes and for those failure modes the failure rates;

j) the hardware fault tolerance;

k) the classification as type A or type B of that part of the compliant item that provides the function (see 7.4.4.1.2 and 7.4.4.1.3);

NOTE 2 Failure modes can only be classified as being safe or dangerous when the application of the compliant item is known in relation to the hazards of the EUC. For example, if a sensor is applied in such a way that a high output is used to signal a hazard of the EUC (for example high pressure), then a failure mode that prevents the correct indication of the hazard (for example output stuck low) would be classified as dangerous whereas a failure mode that causes the sensor output to go high would be classified as safe. This depends on how the sensor signal is interpreted by the safety-related system logic and so cannot be specified without constraining the way that the sensor is applied.

Also, the level of diagnostic coverage claimed for a compliant item may vary from one application to another depending on the extent of any diagnostics in the system logic or external signal processing that may supplement any internal diagnostics of the compliant item.

It follows that any estimate of the hardware fault tolerance or the safe failure fraction can only be made if constraints are placed on the application of the compliant item. These constraints are outside the control of the supplier of the compliant item. Therefore, no claims shall be made in the safety manual, in respect of the hardware fault tolerance or the safe failure fraction or any other functional safety characteristic that is dependent on knowledge of safe and dangerous failure modes, unless the underlying assumptions, as to what constitute safe and dangerous failure modes, are clearly specified.

D.2.3 For every function of the compliant item that is liable to systematic failure, the manual shall contain:

a) the systematic capability of the compliant item or that part of the element that provides the function;

b) any instructions or constraints relating to the application of the compliant item, relevant to the function, that should be observed in order to prevent systematic failures of the compliant item.

NOTE The systematic safety integrity indicated by the systematic capability can be achieved only when the instructions and constraints are observed. Where violations occur, the claim for systematic capability is partially or wholly invalid.

D.2.4 For additional requirements relating to software compliant items see 7.4.2.12 and Annex D of IEC 61508-3.

Annex E
(normative)

Special architecture requirements for integrated circuits (ICs) with on-chip redundancy

E.1 General

This annex is referenced by 7.4.2.2 b).

To allow the use of on-chip redundancy for ICs with one common semi-conductor substrate, a set of requirements is given below. For safety reasons this approach has a conservative nature, for example it is limited up to SIL 3 and a set of restrictive requirements have been specified. The following requirements are related to digital ICs only. For mixed-mode and analogue ICs no general requirements can be given at the moment. Common cause analysis (see IEC 61508-1, 7.6.2.7) may exclude the use of on-chip redundancy for an individual application. On-chip redundancy as used in this standard means a duplication (or triplication etc.) of functional units to establish a hardware fault tolerance greater than zero. According to 7.4.4.1.1 a) in determining the hardware fault tolerance no account is taken of measures that may control the effects of faults such as diagnostics.

A subsystem with a hardware fault tolerance greater than 0 can be realised using one single IC semi-conductor substrate (on-chip redundancy). In this case all of the following requirements a) to q) shall be fulfilled and the design of the E/E/PE system and the IC shall be such as to meet these requirements. An IC with on-chip redundancy shall have its own compliant item safety manual (see Annex D).

a) The highest safety integrity level that can be claimed for a safety function using an IC as described above is limited to SIL 3.

NOTE 1 At the present state of the art, knowledge and experience, it is not feasible to consider and take measures against all effects related to said element (single IC) to gain sufficient confidence for SIL 4.

b) The systematic capability shall not be increased by combination of elements (see 7.4.3.2).

c) To avoid common cause failure(s), the effects of increasing temperature, for example due to random hardware fault(s), shall be considered. At least one of the measures listed in Table E.2, no. 6 shall be applied. In a design where a local fault can cause a safety critical temperature increase, appropriate measures shall be taken.

NOTE 2 While in a power design a local fault can cause a significant temperature increase, the impact of a local short circuit in a logic circuit can be negligible. Examples to be considered in digital circuits include the device pad area and voltage regulators.

d) Separate physical blocks on substratum of the IC shall be established for each channel and each monitoring element such as a watchdog. The blocks shall include bond wires and pin-out. Each channel shall have its own separated inputs and outputs which shall not be routed through another channel/block.

NOTE 3 This does not exclude internal connections between blocks by wiring between output and input cells of different blocks (see also Table E.1, 3a and 3b).

NOTE 4 Input and outputs include, but are not limited to:

– DFT signals (Design for Testability, e.g. scan chains);
– Clock signals and clock enable signals;
– Power supply;
– Reset signals;
– Configuration and mode selection signals;
– Debug and trace signals.

e) Appropriate measures shall be taken to avoid dangerous failure caused by faults of the power supply including common cause failures.

NOTE 5 Faults of the power supply include, but are not limited to:

– noise;

– disturbance propagation over the power supply lines;

– non-simultaneous power supply switch-on, that may cause effects such as latch-up or high in-rush current;

– excessive current-draw resulting from short circuit.

NOTE 6 This requirement can be fulfilled by applying adequate techniques such as:

– providing each block with its own power supply pins so that no block is supplied via the power supply of another block (for example via internal connections) and not connecting wells of separate physical blocks together inside the IC (see also Table E.2, no. 3);

– incorporation of external measures to avoid dangerous failures that may be caused by different voltages of the wells;

– detecting power supply faults by means of voltage monitors;

– using partially increased voltage tolerance;

– considering IR drop problems for the design of power lines.

f) The minimum distance between boundaries of separate physical blocks shall be sufficient to avoid short circuit and cross talk between these blocks.

NOTE 7 Short circuit typically can be caused by electro migration, via migration, contact migration, local defect gate oxide breakdown, latch-up, etc.

NOTE 8 Cross talk typically can be caused by substrate currents, capacitive coupling, etc.

NOTE 9 The minimum distance should be chosen regarding the relevant design rules with a safety factor typically between 10 and 50.

NOTE 10 Potential rings according to Table E.2 are not considered as being part of a block when estimating the distance between separate physical blocks.

g) Short circuit and/or cross-talk between adjacent lines of separate physical blocks shall not lead to a loss of a safety function or an undetected loss of a monitoring function (Table E.2, no. 5).

h) substratum shall be connected to ground whatever the IC design process used (n-well or p-well);

NOTE 11 For p-wells, this means the use of a negative power supply. Negative logic should be avoided since its use may be susceptible to errors in design.

i) The susceptibility of an IC with on-chip redundancy to common cause failures shall be estimated by determining a β-factor according to E.3. This β-factor called β_{IC} shall be used when estimating the achieved safety integrity of the E/E/PE safety-related system according to 7.4.5.1 and will be used for the IC instead of the β-factor determined for example according to Annex D of IEC 61508-6.

j) The detection of a fault (by diagnostic tests, proof tests or by any other means) in an IC with on-chip redundancy shall result in a specified action to achieve or maintain a safe state.

NOTE 12 This requirement does not apply, if the effects of a fault can be controlled, for example by de-energization of a block.

k) The minimum diagnostic coverage of each channel shall be at least 60 %. Where a monitoring element is implemented only once, the minimum diagnostic coverage for this element shall also be at least 60 %.

l) If it is necessary to implement a watchdog, for example for program sequence monitoring and/or to guarantee the required diagnostic coverage or safe failure fraction one channel shall not be used as a watchdog of another channel, except when functionally diverse channels are used.

m) When testing for electromagnetic compatibility without additional safety margin, the function carried out by the IC shall not be interfered (for example performance criterion A

as described in EMC immunity standards, see for example IEC 61000-6-2 or IEC 61326-3-1).

n) When testing for electromagnetic compatibility with additional safety margins, the safety function (including IC) shall comply with the “FS” criterion as defined in IEC 61326-3-1

o) Appropriate measures shall be taken to avoid dangerous failure caused by oscillations of digital input ports connected to external asynchronous digital signals, e.g. introduction of respective multiple clock synchronization stages.

p) The common cause potential of common resources such as boundary scan circuitries and arrays of special function registers shall be analyzed.

q) The requirements a) to p) list common cause initiators specific to ICs with on-chip redundancy. Other relevant common cause initiators shall be considered as specified in this International Standard.

NOTE 13 In general the above requirements restrict the use of on-chip redundancy to ICs designed with a full-custom or semi-custom approach such as ASICs, microcontrollers or other specialised SoCs (systems on chip). Other designs such as Gate Arrays, FPGAs etc. may not meet all requirements.

Use of ICs with on-chip redundancy as described above shall only be permitted if a full common cause analysis (CCA) has been undertaken. This analysis shall cover the complete range of potential common cause failures arising from design, fabrication, construction, procedural and environmental factors. In particular, the loss of physical separation between channels as a result of the use of ICs with on-chip redundancy shall be subject to special scrutiny. The final SIL level assigned to the E/E/PE safety-related system shall be dependent upon the results of this CCA.

NOTE 14 The use of physical separation (i.e. segregation) of “channels” can provide defence against a wide range of common mode failures in redundant systems.

NOTE 15 The CCA methodology proposed is structured into the following steps:

1. Identify potential common cause initiators (CCI). Consider effects listed in this annex and other foreseeable physical CCI and logical CCI (shared resources and signals).
2. Identify the redundant blocks on the IC which will suffer from CCI amongst them.
3. Qualitatively list and evaluate the safety measures against the individual CCI identified in step 1 for each pair of redundant blocks identified in step 2.
4. Quantitatively answer the Tables E.1 and E.2 for each pair of redundant blocks identified in step 2 and evaluate the specific ß factor.
5. Use the specific ß factors in the probabilistic modelling.

E.2 Additional requirements for SIL 3 on-chip redundancy

For SIL 3 on-chip redundancy the following requirements shall be met in addition to the requirements given in E.1:

a) documented evidence that all application specific environmental conditions are in accordance with that taken into account during specification, analysis, verification and validation shall be provided;

b) external measures that can achieve or maintain a safe state of the E/E/PE system. These measures shall achieve medium effectiveness (see also A.3) as minimum. All measures implemented inside the IC to monitor for effects of systematic and/or common cause failures shall use these external measures to achieve or maintain a safe state of the E/E/PE system.

E.3 β-factor

The susceptibility of the IC with on-chip redundancy to common cause failures shall be estimated by determining the β-factor β_{IC}, which is special to ICs with on-chip redundancy (see also E.1, i)). The estimation shall be based upon the following:

a) a basic β-factor called $\beta_{B\text{-}IC}$ of 33 %;

b) estimation of the increase of the basic β-factor, $\beta_{B\text{-}IC}$, by the design using Table E.1; and

c) estimation of the decrease of the basic β-factor, $\beta_{B\text{-}IC}$, by the design using Table E.2.

β_{IC} is estimated by adding $\beta_{B\text{-}IC}$ and all scores from Table E.1 and afterwards subtracting all scores from Table E.2. The estimated final β_{IC} shall not exceed 25 %.

NOTE 1 This β-factor called β_{IC} will be used when estimating the achieved safety integrity of the E/E/PE safety-related system according to 7.4.5.1 and will be used for the IC instead of the β-factor determined for example according to Annex D of IEC 61508-6.

NOTE 2 A specific analysis of the available failure data for the IC design methodology applied should be undertaken to substantiate that the chosen β-factor is conservative. Only ICs with mature design and implementation processes should be used.

Table E.1 – Techniques and measures that increase $\beta_{B\text{-}IC}$

	Technique/measure	Delta β-factor [%]	Remark
1	Watchdog on-chip used as monitoring element	5	Monitoring elements used for watchdog function and necessary to guarantee the required DC or SFF should be realised external to the IC preferably under the aspect of common cause failures. The use of a watchdog(s) on-chip may result in a higher DC or SFF compared to external realization. See also E.2 b).
2	Monitoring elements on-chip other than watchdog, for example clock monitoring	5	Monitoring elements used for example for clock monitoring and necessary to guarantee the required DC or SFF should be realised external to the IC preferably under the aspect of common cause failures. The use of a monitoring element(s) on-chip may result in a higher DC or SFF compared to external realization.
3a	Internal connections between blocks by wiring between output and input cells of separate physical blocks without cross-over in different layers	2	Comparison of conditions and results between separate physical blocks should be realised external to the IC preferably. Analysis of possible common cause failures including FMEA of stuck-at-faults of internal connections is required. Effects of temperature increase due to faults shall be taken into account in particular. Verification of the layout should be carried out by analysis of the final layout, for example with the help of tools.
3b	Internal connections between blocks by wiring between output and input cells of separate physical blocks with cross-over	4	Comparison of conditions and results between separate physical blocks should be realised external to the IC preferably. Analysis of possible common cause failures including FMEA of stuck-at-faults and short circuit of internal connections is required. Effects of temperature increase due to faults shall be taken into account in particular.
Alternate techniques/measures are indicated by a letter following the number. Only one of the alternate techniques/measures can be selected. Techniques and measures listed in this table are not exhaustive. Other techniques and measures may be used, provided evidence is given to support the claimed delta β-factor. If evidence can be provided that measures were taken to mitigate the impact of common cause failures, other delta β-factors may be used. General advice from Annex D of IEC 61508-6 should be observed in such cases. NOTE The interface signals between the redundant blocks are generally composed of multiple layers. Irrespective of the composition of a signal, whether it is solely constructed with only one metal layer or it is a mix of multiple layers, *the whole interface signal will be considered as a single wire*. To minimise possible interference of both channels by one fault none of the interface signals should cross over with the rest of the interface signals.			

Table E.2 – Techniques and measures that decrease β_{B-IC}

	Technique/measure	Delta β-factor [%]	Remark
1a	Diverse measures to control failures in different channels	4	
1b	Diversity in function and measures to control failures in different channels	6	
2	Testing the E/E/PE system for electromagnetic compatibility with additional safety margin not interfering the function of the E/E/PE system (for example performance criterion A)	5	Performance criterion A is described in EMC immunity standards, see for example IEC 61000-6-2 or IEC 61326-3-1.
3	Providing each block with its own power supply pins so that no block is supplied via the power supply of another block (for example via internal connections) and not connecting wells of separate physical blocks inside the IC	6	External measures have to be taken to avoid dangerous failures that might be caused by different voltages of the wells.
4	Structures that isolate and decouple physical locations	2 - 4	Useful to decouple separate physical blocks.
5	Ground pin between pin-out of separate physical blocks	2	If not implemented, short circuit between adjacent lines of separate physical blocks shall be carried out to test for effects of tear-off of bond wiring (see also E.1, g)). The β-factor will not be decreased in this case.
6a	High diagnostic coverage (DC ≥ 99 %) of each channel, failure detection by the technical process and achievement of safe state in adequate short time	7	May be appropriate only in exceptional case.
6b	Temperature sensors between blocks with permanent shut-down (internal or external) to safe state in adequate short time; low effectiveness without diagnostics	2	See also Table A.18, measures against temperature increase.
6c	Temperature sensors between blocks with permanent shut-down (internal or external) to safe state in adequate short time; high effectiveness with diagnostics	9	See also Table A.18, measures against temperature increase.
6d	Analysis/test of the effects of faults (for example increase of temperature). Depending on the result of the analysis/test, comparison between channels, including fault detection and achievement of safe state in adequate short time can be required	9	
6e	Design of the monitoring circuit functional at the increased temperature	7	The design of the monitoring function (e.g. watch dog) shall carry out the safety function under worst case temperature conditions.
Alternate techniques/measures are indicated by a letter following the number. Only one of the alternate techniques/measures can be selected. Techniques and measures listed in this table are not exhaustive. Other techniques and measures may be used, provided evidence is given to support the claimed delta β-factor. NOTE Techniques/measures 6a to 6e aim for controlling effects of temperature rise due to failure.			

Annex F
(informative)

Techniques and measures for ASICs – avoidance of systematic failures

F.1 General

For the design of Application Specific Integrated Circuits (ASICs) the following techniques and measures for the avoidance of failures during the ASIC-development should be applied.

NOTE 1 This informative annex is referenced by 7.4.6.7.

NOTE 2 The following techniques and measures are related to digital ASICs and user programmable ICs only. For mixed-mode and analogue ASICs no general techniques and measures can be given at the moment.

a) All design activities and test arrangements, and tools used for the functional simulation and the results of the simulation, should be documented.

b) All tools, libraries and manufacturing procedures should be proven in use. This includes:

- application of the individual tool (including different versions with equivalent features) over a substantial period of time in projects of similar or greater complexity;

NOTE 3 A substantial period of time might be 2 years in this case.

- application of common or widely used tools to ensure that information about possible bugs and restrictions is known for the given tool and/or the given version, which should be considered during use. Version control and monitoring should be carried out by the manufacturers to track existing faults;
- internal consistency and plausibility checks to avoid faults in the different databases created by different tools.

NOTE 4 User training is very important because of the rapid changes and progress in this field.

c) All activities and their results should be verified, for example by simulation, equivalence checks, timing analysis or checking the technology constraints.

d) Measures for the reproducibility and automation of the design implementation process (script based, automated work and design implementation flow) should be used.

e) For 3rd party soft-cores and hard-cores, only validated macro blocks should be used and these should comply with all constraints and proceedings defined by the macro core provider if practicable. Unless already proven in use, each macro block should be treated as newly written code, for example it should be fully validated.

f) For the design, a problem-oriented and abstract high-level design methodology and design description language should be used.

NOTE 5 The design description should use a hardware description language like VHDL or Verilog. This is the most common hardware description methodology used today in ASIC design. Both languages are defined by IEEE standards and are assumed to satisfy the recommendations for high level programming languages. The hardware description language may be used both for design description and for functional models or test benches. When used for design description, only a subset of the language may be used; this synthesisable code is often referred to as RTL (register transfer level) code. Non synthesisable code, adequate for functional models and test benches is called behavioural code.

g) Adequate testability (for manufacturing test of the full and semi-custom ASIC) should be achieved.

h) Gate and interconnection (wire) delays should be considered during test and ASIC verification steps.

i) Internal gates with tristate outputs should be avoided. If internal tristate outputs are used these outputs should be equipped with pull-ups/downs or bus-holders.

j) Before manufacturing, an adequate verification of the complete ASIC (i.e., including each verification step carried out during design and implementation to ensure correct module and chip functionality) should be carried out.

NOTE 6 The adequacy of ASIC verification depends on the test complexity of the element and the required safety integrity level.

F.2 Guidelines: Techniques and measures

An appropriate group of techniques and measures that are essential to prevent the introduction of faults during the design and development of ASICs should be used. Depending upon the technical realisation, a differentiation between full and semi-custom digital ASICs and user programmable ICs (FPGA/PLD/CPLD) is necessary. Techniques and measures that support the achievement of relevant properties are defined in Table F.1 for full and semi custom ASICs and in Table F.2 for user programmable ICs. The related ASIC development lifecycle is shown in Figure 3.

In Tables F.1 and F2 recommendations are made by safety integrity level, stating firstly the importance of the technique or measure and secondly the effectiveness recommended if it is used. The importance is signified as follows:

– HR*: the technique or measure is highly recommended for this safety integrity level. No design should exclude this technique or measure;
– HR: the technique or measure is highly recommended for this safety integrity level. If this technique or measure is not used, then the rationale behind not using it should be detailed;
– R: the technique or measure is recommended for this safety integrity level. If this technique or measure is not used or none of possible alternatives is used, then the rationale behind not using it should be detailed;
– -: the technique or measure has no recommendation for or against being used;
– NR: the technique or measure is positively not recommended for this safety integrity level. If this technique or measure is used, then the rationale behind using it should be detailed;

The recommended effectiveness is signified as follows.

– Low: if used, the technique or measure should be used to the extent necessary to give at least low effectiveness against systematic failures;
– Medium: if used, the technique or measure should be used to the extent necessary to give at least medium effectiveness against systematic failures;
– High: the technique or measure should be used to the extent necessary to give high effectiveness against systematic failures.

Following the guidelines in this annex does not guarantee by itself the required safety integrity. It is important to consider:

– the consistency of the chosen techniques and measures, and how well they will complement each other;
– which techniques and measures are appropriate, for every phase of the development lifecycle; and
– which techniques and measures are most appropriate for the specific problems encountered during the development of each different E/E/PE safety-related system.

Table F.1 – Techniques and measures to avoid introducing faults during ASIC's design and development – full and semi-custom digital ASICs (see 7.4.6.7)

Design phase	Ref	Technique/Measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Design entry	1	Structured description	E.3	HR high	HR high	HR* high	HR* high
	2	Design description in (V)HDL (see Note)	E.1	HR high	HR high	HR* high	HR* high
	3	Schematic entry	E.2	NR	NR	NR	NR
	4	(V)HDL simulation (see Note)	E.5	HR high	HR high	HR* high	HR* high
	5	Application of proven in use (V)HDL simulators (see Note)	E.4	HR high	HR high	HR* high	HR* high
	6	Functional test on module level (using for example (V)HDL test benches) (see Note)	E.6	HR high	HR high	HR* high	HR* high
	7	Functional test on top level	E.7	HR high	HR high	HR* high	HR* high
	8	Functional test embedded in system environment	E.8	R medium	R medium	HR high	HR high
	9	Restricted use of asynchronous constructs	E.9	HR high	HR high	HR* high	HR* high
	10	Synchronisation of primary inputs and control of metastability	E.10	HR high	HR high	HR* high	HR* high
	11	Design for testability (depending on the test coverage in percent)	E.11	R > 95 %	R > 98 %	R > 99 %	R > 99 %
	12	Modularisation	E.12	R medium	R medium	HR high	HR high
	13	Coverage of the verification scenarios	E.13	R medium	R medium	HR high	HR high
	14	Observation of coding guidelines	E.14	HR high	HR high	HR* high	HR* high
	15	Application of code checker	E.15	R	R	R	R
	16	Defensive programming	E.16	R low	R medium	HR high	HR* high
	17	Documentation of simulation results	E.17	HR low	HR medium	HR high	HR* high
	18a	Code inspection	E.18	R medium	R high	HR high	HR* high
	18b	Walk-through	E.19	R medium	R high	HR high	HR* high
	19a	Application of validated soft-cores	E.20	R medium	R high	HR* high	HR* high
	19b	Validation of soft-cores	E.21	R medium	R high	HR* high	HR* high

Table F.1 *(continued)*

Design phase	Ref	Technique/Measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Synthesis	20a	Simulation of the gate netlist to check timing constraints	E.22	R medium	R medium	R high	R high
	20b	Static analysis of the propagation delay (STA)	E.23	R medium	R medium	R high	R high
	21a	Verification of the gate netlist against a reference model by simulation	E.24	R medium	R medium	HR high	HR high
	21b	Comparison of the gate netlist with the reference model (formal equivalence check)	E.25	R medium	R medium	HR high	HR high
	22	Check of ASIC vendor requirements and constraints	E.26	HR high	HR high	HR* high	HR* high
	23	Documentation of synthesis constraints, results and tools	E.27	HR high	HR high	HR* high	HR* high
	24	Application of proven in use synthesis tools	E.28	HR* high	HR* high	HR* high	HR* high
	25	Application of proven in use target libraries	E.29	HR* high	HR* high	HR* high	HR* high
	26	Script based procedures	E.30	R medium	R medium	HR high	HR high
Test insertion and test pattern generation	27	Implementation of test structures	E.31	R > 95 %	R > 98 %	R > 99 %	R > 99 %
	28a	Estimation of the test coverage by simulation (based on achieved test coverage in percent)	E.32	R > 95 %	R > 98 %	R > 99 %	R > 99 %
	28b	Estimation of the test coverage by application of ATPG tool (based on achieved test coverage in percent)	E.33	R > 95 %	R > 98 %	R > 99 %	R > 99 %
	29a	Simulation of the gate netlist, to check timing constraints	E.22	R medium	R medium	HR high	HR high
	29b	Static analysis of the propagation delay (STA)	E.23	R medium	R medium	HR high	HR high
	30a	Verification of the gate netlist against a reference model by simulation	E.24	R medium	R medium	HR high	HR high
	30b	Comparison of the gate netlist with the reference model (formal equivalence check)	E.25	R medium	R medium	HR high	HR high

Table F.1 *(continued)*

Design phase	Ref	Technique/Measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Placement, routing, layout generation	31a	Justification of proven in use for applied hard cores	E.34	HR high	HR high	HR* high	HR* high
	31b	Application of validated hard cores	E.35	HR high	HR high	HR* high	HR* high
	31c	Online testing of hard cores	E.36	HR high	HR high	HR* high	HR* high
	32a	Simulation of the gate netlist, to check timing constraints	E.22	HR high	HR high	HR* high	HR* high
	32b	Static analysis of the propagation delay (STA)	E.23	HR high	HR high	HR* high	HR* high
	33a	Verification of the gate netlist against a reference model by simulation	E.24	HR high	HR high	HR* high	HR* high
	33b	Comparison of the gate netlist with the reference model (formal equivalence check)	E.25	HR high	HR high	HR* high	HR* high
	34	Design rule check (DRC)	E.37	HR high	HR high	HR high	HR* high
	35	Verification of layout versus schematic (LVS)	E.38	HR high	HR high	HR high	HR* high
	36	Application of a proven in use design environments, application of proven in use cell libraries	E.4	HR* high	HR* high	HR* high	HR* high
	37	Additional slack (>20 %) for process technologies in use for less than 3 years	E.39	HR high	HR high	HR high	HR* high
Chip manu-facturing	38	Application of a proven in use process technology		HR high	HR high	HR* high	HR* high
	39	Proven in use manufacturing process	E.42	HR low	HR medium	HR high	HR* high
	40	Quality assurance for the process technology		HR high	HR high	HR high	HR* high
	41	Quality control of the manufacturing process	E.43	HR high	HR high	HR high	HR* high
	42	Manufacturing quality pass of the device	E.44	R low	R medium	HR high	HR* high
	43	Functional quality pass of the device	E.45	HR high	HR high	HR* high	HR* high
	44	Test coverage of the manufacturing test		> 95 %	> 98 %	> 99 %	> 99 %
	45	Quality standards	E.46	HR low	HR medium	HR high	HR* high
	46	Quality management, for example according to ISO 9000		HR high	HR high	HR high	HR* high
	47	Burn-in test	E.40	R low	R medium	HR high	HR* high

Appropriate techniques/measures should be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. At least one of the alternate or equivalent techniques/measures should be applied.

NOTE The term (V)HDL denotes either the Very high speed integrated circuit Hardware Description Language (VHDL) or Verilog Hardware Description Language.

Table F.2 – Techniques and measures to avoid introducing faults during ASIC design and development: User programmable ICs (FPGA/PLD/CPLD) (see 7.4.6.7)

Design phase	Ref	Technique/Measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Design entry	1	Structured description	E.3	HR high	HR high	HR* high	HR* high
	2	Design description in (V)HDL (see Note)	E.1	HR high	HR high	HR* high	HR* high
	3	Schematic entry	E.2	– high	– high	NR	NR
	4	Design description using boolean equations		R high	R high	NR	NR
	5a	For circuit descriptions that use boolean equations: manual inspection in designs with limited (low) complexity		HR high	HR high	HR* high	HR* high
	5b	For circuit descriptions that use boolean equations: simulation of state transitions in designs with higher complexity		HR high	HR high	HR* high	HR* high
	6	Application of a proven in use design environment	E.4	HR high	HR high	HR* high	HR* high
	7	Application of proven in use (V)HDL simulators (see Note)	E.4	HR high	HR high	HR* high	HR* high
	8	Functional test on module level (using for example (V)HDL test benches) (see Note)	E.6	HR high	HR high	HR* high	HR* high
	9	Restricted use of asynchronous constructs	E.9	HR high	HR high	HR* high	HR* high
	10	Design for testability (depending on the test coverage in percent)	E.11	R > 95 %	R > 98 %	R > 99 %	R > 99 %
	11	Modularisation	E.12	R medium	R medium	HR high	HR high
	12	Coverage of the verification scenarios (test benches)	E.13	R medium	R medium	HR high	HR high
	13	Observation of coding guidelines	E.14	HR high	HR high	HR* high	HR* high
	14	Documentation of simulation results	E.17	HR low	HR medium	HR high	HR* high
	15a	Code inspection	E.18	R medium	R high	HR high	HR* high
	15b	Walk-through	E.19	R medium	R high	HR high	HR* high
	16a	Application of validated soft-cores	E.20	R medium	R high	HR high	HR* high
	16b	Validation of soft-cores	E.21	R medium	R high	HR* high	HR* high

Table F.2 *(continued)*

Design phase	Ref	Technique/Measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Synthesis	17	Internal consistency checks (see for example IEC 61508-7, E.4)		HR high	HR high	HR* high	HR* high
	18a	Simulation of the gate netlist, to check timing constraints	E.22	R medium	R medium	R high	R high
	18b	Static analysis of the propagation delay (STA)	E.23	R medium	R medium	R high	R high
	19a	Verification of the gate netlist against a reference model by simulation	E.24	R medium	R medium	HR high	HR high
	19b	Comparison of the gate netlist with the reference model (formal equivalence check)	E.25	R medium	R medium	HR high	HR high
	20	For PLD/CPLD in complex designs: check of the design by simulation		R medium	R medium	HR high	HR high
	21	Check of IC vendor requirements and constraints	E.26	HR high	HR high	HR* high	HR* high
	22	Documentation of synthesis constraints, results and tools	E.27	HR high	HR high	HR* high	HR* high
	23	Application of proven in use synthesis tools	E.28	HR high	HR high	HR* high	HR* high
	24	Application of proven in use libraries/CPLD technologies	E.29	HR high	HR high	HR* high	HR* high
	25	Script based procedure	E.30	R high	R high	HR high	HR* high
Placement, routing, layout generation	26a	Justification of proven in use for applied hard cores	E.34	HR high	HR high	HR* high	HR* high
	26b	Application of validated hard cores	E.35	HR high	HR high	HR* high	HR* high
	26c	Online testing of hard cores	E.36	HR high	HR high	HR* high	HR* high
	27a	Simulation of the gate netlist, to check timing constraints	E.22	HR high	HR high	HR* high	HR* high
	27b	Static analysis of the propagation delay (STA)	E.23	HR high	HR high	HR* high	HR* high
	28a	Verification of the gate netlist against a reference model by simulation	E.24	HR high	HR high	HR* high	HR* high
	28b	Comparison of the gate netlist with the reference model (formal equivalence check)	E.25	HR high	HR high	HR* high	HR* high
	29	Design rule check (DRC)	E.37	HR high	HR high	HR high	HR* high
	30	Application of a proven in use design environments, application of proven in use cell libraries	E.4	HR* high	HR* high	HR* high	HR* high
	31	Additional slack (>20 %) for process technologies in use for less than 3 years	E.39	HR high	HR high	HR* high	HR* high

Table F.2 *(continued)*

Design phase	Ref	Technique/Measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Manu-facturing	32	Application of a proven in use process technology		HR high	HR high	HR* high	HR* high
	33	Application of proven in use device-series	E.41	HR high	HR high	HR* high	HR* high
	34	Proven in use manufacturing process	E.42	HR low	HR medium	HR high	HR* high
	35	Quality control of the manufacturing process	E.43	HR high	HR high	HR high	HR* high
	36	Manufacturing quality pass of the device	E.44	R low	R medium	HR high	HR* high
	37	Functional quality pass of the device	E.45	HR high	HR high	HR* high	HR* high
	38	Quality standards	E.46	HR low	HR medium	HR high	HR* high
	39	Quality management, for example according to ISO 9000		HR high	HR high	HR high	HR* high
	40	Final verification and validation of the FPGA/PLD prototype in the system		HR high	HR high	HR* high	HR* high
	41	Final verification and validation during mass manufacturing, per-unit-check		R high	R high	HR* high	HR* high
	42	Burn-in test	E.40	R low	R low	R medium	HR* high

Appropriate techniques/measures should be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. At least one of the alternate or equivalent techniques/measures should be applied.

NOTE The term (V)HDL denotes either the Very high speed integrated circuit Hardware Description Language (VHDL) or Verilog Hardware Description Language.

Bibliography

[1] IEC 61511 (all parts), *Functional safety – Safety instrumented systems for the process industry sector*

[2] IEC 62061, *Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems*

[3] IEC 61800-5-2, *Adjustable speed electrical power drive systems – Part 5-2: Safety requirements – Functional*

[4] IEC 61508-5: 2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 5: Examples of methods for the determination of safety integrity* levels

[5] IEC 61508-6:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3*

[6] IEC 60601 (all parts), *Medical electrical equipment*

[7] IEC 61165, *Application of Markov techniques*

[8] IEC 61078, *Analysis techniques for dependability – Reliability block diagram and boolean methods*

[9] IEC 61164, *Reliability growth – Statistical test and estimation methods*

[10] IEC 62308, *Equipment reliability – Reliability assessment methods*

[11] IEC 61000-6-2, *Electromagnetic compatibility (EMC) – Part 6-2: Generic standards – Immunity for industrial environments*

[12] ISO 14224, *Petroleum, petrochemical and natural gas industries – Collection and exchange of reliability and maintenance data for equipment*

[13] IEC 60050-191, *International Electrotechnical Vocabulary – Chapter 191: Dependability and quality of service*

[14] ISO 9000*, Quality management systems – Fundamentals and vocabulary*

[15] IEC 60300-3-2, *Dependability management – Part 3-2: Application guide – Collection of dependability data from the field*

[16] IEEE 352:1987, *IEEE guide for general principles of reliability analysis of nuclear power generating station safety systems*

_____

IEC 61508-3

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

BASIC SAFETY PUBLICATION
PUBLICATION FONDAMENTALE DE SÉCURITÉ

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

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 3: Exigences concernant les logiciels

IEC 61508-3:2010

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.
- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.
- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.
- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:
Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm
Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,…). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.
- Just Published CEI: www.iec.ch/online_news/justpub
Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.
- Electropedia: www.electropedia.org
Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.
- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:
Email: csc@iec.ch
Tél.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61508-3

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

BASIC SAFETY PUBLICATION

PUBLICATION FONDAMENTALE DE SÉCURITÉ

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

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 3: Exigences concernant les logiciels

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE CODE PRIX XE

ICS 25.040.40

ISBN 978-2-88910-526-7

CONTENTS

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) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61508-3 has been prepared by subcommittee 65A: System aspects, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 1998. This edition constitutes a technical revision.

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 standard is based on the following documents:

FDIS	Report on voting
65A/550/FDIS	65A/574/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

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

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

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.

– 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.

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

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.

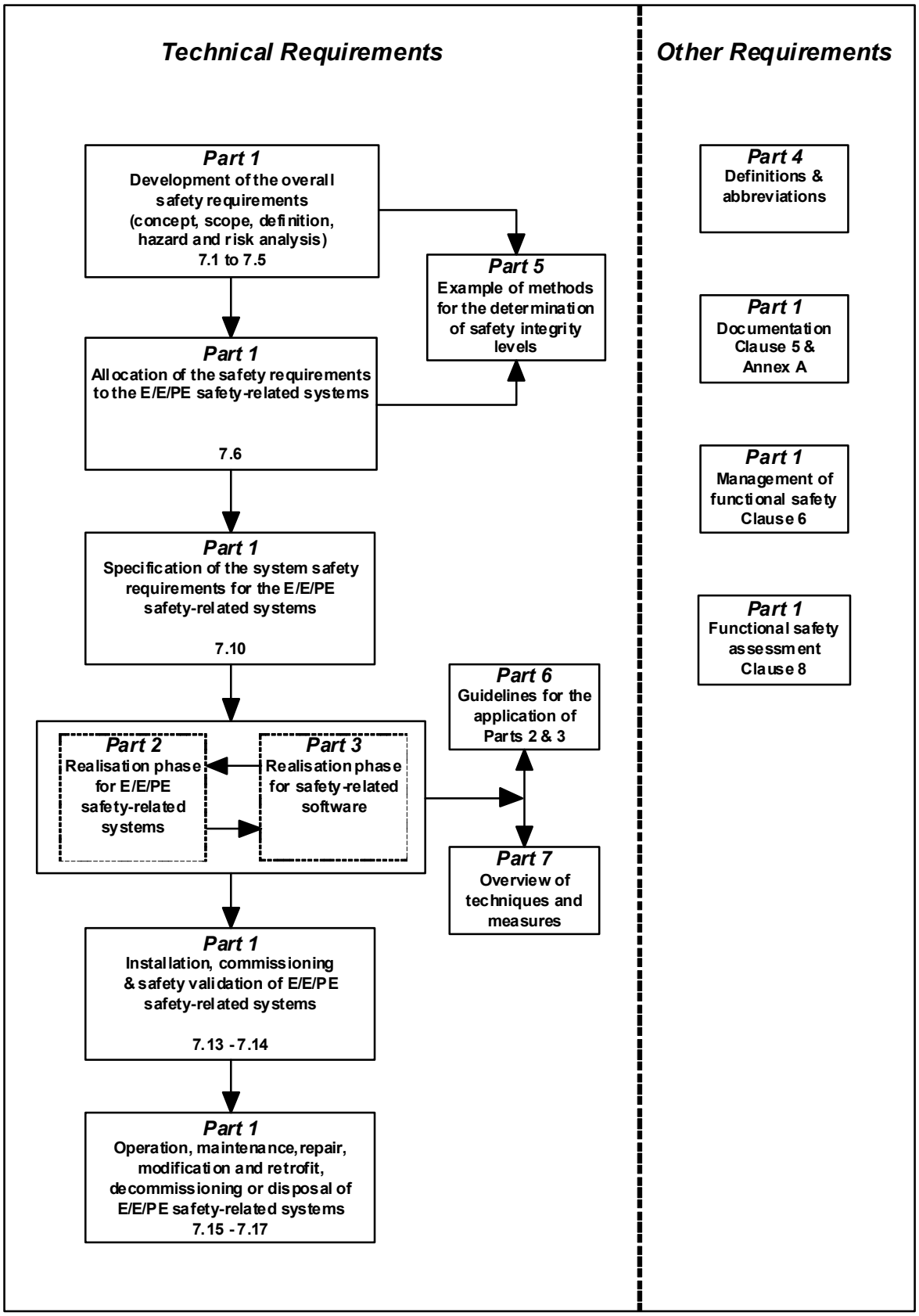


Figure 1 – Overall framework of the IEC 61508 series

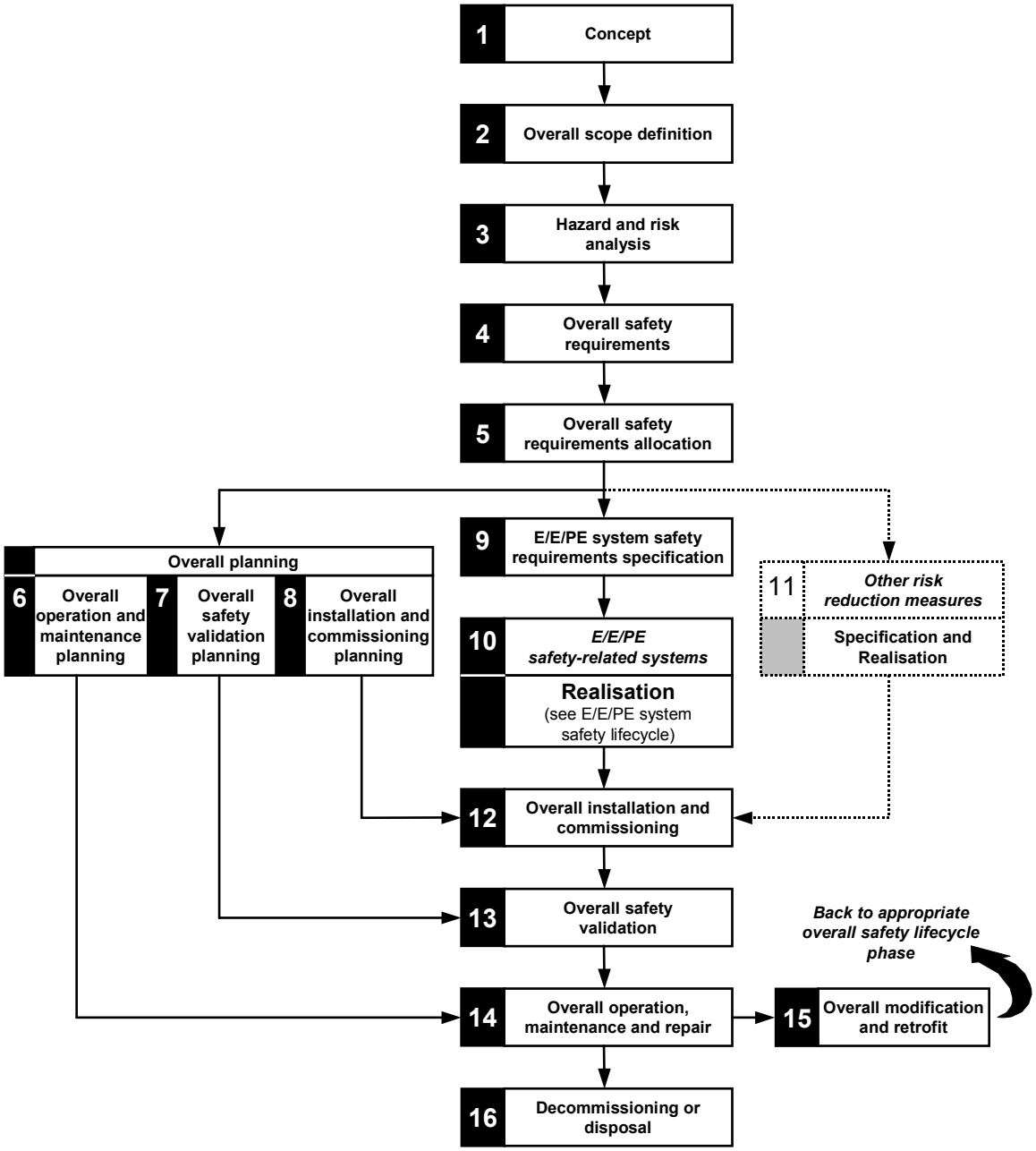


Figure 2 – Overall safety lifecycle

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*

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

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.

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.

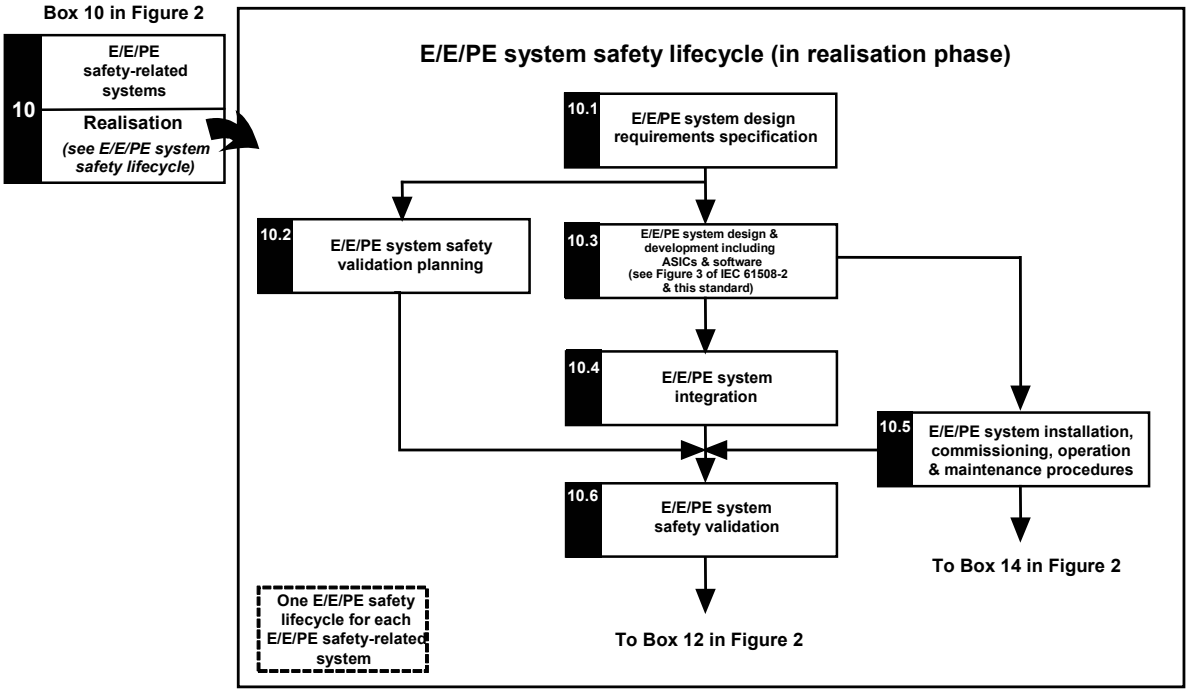


Figure 3 – E/E/PE system safety lifecycle (in realisation phase)

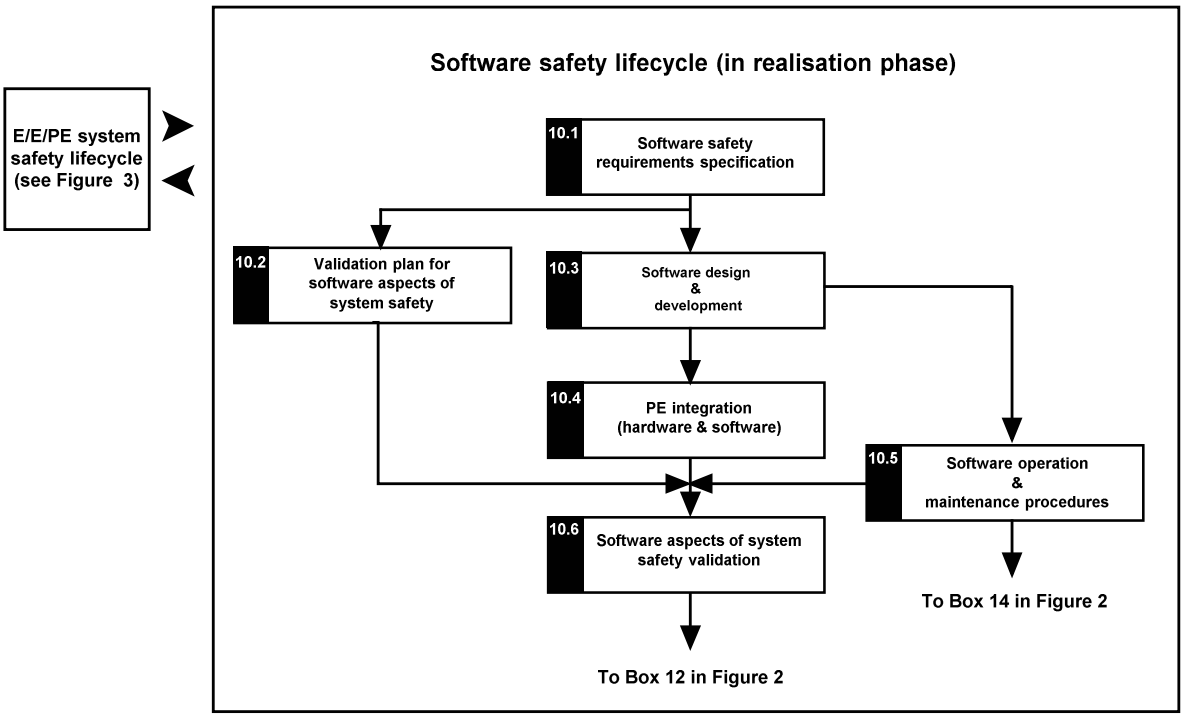


Figure 4 – Software safety lifecycle (in realisation phase)

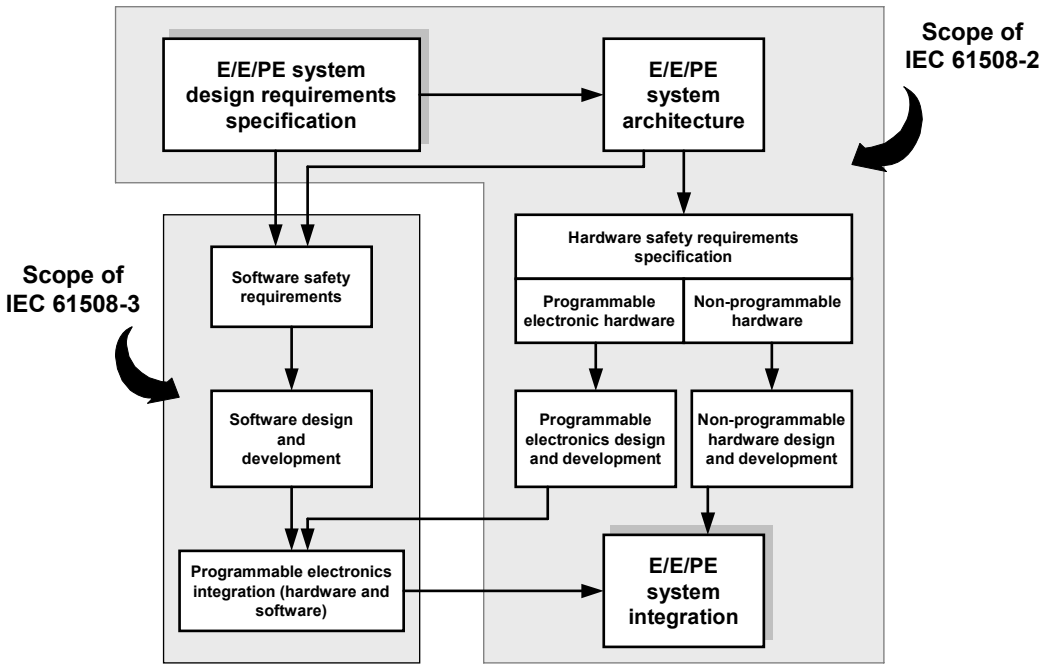


Figure 5 – Relationship and scope for IEC 61508-2 and IEC 61508-3

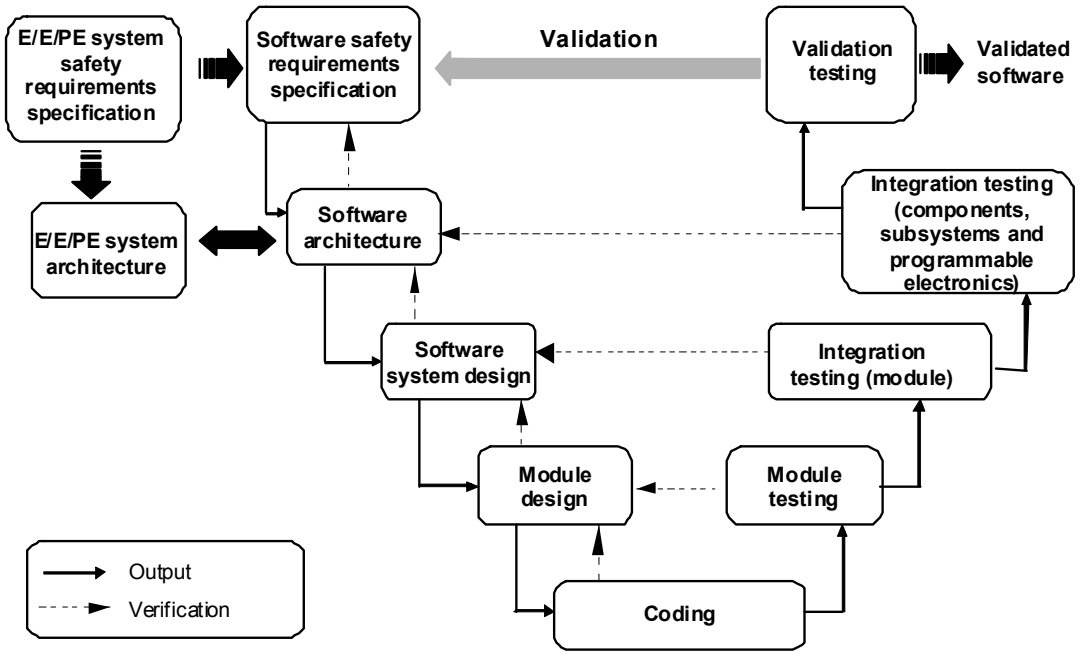


Figure 6 – Software systematic capability and the development lifecycle (the V-model)

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

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

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; 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
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

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.

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);

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

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;

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.

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.

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.

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.

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;

– 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.

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.

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.

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.

However it is still good practice to design the software in a structured way, including organising the software into a modular structure that separates out (as far as possible) safety-related parts; including range checking and other features that provide protection against data input mistakes; using previously verified software modules; and providing a design that facilitates future software modifications.

NOTE 3 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 design and development should be considered:

– completeness with respect to software safety requirements specification;

– correctness with respect to software safety requirements specification;

– freedom from intrinsic design faults;

– simplicity and understandability

– predictability of behaviour;

– verifiable and testable design;

– fault tolerance / fault detection;

– freedom from common cause failure.

7.4.5.1 Depending on the nature of the software development, responsibility for conformance with 7.4.5 can rest with multiple parties. The division of responsibility shall be documented during safety planning (see Clause 6 of IEC 61508-1).

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

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

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

NOTE 1 For pre-existing software elements, see 7.4.2.

NOTE 2 Verification includes testing and analysis.

7.4.5.5 Appropriate software system integration tests shall be specified to ensure that the software system satisfies the software safety requirements specification at the required safety integrity level.

7.4.6 Requirements for code implementation

NOTE To the extent required by the safety integrity level, the source code shall possess the following properties (see Annexes A and B for specific techniques, and see Annex C for guidance on interpretation of properties) of code should be considered:

- be readable, understandable and testable;
- satisfy the specified requirements for software module design (see 7.4.5);
- satisfy the specified requirements of the coding standards (see 7.4.4);
- satisfy all relevant requirements specified during safety planning (see Clause 6).

7.4.6.1 Each module of software code shall be reviewed. Where the code is produced by an automatic tool, the requirements of 7.4.4 shall be met. Where the source code consists of reused pre-existing software, the requirements of 7.4.2 shall be met.

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

7.4.7 Requirements for software module testing

NOTE 1 Testing that the software module correctly satisfies its test specification is a verification activity (see 7.9). It is the combination of code review and software module testing that provides assurance that a software module satisfies its associated specification, i.e. it is verified.

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 module testing should be considered:

– completeness of testing with respect to the software design specification;

– correctness of testing with respect to the software design specification (successful completion);

– repeatability;

– precisely defined testing configuration.

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

NOTE Verification includes testing and analysis.

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

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

NOTE 2 Where the development uses formal methods, formal proofs or assertions, such tests may be reduced in scope. See Annex C of IEC 61508-7 for these techniques.

NOTE 3 Although systematic safety integrity is usually unquantified (see 3.5.6 of IEC 61508-4), quantified statistical evidence (e.g. statistical testing, reliability growth) is acceptable if all the relevant conditions for statistically valid evidence are satisfied e.g. see Annex D of IEC 61508-7.

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

7.4.7.3 The results of the software module testing shall be documented.

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

7.4.8 Requirements for software integration testing

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

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

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

a) the division of the software into manageable integration sets;

b) test cases and test data;

c) types of tests to be performed;

d) test environment, tools, configuration and programs;

e) test criteria on which the completion of the test will be judged;

f) procedures for corrective action on failure of test.

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

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

NOTE 2 Where the development uses formal methods, formal proofs or assertions, such tests may be reduced in scope. See Annex C of IEC 61508-7 for these techniques.

NOTE 3 Although systematic safety integrity is usually unquantified (see 3.5.6 of IEC 61508-4), quantified statistical evidence (e.g. statistical testing, reliability growth) is acceptable if all the relevant conditions for statistically valid evidence are satisfied e.g. see Annex D of IEC 61508-7.

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

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

7.5 Programmable electronics integration (hardware and software)

NOTE This phase is box 10.4 of Figure 4.

7.5.1 Objectives

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

7.5.1.2 The second objective of the requirements of this subclause is 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.

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

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

7.5.2 Requirements

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 the integration should be considered:

– completeness of integration with respect to the design specifications;

– correctness of integration with respect to the design specifications (successful completion);

– repeatability;

– precisely defined integration configuration.

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

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

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

a) the split of the system into integration levels;

b) test cases and test data;

c) types of tests to be performed;

d) test environment including tools, support software and configuration description;

e) test criteria on which the completion of the test will be judged.

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

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

a) merging of the software system on to the target programmable electronic hardware;

b) E/E/PE integration, i.e. adding interfaces such as sensors and actuators;

c) applying the E/E/PE safety-related system to the EUC.

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 and for completeness. They are not normally the responsibility of the software developers.

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

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

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

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

7.6 Software operation and modification procedures

NOTE This phase is box 10.5 of Figure 4.

7.6.1 Objective

The objective of the requirements of this subclause is 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.

7.6.2 Requirements

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

NOTE In this standard software (unlike hardware) is not capable of being maintained: it is always modified.

7.7 Software aspects of system safety validation

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.

7.7.1 Objective

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 safety integrity level.

7.7.2 Requirements

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 safety validation should be considered:

– completeness of validation with respect to the software design specification;

– correctness of validation with respect to the software design specification (successful completion);

– repeatability;

– precisely defined validation configuration.

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 7.7 of IEC 61508-2), then the validation need not be repeated.

7.7.2.2 The validation activities shall be carried out as specified the in validation plan for software aspects of system safety.

7.7.2.3 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 Clause 6 of IEC 61508-1).

7.7.2.4 The results of validating the software aspects of system safety shall be documented.

7.7.2.5 For each safety function, software safety validation shall document the following results:

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.

NOTE The requirements of 7.7.2.2 to 7.7.2.6 are based on the general requirements given in 7.14 of IEC 61508-1.

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

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

b) the software shall be exercised by simulation of:

   1) input signals present during normal operation;

   2) anticipated occurrences;

   3) undesired conditions requiring system action;

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

7.7.2.8 Software tools shall meet the requirements of 7.4.4.

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

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

b) test cases and their results shall be documented for subsequent analysis and independent assessment (see Clause 8 of IEC 61508-1) as required by the safety integrity level;

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

7.8 Software modification

NOTE This phase is Box 10.5 of Figure 4.

7.8.1 Objective

The objective of the requirements of this subclause is to guide corrections, enhancements or adaptations to the validated software, ensuring that the required software systematic capability is sustained.

7.8.2 Requirements

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 the software modification should be considered:

– completeness of modification with respect to its requirements;

– correctness of modification with respect to its requirements;

– freedom from introduction of intrinsic design faults;

– avoidance of unwanted behaviour;

– verifiable and testable design;

– regression testing and verification coverage.

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

NOTE 1 Subclauses 7.8.2.1 to 7.8.2.9 apply primarily to changes occurring during the operational phase of the software. They may also apply during the programmable electronics integration and overall installation and commissioning phases (see 7.13 of IEC 61508-1).

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

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

a) the hazards which may be affected;

b) the proposed modification;

c) the reasons for modification.

NOTE A request for modification could arise from, for example

- functional safety is found to be less than required by the safety requirements specification;
- systematic fault experience;

- new or amended safety legislation;
- modifications to the EUC or its use;
- modification to the overall safety requirements;
- analysis of operations and maintenance performance, indicating that the performance is below target;
- routine functional safety audits.

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

a) to determine whether or not a hazard and risk analysis is required;

b) to determine which software safety lifecycle phases will need to be repeated.

7.8.2.4 The impact analysis results obtained in 7.8.2.3 shall be documented.

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

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

7.8.2.6 The safety planning for the modification of safety-related software shall meet the requirements given in Clause 6 of IEC 61508-1. In particular:

a) identification of staff and specification of their required competency;

b) detailed specification for the modification;

c) verification planning;

d) scope of revalidation and testing of the modification to the extent required by the safety integrity level.

NOTE Depending on the nature of the application, involvement of domain experts may be important.

7.8.2.7 Modification shall be carried out as planned.

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

a) the modification/retrofit request;

b) the results of the impact analysis which assesses the impact of the proposed software modification on the functional safety, and the decisions taken with associated justifications;

c) software configuration management history;

d) deviation from normal operations and conditions;

e) all documented information affected by the modification activity.

7.8.2.9 Information on the details of all modifications shall be documented. The documentation shall include the re-verification and re-validation of data and results.

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

7.9 Software verification

7.9.1 Objective

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

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

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

7.9.2 Requirements

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 the data verification should be considered:

– completeness of verification with respect to the previous phase;

– correctness of verification with respect to the previous phase (successful completion);

– repeatability;

– precisely defined verification configuration.

7.9.2.1 The verification of software shall be planned (see 7.3) concurrently with the development, for each phase of the software safety lifecycle, and shall be documented.

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

a) the evaluation of the safety integrity requirements;

b) the selection and documentation of verification strategies, activities and techniques;

c) the selection and utilisation of verification tools (test harness, special test software, input/output simulators etc.);

d) the evaluation of verification results;

e) the corrective actions to be taken.

7.9.2.3 The software verification shall be performed as planned.

NOTE Selection of techniques, measures for verification and the degree of independence of the verification activities will depend upon a number of factors and may be specified in application sector standards. The factors could include, for example:

- size of project;
- degree of complexity;
- degree of novelty of design;
- degree of novelty of technology.

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

7.9.2.5 After each verification, the verification documentation shall include:

a) identification of items to be verified;

b) identification of the information against which the verification has been done;

NOTE 1 Information against which the verification has been performed includes but is not limited to input from the previous lifecycle phase, design standards, coding standards and tools used.

c) non-conformances.

NOTE 2 Examples of non-conformances include software modules, data structures, and algorithms poorly adapted to the problem.

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

a) adequacy of the specification, design, or code in phase N for:
   1) functionality;
   2) safety integrity, performance and other requirements of safety planning (see Clause 6);
   3) readability by the development team;
   4) testability for further verification;
   5) safe modification to permit further evolution;
b) adequacy of the validation planning and/or tests specified for phase N for specifying and describing the design of phase N;
c) check for incompatibilities between:
   1) the tests specified in phase N, and the tests specified in the previous phase N–1;
   2) the outputs within phase N.

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

a) verification of software safety requirements;
b) verification of software architecture;
c) verification of software system design;
d) verification of software module design;
e) verification of code;
f) verification of data;
g) verification of timing performance;
h) software module testing (see 7.4.7);
i) software integration testing (see 7.4.8);
j) programmable electronics integration testing (see 7.5);
k) software aspects of system safety validation (see 7.7).

NOTE For requirements a) to g) see below.

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

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

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

a) consider whether the software architecture design adequately fulfils the software safety requirements specification;

b) consider whether the integration tests specified in the software architecture design are adequate;

c) consider whether the attributes of each major element/subsystem are adequate with reference to:

   1) feasibility of the safety performance required;

   2) testability for further verification;

   3) readability by the development and verification team;

   4) safe modification to permit further evolution.

d) check for incompatibilities between the following:

   1) the software architecture design, and the software safety requirements specification;

   2) the software architecture design and its integration tests;

   3) the software architecture design integration tests and the validation plan for software aspects of system safety.

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

a) consider whether the software system design (see 7.4.5) adequately fulfils the software architecture design;

b) consider whether the specified tests of the software system integration (see 7.4.5) adequately fulfil the software system design (see 7.4.5);

c) consider whether the attributes of each major element of the software system design specification (see 7.4.5) are adequate with reference to:

   1) feasibility of the safety performance required;

   2) testability for further verification;

   3) readability by the development and verification team;

   4) safe modification to permit further evolution.

   NOTE The software system integration tests may be specified as part of the software architecture integration tests.

d) check for incompatibilities between:

   1) the software system design specification (see 7.4.5), and the software architecture design;

   2) the software system design specification (see 7.4.5), and the software system integration test specification (see.4.5);

   3) the tests required by the software system integration test specification (see 7.4.5) and the software architecture integration test specification (see 7.4.3).

7.9.2.11 Verification of software module design: after the design of each software module has been completed, verification shall:

a) consider whether the software module design specification (see 7.4.5) adequately fulfils the software system design specification (see 7.4.5);

b) consider whether the software module test specification (see 7.4.5) is adequate for the software module design specification (see 7.4.5);

c) consider whether the attributes of each software module are adequate with reference to:

   1) feasibility of the safety performance required (see software safety requirements specification);

2) testability for further verification;

3) readability by the development and verification team;

4) safe modification to permit further evolution.

d) check for incompatibilities between:

1) the software module design specification (see 7.4.5), and the software system design specification (see 7.4.5);

2) (for each software module) the software module design specification (see 7.4.5), and the software module test specification (see 7.4.5);

3) the software module test specification (see 7.4.5), and the software system integration test specification (see 7.4.5).

7.9.2.12 Verification of code: the source code shall be verified by static methods to ensure conformance to the software module design specification (see 7.4.5), the required coding standards (see 7.4.4), and the validation plan for software aspects of system safety.

NOTE In the early phases of the software safety lifecycle, verification is static (for example inspection, review, formal proof, etc). Code verification includes such techniques as software inspections and walk-throughs. It is the combination of the results of code verification and software module testing that provides assurance that each software module satisfies its associated specification. From then onwards testing becomes the primary means of verification.

7.9.2.13 Verification of data.

a) The data structures shall be verified.

b) The application data shall be verified for:

1) consistency with the data structures;

2) completeness against the application requirements;

3) compatibility with the underlying system software (for example, sequence of execution, run-time, etc.); and

4) correctness of the data values.

c) All operational parameters shall be verified against the application requirements.

d) All plant interfaces and associated software (i.e. sensors and actuators and off-line interfaces: see 7.2.2.12) shall be verified for:

1) detection of anticipated interface failures;

2) tolerance to anticipated interface failures.

e) All communication interfaces and associated software shall be verified for an adequate level of:

1) failure detection;

2) protection against corruption;

3) data validation.

7.9.2.14 Verification of timing performance: predictability of behaviour in the time domain shall be verified.

NOTE Timing behaviour may include: performance, resources, response time, worst case execution time, thrashing, dead-lock free, run-time system.

8 Functional safety assessment

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 the functional safety assessment should be considered:

– completeness of functional safety assessment with respect to this standard;

– correctness of functional safety assessment with respect to the design specifications (successful completion);

– traceable closure of all identified issues;

– the ability to modify the functional safety assessment after change without the need for extensive re-work of the assessment;

– repeatability;

– timeliness;

– precisely defined configuration.

8.1 The objective and requirements of Clause 8 of IEC 61508-1 apply to the assessment of safety-related software.

8.2 Unless otherwise stated in application sector international standards, the minimum level of independence of those carrying out the functional safety assessment shall be as specified in Clause 8 of IEC 61508-1.

8.3 An assessment of functional safety may make use of the results of the activities of Table A.10.

NOTE Selecting techniques from Annexes A and B does not guarantee by itself that the required safety integrity will be achieved (see 7.1.2.7). The assessor should also consider:

- 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.

Annex A
(normative)

Guide to the selection of techniques and measures

Some of the subclauses of this standard have an associated table, for example 7.2 (software safety requirements specification) is associated with Table A.1. More detailed tables in Annex B expand upon some of the entries in the tables of Annex A. For example, Table B.2 expands on the topic of dynamic analysis and testing in Table A.5.

See IEC 61508-7 for an overview of the specific techniques and measures referenced in Annexes A and B.

With each technique or measure in the tables there is a recommendation for safety integrity levels 1 to 4. These recommendations are as follows.

HR	the technique or measure is highly recommended for this safety integrity level. If this technique or measure is not used then the rationale behind not using it should be detailed with reference to Annex C during the safety planning and agreed with the assessor.
R	the technique or measure is recommended for this safety integrity level as a lower recommendation to a HR recommendation.
---	the technique or measure has no recommendation for or against being used.
NR	the technique or measure is positively not recommended for this safety integrity level. If this technique or measure is used then the rationale behind using it should be detailed with reference to Annex C during the safety planning and agreed with the assessor.

Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

Other measures and techniques may be applied providing that the requirements and objectives have been met. See Annex C for guidance on selecting techniques.

The ranking of the techniques and measures is linked to the concept of *effectiveness* used in IEC 61508-2. For all other factors being equal, techniques which are ranked HR will be more effective in either preventing the introduction of systematic faults during software development, or (for the case of the software architecture) more effective in controlling residual faults in the software revealed during execution than techniques ranked as R.

Given the large number of factors that affect software systematic capability it is not possible to give an algorithm for combining the techniques and measures that will be correct for any given application. Guidance on a rationale for selecting specific techniques to achieve software systematic capability is given in Annex C.

For a particular application, the appropriate combination of techniques or measures are to be stated during safety planning, with appropriate techniques or measures being selected unless the note attached to the table makes other requirements.

Initial guidance in the form of two worked examples on the interpretation of the tables is given in IEC 61508-6.

Table A.1 – Software safety requirements specification

(See 7.2)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1a	Semi-formal methods	Table B.7	R	R	HR	HR
1b	Formal methods	B.2.2, C.2.4	---	R	R	HR
2	Forward traceability between the system safety requirements and the software safety requirements	C.2.11	R	R	HR	HR
3	Backward traceability between the safety requirements and the perceived safety needs	C.2.11	R	R	HR	HR
4	Computer-aided specification tools to support appropriate techniques/measures above	B.2.4	R	R	HR	HR

NOTE 1 The software safety requirements specification will always require a description of the problem in natural language and any necessary mathematical notation that reflects the application.

NOTE 2 The table reflects additional requirements for specifying the software safety requirements clearly and precisely.

NOTE 3 See Table C.1.

NOTE 4 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

Table A.2 – Software design and development – software architecture design

(see 7.4.3)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
	Architecture and design feature					
1	Fault detection	C.3.1	---	R	HR	HR
2	Error detecting codes	C.3.2	R	R	R	HR
3a	Failure assertion programming	C.3.3	R	R	R	HR
3b	Diverse monitor techniques (with independence between the monitor and the monitored function in the same computer)	C.3.4	---	R	R	----
3c	Diverse monitor techniques (with separation between the monitor computer and the monitored computer)	C.3.4	---	R	R	HR
3d	Diverse redundancy, implementing the same software safety requirements specification	C.3.5	---	---	---	R
3e	Functionally diverse redundancy, implementing different software safety requirements specification	C.3.5	---	---	R	HR
3f	Backward recovery	C.3.6	R	R	---	NR
3g	Stateless software design (or limited state design)	C.2.12	---	---	R	HR
4a	Re-try fault recovery mechanisms	C.3.7	R	R	---	---
4b	Graceful degradation	C.3.8	R	R	HR	HR
5	Artificial intelligence - fault correction	C.3.9	---	NR	NR	NR
6	Dynamic reconfiguration	C.3.10	---	NR	NR	NR
7	Modular approach	Table B.9	HR	HR	HR	HR
8	Use of trusted/verified software elements (if available)	C.2.10	R	HR	HR	HR
9	Forward traceability between the software safety requirements specification and software architecture	C.2.11	R	R	HR	HR
10	Backward traceability between the software safety requirements specification and software architecture	C.2.11	R	R	HR	HR
11a	Structured diagrammatic methods **	C.2.1	HR	HR	HR	HR
11b	Semi-formal methods **	Table B.7	R	R	HR	HR
11c	Formal design and refinement methods **	B.2.2, C.2.4	---	R	R	HR
11d	Automatic software generation	C.4.6	R	R	R	R
12	Computer-aided specification and design tools	B.2.4	R	R	HR	HR
13a	Cyclic behaviour, with guaranteed maximum cycle time	C.3.11	R	HR	HR	HR
13b	Time-triggered architecture	C.3.11	R	HR	HR	HR
13c	Event-driven, with guaranteed maximum response time	C.3.11	R	HR	HR	-
14	Static resource allocation	C.2.6.3	-	R	HR	HR
15	Static synchronisation of access to shared resources	C.2.6.3	-	-	R	HR

NOTE 1 Some of the methods given in Table A.2 are about design concepts, others are about how the design is represented.

NOTE 2 The measures in this table concerning fault tolerance (control of failures) should be considered with the requirements for architecture and control of failures for the hardware of the programmable electronics in IEC 61508-2.

NOTE 3 See Table C.2.

NOTE 4 The group 13 measures apply only to systems and software with safety timing requirements.

NOTE 5 Measure 14. The use of dynamic objects (for example on the execution stack or on a heap) may impose requirements on both available memory and also execution time. Measure 14 does not need to be applied if a compiler is used which ensures a) that sufficient memory for all dynamic variables and objects will be allocated before runtime, or which guarantees that in case of memory allocation error, a safe state is achieved; b) that response times meet the requirements.

NOTE 6 Measure 4a. Re-try fault recovery is often appropriate at any SIL but a limit should be set on the number of retries.

NOTE 7 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

** Group 11, “Structured methods”. Use measure 11a only if 11b is not suited to the domain for SIL 3+4.

Table A.3 – Software design and development – support tools and programming language

(See 7.4.4)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Suitable programming language	C.4.5	HR	HR	HR	HR
2	Strongly typed programming language	C.4.1	HR	HR	HR	HR
3	Language subset	C.4.2	---	---	HR	HR
4a	Certified tools and certified translators	C.4.3	R	HR	HR	HR
4b	Tools and translators: increased confidence from use	C.4.4	HR	HR	HR	HR

NOTE 1 See Table C.3.

NOTE 2 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

Table A.4 – Software design and development – detailed design

(See 7.4.5 and 7.4.6)

(Includes software system design, software module design and coding)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1a	Structured methods **	C.2.1	HR	HR	HR	HR
1b	Semi-formal methods **	Table B.7	R	HR	HR	HR
1c	Formal design and refinement methods **	B.2.2, C.2.4	---	R	R	HR
2	Computer-aided design tools	B.3.5	R	R	HR	HR
3	Defensive programming	C.2.5	---	R	HR	HR
4	Modular approach	Table B.9	HR	HR	HR	HR
5	Design and coding standards	C.2.6 Table B.1	R	HR	HR	HR
6	Structured programming	C.2.7	HR	HR	HR	HR
7	Use of trusted/verified software elements (if available)	C.2.10	R	HR	HR	HR
8	Forward traceability between the software safety requirements specification and software design	C.2.11	R	R	HR	HR

NOTE 1 See Table C.4.

NOTE 2 There is still debate about the suitability of OO software development for safety-related systems. See Annex G of IEC 61508-7 for guidance on object oriented architecture and design.

NOTE 3 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

** Group 1, “Structured methods”. Use measure 1a only if 1b is not suited to the domain for SIL 3+4.

Table A.5 – Software design and development – software module testing and integration

(See 7.4.7 and 7.4.8)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Probabilistic testing	C.5.1	---	R	R	R
2	Dynamic analysis and testing	B.6.5 Table B.2	R	HR	HR	HR
3	Data recording and analysis	C.5.2	HR	HR	HR	HR
4	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	HR	HR	HR
5	Performance testing	Table B.6	R	R	HR	HR
6	Model based testing	C.5.27	R	R	HR	HR
7	Interface testing	C.5.3	R	R	HR	HR
8	Test management and automation tools	C.4.7	R	HR	HR	HR
9	Forward traceability between the software design specification and the module and integration test specifications	C.2.11	R	R	HR	HR
10	Formal verification	C.5.12	---	---	R	R

NOTE 1 Software module and integration testing are verification activities (see Table B.9).

NOTE 2 See Table C.5.

NOTE 3 Technique 9. Formal verification may reduce the amount and extent of module and integration testing required.

NOTE 4 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level.

Table A.6 – Programmable electronics integration (hardware and software)

(See 7.5)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	HR	HR	HR
2	Performance testing	Table B.6	R	R	HR	HR
3	Forward traceability between the system and software design requirements for hardware/software integration and the hardware/software integration test specifications	C.2.11	R	R	HR	HR

NOTE 1 Programmable electronics integration is a verification activity (see Table A.9).

NOTE 2 See Table C.6.

NOTE 3 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level.

Table A.7 – Software aspects of system safety validation

(See 7.7)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Probabilistic testing	C.5.1	---	R	R	HR
2	Process simulation	C.5.18	R	R	HR	HR
3	Modelling	Table B.5	R	R	HR	HR
4	Functional and black-box testing	B.5.1 B.5.2 Table B.3	HR	HR	HR	HR
5	Forward traceability between the software safety requirements specification and the software safety validation plan	C.2.11	R	R	HR	HR
6	Backward traceability between the software safety validation plan and the software safety requirements specification	C.2.11	R	R	HR	HR

NOTE 1 See Table C.7.

NOTE 2 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level.

Table A.8 – Modification

(See 7.8)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Impact analysis	C.5.23	HR	HR	HR	HR
2	Reverify changed software module	C.5.23	HR	HR	HR	HR
3	Reverify affected software modules	C.5.23	R	HR	HR	HR
4a	Revalidate complete system	Table A.7	---	R	HR	HR
4b	Regression validation	C.5.25	R	HR	HR	HR
5	Software configuration management	C.5.24	HR	HR	HR	HR
6	Data recording and analysis	C.5.2	HR	HR	HR	HR
7	Forward traceability between the Software safety requirements specification and the software modification plan (including reverification and revalidation)	C.2.11	R	R	HR	HR
8	Backward traceability between the software modification plan (including reverification and revalidation)and the software safety requirements specification	C.2.11	R	R	HR	HR

NOTE 1 See Table C.8.

NOTE 2 Techniques group 4. Impact analysis is a necessary part of regression validation. See IEC 61508-7.

NOTE 3 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

Table A.9 – Software verification

(See 7.9)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Formal proof	C.5.12	---	R	R	HR
2	Animation of specification and design	C.5.26	R	R	R	R
3	Static analysis	B.6.4 Table B.8	R	HR	HR	HR
4	Dynamic analysis and testing	B.6.5 Table B.2	R	HR	HR	HR
5	Forward traceability between the software design specification and the software verification (including data verification) plan	C.2.11	R	R	HR	HR
6	Backward traceability between the software verification (including data verification) plan and the software design specification	C.2.11	R	R	HR	HR
7	Offline numerical analysis	C.2.13	R	R	HR	HR
Software module testing and integration		See Table A.5				
Programmable electronics integration testing		See Table A.6				
Software system testing (validation)		See Table A.7				

NOTE 1 For convenience all verification activities have been drawn together under this table. However, this does not place additional requirements for the dynamic testing element of verification in Table A.5 and Table A.6 which are verification activities in themselves. Nor does this table require verification testing in addition to software validation (see Table B.7), which in this standard is the demonstration of conformance to the safety requirements specification (end-end verification).

NOTE 2 Verification crosses the boundaries of IEC 61508-1, IEC 61508-2 and IEC 61508-3. Therefore the first verification of the safety-related system is against the earlier system level specifications.

NOTE 3 In the early phases of the software safety lifecycle verification is static, for example inspection, review, formal proof. When code is produced dynamic testing becomes possible. It is the combination of both types of information that is required for verification. For example code verification of a software module by static means includes such techniques as software inspections, walk-throughs, static analysis, formal proof. Code verification by dynamic means includes functional testing, white-box testing, statistical testing. It is the combination of both types of evidence that provides assurance that each software module satisfies its associated specification.

NOTE 4 See Table C.9.

NOTE 5 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level.

Table A.10 – Functional safety assessment

(see Clause 8)

Assessment/Technique *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Checklists	B.2.5	R	R	R	R
2	Decision/truth tables	C.6.1	R	R	R	R
3	Failure analysis	Table B.4	R	R	HR	HR
4	Common cause failure analysis of diverse software (if diverse software is actually used)	C.6.3	---	R	HR	HR
5	Reliability block diagram	C.6.4	R	R	R	R
6	Forward traceability between the requirements of Clause 8 and the plan for software functional safety assessment	C.2.11	R	R	HR	HR
NOTE 1 See Table C.10. NOTE 2 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.						
* Appropriate techniques/measures shall be selected according to the safety integrity level.						

Annex B
(informative)

Detailed tables

Table B.1 – Design and coding standards

(Referenced by Table A.4)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Use of coding standard to reduce likelihood of errors	C.2.6.2	HR	HR	HR	HR
2	No dynamic objects	C.2.6.3	R	HR	HR	HR
3a	No dynamic variables	C.2.6.3	---	R	HR	HR
3b	Online checking of the installation of dynamic variables	C.2.6.4	---	R	HR	HR
4	Limited use of interrupts	C.2.6.5	R	R	HR	HR
5	Limited use of pointers	C.2.6.6	---	R	HR	HR
6	Limited use of recursion	C.2.6.7	---	R	HR	HR
7	No unstructured control flow in programs in higher level languages	C.2.6.2	R	HR	HR	HR
8	No automatic type conversion	C.2.6.2	R	HR	HR	HR

NOTE 1 Measures 2, 3a and 5. The use of dynamic objects (for example on the execution stack or on a heap) may impose requirements on both available memory and also execution time. Measures 2, 3a and 5 do not need to be applied if a compiler is used which ensures a) that sufficient memory for all dynamic variables and objects will be allocated before runtime, or which guarantees that in case of memory allocation error, a safe state is achieved; b) that response times meet the requirements.

NOTE 2 See Table C.11.

NOTE 3 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

Table B.2 – Dynamic analysis and testing

(Referenced by Tables A.5 and A.9)

Technique/Measure *		Ref	SIL 1	SIL 2	SIL 3	SIL 4
1	Test case execution from boundary value analysis	C.5.4	R	HR	HR	HR
2	Test case execution from error guessing	C.5.5	R	R	R	R
3	Test case execution from error seeding	C.5.6	---	R	R	R
4	Test case execution from model-based test case generation	C.5.27	R	R	HR	HR
5	Performance modelling	C.5.20	R	R	R	HR
6	Equivalence classes and input partition testing	C.5.7	R	R	R	HR
7a	Structural test coverage (entry points) 100 % **	C.5.8	HR	HR	HR	HR
7b	Structural test coverage (statements) 100 %**	C.5.8	R	HR	HR	HR
7c	Structural test coverage (branches) 100 %**	C.5.8	R	R	HR	HR
7d	Structural test coverage (conditions, MC/DC) 100 %**	C.5.8	R	R	R	HR

NOTE 1 The analysis for the test cases is at the subsystem level and is based on the specification and/or the specification and the code.

NOTE 2 See Table C.12.

NOTE 3 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level.

** Where 100 % coverage cannot be achieved (e.g. statement coverage of defensive code), an appropriate explanation should be given.

Table B.3 – Functional and black-box testing

(Referenced by Tables A.5, A.6 and A.7)

Technique/Measure *		Ref	SIL 1	SIL 2	SIL 3	SIL 4
1	Test case execution from cause consequence diagrams	B.6.6.2	---	---	R	R
2	Test case execution from model-based test case generation	C.5.27	R	R	HR	HR
3	Prototyping/animation	C.5.17	---	---	R	R
4	Equivalence classes and input partition testing, including boundary value analysis	C.5.7 C.5.4	R	HR	HR	HR
5	Process simulation	C.5.18	R	R	R	R

NOTE 1 The analysis for the test cases is at the software system level and is based on the specification only.

NOTE 2 The completeness of the simulation will depend upon the safety integrity level, complexity and application.

NOTE 3 See Table C.13.

NOTE 4 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level.

Table B.4 – Failure analysis

(Referenced by Table A.10)

Technique/Measure *		Ref	SIL 1	SIL 2	SIL 3	SIL 4
1a	Cause consequence diagrams	B.6.6.2	R	R	R	R
1b	Event tree analysis	B.6.6.3	R	R	R	R
2	Fault tree analysis	B.6.6.5	R	R	R	R
3	Software functional failure analysis	B.6.6.4	R	R	R	R

NOTE 1 Preliminary hazard analysis should have already taken place in order to categorize the software into the most appropriate safety integrity level.

NOTE 2 See Table C.14.

NOTE 3 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

Table B.5 – Modelling

(referenced by Table A.7)

Technique/Measure *		Ref	SIL 1	SIL 2	SIL 3	SIL 4
1	Data flow diagrams	C.2.2	R	R	R	R
2a	Finite state machines	B.2.3.2	---	R	HR	HR
2b	Formal methods	B.2.2, C.2.4	---	R	R	HR
2c	Time Petri nets	B.2.3.3	---	R	HR	HR
3	Performance modelling	C.5.20	R	HR	HR	HR
4	Prototyping/animation	C.5.17	R	R	R	R
5	Structure diagrams	C.2.3	R	R	R	HR

NOTE 1 If a specific technique is not listed in the table, it should not be assumed that it is excluded from consideration. It should conform to this standard.

NOTE 2 Quantification of probabilities is not required.

NOTE 3 See Table C.15.

NOTE 4 The references (which are informative, not normative) “B.x.x.x”, “C.x.x.x” in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

Table B.6 – Performance testing

(referenced by Tables A.5 and A.6)

Technique/Measure *		Ref	SIL 1	SIL 2	SIL 3	SIL 4
1	Avalanche/stress testing	C.5.21	R	R	HR	HR
2	Response timings and memory constraints	C.5.22	HR	HR	HR	HR
3	Performance requirements	C.5.19	HR	HR	HR	HR
NOTE 1 See Table C.16. NOTE 2 The references (which are informative, not normative) "B.x.x.x", "C.x.x.x" in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.						
* Appropriate techniques/measures shall be selected according to the safety integrity level.						

Table B.7 – Semi-formal methods

(Referenced by Tables A.1, A.2 and A.4)

Technique/Measure *		Ref	SIL 1	SIL 2	SIL 3	SIL 4
1	Logic/function block diagrams	See Note 1	R	R	HR	HR
2	Sequence diagrams	see Note 1	R	R	HR	HR
3	Data flow diagrams	C.2.2	R	R	R	R
4a	Finite state machines/state transition diagrams	B.2.3.2	R	R	HR	HR
4b	Time Petri nets	B.2.3.3	R	R	HR	HR
5	Entity-relationship-attribute data models	B.2.4.4	R	R	R	R
6	Message sequence charts	C.2.14	R	R	R	R
7	Decision/truth tables	C.6.1	R	R	HR	HR
8	UML	C.3.12	R	R	R	R
NOTE 1 Logic/function block diagrams and sequence diagrams are described in IEC 61131-3. NOTE 2 See Table C.17. NOTE 3 The references "B.x.x.x", "C.x.x.x" in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.						
* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.						

Table B.8 – Static analysis

(Referenced by Table A.9)

Technique/Measure *		Ref	SIL 1	SIL 2	SIL 3	SIL 4
1	Boundary value analysis	C.5.4	R	R	HR	HR
2	Checklists	B.2.5	R	R	R	R
3	Control flow analysis	C.5.9	R	HR	HR	HR
4	Data flow analysis	C.5.10	R	HR	HR	HR
5	Error guessing	C.5.5	R	R	R	R
6a	Formal inspections, including specific criteria	C.5.14	R	R	HR	HR
6b	Walk-through (software)	C.5.15	R	R	R	R
7	Symbolic execution	C.5.11	---	---	R	R
8	Design review	C.5.16	HR	HR	HR	HR
9	Static analysis of run time error behaviour	B.2.2, C.2.4	R	R	R	HR
10	Worst-case execution time analysis	C.5.20	R	R	R	R

NOTE 1 See Table C.18.

NOTE 2 The references "B.x.x.x", "C.x.x.x" in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

Table B.9 – Modular approach

(Referenced by Table A.4)

Technique/Measure *		Ref	SIL 1	SIL 2	SIL 3	SIL 4
1	Software module size limit	C.2.9	HR	HR	HR	HR
2	Software complexity control	C.5.13	R	R	HR	HR
3	Information hiding/encapsulation	C.2.8	R	HR	HR	HR
4	Parameter number limit / fixed number of subprogram parameters	C.2.9	R	R	R	R
5	One entry/one exit point in subroutines and functions	C.2.9	HR	HR	HR	HR
6	Fully defined interface	C.2.9	HR	HR	HR	HR

NOTE 1 See Table C.19.

NOTE 2 The references "B.x.x.x", "C.x.x.x" in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. No single technique is likely to be sufficient. All appropriate techniques shall be considered.

Annex C
(informative)

Properties for software systematic capability

C.1 Introduction

Given the large number of factors that affect software systematic capability it is not possible to give an algorithm for combining the techniques and measures that will be correct for any given application. The purpose of Annex C is:

– to give guidance on selecting specific techniques from Annexes A and B to achieve software systematic capability;

– to outline a rationale for justifying the use of techniques that are not explicitly listed in Annexes A and B.

Annex C is supplementary to Annexes A and B tables.

C.1.1 Structure of Annex C, relating to Annexes A and B

The outputs from each phase of the software safety lifecycle are defined in Table 1. For example, consider the software safety requirements specification.

Table A.1 (“Software safety requirements specification”) of Annex A recommends specific techniques for developing the software safety requirements specification.

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1a	Semi-formal methods	Table B.7	R	R	HR	HR
1b	Formal methods	B.2.2, C.2.4	---	R	R	HR
2	Forward traceability between the system safety requirements and the software safety requirements	C.2.11	R	R	HR	HR
3	Backward traceability between the safety requirements and the perceived safety needs	C.2.11	R	R	HR	HR
4	Computer-aided specification tools to support appropriate techniques/measures above	B.2.4	R	R	HR	HR

Annex C Table C.1 (“Properties for systematic safety integrity – Software safety requirements specification”) states that the software safety requirements specification is characterized by the following desirable properties (which are informally defined in Annex F of IEC 61508-7):

Properties					
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

Annex C Table C.1 also ranks on an informal scale R1/R2/R3 the effectiveness of specific techniques in achieving these desirable properties.

Technique/ Measure		Properties					
		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
1a	Semi-formal methods	R1 Application-friendly or domain specific specification method and notation used by domain experts	R1 Application-friendly or domain specific specification method and notation used by domain experts R2 Verification of specification according to coverage criteria	R1 Method and notation that helps avoid or detect internal inconsistency, missing behaviour or mathematically inconsistent expressions. R2 Verification of specification according to coverage criteria R3 Verification of specification based on systematic analysis, and / or systematic avoidance of particular types of intrinsic specification faults	R1 Defined notation that restricts opportunity for misunderstanding R2 Application of complexity limits in specification	_	R2 Defined notation that reduces ambiguity in specification

The confidence that can be placed in the software safety requirements specification as a basis for safe software depends on the rigour of the techniques by which the desirable properties of the software safety requirements specification have been achieved. The rigour of a technique is informally ranked on a scale R1 to R3, where R1 is the least rigorous and R3 the most rigorous.

R1	without objective acceptance criteria, or with limited objective acceptance criteria. E.g., black-box testing based on judgement, field trials.
R2	with objective acceptance criteria that can give a high level of confidence that the required property is achieved (exceptions to be identified & justified); e.g., test or analysis techniques with coverage metrics, coverage of checklists.
R3	with objective, systematic reasoning that the required property is achieved. E.g. formal proof, demonstrated adherence to architectural constraints that guarantee the property.
_	this technique is not relevant to this property.

A technique may achieve one of several R1/R2/R3 rankings relating to a particular property, depending on the level of rigour that the technique satisfies.

Technique/ Measure		Properties					
		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
1a	Semi-formal methods				R1 Defined notation that restricts opportunity for misunderstanding R2 Application of complexity limits in specification		

In this example, a semi-formal method achieves rigour R1 by providing a restricted notation that improves accurate expression, and achieves R2 by further restricting the complexity of specification which might otherwise cause confusion.

C.1.2 Method of use – 1

For guidance purposes, if it can be convincingly demonstrated that the desirable properties have been achieved in the development of the software safety requirements specification, then confidence is justified that the software safety requirements specification is an adequate basis for developing software that has sufficient systematic safety integrity.

Annex C Table C.1 says that each of the Annex A Table A.1 techniques typically achieves, to a greater or lesser extent, one or more of the above Table C.1 properties that are relevant to the software safety requirements specification.

However, it is important to note that although Annex A Table A.1 recommends specific techniques, these recommendations are not prescriptive, and in fact Annex A states clearly that “Given the large number of factors that affect software systematic capability it is not possible to give an algorithm for combining the techniques and measures that will be correct for any given application”.

In practice the techniques by which the software safety requirements specification is developed are selected subject to several practical constraints (see 7.1.2.7) in addition to the inherent capabilities of the techniques. Such constraints may include:

– 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.

Table C.1 may be used to compare the relative effectiveness of the specific Annex A Table A.1 techniques in achieving the desirable properties of the software safety requirements specification lifecycle, while at the same time factoring in the practical constraints of the particular development project.

For example, a formal method is capable of giving a better basis (R3) for verification and validation than is a semi-formal method (R2), but other project constraints (e.g. the availability of sophisticated computer support tools, or the very specialized expressiveness of a formal notation) may favour a semi-formal approach.

In this way, the Table C.1 desirable properties can provide the basis of a reasoned and practical comparison of the alternative techniques that Annex A Table A.1 recommends for developing the software safety requirements specification. Or more generally, a reasoned selection from the several alternative techniques recommended by Annex A for a particular lifecycle phase can be made by considering the desirable properties listed in the corresponding Annex C table.

But note carefully that due to the nature of systematic behaviour, these Annex C properties may not be achievable or demonstrable with the highest rigour. Rather, they are goals to be aimed for. Their achievement may even necessitate trade-offs between different properties e.g. between defensive design and simplicity.

Finally, in addition to defining R1/R2/R3 criteria, it is useful for guidance purposes to make an informal link between (1) the increasing level of rigour of the R1 to R3 progression and (2) an increased confidence in the correctness of the software. As a general and informal recommendation, the following minimum levels of rigour should be aimed for when Annex A requires the corresponding SIL performance:

SIL	Rigour R
1 / 2	R1
3	R2 where available
4	highest rigour available

C.1.3 Method of use – 2

Although Annex A recommends specific techniques, it is also permitted to apply other measures and techniques, providing that the requirements and objectives of the lifecycle phase have been met.

It has already been noted that many factors affect software systematic capability, and it is not possible to give an algorithm for selecting and combining the techniques in a way that is guaranteed in any given application to achieve the desirable properties.

There may be several effective ways to achieve the desirable properties, and it should be recognized that system developers may be able to provide alternative evidence. The information in these Annex C tables can be used as the basis of a reasoned argument to justify the selection of techniques other than those given in the Annex A tables.

C.2 Properties for systematic safety integrity

The guidance here and in IEC 61508-7 indicates specific techniques for achieving the systematic safety integrity properties and for generating convincing evidence. Where a method does not contribute to the achievement of a property, this is shown in the following tables by a dash. Where a method may have adverse effects on some properties and positive effects on others, a note is provided in the relevant table below.

Table C.1 – Properties for systematic safety integrity – Software safety requirements specification

(See 7.2. Referenced by Table A.1)

Technique/Measure		Properties					
		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**
1a	Semi-formal methods	R1 Application-friendly or domain specific specification method and notation used by domain experts	R1 Application-friendly or domain specific specification method and notation used by domain experts R2 Verification of specification according to coverage criteria	R1 Method and notation that helps avoid or detect internal inconsistency, missing behaviour or mathematically inconsistent expressions. R2 Verification of specification according to coverage criteria R3 Verification of specification based on systematic analysis, and / or systematic avoidance of particular types of intrinsic specification faults	R1 Defined notation that restricts opportunity for misunderstanding R2 Application of complexity limits in specification	–	R2 Defined notation that reduces ambiguity in specification

Technique/Measure		Properties					
		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**
1b	Formal methods	R1 Application-friendly or domain specific specification method and notation used by domain experts	R1 Application-friendly or domain specific specification method and notation used by domain experts. R2 Verification of specification according to coverage criteria R3 Guarantee of correctness on limited aspects of behaviour	R1 Method and notation that help avoid or detect internal inconsistency, missing behaviour or mathematically inconsistent expressions. R2 Verification of specification according to coverage criteria R3 Verification of specification based on systematic analysis, and / or Systematic avoidance of particular types of intrinsic specification faults	– Note: May complicate the achievement of this property if the method is not application-friendly or domain specific.	–	R3 Reduces ambiguity in specification.
2	Forward traceability between the system safety requirements specification and the software safety requirements	R1 Confidence that the Software safety requirements specification addresses the system safety requirements	–	–	–	–	–
3	Backward traceability between the software safety requirements specification and the perceived safety needs	–	R1 Confidence that the software safety requirements specification contains no unnecessary complexity	–	RI Traceability to the EUC safety needs enhances understandability	R1	R1

Technique/Measure		Properties					
		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**
4	Computer-aided specification tools to support appropriate techniques/measures above	R1 Encapsulation of domain knowledge of the EUC and of the software environment R2 If checklist of issues to be taken into consideration is defined, justified and covered	R1 Functional simulation techniques R2 Functional simulation according to defined and justified coverage criteria	R2 Semantic and syntactic checks to ensure that the relevant rules are satisfied	R1 Animation of, or browsing through the specification	R1 Identification of safety and non-safety functions	R1 Assists traceability and coverage R2 Measurement of traceability and coverage

Table C.2 – Properties for systematic safety integrity – Software design and development – software Architecture Design

(See 7.4.3. Referenced by Table A.2)

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance**	**Defence against common cause failure from external events**
1	Fault detection	_	_	_	_ May complicate the achievement of this property	R1 Logical program flow monitoring provides for predictability	_	R1 (R2 if coverage targets are defined, justified and met)	R1 or _
2	Error detecting codes	_	_	_	_ May complicate the achievement of this property.	_ May complicate the achievement of this property.	_	R1 (R2 if coverage targets are defined, justified and met) Effective for specific application areas e.g. data comms	R1 Effective for specific application areas e.g. data comms
3a	Failure assertion programming	_	R2 Post-assertions may check compliance with detailed requirements	_	R2 Pre-assertions limit the input space	R2 Post-assertions check for expected / acceptable outputs	R2 Pre-assertions limit the input space and hence the required test space	R3 Effective for the targeted failures	R3 Effective for the targeted failures
3b	Diverse monitor techniques (with independence between the monitor and the monitored function in the same computer)	_	_	R2 Diverse monitor implements only the minimum safety requirements	R2 Diverse monitor provides for implicit diversity	R2 Diverse monitor implements in a simple manner only the minimum safety requirements	R2 Diverse monitor implements only the minimum safety requirements	R1 (R2 if coverage targets are defined, justified and met)	R1 (R2 if coverage targets are defined, justified and met)

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance**	**Defence against common cause failure from external events**
3c	Diverse monitor techniques (with separation between the monitor computer and the monitored computer)	-	_	R2 Diverse monitor implements only the minimum safety requirements	R2 Diverse monitor provides for implicit diversity	R2 Diverse monitor implements in a simple manner only the minimum safety requirements)	R2 Diverse monitor implements only the minimum safety requirements	R1 (R2 if coverage targets are defined, justified and met)	R1 (R2 if coverage targets are defined, justified and met)
3d	Diverse redundancy, implementing the same software safety requirements specification	_	_	_	_ Note: May complicate the achievement of this property if done within the same executable software.	_	_	R1 If the failure of one program does not adversely affect the others R2 If coverage targets are defined, justified and met Does not protect against requirements specification faults	R1 If the failure of one program does not adversely affect the others R2 If coverage targets are defined, justified and met Does not protect against requirements specification faults

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance**	**Defence against common cause failure from external events**
3e	Functionally diverse redundancy, implementing different software safety requirements specification. This will typically require sensors operating on different physical principles	–	–	R1	– Note: May complicate the achievement of this property if done within the same executable software.	–	–	R1 If the failure of one program does not adversely affect the others.	R1 If the failure of one program does not adversely affect the others. Protects against specification faults
3f	Backward recovery	–	–	– Note: May complicate the achievement of this property.	–	– Note: May complicate the achievement of this property.	–	R2	R1 (R2 if coverage targets are defined, justified and met)
3g	Stateless design (or limited state design)	R2 Provided safety requirements are also stateless or limited state	R2 Provided safety requirements are also stateless or limited state	R2 Provided safety requirements are also stateless or limited state	R1 R2 If limits are defined, justified and met regarding the possible number of states	R1 R2 If limits are defined, justified and met regarding the possible number of states	R1 R2 If targets are defined, justified and met for the verification / test coverage of the possible states	R1 If this leads to a self-healing design R2 If targets are defined, justified and met for self-healing	R1 If this leads to a self-healing design R2 If targets are defined, justified and met for self-healing
4a	Re-try fault recovery mechanisms	–	–	–	–	May complicate the achievement of this property	–	R1 (R2 if coverage targets are defined, justified and met)	R1 (R2 if coverage targets are defined, justified and met)

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance**	**Defence against common cause failure from external events**
4b	Graceful degradation	–	–	– Note: May complicate the achievement of this property.	–	–	–	R1 R2 if coverage targets are defined, justified and met	R1 R2 if coverage targets are defined, justified and met
5	Artificial intelligence - fault correction	–	– Note: May complicate the achievement of this property.	– Note: May complicate the achievement of this property.	– Note: May complicate the achievement of this property.	– Note: May complicate the achievement of this property.	– Note: May complicate the achievement of this property.	–	–
6	Dynamic reconfiguration	–	– Note: May complicate the achievement of this property.	– Note: May complicate the achievement of this property.	– Note: May complicate the achievement of this property.	– Note: May complicate the achievement of this property.	– Note: May complicate the achievement of this property.	–	–

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance**	**Defence against common cause failure from external events**
7	Modular approach	_	R1 R2 R2 is achieved If modularity targets are defined, justified and met. Otherwise, only R1 is achieved.	R1 If freedom from particular types of intrinsic design faults can be verified independently for each module R3 If freedom from particular types of intrinsic design faults can be supported by a rigorous reasoning based on modular design	R1 R2 If modularity targets are defined, justified and met	R1 R2 If modularity targets are defined, justified and met	R1 R2 If modularity targets are defined, justified and met	R1 If modules not affected by the failure of a module contribute in mitigation / recovery R3 If tolerance to particular faults can be supported by a rigorous reasoning	R1 If modules that can be influenced by external events that can affect multiple channels concurrently, are identified and subject to thorough verification R3 If tolerance to particular external events can be supported by a rigorous reasoning

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance**	**Defence against common cause failure from external events**
8	Use of trusted/verified software modules and elements (if available)	–	R1 R2 R3 If the element significantly contributes to particular safety requirements, and is correctly used	R1 R2 R3 Re-uses proven elements. Such capability shall be justified for the element	R1 Modular approach decomposes overall complexity into understandable units	R1 R2 R3 Reuses proven elements	–	R1R2 R2 If fault tolerance capabilities are readily provided by the element and are correctly used, or if a fault tolerance layer is built around the element	R1R2 R2 If defences against external events that could affect concurrently multiple channels are readily provided by the element and are correctly used, or if a defensive layer is built around the element
9	Forward traceability between the software safety requirements specification and software architecture	R1 Confidence that the architecture addresses the software safety requirements	–	–	–	–	–	–	–
10	Backward traceability between the software architecture and the software safety requirements specification	–	R1 Confidence that the architecture contains no unnecessary complexity	–	–	–	–	–	–
11a	Structured diagrammatic methods	–	R1	–	R1 (Graphical descriptions are easier to understand)	–	R1 (Structured designs are easier to verify and test)	–	–

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance**	**Defence against common cause failure from external events**
11b	Semi-formal methods	R1 An application-friendly or domain specific specification method and notation	R1 An application-friendly or domain specific specification method and notation	R2 Can detect internal inconsistency or missing behaviour or mathematically inconsistent expressions	–	R2 (Provides evidence for predictability)	R2 (Provides evidence for inner consistence of the design model)	–	–
11c	Formal design and refinement methods	R1 An application-friendly or domain specific specification method and notation	R1 Provides precise definition of limited aspects of behaviour which needs to be appropriate to the domain	R3 Can detect internal inconsistency or missing behaviour or mathematically inconsistent expressions	– Note: May complicate the achievement of this property.	R2 Provides proof for predictability	R2	–	–
11d	Automatic software generation	R1 If executable software is automatically generated from requirements specification, or from a design that has been shown to be complete R2 If the generation tools are shown to have appropriate pedigree	R1 If executable software is automatically generated from requirements specification, or from a design that has been shown to be correct R2 If the generation tools are shown to have appropriate pedigree	R1 If the generation tools guarantee avoidance of particular intrinsic design faults R2 If the generation tools are shown to have appropriate pedigree	–	–	–	R1 R2 R3 If fault tolerance capabilities are automatically generated	–

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance**	**Defence against common cause failure from external events**
12	Computer-aided specification and design tools	R1 Encapsulation of domain knowledge of the EUC and of the software environment R2 If checklist of issues to be taken into consideration are defined, justified and covered	R1 Enforcement of backward requirements traceability Functional simulation techniques R2 Functional simulation according to defined and justified coverage criteria	R2 Semantic and syntactic checks to ensure that the relevant rules are satisfied	R1 Animation and browsing	–	R2 Semantic and syntactic checks to ensure that the relevant rules are satisfied	–	–
13a	Cyclic behaviour, with guaranteed maximum cycle time	–	R1 for timing aspects of specification R3 If maximum cycle time established by rigorous reasoning	R1 for timing aspects of specification R3 If maximum cycle time established by rigorous reasoning	–	R1 for timing aspects of specification R3 If maximum cycle time established by rigorous reasoning	R1 for timing aspects of specification R3 If maximum cycle time established by rigorous reasoning	–	–

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance**	**Defence against common cause failure from external events**
13b	Time-triggered architecture	R3 Completeness is guaranteed by allocation (only for timing properties)	R3 Correctness is guaranteed by allocation (only for timing properties)	R3 Rigorous guarantee against intrinsic timing faults	R1 Defined notation reduces misunderstanding considerably, predictability as approach	R3 Adverse interference: total separation in time domain, no interference	R3 Greatly reduces the effort required for testing and certifying the system	R2 Transparent implementation of fault-tolerance	R3 External interrupts cannot interfere with the time-triggered schedule which gives priority to safety-critical tasks
13c	Event-driven, with guaranteed maximum response time	–	–	–	R1 Event driven architectures may hinder understandibility	R2 Event driven architectures may hinder understandibilit y	R1 Makes testing more predictable	–	–
14	Static resource allocation	R1	R1	R1	R1 Makes the design more understandable	R2 With architecture defining resource usage	R1 Makes testing more predictable	–	–
15	Static synchronisation of access to shared resources	–	R1 Gives predictability in resource access	R1 R3 if supported by rigorous reasoning as to correctness of synchronisation	R1 Makes the design more understandable	R1 R3 if supported by rigorous reasoning as to correctness of synchronisation	–	–	–

Table C.3 – Properties for systematic safety integrity – Software design and development – support tools and programming language

(See 7.4.4. Referenced by Table A.3)

Technique/Measure		Properties		
		Support the production of software with the required software properties	**Clarity of the operation and functionality of the tool**	**Correctness and repeatability of output**
1	Suitable programming language	R2 if strong typing, restricted type conversion. R3 if defined semantics for rigorous reasoning	–	–
2	Strongly typed programming language	R2	–	–
3	Language subset	R2 Depending on chosen subset	R1	R2 Depending on chosen subset
4a	Certificated tools	–	R2	R2
4b	Tools: increased confidence from use	R1 If the class of detected program errors is systematically defined R2 If there is objective validation evidence for the tool performance.	R1 If the tool support is non-specific to the problem domain. R2 If the tool support is significantly specialized to the problem domain.	R1 R2 If there is objective validation evidence for the tool performance e.g. a complier validation suite.

Table C.4 – Properties for systematic safety integrity – Software design and development – detailed design (includes software system design, software module design and coding)

(See 7.4.5 and 7.4.6. Referenced by Table A.4)

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance / Fault detection**	**Freedom from common cause failure**
1a	Structured methods	R2	R1	R1	–	–	R1 Structured designs are more readily verifiable and testable	–	–
1b	Semi-formal methods	R2	R2	R2	–	R2	R2	–	–
1c	Formal design and refinement methods	–	R3	R3	– Note: May complicate the achievement of this property.	R3 Provides evidence for predictability	R2	–	–
2	Computer-aided design tools	R2 Dependent upon the Computer aided specification tool applying semantic and syntactic checks to ensure that the relevant rules are satisfied	R1	R2 Dependent upon the computer aided specification tool applying semantic and syntactic checks to ensure that the relevant rules are satisfied	–	–	R2 Dependant upon CASE tool to support test coverage and static verification	–	–

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance / Fault detection**	**Freedom from common cause failure**
3	Defensive programming	–	–	–	– Note: May complicate the achievement of this property.	– Note: May complicate the achievement of this property.	–	R1 (R2 if coverage targets are defined, justified and met)	R1 (R2 if coverage targets are defined, justified and met)
4	Modular approach	–	–	R1	R1	R1	R1	–	–
5	Design and coding standards	–	–	R1	R1	R1	R1	–	–
6	Structured programming	–	R1	R1	R1	R1	R1	–	–
7	Use of trusted/verified software modules and elements (if available)	–	–	R1 Reuses proven elements	R1 Modular approach decomposes overall complexity into understandable units	R1 The behaviour of the element is already known	–	–	–
8	Forward traceability between the software safety requirements specification and software design	R1 Confidence that the design addresses the software safety requirements	–	–	–	–	–	–	–

Table C.5 – Properties for systematic safety integrity – Software design and development – software module testing and integration

(See 7.4.7 and 7.4.8. Referenced by Table A.5)

Technique/Measure		Properties			
		Completeness of testing and integration with respect to the software design specification	**Correctness of testing and integration with respect to the software design specification (successful completion)**	**Repeatability**	**Precisely defined testing configuration**
1	Probabilistic testing	R1 (R2 if operational profile coverage targets are defined, justified and met)	R1 (R2 if required outputs are defined, justified and met)	–	–
2	Dynamic analysis and testing	R1 (R2 if structural coverage targets are defined, justified and met)	R1 (R2 if required outputs are defined, justified and met)	–	–
3	Data recording and analysis	–	R1	R1 Promotes consistency in testing procedures	R2 If fault records/test logs include details of software baseline
4	Functional and black box testing	R1 (R2 if operational profile coverage targets are defined, justified and met)	R1 (R2 if required outputs are defined, justified and met)	–	–
5	Performance testing	–	R1 (R2 if required outputs are defined, justified and met)	–	–

Technique/Measure		Properties			
		Completeness of testing and integration with respect to the software design specification	**Correctness of testing and integration with respect to the software design specification (successful completion)**	**Repeatability**	**Precisely defined testing configuration**
6	Model based testing (MBT)	R2 MBT allows early exposure of ambiguities in specification and design, the MBT process starts with requirements R3 If rigorous reasoning is applied to modelling, and test case generation (TCG) is used	R2 Evaluation of results and regression test suites is a key benefit of MBT R3 If rigorous modelling approach is applied, then objective evidence of coverage is possible	R3 MBT (with TCG) aims at automatic execution of generated tests	R2 MBT is automated, testing configuration has to be precisely defined; execution of the generated tests is similar to black box testing with the possibility to be combined with source code level coverage measurement
7	Interface testing	–	R1 (R2 if required outputs are defined, justified and met)	–	–
8	Test management and automation tools	R1 (R2 if test coverage targets are defined, justified and met)	–	R1 Automation promotes consistency	R2 Gives repeatability of testing
9	Forward traceability between the software safety requirements specification and the module and integration test specifications	R1 Confidence that the test specification addresses the software safety requirements	–	–	R2 Confidence in a clear baseline of requirements under test
10	Formal verification	R3 If rigorous reasoning is applied to construction of test cases to show that all aspects of design have been exercised	R3 Gives objective evidence of meeting all of the software safety requirements	R1 If support tools unavailable R2 If tool supported	–

Table C.6 – Properties for systematic safety integrity – Programmable electronics integration (hardware and software)

(See 7.5. Referenced by Table A.6)

	Technique/Measure	Properties			
		Completeness of integration with respect to the design specifications	**Correctness of integration with respect to the design specifications (successful completion)**	**Repeatability**	**Precisely defined integration configuration**
1	Functional and black box testing	R1 (R2 if operational profile coverage targets are defined, justified and met)	R1 (R2 if required outputs are defined, justified and met)	–	–
2	Performance testing	–	R1 (R2 if required outputs are defined, justified and met)	–	–
3	Forward traceability between the system and software design requirements for hardware/software integration and the hardware/software integration test specifications	R1 Confidence that the hardware/software integration test specifications addresses the integration requirements	–	–	R2 Confidence with a clear baseline of requirements under test

Table C.7 – Properties for systematic safety integrity – Software aspects of system safety validation

(See 7.7. Referenced by Table A.7)

Technique/Measure		Properties			
		Completeness of validation with respect to the software Design Specification	**Correctness of validation with respect to the software Design Specification (successful completion)**	**Repeatability**	**Precisely defined validation configuration**
1	Probabilistic testing	R1 (R2 if operational profile coverage targets are defined, justified and met)	R1 (R2 if required outputs are defined, justified and met)	–	–
2	Process simulation	R1	R1 (R2 if required outputs are defined, justified and met)	–	R2 Gives a definition of the external environment
3	Functional and black-box testing	R1 (R2 if operational profile coverage targets are defined, justified and met)	R1 (R2 if required outputs are defined, justified and met)	–	–
4	Forward traceability between the software safety requirements specification and the software safety validation plan	R1 Confidence that the software safety validation plan addresses the software safety requirements	–	–	R2 Confidence with a clear baseline of requirements under test
5	Backward traceability between the software safety validation plan and the software safety requirements specification	–	R1 Confidence that software safety validation plan contains no unnecessary complexity	–	R2 Confidence with a clear baseline of requirements under test

Table C.8 – Properties for systematic safety integrity – Software modification

(See 7.8. Referenced by Table A.8)

Technique/Measure		Properties					
		Completeness of modification with respect to its requirements	**Correctness of modification with respect to its requirements**	**Freedom from introduction of intrinsic design faults**	**Avoidance of unwanted behaviour**	**Verifiable and testable design**	**Regression testing and verification coverage**
1	Impact analysis	_	_	_	R1	R1	R1
2	Re-verify changed software module	R1 (R2 if objective verification targets)	R1 (R2 if objective verification targets)	R1 (R2 if objective verification targets)	_	_	R1R2 (R2 if objective verification targets)
3	Re-verify affected software modules	R1 (R2 if objective verification targets)	R1 (R2 if objective verification targets)	R1 (R2 if objective verification targets)	_	_	R1 (R2 if objective verification targets)
4a	Revalidate complete system	R1 (R2 if objective verification targets)	R1 (R2 if objective verification targets)	_	R1 (R2 if objective verification targets)	_	R1 (R2 if objective verification targets)
4b	Regression validation	R1 (R2 if objective verification targets)	R1 (R2 if objective verification targets)	_	R1 (R2 if objective verification targets)	_	R1 (R2 if objective verification targets)
5	Software configuration management	_	_	_	_	_	R1
6	Data recording and analysis	R1	R1	_	_	_	_

Technique/Measure		Properties					
		Completeness of modification with respect to its requirements	**Correctness of modification with respect to its requirements**	**Freedom from introduction of intrinsic design faults**	**Avoidance of unwanted behaviour**	**Verifiable and testable design**	**Regression testing and verification coverage**
7	Forward traceability between the software safety requirements specification and the software modification plan (including reverification and revalidation)	R1 Confidence that the software modification plan (including re-verification and revalidation) addresses the software safety requirements	–	–	–	–	–
8	Backward traceability between the software modification plan (including reverification and revalidation) and the software safety requirements specification	–	R1 Confidence that software modification plan (including re-verification and revalidation) contains no unnecessary complexity	–	–	–	–

Table C.9 – Properties for systematic safety integrity – Software verification

(See 7.9. Referenced by Table A.9)

Technique/Measure		Properties			
		Completeness of verification with respect to the previous phase	**Correctness of verification with respect to the previous phase (successful completion)**	**Repeatability**	**Precisely defined verification configuration**
1	Formal proof	–	R3	–	–
2	Animation of specification and design	R1	R1	–	–
3	Static analysis	–	R1/R2/R3 (Rigour may range from language subset enforcement to mathematical formal analysis)	–	–
4	Dynamic analysis and testing	R1 (R2 if structural coverage targets are defined, justified and met)	R1 (R2 if required outputs are defined, justified and met)	–	–
5	Forward traceability between the software Design Specification and the software verification (including data verification) plan.	R1 Confidence that the software verification (including data verification) plan addresses the software safety requirements	–	–	R2 Confidence with a clear baseline of requirements under test
6	Backward traceability between the software verification (including data verification) plan and the software design specification	–	R1 Confidence that software verification (including data verification) plan contains no unnecessary complexity	–	R2 Confidence with a clear baseline of requirements under test
7	Offline numerical analysis	–	R1 Increased confidence in the expected numerical accuracy of well-conditioned calculations (R2 with objective acceptance criteria. R3 if used in conjunction with objective systematic reasoning to justify the acceptance criteria)	–	–

Table C.10 – Properties for systematic safety integrity – Functional safety assessment

(See Clause 8. Referenced by Table A.10)

Technique/Measure		Properties						
		Completeness of functional safety assessment with respect to this standard	**Correctness of functional safety assessment with respect to the design specifications (successful completion)**	**Traceable closure of all identified issues**	**The ability to modify the functional safety assessment after change without the need for extensive re-work of the assessment**	**Repeatability**	**Timeliness**	**Precisely defined configuration**
1	Checklists	R1	R1	R1	–	R1	–	–
2	Decision/truth tables	R1	R2	–	–	R2	–	–
3	Failure analysis	R2	R2	R1 (The failure analysis is based on agreed failure lists)	–	R1 (The failure analysis is based on agreed failure lists)	–	–
4	Common cause failure analysis of diverse software (if diverse software is actually used)	R2	R2	R1 (Provided the CCF analysis is based on agreed CC initiator lists)	–	R1 (Provided the CCF analysis is based on agreed CC initiator lists)	–	–
5	Reliability block diagram	R1	R1	–	–	–	–	–
6	Forward traceability between the requirements of IEC 61508-3 Clause 8 and the plan for software functional safety assessment	R1 Confidence that the plan for software functional safety assessment addresses the requirements of 61508-3 Clause 8	–	–	–	–	–	–

C.3 Properties for systematic safety integrity – Detailed tables

Table C.11 – Detailed properties – Design and coding standards

(Referenced by Table B.1)

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance / Fault detection**	**Freedom from common cause failure**
1	Use of coding standard to reduce likelihood of errors	–	–	R1	R1 Eliminates selected language constructs	R1	R1	–	–
2	No dynamic objects	–	–	R1/R2/R3 Depending on language used	–	R1/R2/R3 Depending on language used	R1/R2 Depending on language used	–	–
3a	No dynamic variables	–	–	R1/R2/R3 Depending on language used	–	R1/R2/R3 Depending on language used	R1/R2 Depending on language used	–	–
3b	Online checking of the installation of dynamic variables	–	–	R1/R2/R3 Depending on language used	–	R1/R2/R3 Depending on language used	R1/R2 Depending on language used	–	–
4	Limited use of interrupts	–	–	R1/R2 Depending on language used	R1 Increases clarity of logic and event sequences	R1/R2 Depending on language used	R1/R2 Depending on language used	–	–
5	Limited use of pointers	–	–	R1/R2 Depending on language used	R1 Increases clarity of logic	R1/R2 Depending on language used	R1/R2 Depending on language used	–	–
6	Limited use of recursion	–	–	R1/R2 Depending on language used	–	R1/R2 Depending on language used	R1/R2 Depending on language used	–	–

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance / Fault detection**	**Freedom from common cause failure**
7	No unstructured control flow in programs in higher level languages	–	–	R1/R2 Depending on language used	R1 Increases clarity of logic	R1/R2 Depending on language used	R1/R2 Depending on language used	–	–
8	No automatic type conversion	–	R2 Prevents rounding errors	R2 Prevents rounding errors	R1	R1	–	–	–

Table C.12 – Detailed properties – Dynamic analysis and testing

(Referenced by Table B.2)

Technique/Measure		Properties			
		Completeness of testing and verification with respect to the software design specifications	**Correctness of testing and verification with respect to the software design specifications (successful completion)**	**Repeatability**	**Precisely defined testing and verification configuration**
1	Test case execution from boundary value analysis	–	R1 (R2 if objective criteria for boundary results)	–	–
2	Test case execution from error guessing	–	R1	–	–
3	Test case execution from error seeding	–	R1	–	–
4	Test case execution from model-based test case generation	R2 The MBT process starts with requirements and facilitates early finding of errors during software design and development R3 If rigorous reasoning is applied to modelling, and TCG (Test Case Generation) is used	R2 Evaluation of results and regression test suites is a key benefit of MBT, it further facilitates understanding of consequences of specified requirements R3 If rigorous modelling approach is applied, then objective evidence of coverage is possible	R3 MBT (with TCG) aims at automatic execution of generated tests	R2 MBT is automated, testing configuration has to be precisely defined; execution of the generated tests is similar to black box testing with the possibility to be combined with source code level coverage measurement
5	Performance modelling	–	R1 (R2 if objective performance requirements)	–	–
6	Equivalence classes and input partition testing	R1 (If the input data profile is well defined and is manageably simple in structure)	R1 (If the partitions plausibly contain no non-linearities i.e. all members of a class are truly equivalent)	–	–
7	Structure-based testing	–	R1 (R2 is objective structural coverage targets)	–	–

Table C.13 – Detailed properties – Functional and black-box testing

(Referenced by Table B.3)

Technique/Measure		Properties			
		Completeness of testing, integration and validation with respect to the design specifications	**Correctness of testing, integration and validation with respect to the design specifications (successful completion)**	**Repeatability**	**Precisely defined testing, integration and validation configuration**
1	Test case execution from cause consequence diagrams	R1	R1	–	–
2	Test case execution from model-based test case generation	R2 MBT Model-based Testing is the automatic generation of efficient test cases/procedures using models of system requirements and specified functionality, it facilitates early error disclosure and understanding of consequences of specified requirements R3 If rigorous reasoning is applied to modelling, and TCG is used	R2 MBT is based on system models derived from (mainly functional/behavioural) requirements. R3 If rigorous modelling approach is applied, then objective evidence of coverage is possible	R3 MBT (with TCG) aims at automatic execution of generated tests	R2 MBT is automated, testing configuration has to be precisely defined
3	Prototyping/animation	–	R1	–	–
4	Equivalence classes and input partition testing, including boundary value analysis	R1 (If the input data profile is well defined and is manageably simple in structure)	R1 (If the partitions plausibly contain no non-linearities i.e. all members of a class are truly equivalent)	–	–
5	Process simulation	–	R1	–	R2 Gives a definition of the external environment

Table C.14 – Detailed properties – Failure analysis

(Referenced by Table B.4)

Technique/Measure		Properties						
		Completeness of functional safety assessment with respect to this standard	**Correctness of functional safety assessment with respect to the design specifications (successful completion)**	**Traceable closure of all identified issues**	**The ability to modify the functional safety assessment after change without the need for extensive re-work of the assessment**	**Repeatability**	**Timeliness**	**Precisely defined configuration**
1a	Cause consequence diagrams	R2	R2	–	–	–	–	–
1b	Event tree analysis	R2	R2	–	–	–	–	–
2	Fault tree analysis	R2	R2	–	–	–	–	–
3	Software functional failure analysis	R2	R2	–	–	–	–	–

Table C.15 – Detailed properties – Modelling

(Referenced by Table B.5)

Technique/Measure		Properties			
		Completeness of validation with respect to the software design specification	**Correctness of validation with respect to the software design specification (successful completion)**	**Repeatability**	**Precisely defined validation configuration**
1	Data flow diagrams	_	R1	_	_
2a	Finite state machines	R3	R3	_	_
2b	Formal methods	R3	R3	_	_
2c	Time Petri nets	_	R1	_	_
3	Performance modelling	_	R1	_	_
4	Prototyping/animation	_	R1	_	_
5	Structure diagrams	_	R1	_	_

Table C.16 – Detailed properties – Performance testing

(Referenced by Table B.6)

Technique/Measure		Properties			
		Completeness of testing and integration with respect to the design specifications	**Correctness of testing and integration with respect to the design specifications (successful completion)**	**Repeatability**	**Precisely defined testing and integration configuration**
1	Avalanche/stress testing	–	R1 (R2 if objective targets are set)	–	–
2	Response timings and memory constraints	–	R1 (R2 if objective targets are set)	–	–
3	Performance requirements	–	R1 (R2 if objective targets are set)	–	–

Table C.17 – Detailed properties – Semi-formal methods

(Referenced by Table B.7)

Technique/Measure		Properties									
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Understandable safety requirements**	**Freedom from adverse interference of non-safety functions with the safety needs to be addressed by software**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance / Fault detection**	**Freedom from common cause failure from external events**
1	Logic/function block diagrams	R2	R2	R2	–	–	R1	R2	–	–	R1
2	Sequence diagrams	R2	R2	R2	–	–	R1	R2	–	–	R2
3	Data flow diagrams	R1	R1	R1	–	–	R1 Suitable for transaction processing	–	–	–	R1
4a	Finite state machines/state transition diagrams	R2	R2	R2	–	–	R1 Mathematically complete specification of event sequences	R2	–	–	R2
4b	Time Petri nets	R2	R2	R2	–	–	R1 Specifies real-time interactions	R2	–	–	R2
5	Entity-relationship-attribute data models	R1	R1	R1	–	–	R1	–	–	–	R1
6	Message sequence charts	R2	R2	R2	–	–	R1	R2	–	–	R2
7	Decision/truth tables	R2	R2	R2	–	–	R1 For combinatorial logic	R2	–	–	R2

Table C.18 – Properties for systematic safety integrity – Static analysis

(Referenced by Table B.8)

Technique/Measure		Properties			
		Completeness of verification with respect to the previous phase	**Correctness of verification with respect to the previous phase (successful completion)**	**Repeatability**	**Precisely defined verification configuration**
1	Boundary value analysis	_	R1 (R2 if objective criteria for boundary results)	_	_
2	Checklists	_	R1	_	R1
3	Control flow analysis	_	R1	_	_
4	Data flow analysis	_	R1	_	_
5	Error guessing	_	R1	_	_
6a	Formal inspections, including specific criteria	R2	R2	_	R2
6b	Walk-through (software)	R1	R1	_	R1
7	Symbolic execution	_	R2 R3 if used in the context formally defined preconditions and postconditions and performed by a tool using a mathematically rigorous algorithm	_	_
8	Design review	R2	R1 R2 (with objective criteria)	_	R2
9	Static analysis of run time error behaviour	_	R1 R3 for certain classes of error if performed by a tool using a mathematically rigorous algorithm	_	_
10	Worst-case execution time analysis	R1	R3	_	R2

Table C.19 – Detailed properties – Modular approach

(Referenced by Table B.9)

Technique/Measure		Properties							
		Completeness with respect to software safety requirements specification	**Correctness with respect to software safety requirements specification**	**Freedom from intrinsic design faults**	**Simplicity and understandability**	**Predictability of behaviour**	**Verifiable and testable design**	**Fault tolerance / Fault detection**	**Freedom from common cause failure**
1	Software module size limit	–	–	R1	R1	R1	R1	–	–
2	Software complexity control	–	–	R1	R1	R1	R1	–	–
3	Information hiding/encapsulation	–	–	R1	R1	R1	R1	–	–
4	Parameter number limit / fixed number of subprogram parameters	–	–	R1	R1	R1	R1	–	–
5	One entry/one exit point in subroutines and functions	–	–	R1	R1	R1	R1	–	–
6	Fully defined interface	–	–	R2	R1	R1	R1	–	–

Annex D
(normative)

Safety manual for compliant items – additional requirements for software elements

D.1 Purpose of the safety manual

D.1.1 When an element is re-used or is intended to be re-used in one or more other system developments, it is necessary to ensure that the element is accompanied by a sufficiently precise and complete description (i.e. functions, constraints and evidence), to make possible an assessment of the integrity of a specific safety function that depends wholly or partly on the element. This shall be implemented by means of a safety manual.

D.1.2 The safety manual may consist of the element supplier's documentation if this is adequate to meet the requirements of Annex D of IEC 61508-2 and of this annex. Otherwise it should be created as part of the design of the safety related system.

D.1.3 The safety manual shall define the attributes of an element, which may comprise hardware constraints and/or software of which the integrator shall be aware and take into consideration during application. In particular it forms the vehicle for informing the integrator of its properties and what the element was designed for, its behaviour and characteristics.

NOTE 1 The scope and time of delivery of the safety manual will be dependent upon who it applies to, the type of integrator, the purpose of the element and who provides and maintains it.

NOTE 2 The person or department or organization that integrates software is called the integrator.

D.2 Contents of the safety manual for a software element

D.2.1 The safety manual shall contain all the information required by IEC 61508-2 Annex D, that is relevant to the element. E.g. the hardware-related items of IEC 61508-2 Annex D are not relevant to a purely software element.

D.2.2 The element shall be identified and all necessary instructions for its use shall be available to the integrator.

NOTE For software this can be demonstrated by clearly identifying the element and demonstrating that its content is unchanged.

D.2.3 Element configuration:

a) The configuration of the software element, the software and hardware run-time environment and if necessary the configuration of the compilation / link system shall be documented in the safety manual.
b) The recommended configuration of the software element shall be documented in the safety manual and that configuration shall be used in safety application.
c) The safety manual shall include all the assumptions made on which the justification for use of the element depends.

D.2.4 The following shall be included in the safety manual:

a) Competence: The minimum degree of knowledge expected of the integrator of the element should be specified, i.e. knowledge of specific application tools.
b) Degree of reliance placed on the element: Details of any certification of the element, independent assessment performed, integrity to which the integrator may place on the

pre-existing element. This should include the integrity to which the element was designed, the standards that were followed during the design process, and any constraints passed to the integrator which shall be implemented in support of the systematic capability claimed. (depending on the functionality of the element, it is conceivable that some requirements may only be met at the integration phase of a system. In such circumstances, these requirements shall be identified for further progression by the integrator. Requirements pertaining to response times and performance are two such examples).

NOTE Unlike IEC 61508-2, IEC 61508-3 does not require software failure modes or quantitative failure rates in safety manual for compliant items, because the causes of software errors are fundamentally different from the causes of the random hardware failures of interest in IEC 61508-2 Annex D.

c) Installation instructions: Details of, or reference to, how to install the pre-existing element into the integrated system.

d) The reason for release of the element: Details of whether the pre-existing element has been subject to release to clear outstanding anomalies, or inclusion of additional functionality.

e) Outstanding anomalies: Details of all outstanding anomalies should be given, with explanation of the anomaly, how it occurs and the mechanisms that the integrator shall take to mitigate the anomaly should the particular functions be used.

f) Backward compatibility: Details of whether the element is compatible with previous releases of the sub-system, and if not, details of the process providing the upgrade path to be followed.

g) Compatibility with other systems: A pre-existing element may be dependent upon a specially developed operating system. In such circumstances, details of the version of the specially developed operating system should be detailed.

The build standard should also be specified incorporating compiler identification and version, tools used in creation of the pre-existing element (identification and version), and test pre-existing element used (again identification and version).

h) Element configuration: Details of the pre-existing element name(s) and description(s) should be given, including the version / issue / modification state.

i) Change control: The mechanism by which the integrator can initiate a change request to the producer of the software.

j) Requirements not met: It is conceivable that there may exist specific requirements that have been specified, but have not been met in the current revision of the element. In such circumstances, these requirements should be identified for the integrator to consider.

k) Design safe state: In certain circumstances, upon controlled failure of the system application, the element may revert to a design safe state. In such circumstances, the precise definition of design safe state should be specified for consideration by the integrator.

l) Interface constraints: Details of any specific constraints, in particular user interface requirements shall be identified.

m) Details of any security measures that may have been implemented against listed threats and vulnerabilities.

n) Configurable elements: details of the configuration method or methods available for the element, their use and any constraints on their use shall be provided.

D.3 Justification of claims in the safety manual for compliant items

D.3.1 All claims in the safety manual for compliant items shall be justified by adequate supporting evidence. See 7.4.9.7 of IEC 61508-2.

NOTE 1 It is essential that the claimed safety performance of an element is supported by sufficient evidence. Unsupported claims do not help establish the correctness and integrity of the safety function to which the element contributes.

NOTE 2 The supporting evidence 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 by the developer of the safety related system or by third parties.

NOTE 3 There may be commercial or legal restrictions on the availability of the evidence (e.g. copyright or intellectual property rights). These restrictions are outside the scope of this standard.

D.3.2 The supporting evidence that justifies the claims in the safety manual for compliant items is distinct from the element safety manual.

D.3.3 Where the evidence cannot be made available to facilitate functional safety assessment, then the element is not suitable for use in E/E/PE safety-related systems.

Annex E
(informative)

Relationships between IEC 61508-2 and IEC 61508-3

The following table helps finding which clauses of IEC 61508-2 need consideration by those who are dealing with software only and which clauses can be neglected. It is well known that almost all clauses address hardware issues. Therefore this is not repeated here. Important software aspects are treated by IEC 61508-3, many software-related requirements do however also occur in IEC 61508-2, mostly overlapping IEC 61508-3 requirements. Knowledge of IEC 61508-2 is mainly needed for those software specialists who seek compatibility between hardware and software. The IEC 61508-2 requirements are grouped into the following categories:

Table E.1 – Categories of IEC 61508-2 requirements

Software	Both for users of the standard dealing with hardware and for users dealing with software.
Application software	Users dealing with software that is for solving a related safety function as such; not for operating system software or library functions.
System software	For users dealing primarily with operating system software, library functions and the like.
Hardware only	Not for those interested in software only.
Mainly hardware	Concerns software only marginally.

Table E.2 – Requirements of IEC 61508-2 for software and their typical relevance to certain types of software

IEC 61508-2 Requirement	Important to users dealing with	Remarks
7.2	Software	
7.2.3.1	Application software	
7.2.3.2 to 7.2.3.6	Software	
7.2.3.3	Hardware only	
7.3	Software	7.3.2.2 f) Hardware only
7.4	Software	
7.4.2.1 to 7.4.2.12	Software	
7.4.2.13, 7.4.2.14	Hardware only	
7.4.3.1 to 7.4.3.3	Software	
7.4.3.4	Hardware only	
7.4.4	Hardware only	
7.4.5	Hardware only	
7.4.6	Software	7.4.6.7 Hardware only
7.4.7	Software	7.4.7.1 a), b) Hardware only
7.4.8	Hardware only	
7.4.9.1 to 7.4.9.3	Software	
7.4.9.4, 7.4.9.5	Hardware only	
7.4.9.6, 7.4.9.7	Software	
7.4.10	Software	Mainly system software

IEC 61508-2 Requirement	Important to users dealing with	Remarks
7.4.11	Hardware only	
7.5	Software	
7.6	Software	
7.6.2.1 a)	Hardware	
7.6.2.4	Mainly hardware	
7.7	Software	7.7.2.3, 7.7.2.4 Mainly application software
7.8	Software	
7.9	Mainly Application software	
8	Software	
Annex A.1	Mainly hardware	
Annex A.2 and tables	Mainly hardware	Table A.10 Software
Annex A.3	Mainly hardware	Tables A.16, A.17, A.18 Contain some software aspects
Annex B, all tables	Software	
Annex C	Hardware	
Annex D	Software	D.2.3 Hardware only
Annex E	Hardware only	
Annex F	Hardware only	

Annex F
(informative)

Techniques for achieving non-interference between software elements on a single computer

F.1 Introduction

Independence of execution between software elements which are hosted on a single computer system (consisting of one or more processors together with memory and other hardware devices shared between those processors) can be achieved and demonstrated by means of a number of different methods. This annex sets out some techniques which can be used to achieve non-interference (between elements of differing systematic capability, between elements which are designed to achieve or contribute to the same safety function, or between software contributing to a safety function and non-safety related software on the same computer).

NOTE The term “independence of execution” means that elements will not adversely interfere with each other’s execution behaviour such that a dangerous failure would occur. It is used to distinguish other aspects of independence which may be required between elements, in particular diversity, to meet other requirements of the standard.

F.2 Domains of behaviour

Independence of execution should be achieved and demonstrated both in the spatial and temporal domains.

Spatial: the data used by a one element shall not be changed by a another element. In particular, it shall not be changed by a non-safety related element.

Temporal: one element shall not cause another element to function incorrectly by taking too high a share of the available processor execution time, or by blocking execution of the other element by locking a shared resource of some kind.

F.3 Causal factor analysis

To demonstrate independence of execution, an analysis of the proposed design should be undertaken to identify all possible causes of execution interference between the notionally independent (non-interfering) elements in the spatial and temporal domains. The analysis should consider both normal operation and operation under failure conditions, and should include (but need not be limited to) the following:

a) shared use of random access memory;

b) shared use of peripheral devices;

c) shared use of processor time (where two or more elements are executed by a single processor);

d) communications between the elements necessary to achieve the overall design;

e) the possibility that a failure in one element (such as an overflow, or divide by zero exception, or an incorrect pointer calculation) may cause a consequent failure in other elements.

The achievement and justification of independence of execution will then have to address all these identified sources of interference.

F.4 Achieving spatial independence

Techniques for achieving and demonstrating spatial independence include the following:

a) Use of hardware memory protection between different elements, including elements of differing systematic capability.

b) Use of an operating system which permits each element to execute in its own process with its own virtual memory space, supported by hardware memory protection.

c) Use of rigorous design, source code and possibly object code analysis to demonstrate that no explicit or implicit memory references are made from between software elements which can result in data belonging to another element being overwritten (for the case where hardware memory protection is not available).

d) Software protection of the data of a higher integrity element from illegal modification by a lower integrity element.

Data should not be passed from a lower to a higher integrity element unless the higher integrity element can verify that the data is of sufficient integrity.

Where data has to be passed between elements which are required to be independent, uni-directional interfaces such as messages or pipes should be used in preference to shared memory.

NOTE Ideally the independent elements would not communicate with each other. However, where the design of the system requires that one element should send data to another element, the design of the communication mechanism should be such that neither the sending nor the receiving elements should fail or be blocked in execution if data transmission ceases or is delayed.

Any data resident on permanent storage devices such as magnetic discs shall be taken into account for spatial partitioning, in addition to transient data in random access memory. For example, file access protection implemented by an operating system could be used to prevent one element writing to data areas belonging to another element.

F.5 Achieving temporal independence

Techniques for ensuring temporal independence include

a) Deterministic scheduling methods. For example,
   - a cyclic scheduling algorithm which gives each element a defined time slice supported by worst case execution time analysis of each element to demonstrate statically that the timing requirements for each element are met;
   - time triggered architectures.

b) Strict priority based scheduling implemented by a real-time executive with a means of avoiding priority inversion.

c) Time fences which will terminate the execution of an element if it over-runs its allotted execution time or deadline (in such a case, hazard analysis shall be undertaken to show that termination of an element will not result in a dangerous failure, so this technique may be best employed for a non-safety related element).

d) An operating system which guarantees that no process can be starved of processor time, for example by means of time slicing. Such an approach may only be applicable where there are no hard real time requirements to be met by the safety related elements, and it is shown that the scheduling algorithm will not result in undue delays to any element.

Where a resource (such as a peripheral device) is shared between elements, the design shall ensure that the elements will not function incorrectly because the shared resource is locked by another element. The time required to access a shared resource shall be taken into account in determining temporal non-interference.

F.6 Requirements for supporting software

If an operating system, a real-time executive, memory management, timer management or any other such software is to be used to provide spatial or temporal independence, or both, then such software shall be of the highest systematic capability of any of the elements which are required to be independent.

NOTE It is clear that any such software represents a potential common cause of failure of the independent elements.

F.7 Independence of software modules – programming language aspects

The following Table F.1 is an informal definition of relevant terms.

Table F.1 – Module coupling – definition of terms

Term	Informal definition
Cohesion	measure of tightness of the connections between data and subprograms within one module
Coupling	measure for the tightness of connections between modules
Encapsulation	hiding of internal (private) data and subprograms from external access; term primarily used with object oriented programs
Independence	measure of decoupling of software parts; complement of coupling
Module	confined software part that performs something and that may have data of its own; `Class`, hierarchy of classes, subprogram, `unit`, `module`, `package`, ... according to programming language
Interface	well defined set of heads of subprograms that provide access to a module
Tramp data	data that is not used in the receiving module, but only transferred to another module

As a general rule, module independence is enhanced if there is loose coupling between modules and high cohesion within modules. High cohesion encourages the situation where identifiable units of functionality correspond clearly with identifiable units of implementing code, while loose module coupling promotes low interaction and thus high independence between functionally unrelated modules.

Loose module coupling usually results from achieving high cohesion within modules by putting the code and data together that are used to perform one particular function. Low cohesion results, if code and data are assembled in modules only arbitrarily, or because of some timing sequence or due to some sequence in the control flow.

Several aspects of module coupling can be distinguished, see Table F.2 below.

Table F.2 – Types of module coupling

Coupling	Definition	Explanation	Rationale	Remark
Interface coupling, encapsulation	Coupling only via a well defined set of subprograms.	Access to the module or its data only via subprograms; any change of a value of a variable, any question about the value of such a variable, or any other service required from the module is routed via a subprogram call.	The heads of the subprograms (signatures) of a module explain the available services. If any changes of a module are required, a large amount of these changes can be done within that module, without affecting other modules. Promotes loose coupling, recommended in general.	Mainly for object oriented programs, classes, hierarchies of classes, packages of libraries; not for subprograms.
Data coupling via parameter list	Data transfer only via the parameter list or the identifier of subprograms.	Access to the module or its data only via variables or objects that are indicated in the head of the subprogram; any change of a value of a variable, any question about the value of such a variable is visible.	The head of the subprograms exhibits the data or objects involved with a call of that subprogram. Promotes loose coupling, recommended in general.	Within classes of object oriented programs this principle is normally not observed. Local variables may be accessed directly. Strict adherence to that principle may also lead to tramp data. The principle should be violated to avoid this type of data.
Structure coupling	Data transfer contains more data than necessary.	More data are transferred to the receiving subprogram than necessary for performing the required function.	The superfluous data provide another module with information that it does not require for fulfilling its purpose. These data may lead to misunderstanding the cooperation between the modules. It is, however, not deprecated.	The deficiency can normally easily be corrected.
Control coupling	Coupling that exercises immediate control on the receiving module.	Data transfer that can only cause a branching reaction in the other module; in many cases characterized by transfer of a single bit.	Tighter than the couplings above, as it requires immediate action, prescribing the receiving subprogram to do something. To be handled cautiously; to be avoided, if possible. Not recommended in general.	Cannot always be avoided. May be necessary, e.g. if the completion of an action is announced, or the validity of a value.
Global coupling	Coupling via global data.	Modules can access data that are directly accessible by other modules, or one module can directly access data belonging to another module.	The heads of the subprograms do not indicate, which data are used and from where. It is difficult to understand the subprograms' functions and to predict the effects of any changes to code.	Deprecated in general. May be necessary exceptionally, e.g. to avoid tramp data. To be used only in very limited way that conforms to a clearly defined and documented coding standard.
Content coupling	Jumping directly into other modules, influencing branching goals in other modules, or accessing data in other modules directly.	Feasible in assembly language programs; not possible in all higher level languages. Can accelerate program execution and reduce coding effort.	Deprecated. One module can only be understood by understanding its connected modules as well. Makes a program extremely difficult to understand and extremely difficult to change.	In some programming languages not even possible. Can always be avoided.

Code reading or code review (see 7.9.2.12) should verify whether or not the program modules are loosely coupled. This analysis normally requires some sort of understanding of the modules' purpose and their way of working. Proper coupling can therefore be assessed only by reading the code and its documentation.

Content coupling should be avoided. Global coupling may be used only exceptionally. Control coupling and procedural coupling should be avoided. If ever possible, modules should be connected by interface coupling (encapsulation) and/or data coupling.

Annex G
(informative)

Guidance for tailoring lifecycles associated with data driven systems

G.1 Data driven – system part and application part

Many systems are written in two parts. One part provides the underlying system capability. The other part adapts the system to the specific requirements of the intended application. The application part may be written in the form of data, that configures the system part. This is termed “data driven” in this Annex.

The application specific part of the software, may be developed using a variety of programming tools and programming languages. These languages and tools may constrain the way the application program can be written.

For instance, where a programming language supports the developer/configurer in describing the functionality (e.g. the use of ladder logic for simple interlock systems), then the application software programming task is likely to be fairly simple. However, where the programming language allows the developer/configurer to describe complex application behaviour, then the application software programming task is likely to be complex. Where very simple application software is developed, detailed design may be considered as configuring rather than programming.

The degree of rigour necessary to achieve the required safety integrity is dependent upon the degree of configuration complexity available to the developer/configurer and the complexity of behaviour to be represented in the application. This is represented diagrammatically on the axes of Figure G.1.

For simplicity the axes have been further divided into classes of complexity as:

a) Variability allowed by the language:
   - fixed program;
   - limited variability (some industries view the application program as ‘data’ which is interpreted by the system part);
   - full variability (whilst not normally considered as data driven this type of system may also be used for application development and is included in this annex for completeness).

b) Ability to configure application:
   - limited;
   - full.

In reality a particular system may comprise different levels of complexity and configurability. Further, the complexity may exhibit a sliding scale along the continuum of the two axes. When attempting to tailor the software lifecycle, the relevant level of complexity should be identified and the degree of tailoring should be justified.

A description of the typical types of system for each level of complexity is given below. Guidance on suggested techniques for implementing each type of system is given in IEC 61508-7.

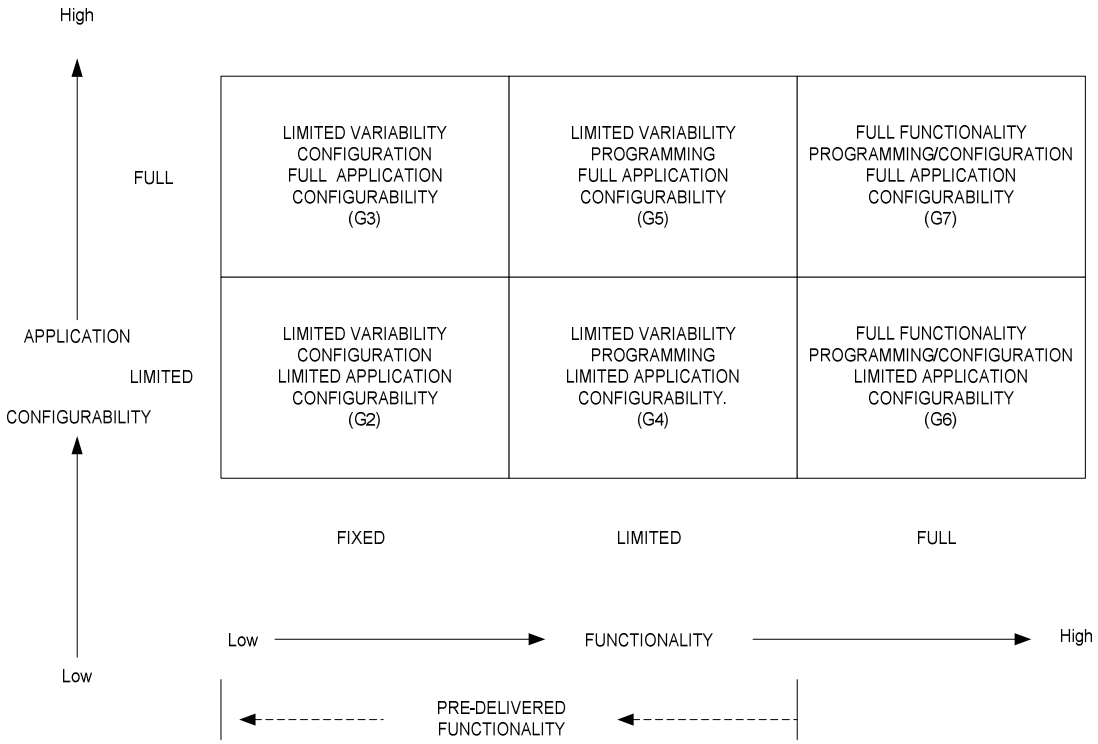


Figure G.1 – Variability in complexity of data driven systems

Typical systems in each class of complexity are described in G.2.

G.2 Limited variability configuration, limited application configurability

A proprietary configuration language used with an IEC 61508 compliant system with fixed pre-delivered functionality.

The configuration language does not allow the programmer to alter the function of the system. Instead configuration is limited to adjustment of a few (data) parameters to enable the system to be matched to its application. Examples may include smart sensors and actuators whereupon specific parameters are entered, network controllers, sequence controllers, small data logging systems and smart instruments.

The justification of the tailoring of the safety lifecycle should include, but not be limited to, the following:

a) specification of the input parameters for this application;

b) verification that the parameters have been correctly implemented in the operational system;

c) validation of all combinations of input parameters;

d) consideration of special and specific modes of operation during configuration;

e) human factors / ergonomics;

f) interlocks, e.g. ensuring that operational interlocks are not invalidated during the configuration process;

g) Inadvertent re-configuration, e.g. key switch access, protection devices.

G.3 Limited variability configuration, full application configurability

A proprietary configuration language used with an IEC 61508 compliant system with fixed pre-delivered functionality.

The configuration language does not allow the programmer to alter the function of the system. Instead, configuration is constrained to creation of extensive static data parameters to enable the system to be matched to its application. An example may be an air traffic control system consisting of data with large numbers of data entities each with one or more attributes. An essential characteristic of the data is that it contains no explicit sequencing, ordering or branching constructs in the data and does not contain any representation of the combinatorial states of the application.

In addition to the considerations given in G.2, the justification of the tailoring of the safety lifecycle should include, but not be limited to, the following:

a) automation tools for creation of data;

b) consistency checking, e.g. the data is self compatible;

c) rules checking, e.g. to ensure the generation of the data meets the defined constraints;

d) validity of interfaces with the data preparation systems.

G.4 Limited variability programming, limited application configurability

A problem-oriented language, used with an IEC 61508 compliant system, where the language statements contain or resemble the terminology of the application of the user for systems with limited pre-delivered functionality.

These languages allow the user limited flexibility to customize the functions of the system to their own specific requirements, based on a range of hardware and software elements.

An essential characteristic of limited variability programming is that data may contain explicit sequencing, ordering or branching constructs and may invoke combinatorial states of the application. Examples may include functional block programming, ladder logic, spreadsheet based systems, and graphical systems.

In addition to the considerations given in G.3, the following elements should be included, but not limited to:

a) the specification of the application requirements;

b) the permitted language sub-sets for this application;

c) the design methods for combining the language sub-sets;

d) the coverage criteria for verification addressing the combinations of potential system states.

G.5 Limited variability programming, full application configurability

A problem-oriented language, used with an IEC 61508 compliant system, where the language statements contain or resemble the terminology of the application of the user for system with limited pre-delivered functionality.

The essential difference from limited variability programming, limited application configurability is the complexity of the configuration of the application. Examples may include graphical systems and SCADA-based batch control systems.

In addition to the considerations given in G.4, the following elements should be included but not limited to:

a) the architectural design of the application;

b) the provision of templates;

c) the verification of the individual templates;

d) the verification and validation of the application.

The aspect of the lifecycle outlined in this standard which is most likely to be unnecessary (depending on the language used) is the lowest level module implementation and testing.

G.6 Full functionality programming/configuration, limited application configurability

See G.7 below.

G.7 Full functionality programming/configuration, full application configurability

For these systems the full lifecycle requirements of this standard apply.

Full variability parts of systems are based on general purpose programming languages or general purpose database languages, or general scientific and simulation packages. Typically, these parts will be used in conjunction with a computer-based system, equipped with an operating system which provides system resource allocation and a real time multi-programming environment. Examples of systems that may be written in full variability languages may include for example: a dedicated machinery control system, specially developed flight control systems, or web services for management of safety related services.

Bibliography

[1] IEC 61511 (all parts), *Functional safety – Safety instrumented systems for the process industry sector*

[2] IEC 62061, *Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems*

[3] IEC 61800-5-2, *Adjustable speed electrical power drive systems – Part 5-2: Safety requirements – Functional*

[4] IEC 61508-5: 2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 5: Examples of methods for the determination of safety integrity levels*

[5] IEC 61508-6: 2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3*

[6] IEC 61508-7: 2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures*

[7] IEC 60601 (all parts), *Medical electrical equipment*

[8] IEC 61131-3, *Programmable controllers – Part 3: Programming languages*

_____

IEC 61508-4

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

BASIC SAFETY PUBLICATION

PUBLICATION FONDAMENTALE DE SÉCURITÉ

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 4: Definitions and abbreviations

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 4: Définitions et abréviations

IEC 61508-4:2010

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:
Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm

Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,…). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.

- Just Published CEI: www.iec.ch/online_news/justpub

Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.

- Electropedia: www.electropedia.org

Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.

- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:
Email: csc@iec.ch
Tél.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61508-4

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

BASIC SAFETY PUBLICATION

PUBLICATION FONDAMENTALE DE SÉCURITÉ

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 4: Definitions and abbreviations

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 4: Définitions et abréviations

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX

V

ICS 25.040.40; 29.020

ISBN 978-2-88910-527-4

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

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

Part 4: Definitions and abbreviations

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) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61508-4 has been prepared by subcommittee 65A: System aspects, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 1998. This edition constitutes a technical revision.

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 standard is based on the following documents:

FDIS	Report on voting
65A/551/FDIS	65A/575/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2

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

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

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, though 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.

– 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.

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

Part 4: Definitions and abbreviations

1 Scope

1.1 This part of IEC 61508 contains the definitions and explanation of terms that are used in parts 1 to 7 of the IEC 61508 series of standards.

1.2 The definitions are grouped under general headings so that related terms can be understood within the context of each other. However, it should be noted that these headings are not intended to add meaning to the definitions.

1.3 IEC 61508-1, IEC 61508-2, 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 function of this international standard does not apply to medical equipment in compliance with the IEC 60601 series.

1.4 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.

1.5 Figure 1 shows the overall framework of the IEC 61508 series and indicates the role that IEC 61508-4 plays in the achievement of functional safety for E/E/PE safety-related systems.

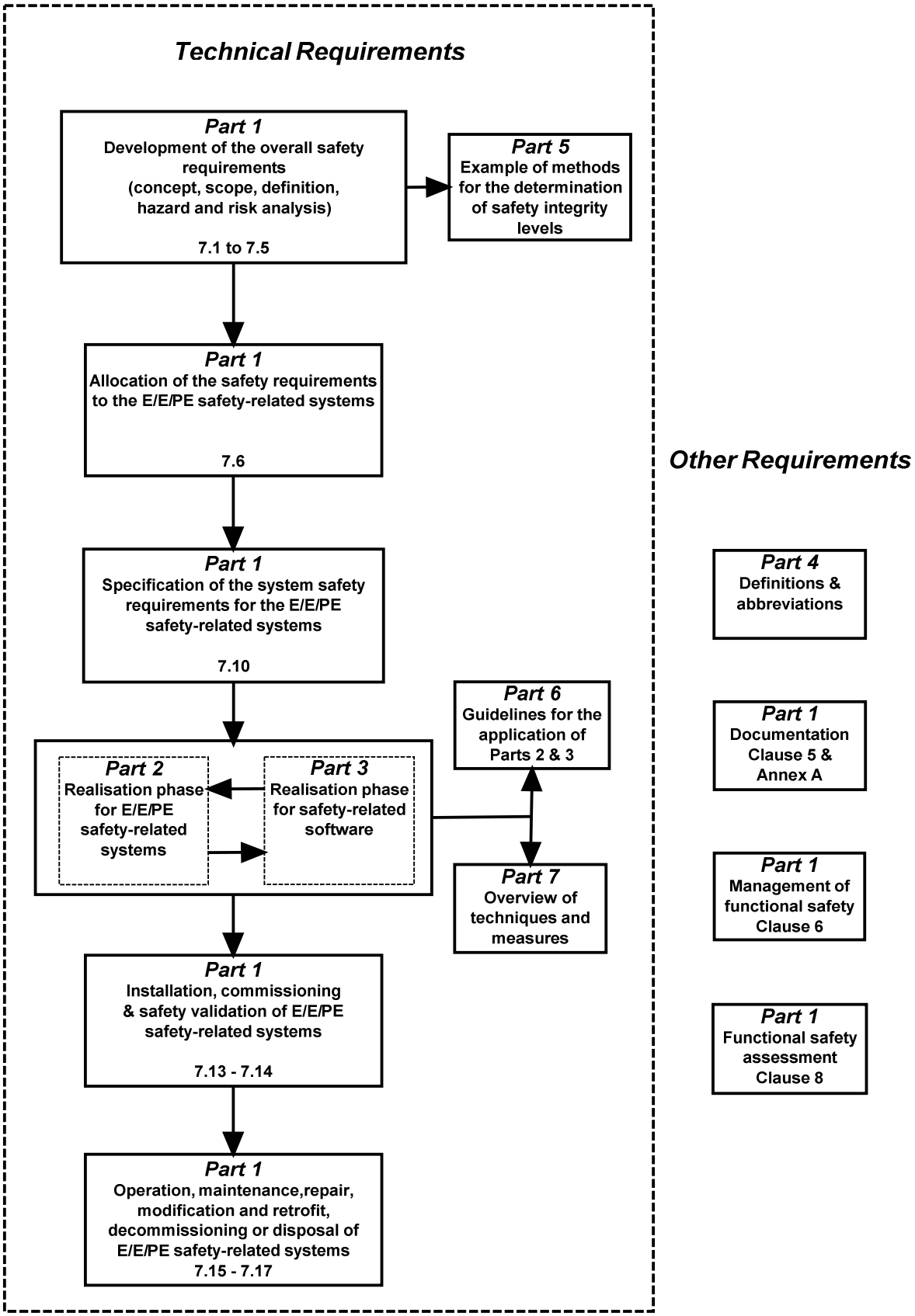


Figure 1 – Overall framework of the IEC 61508 series

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 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*

3 Definitions and abbreviations

For the purposes of this document, the definitions and the abbreviations given in Table 1 below, as well as the following apply.

Table 1 – Abbreviations used in this standard

Abbreviation	Full expression	Definition and/or explanation of term
ALARP	As Low As Reasonably Practicable	IEC 61508-5, Annex C
ASIC	Application Specific Integrated Circuit	3.2.15
CCF	Common Cause Failure	3.6.10
CPLD	Complex Programmable Logic Device	
DC	Diagnostic Coverage	3.8.6
(E)EPLD	(Electrically) Erasable Programmable Logic Device	
E/E/PE	Electrical/Electronic/Programmable Electronic	3.2.13, example: E/E/PE safety-related system
E/E/PE (system)	Electrical/Electronic/Programmable Electronic System	3.3.2
EEPROM	Electrically Erasable Programmable Read-Only Memory	
EPROM	Erasable Programmable Read-Only Memory	
EUC	Equipment Under Control	3.2.1
FPGA	Field Programmable Gate Array	
GAL	Generic Array Logic	
HFT	Hardware Fault Tolerance	7.4.4 of IEC 61508-2
MooN	M out of N channel architecture (for example 1oo2 is 1 out of 2 architecture, where either of the two channels can perform the safety function)	IEC 61508-6, Annex B
MooND	M out of N channel architecture with Diagnostics	IEC 61508-6, Annex B
MTBF	Mean Time Between Failures	3.6.19, NOTE 3
MTTR	Mean Time To Repair	3.6.21
MRT	Mean Repair Time	3.6.22
PAL	Programmable Array Logic	
PE	Programmable Electronic	3.2.12
PE(system)	Programmable Electronic	3.3.1
PFD	Probability of Dangerous Failure on Demand	3.6.17
PFD_{avg}	Average Probability of dangerous Failure on Demand	3.6.18
PFH	Average frequency of dangerous failure [h^{-1}]	3.6.19
PLA	Programmable Logic Array	

Abbreviation	Full expression	Definition and/or explanation of term
PLC	Programmable Logic Controller	IEC 61508-6, Annex E
PLD	Programmable Logic Device	
PLS	Programmable Logic Sequencer	
PML	Programmable Macro Logic	
RAM	Random Access Memory	
ROM	Read-Only Memory	
SFF	Safe Failure Fraction	3.6.15
SIL	Safety Integrity Level	3.5.8
VHDL	Very High Speed Integrated Circuit Hardware Description Language	IEC 61508-2, Annex F, Note 5

3.1 Safety terms

3.1.1
harm
physical injury or damage to the health of people or damage to property or the environment

[ISO/IEC Guide 51:1999, definition 3.3]

3.1.2
hazard
potential source of harm

[ISO/IEC Guide 51:1999, definition 3.5]

NOTE The term includes danger to persons arising within a short time scale (for example, fire and explosion) and also those that have a long-term effect on a person's health (for example, release of a toxic substance).

3.1.3
hazardous situation
circumstance in which people, property or the environment are exposed to one or more hazards

[ISO/IEC Guide 51:1999, definition 3.6, modified]

3.1.4
hazardous event
event that may result in harm

NOTE Whether or not a hazardous event results in harm depends on whether people, property or the environment are exposed to the consequence of the hazardous event and, in the case of harm to people, whether any such exposed people can escape the consequences of the event after it has occurred.

3.1.5
harmful event
occurrence in which a hazardous situation or hazardous event results in harm

NOTE Adapted from ISO/IEC Guide 51, definition 3.4, to allow for a hazardous event.

3.1.6
risk
combination of the probability of occurrence of harm and the severity of that harm

[ISO/IEC Guide 51:1999, definition 3.2]

NOTE For more discussion on this concept see Annex A of IEC 61508-5.

3.1.7
tolerable risk
risk which is accepted in a given context based on the current values of society

[ISO/IEC Guide 51:1999, definition 3.7]

NOTE See Annex C of IEC 61508-5.

3.1.8
residual risk
risk remaining after protective measures have been taken

[ISO/IEC Guide 51:1999, definition 3.9]

3.1.9
EUC risk
risk arising from the EUC or its interaction with the EUC control system

NOTE 1 The risk in this context is that associated with the specific harmful event in which E/E/PE safety-related systems and other risk reduction measures are to be used to provide the necessary risk reduction, (i.e. the risk associated with functional safety).

NOTE 2 The EUC risk is indicated in Figure A.1 of IEC 61508-5. The main purpose of determining the EUC risk is to establish a reference point for the risk without taking into account E/E/PE safety-related systems and other risk reduction measures.

NOTE 3 Assessment of this risk will include associated human factor issues.

3.1.10
target risk
risk that is intended to be reached for a specific hazard taking into account the EUC risk together with the E/E/PE safety-related systems and the other risk reduction measures

3.1.11
safety
freedom from unacceptable risk

[ISO/IEC Guide 51:1999, definition 3.1]

3.1.12
functional safety
part of the overall safety relating to the EUC and the EUC control system that depends on the correct functioning of the E/E/PE safety-related systems and other risk reduction measures

3.1.13
safe state
state of the EUC when safety is achieved

NOTE In going from a potentially hazardous condition to the final safe state, the EUC may have to go through a number of intermediate safe states. For some situations a safe state exists only so long as the EUC is continuously controlled. Such continuous control may be for a short or an indefinite period of time.

3.1.14
reasonably foreseeable misuse
use of a product, process or service in a way not intended by the supplier, but which may result from readily predictable human behaviour

[ISO/IEC Guide 51:1999, definition 3.14]

3.2 Equipment and devices

3.2.1
equipment under control
EUC
equipment, machinery, apparatus or plant used for manufacturing, process, transportation, medical or other activities

NOTE The EUC control system is separate and distinct from the EUC.

3.2.2
environment
all relevant parameters that can affect the achievement of functional safety in the specific application under consideration and in any safety lifecycle phase

NOTE This would include, for example, physical environment, operating environment, legal environment and maintenance environment.

3.2.3
functional unit
entity of hardware or software, or both, capable of accomplishing a specified purpose
[ISO/IEC 2382-1, 01-01-40]

NOTE In IEV 191-01-01 the more general term “item” is used in place of functional unit. An item may sometimes include people.

3.2.4
application
task related to the EUC rather than to the E/E/PE system

3.2.5
software
intellectual creation comprising the programs, procedures, data, rules and any associated documentation pertaining to the operation of a data processing system

NOTE 1 Software is independent of the medium on which it is recorded.

NOTE 2 This definition without Note 1 differs from ISO/IEC 2382-1 (reference [7] in the Bibliography) by the addition of the word data.

3.2.6
system software
part of the software of a PE system that relates to the functioning of, and services provided by, the programmable device itself, as opposed to the application software that specifies the functions that perform a task related to the safety of the EUC

NOTE Refer to IEC 61508-7 for examples.

3.2.7
application software
application data
configuration data
part of the software of a programmable electronic system that specifies the functions that perform a task related to the EUC rather than the functioning of, and services provided by the programmable device itself

3.2.8
pre-existing software
software element which already exists and is not developed specifically for the current project or safety-related system.

NOTE The 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.

3.2.9
data
information represented in a manner suitable for communication, interpretation, or processing by computers

NOTE 1 Data may take the form of static information (for example configuration of a set point or a representation of geographical information) or it may take the form of instructions to specify a sequence of pre-existing functions.

NOTE 2 Refer to IEC 61508-7 for examples.

3.2.10
software on-line support tool
software tool that can directly influence the safety-related system during its run time

3.2.11
software off-line support tool
software tool that supports a phase of the software development lifecycle and that cannot directly influence the safety-related system during its run time. Software off-line tools may be divided into the following classes:

– **T1**
generates no outputs which can directly or indirectly contribute to the executable code (including data) of the safety related system;

NOTE 1 T1 examples include: a text editor or a requirements or design support tool with no automatic code generation capabilities; configuration control tools.

– **T2**
supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software;

NOTE 2 T2 examples include: a test harness generator; a test coverage measurement tool; a static analysis tool.

– **T3**
generates outputs which can directly or indirectly contribute to the executable code of the safety related system.

NOTE 3 T3 examples include: an optimising compiler where the relationship between the source code program and the generated object code is not obvious; a compiler that incorporates an executable run-time package into the executable code.

3.2.12
programmable electronic
PE
based on computer technology which may be comprised of hardware, software, and of input and/or output units

NOTE This term covers microelectronic devices based on one or more central processing units (CPUs) together with associated memories, etc.

EXAMPLE The following are all programmable electronic devices:

– microprocessors;

– micro-controllers;

– programmable controllers;

– application specific integrated circuits (ASICs);

– programmable logic controllers (PLCs);

– other computer-based devices (for example smart sensors, transmitters, actuators).

3.2.13
electrical/electronic/programmable electronic
E/E/PE
based on electrical (E) and/or electronic (E) and/or programmable electronic (PE) technology

NOTE The term is intended to cover any and all devices or systems operating on electrical principles.

EXAMPLE Electrical/electronic/programmable electronic devices include:

– electro-mechanical devices (electrical);

– solid-state non-programmable electronic devices (electronic);

– electronic devices based on computer technology (programmable electronic); see 3.2.12.

3.2.14
limited variability language
software programming language, whose notation is textual or graphical or has characteristics of both, for commercial and industrial programmable electronic controllers with a range of capabilities limited to their application

EXAMPLE The following are limited variability languages, from IEC 61131-3 (reference [8] in the Bibliography) and other sources, which are used to represent the application program for a PLC system:

– ladder diagram: a graphical language consisting of a series of input symbols (representing behaviour similar to devices such as normally open and normally closed contacts) interconnected by lines (to indicate the flow of current) to output symbols (representing behaviour similar to relays);

– Boolean algebra: a low-level language based on Boolean operators such as AND, OR and NOT with the ability to add some mnemonic instructions;

– function block diagram: in addition to Boolean operators, allows the use of more complex functions such as data transfer file, block transfer read/write, shift register and sequencer instructions;

– sequential function chart: a graphical representation of a sequential program consisting of interconnected steps, actions and directed links with transition conditions.

3.2.15
application specific integrated circuit
ASIC
integrated circuit designed and manufactured for specific function, where its functionality is defined by the product developer

NOTE The term ASIC as a stand-alone covers all types of the following integrated circuits:

– Full custom ASIC: ASIC where design and production is similar to a standard integrated circuit with the functionality defined by the product developer.

  A standard integrated circuit is manufactured in large quantities and can be used for different applications. Functionality, validation, production and production test are solely in the hand of the semiconductor vendor. Manual manipulations and optimisations at layout level are frequently used to reduce required area. They are not designed for safety-related systems. Frequent changes in production process, process technology and layout are likely for cost and yield optimisation. The number of components manufactured using a certain process or mask revision are not publicly known.

– Core based ASIC: ASIC based on a pre-layout, designed or generated macro cores, supported by additional logic.

  EXAMPLE 1 Examples for pre-layout macros are standard microprocessor cores, peripheral components, communication interfaces, analogue blocks, special function I/O cells.

  EXAMPLE 2 Examples for pre-designed macros known as Intellectual Property (IP) are a variety of similar components as mentioned in Example 1, with the difference that the design data consists of a high level hardware description language (VHDL, Verilog) as described for cell based ASIC.

  EXAMPLE 3 Examples for generated macros include embedded RAM, ROM, EEPROM or FLASH (flash memory). Generated blocks are assumed to be correct by construction, based on design rules. Pre-layout or generated macros are process specific but may be ported to different technologies. In most cases, the macro cores are not identical to the original discrete off-the-shelf components (different process, provided by a third party).

– Cell based ASIC: ASIC based on logic primitives (like AND, OR, Flip-Flop, Latch) taken from a cell library.

  The gate-level netlist containing the logic primitives and the interconnections is usually created from a high level hardware description language (VHDL, Verilog HDL) using synthesis tools. The functional and timing

characteristics of the logic primitives is characterised in the cell library; these parameters are used to drive the synthesis tool and are also used for simulation. In addition, layout tools are used to place the cells and to route the interconnects.

– Gate array: pre-manufactured silicon masters with a fixed number of cells that provide a common starting point for different components.

  The functionality is defined by the interconnection matrix (metal layer) between the pre-manufactured cells. The design process is very similar to that of a cell based ASIC, while the layout step is replaced by a routing step to connect the already existing cells.

– Field programmable gate array (FPGA): standard integrated circuit, using one-time programmable or re-programmable elements to define the connection between functional blocks and to configure the functionality of the individual blocks.

  It is not possible to test one-time programmable FPGAs completely during production due to the nature of the programmable element.

– Programmable logic device (PLD): standard integrated circuit, with low to medium complexity, using one-time programmable or electrical erasable elements (fuses) to define combinatorial logic – typically based on AND or OR product terms – and configurable storage elements.

  PLDs provide predictable timing and guaranteed maximum operating frequency in synchronous design due to their regular structure.

  Type of PLD are for example PAL, GAL, PML, (E)EPLD, PLA, PLS.

– Complex programmable logic device (CPLD): multiple PLD-like blocks on a single chip, connected by a programmable interconnection matrix (crossbar).

  The programmable logic element is re-programmable (EPROM or EEPROM) in most cases.

3.3 Systems – general aspects

3.3.1
programmable electronic system
PE system
system for control, protection or monitoring based on one or more programmable electronic devices, including all elements of the system such as power supplies, sensors and other input devices, data highways and other communication paths, and actuators and other output devices (see Figure 2)

NOTE The structure of a PES is shown in Figure 2 a). Figure 2 b) illustrates the way in which a PES is represented in this International Standard, with the programmable electronics shown as a unit distinct from sensors and actuators on the EUC and their interfaces, but the programmable electronics could exist at several places in the PES. Figure 2 c) illustrates a PES with two discrete units of programmable electronics. Figure 2 d) illustrates a PES with dual programmable electronics (i.e. two-channel), but with a single sensor and a single actuator

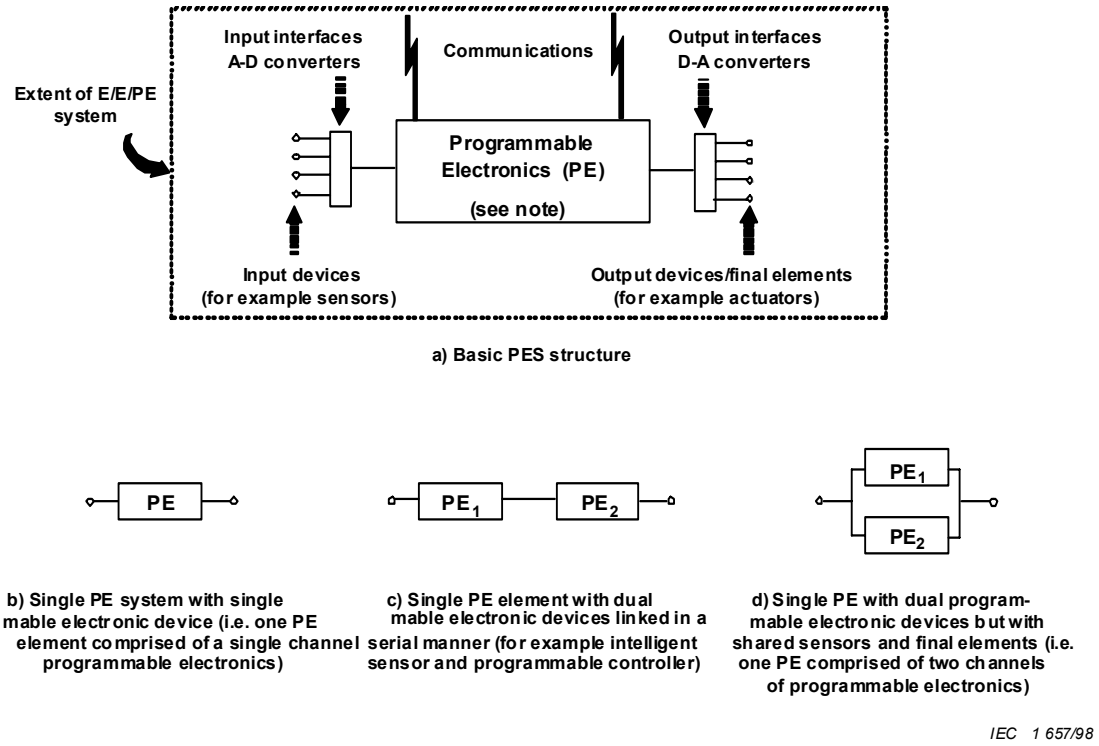
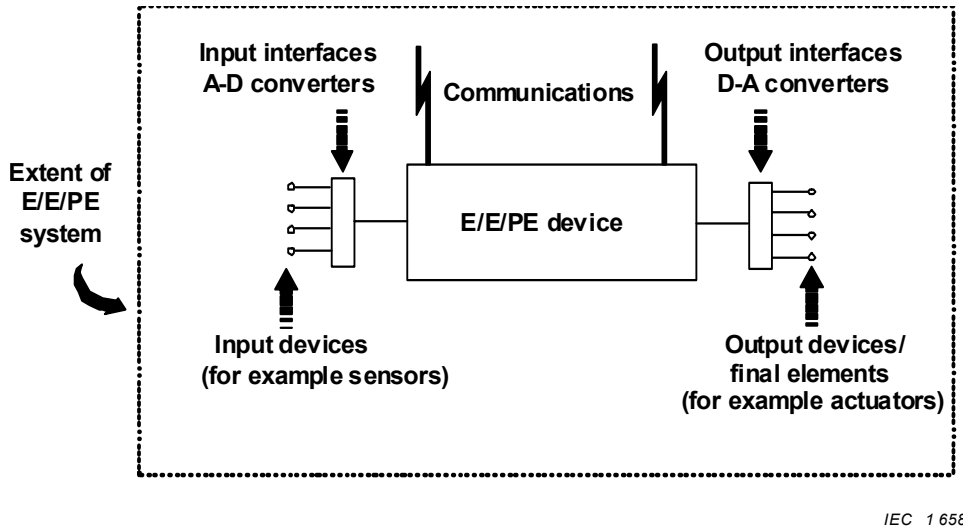


Figure 2 – Programmable electronic system

3.3.2
electrical/electronic/programmable electronic system
E/E/PE system
system for control, protection or monitoring based on one or more electrical/electronic programmable electronic (E/E/PE) devices, including all elements of the system such as power supplies, sensors and other input devices, data highways and other communication paths, and actuators and other output devices (see Figure 3)



NOTE THE E/E/PE device is shown centrally located but such device(s) could exist at several places in the E/E/PE system.

Figure 3 – Electrical/electronic/programmable electronic system (E/E/PE system) – structure and terminology

3.3.3
EUC control system
system that responds to input signals from the process and/or from an operator and generates output signals causing the EUC to operate in the desired manner

NOTE The EUC control system includes input devices and final elements.

3.3.4
architecture
specific configuration of hardware and software elements in a system

3.3.5
software module
construct that consists of procedures and/or data declarations and that can also interact with other such constructs

3.3.6
channel
element or group of elements that independently implement an element safety function

EXAMPLE A two-channel (or dual-channel) configuration is one with two channels that independently perform the same function.

NOTE The term can be used to describe a complete system, or a portion of a system (for example, sensors or final elements).

3.3.7
diversity
different means of performing a required function

NOTE Diversity may be achieved by different physical methods or different design approaches.

3.4 Systems – safety-related aspects

3.4.1
safety-related system
designated system that both

– implements the required safety functions necessary to achieve or maintain a safe state for the EUC; and
– is intended to achieve, on its own or with other E/E/PE safety-related systems and other risk reduction measures, the necessary safety integrity for the required safety functions

NOTE 1 The term refers to those systems, designated as safety-related systems, that are intended to achieve, together with the other risk reduction measures (see 3.4.2), the necessary risk reduction in order to meet the required tolerable risk (see 3.1.7). See also Annex A of IEC 61508-5.

NOTE 2 Safety-related systems are designed to prevent the EUC from going into a dangerous state by taking appropriate action on detection of a condition which may lead to a hazardous event. The failure of a safety-related system would be included in the events leading to the determined hazard or hazards. Although there may be other systems having safety functions, it is the safety-related systems that have been designated to achieve, in their own right, the required tolerable risk. Safety-related systems can broadly be divided into safety-related control systems and safety-related protection systems.

NOTE 3 Safety-related systems may be an integral part of the EUC control system or may interface with the EUC by sensors and/or actuators. That is, the required safety integrity level may be achieved by implementing the safety functions in the EUC control system (and possibly by additional separate and independent systems as well) or the safety functions may be implemented by separate and independent systems dedicated to safety.

NOTE 4 A safety-related system may

a) be designed to prevent the hazardous event (i.e. if the safety-related systems perform their safety functions then no harmful event arises);
b) be designed to mitigate the effects of the harmful event, thereby reducing the risk by reducing the consequences;

c) be designed to achieve a combination of a) and b).

NOTE 5 A person can be part of a safety-related system. For example, a person could receive information from a programmable electronic device and perform a safety action based on this information, or perform a safety action through a programmable electronic device.

NOTE 6 A safety-related system includes all the hardware, software and supporting services (for example, power supplies) necessary to carry out the specified safety function (sensors, other input devices, final elements (actuators) and other output devices are therefore included in the safety-related system).

NOTE 7 A safety-related system may be based on a wide range of technologies including electrical, electronic, programmable electronic, hydraulic and pneumatic.

3.4.2
other risk reduction measure
measure to reduce or mitigate risk that is separate and distinct from, and does not use, E/E/PE safety-related systems

EXAMPLE A relief valve is an other risk reduction measure.

3.4.3
low complexity E/E/PE safety-related system
E/E/PE safety-related system (see 3.2.13 and 3.4.1), in which

– the failure modes of each individual component are well defined;

– the behaviour of the system under fault conditions can be completely determined.

NOTE Behaviour of the system under fault conditions may be determined by analytical and/or test methods.

EXAMPLE A system comprising one or more limit switches, operating, possibly via interposing electro-mechanical relays, one or more contactors to de-energise an electric motor is a low-complexity E/E/PE safety-related system.

3.4.4
subsystem
entity of the top-level architectural design of a safety-related system where a dangerous failure according to 3.6.7 (a) of the subsystem results in dangerous failure of a safety function according to 3.6.7 (a)

3.4.5
element
part of a subsystem comprising a single component or any group of components that performs one or more element safety functions.

[IEC 62061, definition 3.2.6, modified]

NOTE 1 An element may comprise hardware and/or software.

NOTE 2 A typical element is a sensor, programmable controller or final element

3.4.6
redundancy
the existence of more than one means for performing a required function or for representing information.

[based on IEC 62059-11]

EXAMPLE Duplicated functional components and the addition of parity bits are both instances of redundancy.

NOTE 1 Redundancy is used primarily to improve reliability (probability of functioning properly over a given period of time) or availability (probability of functioning at given instant). It may also be used in order to minimize spurious actions through architectures such as 2oo3.

NOTE 2 The definition in IEV 191-15-01 is less complete.

NOTE 3 Redundancy may be "hot" or "active" (all redundant item running at the same time), "cold" or "stand-by" (only one of the redundant item working at the same time), "mixed" (one or several items running and one or several items in stand-by at the same time).

3.5 Safety functions and safety integrity

3.5.1
safety function
function to be implemented by an E/E/PE safety-related system or other risk reduction measures, that is intended to achieve or maintain a safe state for the EUC, in respect of a specific hazardous event (see 3.4.1 and 3.4.2)

EXAMPLE Examples of safety functions include:

– functions that are required to be carried out as positive actions to avoid hazardous situations (for example switching off a motor); and

– functions that prevent actions being taken (for example preventing a motor starting).

3.5.2
overall safety function
means of achieving or maintaining a safe state for the EUC, in respect of a specific hazardous event

3.5.3
element safety function
that part of a safety function (see 3.5.1) which is implemented by an element

3.5.4
safety integrity
probability of an E/E/PE safety-related system satisfactorily performing the specified safety functions under all the stated conditions within a stated period of time

NOTE 1 The higher the level of safety integrity, the lower the probability that the safety-related system will fail to carry out the specified safety functions or will fail to adopt a specified state when required.

NOTE 2 There are four levels of safety integrity (see 3.5.8).

NOTE 3 In determining safety integrity, all causes of failures (both random hardware failures and systematic failures) that lead to an unsafe state should be included, for example hardware failures, software induced failures and failures due to electrical interference. Some of these types of failure, in particular random hardware failures, may be quantified using such measures as the average frequency of failure in the dangerous mode of failure or the probability of a safety-related protection system failing to operate on demand. However, safety integrity also depends on many factors that cannot be accurately quantified but can only be considered qualitatively.

NOTE 4 Safety integrity comprises hardware safety integrity (see 3.5.7) and systematic safety integrity (see 3.5.6).

NOTE 5 This definition focuses on the reliability of the safety-related systems to perform the safety functions (see IEV 191-12-01 for a definition of reliability).

3.5.5
software safety integrity
part of the safety integrity of a safety-related system relating to systematic failures in a dangerous mode of failure that are attributable to software

3.5.6
systematic safety integrity
part of the safety integrity of a safety-related system relating to systematic failures in a dangerous mode of failure

NOTE Systematic safety integrity cannot usually be quantified (as distinct from hardware safety integrity which usually can).

3.5.7
hardware safety integrity
part of the safety integrity of a safety-related system relating to random hardware failures in a dangerous mode of failure

NOTE The term relates to failures in a dangerous mode, that is, those failures of a safety-related system that would impair its safety integrity. The two parameters that are relevant in this context are the average frequency of dangerous failure and the probability of failure to operate on demand. The former reliability parameter is used when it is necessary to maintain continuous control in order to maintain safety, the latter reliability parameter is used in the context of safety-related protection systems.

3.5.8
safety integrity level
SIL
discrete level (one out of a possible four), corresponding to a range of safety integrity values, where safety integrity level 4 has the highest level of safety integrity and safety integrity level 1 has the lowest

NOTE 1 The target failure measures (see 3.5.17) for the four safety integrity levels are specified in Tables 2 and 3 of IEC 61508-1.

NOTE 2 Safety integrity levels are used for specifying the safety integrity requirements of the safety functions to be allocated to the E/E/PE safety-related systems.

NOTE 3 A safety integrity level (SIL) is not a property of a system, subsystem, element or component. The correct interpretation of the phrase "SIL n safety-related system" (where n is 1, 2, 3 or 4) is that the system is potentially capable of supporting safety functions with a safety integrity level up to n.

3.5.9
systematic capability
measure (expressed on a scale of SC 1 to SC 4) of the confidence that the systematic safety integrity of an element meets the requirements of the specified SIL, in respect of the specified element safety function, when the element is applied in accordance with the instructions specified in the compliant item safety manual for the element

NOTE 1 Systematic capability is determined with reference to the requirements for the avoidance and control of systematic faults (see IEC 61508-2 and IEC 61508-3).

NOTE 2 What is a relevant systematic failure mechanism will depend on the nature of the element. For example, for an element comprising solely software, only software failure mechanisms will need to be considered. For an element comprising hardware and software, it will be necessary to consider both systematic hardware and software failure mechanisms.

NOTE 3 A Systematic capability of SC N for an element, in respect of the specified element safety function, means that the systematic safety integrity of SIL N has been met when the element is applied in accordance with the instructions specified in the compliant item safety manual for the element.

3.5.10
software safety integrity level
systematic capability of a software element that forms part of a subsystem of a safety-related system

NOTE SIL characterises the overall safety function, but not any of the distinct subsystems or elements that support that safety function. In common with any element, software therefore has no SIL in its own right. However, it is convenient to talk about "SIL N software" meaning "software in which confidence is justified (expressed on a scale of 1 to 4) that the (software) element safety function will not fail due to relevant systematic failure mechanisms when the (software) element is applied in accordance with the instructions specified in the compliant item safety manual for the element".

3.5.11
E/E/PE system safety requirements specification
specification containing the requirements for the safety functions and their associated safety integrity levels

3.5.12
E/E/PE system safety functions requirements specification
specification containing the requirements for the safety functions that have to be performed by the safety-related systems

NOTE 1 This specification is one part (the safety functions part) of the E/E/PE system safety requirements specification (see 7.10 and 7.10.2.6 of IEC 61508-1) and contains the precise details of the safety functions that have to be performed by the safety-related systems.

NOTE 2 Specifications may be documented in text, flow diagrams, matrices, logic diagrams, etc., providing that the safety functions are clearly conveyed.

3.5.13
E/E/PE system safety integrity requirements specification
specification containing the safety integrity requirements of the safety functions that have to be performed by the safety-related systems

NOTE This specification is one part (the safety integrity part) of the E/E/PE system safety requirements specification (see 7.10 and 7.10.2.7 of IEC 61508-1)

3.5.14
E/E/PE system design requirements specification
specification containing the design requirements for the E/E/PE safety-related system in terms of the subsystems and elements

3.5.15
safety-related software
software that is used to implement safety functions in a safety-related system

3.5.16
mode of operation
way in which a safety function operates, which may be either

– **low demand mode:** where the safety function is only performed on demand, in order to transfer the EUC into a specified safe state, and where the frequency of demands is no greater than one per year; or

NOTE The E/E/PE safety-related system that performs the safety function normally has no influence on the EUC or EUC control system until a demand arises. However, if the E/E/PE safety-related system fails in such a way that it is unable to carry out the safety function then it may cause the EUC to move to a safe state (see 7.4.6 of IEC 61508-2).

– **high demand mode:** where the safety function is only performed on demand, in order to transfer the EUC into a specified safe state, and where the frequency of demands is greater than one per year; or

– **continuous mode:** where the safety function retains the EUC in a safe state as part of normal operation

3.5.17
target failure measure
target probability of dangerous mode failures to be achieved in respect of the safety integrity requirements, specified in terms of either

– the average probability of a dangerous failure of the safety function on demand, (for a low demand mode of operation);

– the average frequency of a dangerous failure [h^{-1}] (for a high demand mode of operation or a continuous mode of operation)

NOTE The numerical values for the target failure measures are given in Tables 2 and 3 of IEC 61508-1.

3.5.18
necessary risk reduction
risk reduction to be achieved by the E/E/PE safety-related systems and/or other risk reduction measures in order to ensure that the tolerable risk is not exceeded

3.6 Fault, failure and error (see Figure 4)

3.6.1
fault
abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function

[ISO/IEC 2382-14, 14-01-10]

NOTE IEV 191-05-01 defines “fault” as a state characterised by the inability to perform a required function, excluding the inability during preventative maintenance or other planned actions, or due to lack of external resources. See Figure 4 for an illustration of these two points of view.

3.6.2
fault avoidance
use of techniques and procedures that aim to avoid the introduction of faults during any phase of the safety lifecycle of the safety-related system

3.6.3
fault tolerance
ability of a functional unit to continue to perform a required function in the presence of faults or errors

[ISO/IEC 2382-14, 14-04-06]

NOTE The definition in IEV 191-15-05 refers only to sub-item faults. See the Note for the term “fault” in 3.6.1.

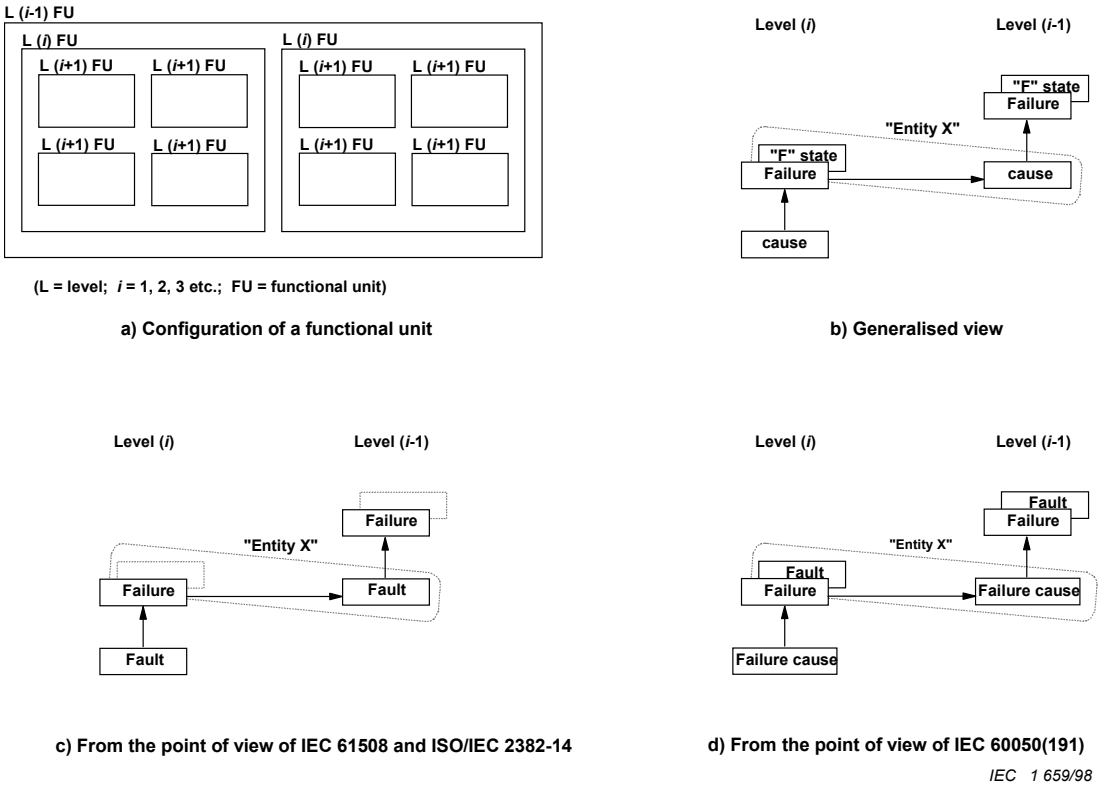
3.6.4
failure
termination of the ability of a functional unit to provide a required function or operation of a functional unit in any way other than as required

NOTE 1 This is based on IEV 191-04-01 with changes to include systematic failures due to, for example, deficiencies in specification or software.

NOTE 2 See Figure 4 for the relationship between faults and failures, both in the IEC 61508 series and IEC 60050-191.

NOTE 3 Performance of required functions necessarily excludes certain behaviour, and some functions may be specified in terms of behaviour to be avoided. The occurrence of such behaviour is a failure.

NOTE 4 Failures are either random (in hardware) or systematic (in hardware or software), see 3.6.5 and 3.6.6.



NOTE 1 As shown in a), a functional unit can be viewed as a hierarchical composition of multiple levels, each of which can in turn be called a functional unit. In level (*i*), a "cause" may manifest itself as an error (a deviation from the correct value or state) within this level (*i*) functional unit, and, if not corrected or circumvented, may cause a failure of this functional unit, as a result of which it falls into an "F" state where it is no longer able to perform a required function (see b)). This "F" state of the level (*i*) functional unit may in turn manifest itself as an error in the level (*i*-1) functional unit and, if not corrected or circumvented, may cause a failure of this level (*i*-1) functional unit.

NOTE 2 In this cause and effect chain, the same thing ("Entity X") can be viewed as a state ("F" state) of the level (*i*) functional unit into which it has fallen as a result of its failure, and also as the cause of the failure of the level (*i*-1) functional unit. This "Entity X" combines the concept of "fault" in IEC 61508 and ISO/IEC 2382-14, which emphasizes its cause aspect as illustrated in c), and that of "fault" in IEC 60050-191, which emphasizes its state aspect as illustrated in d). The "F" state is called fault in IEC 60050-191, whereas it is not defined in the IEC 61508 series and ISO/IEC 2382-14.

NOTE 3 In some cases, a failure or an error may be caused by an external event such as lightning or electrostatic noise, rather than by an internal fault. Likewise, a fault (in both vocabularies) may exist without a prior failure. An example of such a fault is a design fault.

Figure 4 – Failure model

3.6.5
random hardware failure
failure, occurring at a random time, which results from one or more of the possible degradation mechanisms in the hardware

NOTE 1 There are many degradation mechanisms occurring at different rates in different components and, since manufacturing tolerances cause components to fail due to these mechanisms after different times in operation, failures of equipment comprising many components occur at predictable rates but at unpredictable (i.e. random) times.

NOTE 2 A major distinguishing feature between random hardware failures and systematic failures (see 3.6.6), is that system failure rates (or other appropriate measures), arising from random hardware failures, can be predicted with reasonable accuracy but systematic failures, by their very nature, cannot be accurately predicted. That is, system failure rates arising from random hardware failures can be quantified with reasonable accuracy but those arising from systematic failures cannot be accurately statistically quantified because the events leading to them cannot easily be predicted.

3.6.6
systematic failure
failure, related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors

[IEV 191-04-19]

NOTE 1 Corrective maintenance without modification will usually not eliminate the failure cause.

NOTE 2 A systematic failure can be induced by simulating the failure cause.

NOTE 3 Examples of causes of systematic failures include human error in

– the safety requirements specification;

– the design, manufacture, installation, operation of the hardware;

– the design, implementation, etc. of the software.

NOTE 4 In this standard, failures in a safety-related system are categorized as random hardware failures (see 3.6.5) or systematic failures.

3.6.7
dangerous failure
failure of an element and/or subsystem and/or system that plays a part in implementing the safety function that:

a) prevents a safety function from operating when required (demand mode) or causes a safety function to fail (continuous mode) such that the EUC is put into a hazardous or potentially hazardous state; or

b) decreases the probability that the safety function operates correctly when required

3.6.8
safe failure
failure of an element and/or subsystem and/or system that plays a part in implementing the safety function that:

a) results in the spurious operation of the safety function to put the EUC (or part thereof) into a safe state or maintain a safe state; or

b) increases the probability of the spurious operation of the safety function to put the EUC (or part thereof) into a safe state or maintain a safe state

3.6.9
dependent failure
failure whose probability cannot be expressed as the simple product of the unconditional probabilities of the individual events that caused it

NOTE Two events A and B are dependent, only if: $P(\text{A and B}) > P(\text{A}) \times P(\text{B})$.

3.6.10
common cause failure
failure, that is the result of one or more events, causing concurrent failures of two or more separate channels in a multiple channel system, leading to system failure

3.6.11
error
discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition

[IEV 191-05-24, modified]

3.6.12
soft-error
erroneous changes to data content but no changes to the physical circuit itself

NOTE 1 When a soft error has occurred and the data is rewritten, the circuit will be restored to its original state.

NOTE 2 Soft errors can occur in memory, digital logic, analogue circuits, and on transmission lines, etc and are dominant in semiconductor memory, including registers and latches. Data may be obtained, for example, from manufactures.

NOTE 3 Soft errors are transient and should not be confused with software programming errors.

3.6.13
no part failure
failure of a component that plays no part in implementing the safety function

NOTE The no part failure is not used for SFF calculations

3.6.14
no effect failure
failure of an element that plays a part in implementing the safety function but has no direct effect on the safety function

NOTE 1 The no effect failure has by definition no effect on the safety function so it cannot contribute to the failure rate of the safety function.

NOTE 2 The no effect failure is not used for SFF calculations.

3.6.15
safe failure fraction
SFF
property of a safety related element that is defined by the ratio of the average failure rates of safe plus dangerous detected failures and safe plus dangerous failures. This ratio is represented by the following equation:

$SFF = (\Sigma\lambda_{S\ avg} + \Sigma\lambda_{Dd\ avg})/(\Sigma\lambda_{S\ avg} + \Sigma\lambda_{Dd\ avg} + \Sigma\lambda_{Du\ avg})$

when the failure rates are based on constant failure rates the equation can be simplified to:

$SFF = (\Sigma\lambda_{S} + \Sigma\lambda_{Dd})/(\Sigma\lambda_{S} + \Sigma\lambda_{Dd} + \Sigma\lambda_{Du})$

3.6.16
failure rate
reliability parameter ($\lambda(t)$) of an entity (single components or systems) such that $\lambda(t).dt$ is the probability of failure of this entity within [t, t+dt] provided that it has not failed during [0, t]

NOTE 1 Mathematically, $\lambda(t)$ is the conditional probability of failure per unit of time over [t, t+dt]. It is in strong relationship with the reliability function (i.e. probability of no failure from 0 to t) by the general formula $R(t) = \exp(-\int_0^t \lambda(\tau)d\tau$. Reversely it is defined from the reliability function by $\lambda(t) = -\frac{dR(t)}{dt}\frac{1}{R(t)}$.

NOTE 2 Failure rates and their uncertainties can be estimated from field feedback by using conventional statistics. During the "useful life" (i.e. after burn-in and before wear-out), the failure rate of a simple item is more or less constant, $\lambda(t) \equiv \lambda$.

NOTE 3 The average of $\lambda(t)$ over a given period [0, T], $\lambda_{avg}(T) = (\int_0^T \lambda(\tau)d\tau)/T$, is not a failure rate because it cannot be used for calculating R(t) as shown in Note 1. However it may be interpreted as the *average frequency* of failure over this period (i.e. the PFH, see Annex B of IEC 61508-6).

NOTE 4 The failure rate of a series of items is the sum of the failure rates of each item.

NOTE 5 The failure rate of redundant systems is generally non constant. Nevertheless when all failures are quickly revealed, independent and quickly repaired, $\lambda(t)$ converges quickly to an asymptotic value λ_{as} which is the *equivalent failure rate* of the systems. It should not be confused with the average failure rate described in Note 3 which doesn't necessarily converge to an asymptotic value.

3.6.17
probability of dangerous failure on demand
PFD
safety unavailability (see IEC 60050-191) of an E/E/PE safety-related system to perform the specified safety function when a demand occurs from the EUC or EUC control system

NOTE 1 The [instantaneous] unavailability (as per IEC 60050-191) is the probability that an item is not in a state to perform a required function under given conditions at a given instant of time, assuming that the required external resources are provided. It is generally noted by U (t).

NOTE 2 The [instantaneous] availability does not depend on the states (running or failed) experienced by the item before t. It characterizes an item which only has to be able to work when it is required to do so, for example, an E/E/PE safety related system working in low demand mode

NOTE 3 If periodically tested, the PFD of an E/E/PE safety-related system is, in respect of the specified safety function, represented by a saw tooth curve with a large range of probabilities ranging from low, just after a test, to a maximum just before a test.

3.6.18
average probability of dangerous failure on demand
PFD_{avg}
mean unavailability (see IEC 60050-191) of an E/E/PE safety-related system to perform the specified safety function when a demand occurs from the EUC or EUC control system

NOTE 1 The mean unavailability over a given time interval [t1, t2] is generally noted by U (t1, t2).

NOTE 2 Two kind of failures contribute to PFD and PFD_{avg}: the *dangerous undetected failures* occurred since the last proof test and genuine *on demand failures* caused by the demands (proof tests and safety demands) themselves. The first one is *time dependent* and characterized by their dangerous failure rate $\lambda_{DU}(t)$ whilst the second one is dependent only on the number of demands and is characterized by a *probability of failure per demand* (denoted by γ).

NOTE 3 As genuine on demand failures cannot be detected by tests, it is necessary to identify them and take them into consideration when calculating the target failure measures.

3.6.19
average frequency of a dangerous failure per hour
PFH
average frequency of a dangerous failure of an E/E/PE safety related system to perform the specified safety function over a given period of time

NOTE 1 The term “probability of dangerous failure per hour” is not used in this standard but the acronym PFH has been retained but when it is used it means “average frequency of dangerous failure [h]".

NOTE 2 From a theoretical point of view, the PFH is the average of the *unconditional failure intensity,* also called *failure frequency,* and which is generally designated w(t). It should not be confused with a failure rate (see Annex B of IEC 61508-6).

NOTE 3 When the E/E/PE safety-related system is the ultimate safety layer, the PFH should be calculated from its *unreliability* F(T)=1-R(t) (see “failure rate” above). When it is not the ultimate safety-related system its PFH should be calculated from its *unavailability* U(t) (see PFD above). PFH approximations are given by F(T)/T and 1/MTTF in the first case and 1/MTBF in the second case.

NOTE 4 When the E/E/PE safety-related system implies only quickly repaired revealed failures then an asymptotic failure rate λ_{as} is quickly reached. It provides an estimate of the PFH.

3.6.20
process safety time
period of time between a failure, that has the potential to give rise to a hazardous event, occurring in the EUC or EUC control system and the time by which action has to be completed in the EUC to prevent the hazardous event occurring

3.6.21
mean time to restoration
MTTR
expected time to achieve restoration

NOTE MTTR encompasses:

- the time to detect the failure (a); and,
- the time spent before starting the repair (b); and,
- the effective time to repair (c); and,
- the time before the component is put back into operation (d)

The start time for (b) is the end of (a); the start time for (c) is the end of (b); the start time for (d) is the end of (c).

3.6.22
mean repair time
MRT
expected overall repair time

NOTE MRT encompasses the times (b), (c) and (d) of the times for MTTR (see 3.6.21).

3.7 Lifecycle activities

3.7.1
safety lifecycle
necessary activities involved in the implementation of safety-related systems, occurring during a period of time that starts at the concept phase of a project and finishes when all of the E/E/PE safety-related systems and other risk reduction measures are no longer available for use

NOTE 1 The term “functional safety lifecycle” is more accurate, but the adjective “functional” is not considered necessary in this case within the context of this standard.

NOTE 2 The safety lifecycle models used in this standard are specified in Figures 2, 3 and 4 of IEC 61508-1.

3.7.2
software lifecycle
activities occurring during a period of time that starts when software is conceived and ends when the software is permanently decommissioned

NOTE 1 A software lifecycle typically includes a requirements phase, development phase, test phase, integration phase, installation phase and a modification phase.

NOTE 2 Software is not capable of being maintained; rather, it is modified.

3.7.3
configuration management
discipline of identifying the components of an evolving system for the purposes of controlling changes to those components and maintaining continuity and traceability throughout the lifecycle

NOTE For details on software configuration management see C.5.24 of IEC 61508-7.

3.7.4
configuration baseline
information that allows the software release to be recreated in an auditable and systematic way, including: all source code, data, run time files, documentation, configuration files, and installation scripts that comprise a software release; information about compilers, operating systems, and development tools used to create the software release

3.7.5
impact analysis
activity of determining the effect that a change to a function or component in a system will have to other functions or components in that system as well as to other systems

NOTE In the context of software, see C.5.23 of IEC 61508-7.

3.8 Confirmation of safety measures

3.8.1
verification
confirmation by examination and provision of objective evidence that the requirements have been fulfilled

[ISO 8402, definition 2.17, modified]

NOTE In the context of this standard, verification is the activity of demonstrating for each phase of the relevant safety lifecycle (overall, E/E/PE system and software), by analysis, mathematical reasoning and/or tests, that, for the specific inputs, the outputs meet in all respects the objectives and requirements set for the specific phase.

EXAMPLE Verification activities include

– reviews on outputs (documents from all phases of the safety lifecycle) to ensure compliance with the objectives and requirements of the phase, taking into account the specific inputs to that phase;

– design reviews;

– tests performed on the designed products to ensure that they perform according to their specification;

– integration tests performed where different parts of a system are put together in a step-by-step manner and by the performance of environmental tests to ensure that all the parts work together in the specified manner.

3.8.2
validation
confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled

[ISO 8402, definition 2.18, modified]

NOTE 1 In this standard there are three validation phases:

– overall safety validation (see Figure 2 of IEC 61508-1);

– E/E/PE system validation (see Figure 3 of IEC 61508-1);

– software validation (see Figure 4 of IEC 61508-1).

NOTE 2 Validation is the activity of demonstrating that the safety-related system under consideration, before or after installation, meets in all respects the safety requirements specification for that safety-related system. Therefore, for example, software validation means confirming by examination and provision of objective evidence that the software satisfies the software safety requirements specification.

3.8.3
functional safety assessment
investigation, based on evidence, to judge the functional safety achieved by one or more E/E/PE safety-related systems and/or other risk reduction measures

3.8.4
functional safety audit
systematic and independent examination to determine whether the procedures specific to the functional safety requirements to comply with the planned arrangements are implemented effectively and are suitable to achieve the specified objectives

NOTE A functional safety audit may be carried out as part of a functional safety assessment.

3.8.5
proof test
periodic test performed to detect dangerous hidden failures in a safety-related system so that, if necessary, a repair can restore the system to an “as new” condition or as close as practical to this condition

NOTE 1 In this standard the term “proof test” is used but it is recognised that a synonymous term is “periodical test”.

NOTE 2 The effectiveness of the proof test will be dependent both on failure coverage and repair effectiveness. In practice detecting 100 % of the hidden dangerous failures is not easily achieved for other than low-complexity E/E/PE safety-related systems. This should be the target. As a minimum, all the safety functions which are executed are checked according to the E/E/EP system safety requirements specification. If separate channels are used, these tests are done for each channel separately. For complex elements, an analysis may need to be performed in order to demonstrate that the probability of hidden dangerous failure not detected by proof tests is negligible over the whole life duration of the E/E/EP safety related system.

NOTE 3 A proof test needs some time to be achieved. During this time the E/E/PE safety related system may be inhibited partially or completely. The proof test duration can be neglected only if the part of the E/E/PE safety related system under test remains available in case of a demand for operation or if the EUC is shut down during the test.

NOTE 4 During a proof test, the E/E/PE safety related system may be partly or completely unavailable to respond to a demand for operation. The MTTR can be neglected for SIL calculations only if the EUC is shut down during repair or if other risk measures are put in place with equivalent effectiveness.

3.8.6
diagnostic coverage
DC
fraction of dangerous failures detected by automatic on-line diagnostic tests. The fraction of dangerous failures is computed by using the dangerous failure rates associated with the detected dangerous failures divided by the total rate of dangerous failures

NOTE 1 The dangerous failure diagnostic coverage is computed using the following equation, where *DC* is the diagnostic coverage, λ_{DD} is the detected dangerous failure rate and $\lambda_{D\ total}$ is the total dangerous failure rate:

$$DC = \frac{\sum \lambda_{DD}}{\sum \lambda_{Dtotal}}$$

NOTE 2 This definition is applicable providing the individual components have constant failure rates.

3.8.7
diagnostic test interval
interval between on-line tests to detect faults in a safety-related system that has a specified diagnostic coverage

3.8.8
detected
revealed
overt
in relation to hardware, detected by the diagnostic tests, proof tests, operator intervention (for example physical inspection and manual tests), or through normal operation

EXAMPLE These adjectives are used in detected fault and detected failure.

NOTE A dangerous failure detected by diagnostic test is a revealed failure and can be considered a safe failure only if effective measures, automatic of manual, are taken.

3.8.9
undetected
unrevealed
covert
in relation to hardware, undetected by the diagnostic tests, proof tests, operator intervention (for example physical inspection and manual tests), or through normal operation

EXAMPLE These adjectives are used in undetected fault and undetected failure.

3.8.10
assessor
person, persons or organization that performs the functional safety assessment in order to arrive at a judgement on the functional safety achieved by the E/E/PE safety-related systems and other risk reduction measures

NOTE See also Clause 8 of IEC 61508-1.

3.8.11
independent person
person who is separate and distinct from the activities which take place during the specific phase of the overall, E/E/PE system or software safety lifecycle that is subject to the functional safety assessment or validation, and does not have direct responsibility for those activities

3.8.12
independent department
department that is separate and distinct from the departments responsible for the activities which take place during the specific phase of the overall, E/E/PE system or software safety lifecycle that is subject to the functional safety assessment or validation

3.8.13
independent organisation
organisation that is separate and distinct, by management and other resources, from the organisations responsible for the activities that take place during the specific phase of the overall, E/E/PE system or software safety lifecycle that is subject to the functional safety assessment or validation

3.8.14
animation
simulated operation of the software system (or of some significant portion of the system) to display significant aspects of the behaviour of the system, for instance applied to a requirements specification in an appropriate format or an appropriate high-level representation of the system design

NOTE Animation can give extra confidence that the system meets the real requirements because it improves human recognition of the specified behaviour.

3.8.15
dynamic testing
executing software and/or operating hardware in a controlled and systematic way, so as to demonstrate the presence of the required behaviour and the absence of unwanted behaviour

NOTE Dynamic testing contrasts with static analysis, which does not require the software to be executed or hardware to be in operation.

3.8.16
test harness
facility that is capable of simulating (to some useful degree) the operating environment of software or hardware under development, by applying test cases to the software and recording the response

NOTE The test harness may also include test case generators and facilities to verify the test results (either automatically against values that are accepted as correct or by manual analysis).

3.8.17
safety manual for compliant items
document that provides all the information relating to the functional safety of an element, in respect of specified element safety functions, that is required to ensure that the system meets the requirements of IEC 61508 series

3.8.18
proven in use
demonstration, based on an analysis of operational experience for a specific configuration of an element, that the likelihood of dangerous systematic faults is low enough so that every safety function that uses the element achieves its required safety integrity level

Bibliography

[1] IEC 61511 (all parts), *Functional safety – Safety instrumented systems for the process industry sector*

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

[3] IEC 61800-5-2, *Adjustable speed electrical power drive systems – Part 5-2: Safety requirements – Functional*

[4] IEC 61508-5:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 5: Examples of methods for the determination of safety integrity levels*)

[5] IEC 61508-6:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3*

[6] IEC 61508-7:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures*

[7] ISO/IEC 2382-1:1993, *Information technology – Vocabulary – Part 1: Fundamental terms*

[8] IEC 61131-3:2003, *Programmable controllers – Part 3: Programming languages*

[9] IEC/TR 62059-11, *Electricity metering equipment – Dependability – Part 11:General concepts*

[10] ISO 8402:1994, *Quality management and quality assurance – Vocabulary*

[11] IEC 60601 (all parts), *Medical electrical equipment*

[12] IEC 60050-191:1990, *International Electrotechnical Vocabulary – Chapter 191: Dependability and quality of service*

[13] IEC 60050-351:2006, *International Electrotechnical Vocabulary – Part 351: Control technology*

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

[15] 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*

[16] IEC 61508-3:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements*

[17] ISO/IEC 2382-14:1997, *Information technology – Vocabulary – Part 14: Reliability, maintainability and availability*

[18] ISO 9000:2005, *Quality management systems – Fundamentals and vocabulary*

Index

IEC 61508-5

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 5: Examples of methods for the determination of safety integrity levels

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 5: Exemples de méthodes pour la détermination des niveaux d'intégrité de sécurité

IEC 61508-5:2010

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.
- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.
- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.
- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:
Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm
Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,…). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.
- Just Published CEI: www.iec.ch/online_news/justpub
Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.
- Electropedia: www.electropedia.org
Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.
- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:
Email: csc@iec.ch
Tél.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61508-5

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 5: Examples of methods for the determination of safety integrity levels

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 5: Exemples de méthodes pour la détermination des niveaux d'intégrité de sécurité

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX X

ICS 25.040.40

ISBN 978-2-88910-528-1

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

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

Part 5: Examples of methods for the determination of safety integrity levels

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) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61508-5 has been prepared by subcommittee 65A: System aspects, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 1998. This edition constitutes a technical revision.

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

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

FDIS	Report on voting
65A/552/FDIS	65A/576/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2

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

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

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, though 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.

– 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.

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

Part 5: Examples of methods for the determination of safety integrity levels

1 Scope

1.1 This part of IEC 61508 provides information on

– the underlying concepts of risk and the relationship of risk to safety integrity (see Annex A);

– a number of methods that will enable the safety integrity levels for the E/E/PE safety-related systems to be determined (see Annexes C, D, E, F and G).

The method selected will depend upon the application sector and the specific circumstances under consideration. Annexes C, D, E, F and G illustrate quantitative and qualitative approaches and have been simplified in order to illustrate the underlying principles. These annexes have been included to illustrate the general principles of a number of methods but do not provide a definitive account. Those intending to apply the methods indicated in these annexes should consult the source material referenced.

NOTE For more information on the approaches illustrated in Annexes B, and E, see references [5] and [8] in the Bibliography. See also reference [6] in the Bibliography for a description of an additional approach.

1.2 IEC 61508-1, IEC 61508-2, 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 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 61508-5 plays in the achievement of functional safety for E/E/PE safety-related systems.

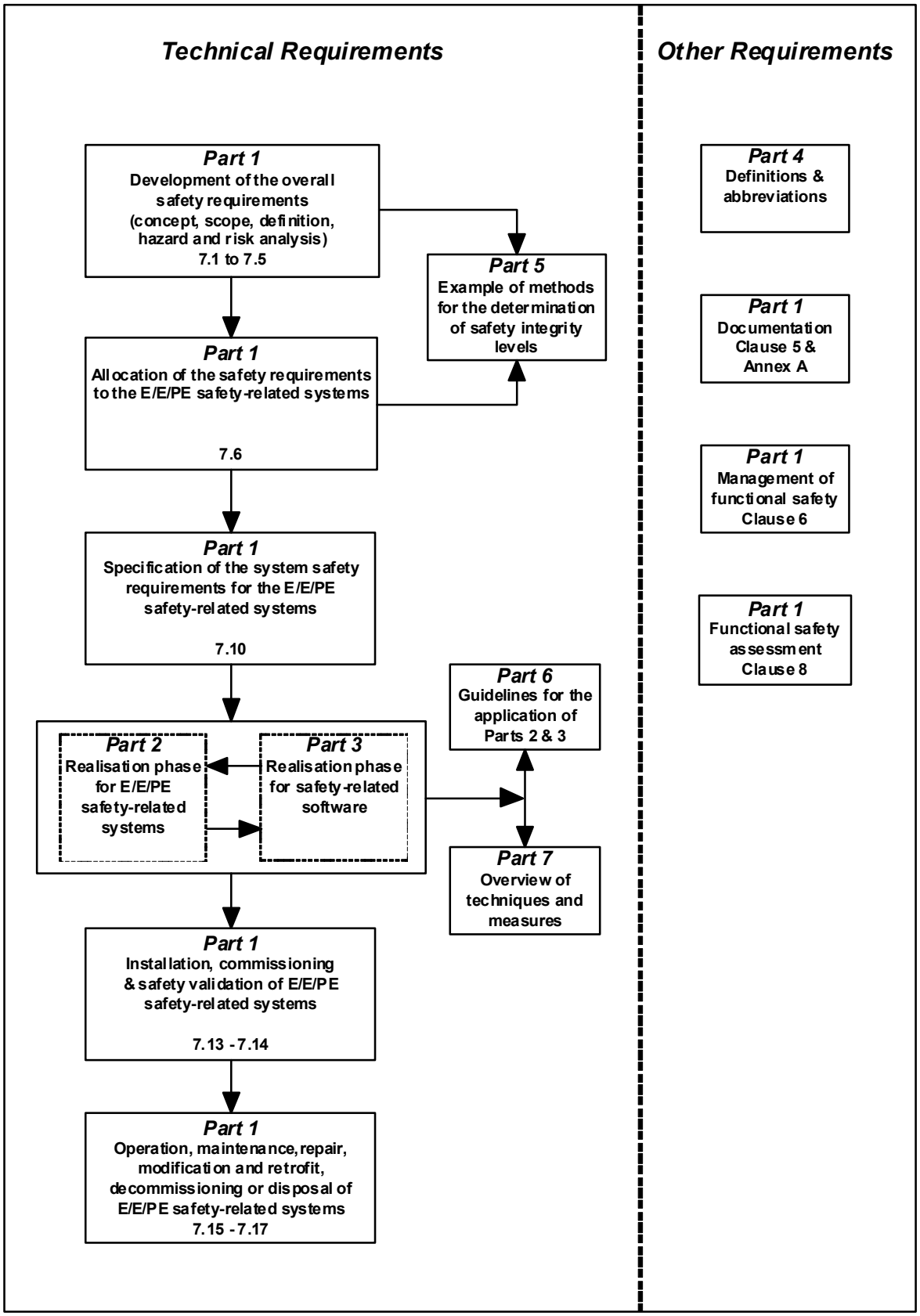


Figure 1 – Overall framework of the IEC 61508 series

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*

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

3 Definitions and abbreviations

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

Annex A
(informative)

Risk and safety integrity – General concepts

A.1 General

This annex provides information on the underlying concepts of risk and the relationship of risk to safety integrity.

A.2 Necessary risk reduction

The necessary risk reduction (see 3.5.18 of IEC 61508-4) is the reduction in risk that has to be achieved to meet the tolerable risk for a specific situation (which may be stated either qualitatively[1] or quantitatively[2]). The concept of necessary risk reduction is of fundamental importance in the development of the safety requirements specification for the E/E/PE safety-related systems (in particular, the safety integrity requirements part of the safety requirements specification). The purpose of determining the tolerable risk for a specific hazardous event is to state what is deemed reasonable with respect to both the frequency (or probability) of the hazardous event and its specific consequences. Safety-related systems are designed to reduce the frequency (or probability) of the hazardous event and/or the consequences of the hazardous event.

The tolerable risk will depend on many factors (for example, severity of injury, the number of people exposed to danger, the frequency at which a person or people are exposed to danger and the duration of the exposure). Important factors will be the perception and views of those exposed to the hazardous event. In arriving at what constitutes a tolerable risk for a specific application, a number of inputs are considered. These include:

– legal requirements, both general and those directly relevant to the specific application;
– guidelines from the appropriate safety regulatory authority;
– discussions and agreements with the different parties involved in the application;
– industry standards and guidelines;
– international discussions and agreements; the role of national and international standards is becoming increasingly important in arriving at tolerable risk criteria for specific applications;
– the best independent industrial, expert and scientific advice from advisory bodies.

In determining the safety integrity requirements of the E/E/PE safety-related system(s) and other risk reduction measures, in order to meet the tolerable frequency of a hazardous event, account needs to be taken of the characteristics of the risk that are relevant to the application. The tolerable frequency will depend on the legal requirements in the country of application and on the criteria specified by the user organisation. Issues that may need to be considered together with how they can be applied to E/E/PE safety-related systems are discussed below.

---

1 In achieving the tolerable risk, the necessary risk reduction will need to be established. Annexes E and G of this document outline qualitative methods, although in the examples quoted the necessary risk reduction is incorporated implicitly by specification of the SIL requirement rather than stated explicitly by a numeric value of risk reduction required.

2 For example, that the hazardous event, leading to a specific consequence, shall not occur with a frequency greater than one in 10^8 h.

A.2.1 Individual risk

Different targets are usually defined for employees and members of the public. The target for individual risk for employees is applied to the most exposed individual and may be expressed as the total risk per year arising from all work activities. The target is applied to a hypothetical person and therefore needs to take into account the percentage of time that the individual spends at work. The target applies to all risks to the exposed person and the tolerable risk for an individual safety function will need to take account of other risks.

Assurance that the total risk is reduced below a specified target can be done in a number of ways. One method is to consider and sum all risks to the most exposed individual. This may be difficult in cases where a person is exposed to many risks and early decisions are needed for system development. An alternative approach is to allocate a percentage of the overall individual risk target to each safety function under consideration. The percentage allocated can usually be decided from previous experience of the type of facility under consideration.

The target applied to an individual safety function should also take into account the conservatism of the method of risk analysis used. All qualitative methods such as risk graphs involve some evaluation of the critical parameters that contribute to risk. The factors that give rise to risk are the consequence of the hazardous event and its frequency. In determining these factors a number of risk parameters may need to be taken into account such as a vulnerability to the hazardous event, number of people who may be affected by the hazardous event, the probability that a person is present when the hazardous event occurs (i.e. occupancy) and probability of avoiding the hazardous event.

Qualitative methods generally involve deciding if a parameter lies within a certain range. The descriptions of the criteria when using such methods will need to be such that there can be a high level of confidence that the target for risks is not exceeded. This can involve setting range boundaries for all parameters so applications with all parameters at the boundary condition will meet the specified risk criteria for safety. This approach to setting the range boundaries is very conservative because there will be very few applications where all parameters will be at the worst case of the range. If members of the public are to be exposed to risk from failure of a E/E/PE safety-related system then a lower target will normally apply.

A.2.2 Societal risk

This arises where multiple fatalities are likely to arise from single events. Such events are called societal because they are likely to provoke a socio-political response. There can be significant public and organisational aversion to high consequence events and this will need to be taken into consideration in some cases. The criterion for societal risk is often expressed as a maximum accumulated frequency for fatal injuries to a specified number of persons. The criterion is normally specified in the form of one or more lines on an F/N plot where F is the cumulative frequency of hazards and N the number of fatalities arising from the hazards. The relationship is normally a straight line when plotted on logarithmic scales. The slope of the line will depend on the extent to which the organisation is risk averse to higher levels of consequence. The requirement will be to ensure the accumulated frequency for a specified number of fatalities is lower than the accumulated frequency expressed in the F/N plot. (see reference [7] in the Bibliography)

A.2.3 Continuous improvement

The principles of reducing risk to as low as reasonably practicable are discussed in Annex C.

A.2.4 Risk profile

In deciding risk criteria to be applied for a specific hazard, the risk profile over the life of the asset may need to be considered. Residual risk will vary from low just after a proof test or a repair has been performed to a maximum just prior to proof testing. This may need to be taken into consideration by organisations that specify the risk criteria to be applied. If proof test intervals are significant, then it may be appropriate to specify the maximum hazard

probability that can be accepted just prior to proof testing or that the PFD(t) or PFH(t) is lower than the upper SIL boundary more than a specified percentage of the time (e.g. 90 %).

A.3 Role of E/E/PE safety-related systems

E/E/PE safety-related systems contribute towards providing the necessary risk reduction in order to meet the tolerable risk.

A safety-related system both

– implements the required safety functions necessary to achieve a safe state for the equipment under control or to maintain a safe state for the equipment under control; and

– is intended to achieve, on its own or with other E/E/PE safety-related systems or other risk reduction measures, the necessary safety integrity for the required safety functions (3.5.1 of IEC 61508-4).

NOTE 1 The first part of the definition specifies that the safety-related system must perform the safety functions which would be specified in the safety functions requirements specification. For example, the safety functions requirements specification may state that when the temperature reaches x, valve y shall open to allow water to enter the vessel.

NOTE 2 The second part of the definition specifies that the safety functions must be performed by the safety-related systems with the degree of confidence appropriate to the application, in order that the tolerable risk will be achieved.

A person could be an integral part of an E/E/PE safety-related system. For example, a person could receive information, on the state of the EUC, from a display screen and perform a safety action based on this information.

E/E/PE safety-related systems can operate in a low demand mode of operation or high demand or continuous mode of operation (see 3.5.16 of IEC 61508-4).

A.4 Safety integrity

Safety integrity is defined as the probability of a safety-related system satisfactorily performing the required safety functions under all the stated conditions within a stated period of time (3.5.4 of IEC 61508-4). Safety integrity relates to the performance of the safety-related systems in carrying out the safety functions (the safety functions to be performed will be specified in the safety functions requirements specification).

Safety integrity is considered to be composed of the following two elements.

– Hardware safety integrity; that part of safety integrity relating to random hardware failures in a dangerous mode of failure (see 3.5.7 of IEC 61508-4). The achievement of the specified level of safety-related hardware safety integrity can be estimated to a reasonable level of accuracy, and the requirements can therefore be apportioned between subsystems using the normal rules for the combination of probabilities. It may be necessary to use redundant architectures to achieve adequate hardware safety integrity.

– Systematic safety integrity; that part of safety integrity relating to systematic failures in a dangerous mode of failure (see 3.5.6 of IEC 61508-4). Although the mean failure rate due to systematic failures may be capable of estimation, the failure data obtained from design faults and common cause failures means that the distribution of failures can be hard to predict. This has the effect of increasing the uncertainty in the failure probability calculations for a specific situation (for example the probability of failure of a safety-related protection system). Therefore a judgement has to be made on the selection of the best techniques to minimise this uncertainty. Note that it is not the case that measures to reduce the probability of random hardware failure will have a corresponding effect on the probability of systematic failure. Techniques such as redundant channels of identical hardware, which are very effective at controlling random hardware failures, are of little use in reducing systematic failures such as software errors.

A.5 Modes of operation and SIL determination

The mode of operation relates to the way in which a safety function is intended to be used with respect to the frequency of demands made upon it which may be either:

– **low demand mode:** where frequency of demands for operation made on the safety function is no greater than one per year; or

– **high demand mode:** where frequency of demands for operation made on the safety function is greater than one per year; or

– **continuous mode:** where demand for operation of the safety function is continuous.

Tables 2 and 3 of IEC 61508-1 detail the target failure measures associated with the four safety integrity levels for each of the modes of operation. The modes of operation are explained further in the following paragraphs.

A.5.1 Safety integrity and risk reduction for low demand mode applications

The required safety integrity of the E/E/PE safety-related systems and other risk reduction measures shall be of such a level so as to ensure that:

– the average probability of failure on demand of the safety-related systems is sufficiently low to prevent the hazardous event frequency exceeding that required to meet the tolerable risk; and/or

– the safety-related systems modify the consequences of failure to the extent required to meet the tolerable risk.

Figure A.1 illustrates the general concepts of risk reduction. The general model assumes that:

– there is an EUC and a control system;

– there are associated human factor issues;

– the safety protective features comprise:

  – E/E/PE safety-related systems;

  – other risk reduction measures.

NOTE Figure A.1 is a generalised risk model to illustrate the general principles. The risk model for a specific application will need to be developed taking into account the specific manner in which the necessary risk reduction is actually being achieved by the E/E/PE safety-related systems and/or other risk reduction measures. The resulting risk model may therefore differ from that shown in Figure A.1.

The various risks indicated in Figure A.1 and A.2 are as follows:

– EUC risk: the risk existing for the specified hazardous events for the EUC, the EUC control system and associated human factor issues: no designated safety protective features are considered in the determination of this risk (see 3.1.9 of IEC 61508-4);

– tolerable risk; the risk which is accepted in a given context based on the current values of society (see 3.1.7 of IEC 61508-4);

– residual risk: in the context of this standard, the residual risk is that remaining for the specified hazardous events for the EUC, the EUC control system, human factor issues but with the addition of, E/E/PE safety-related systems and other risk reduction measures (see also 3.1.7 of IEC 61508-4).

The EUC risk is a function of the risk associated with the EUC itself but taking into account the risk reduction brought about by the EUC control system. To prevent unreasonable claims for the safety integrity of the EUC control system, this standard places constraints on the claims that can be made (see 7.5.2.5 of IEC 61508-1).

The necessary risk reduction is achieved by a combination of all the safety protective features. The necessary risk reduction to achieve the specified tolerable risk, from a starting

point of the EUC risk, is shown in Figure A.1 (relevant for a safety function operating in low demand mode of operation).

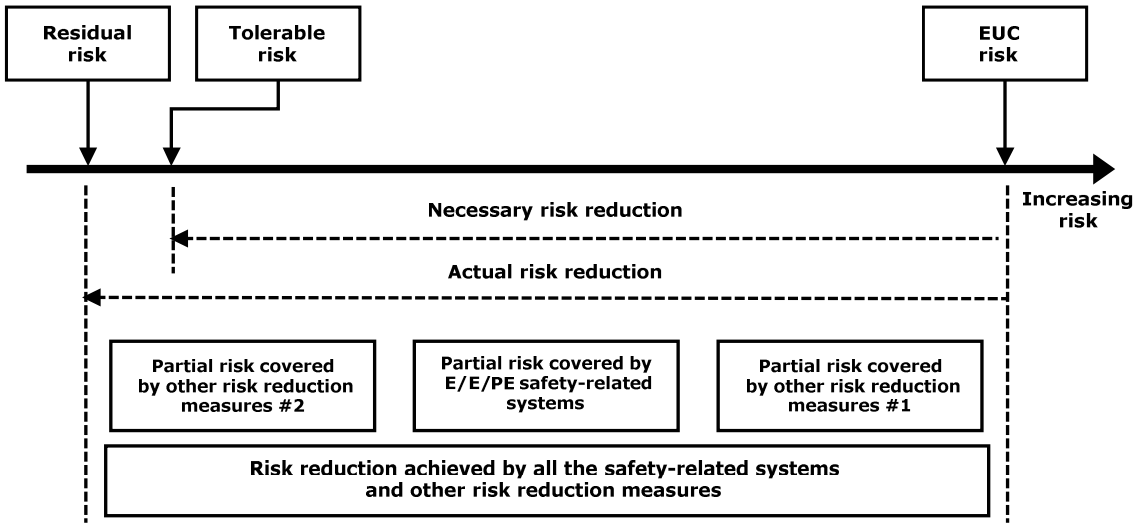


Figure A.1 – Risk reduction – general concepts (low demand mode of operation)

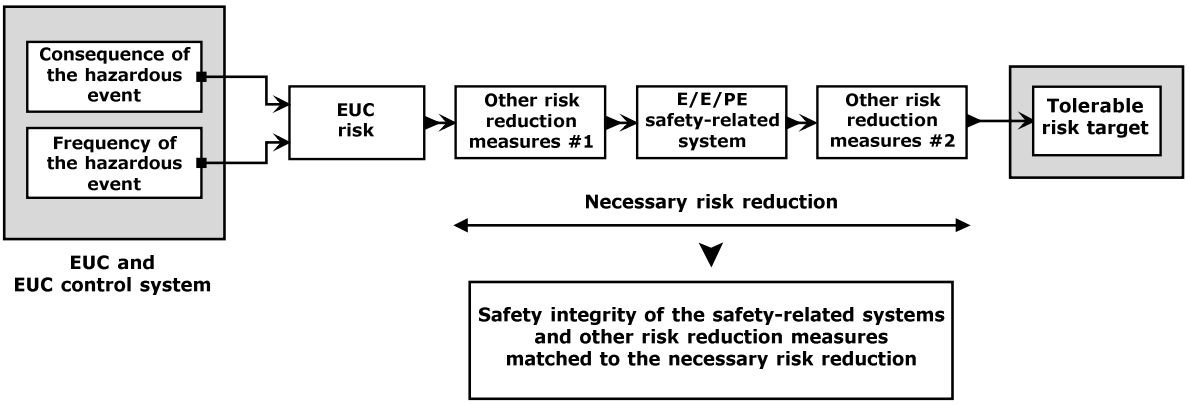


Figure A.2 – Risk and safety integrity concept

A.5.2 Safety integrity for high demand mode applications

The required safety integrity of the E/E/PE safety-related systems and other risk reduction measures shall be of such a level to ensure that:

– the average probability of failure on demand of the safety-related systems is sufficiently low to prevent the hazardous event frequency exceeding that required to meet the tolerable risk; and/or

– the average probability of failure per hour of the safety-related system is sufficiently low to prevent the hazardous event frequency exceeding that required to meet the tolerable risk.

Figure A.3 illustrates the general concepts of high demand applications. The general model assumes that:

– there is a EUC and a control system;

– there are associated human factor issues;
– the safety protective features comprise:
  – E/E/PE safety-related system operating in high demand mode;
  – other risk reduction measures.

Various demands on the E/E/PE safety related systems can occur as follows:

– general demands from the EUC;
– demands arising from failures in the EUC control system;
– demands arising from human failures.

If the total demand rate arising from all the demands on the system exceeds 1 per year then the critical factor is the dangerous failure rate of the E/E/PE safety-related system. Residual hazard frequency can never exceed the dangerous failure rate of the E/E/PE safety-related system. It can be lower if other risk reduction measures reduce the probability of harm.

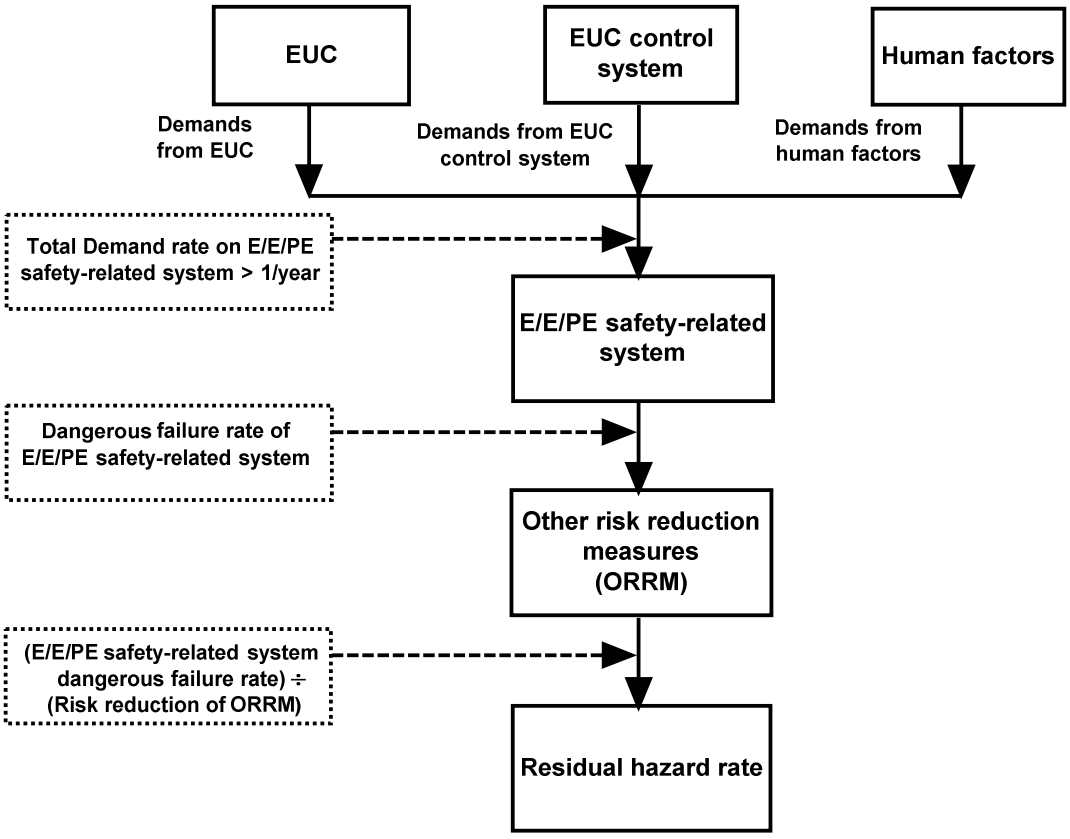


Figure A.3 – Risk diagram for high demand applications

A.5.3 Safety integrity for continuous mode applications

The required safety integrity of the E/E/PE safety-related systems and any other risk reduction measures shall be of such a level to ensure that the average probability of a dangerous failure per hour of the safety-related system is sufficiently low to prevent the hazardous event frequency exceeding that required to meet the tolerable risk.

With an E/E/PE safety-related system operating in continuous mode, other risk reduction measures can reduce the residual hazard frequency according to the risk reduction provided The model is shown in Figure A.4.

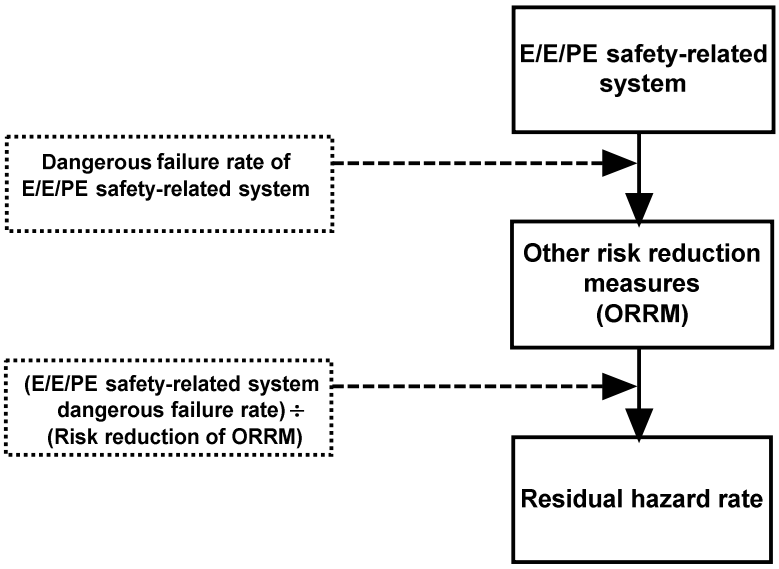


Figure A.4 – Risk diagram for continuous mode operation

A.5.4 Common cause and dependency failures

During the determination of the safety integrity levels it is important to take account of common cause and dependency failures. The models shown above in Figures A.1, A.2, A.3 and A.4 are drawn on the basis that each safety system relevant to the same hazard is fully independent. There are many applications where this is not the case. Examples include the following:

1) Where a dangerous failure of an element within the EUC control system can cause a demand on a safety-related system and the safety-related system uses an element subject to failure from the same cause. An example of this could be where the control and protection system sensors are separate but common cause could lead to failure of both (see Figure A.5).
2) Where more than one safety-related system is used and some of the same type of equipment is used within each safety-related system and each is subject to failure from the same common cause. An example would be where the same type of sensor is used in two separate protection systems both providing risk reduction for the same hazard (see Figure A.6).
3) Where more than one protection system is used, the protection systems are diverse but proof testing is carried out on all the systems on a synchronous basis. In such cases the actual PFD_{avg} achieved by the combination of multiple systems will be significantly higher than the PFD_{avg} suggested by the multiplication of the PFD_{avg} of the individual systems.
4) Where the same individual element is used as part of the control system and the safety-related system.
5) Where more than one protection system is used and where the same individual element is used as part of more than one system.

In such cases the effect of common cause/dependency will need to be considered. Consideration should be given as to whether the final arrangement is capable of meeting the necessary systematic capability and the necessary probability of dangerous random hardware failure rates relating to the overall risk reduction required. The effect of common cause failures is difficult to determine and often requires the construction of special purpose models (e.g. fault tree or Markov models).

The effect of common cause is likely to be more significant in applications involving high safety integrity levels. In some applications it may be necessary to incorporate diversity so that common cause effects are minimised. It should however be noted that incorporation of diversity can lead to problems during design, maintenance and modification. Introducing diversity can lead to errors due to the unfamiliarity and lack of operation experience with the diverse devices.

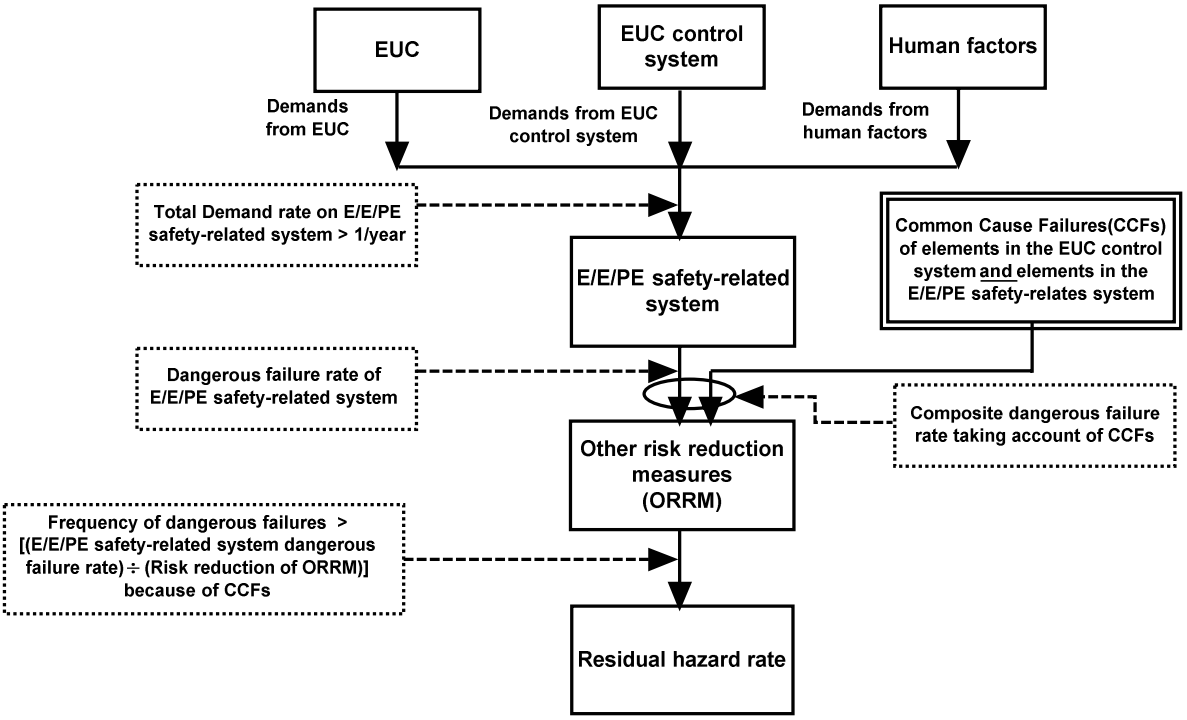


Figure A.5 – Illustration of common cause failures (CCFs) of elements in the EUC control system and elements in the E/E/PE safety-related system

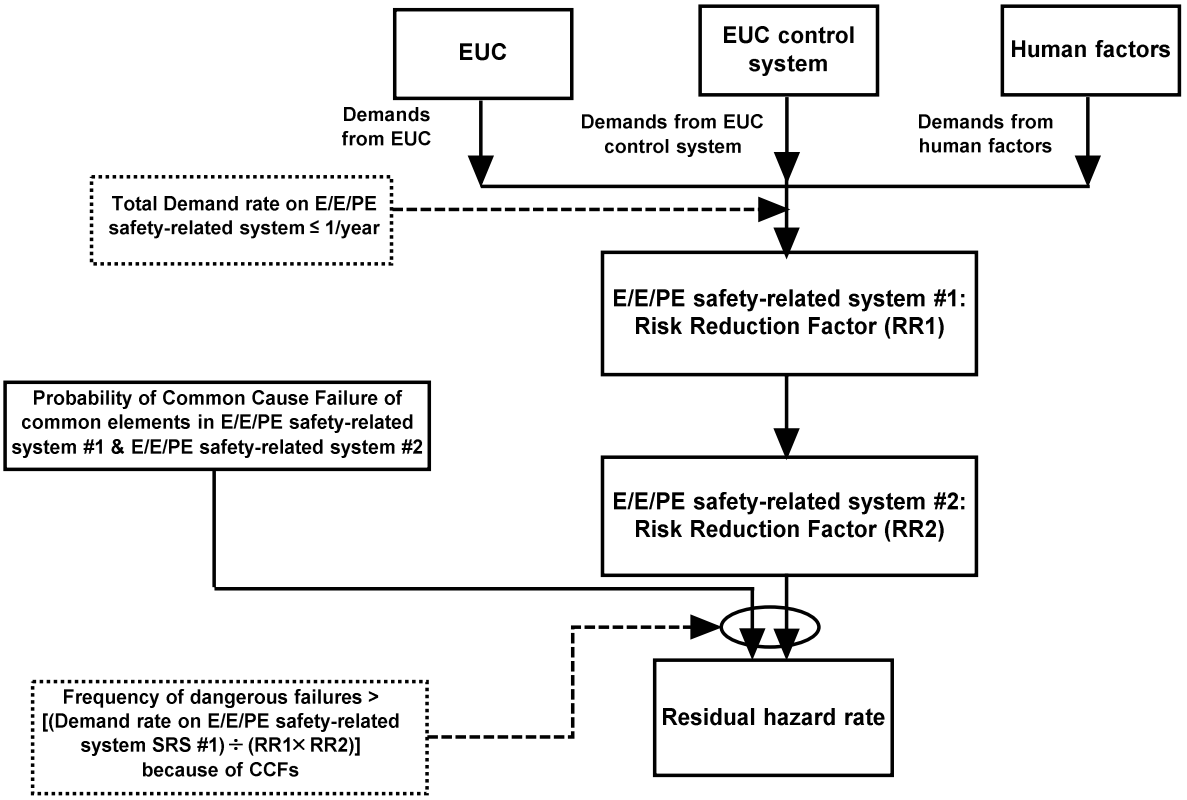


Figure A.6 – Common cause between two E/E/PE safety-related systems

A.5.5 Safety Integrity levels when multiple layers of protection are used

When multiple layers of protection are used to achieve a tolerable risk there may be interactions between systems themselves and also between systems and causes of demand. As discussed above in A.5.4 there are always concerns about test (de)synchronisation and common cause failures since these can be significant factors when overall risk reduction requirements are high or where demand frequency is low. Evaluation of the interactions between safety layers and between safety layers and causes of demand can be complex and may need the development of a holistic model (e.g. as described in ISO/IEC 31010) and based, for example on a top down approach with the top event specified as the tolerable hazard frequency. The model may include all safety layers for calculating the actual risk reduction and all causes of demand for calculating the actual frequency of accident. This allows the identification of minimal cut sets (i.e. failure scenarios), reveals the weak points (i.e. the shortest minimal cut sets: single, double failures, etc.) in the arrangement of systems and facilitates system improvement through sensitivity analysis.

A.6 Risk and safety integrity

It is important that the distinction between risk and safety integrity be fully appreciated. Risk is a measure of the probability and consequence of a specified hazardous event occurring. This can be evaluated for different situations (EUC risk, risk reduction required to meet the tolerable risk, actual risk (see Figure A.1). The tolerable risk is determined by consideration of the issues described in A.2. Safety integrity applies solely to the E/E/PE safety-related systems and other risk reduction measures and is a measure of the likelihood of those systems/facilities satisfactorily achieving the necessary risk reduction in respect of the specified safety functions. Once the tolerable risk has been set, and the necessary risk reduction estimated, the safety integrity requirements for the safety-related systems can be allocated (see 7.4, 7.5 and 7.6 of IEC 61508-1).

NOTE The allocation is necessarily iterative in order to optimize the design to meet the various requirements.

A.7 Safety integrity levels and software systematic capability

To cater for the wide range of necessary risk reductions that the safety-related systems have to achieve, it is useful to have available a number of safety integrity levels as a means of satisfying the safety integrity requirements of the safety functions allocated to the safety-related systems. Software systematic capability is used as the basis of specifying the safety integrity requirements of the safety functions implemented in part by safety-related software. The safety integrity requirements specification should specify the safety integrity levels for the E/E/PE safety-related systems.

In this standard, four safety integrity levels are specified, with safety integrity level 4 being the highest level and safety integrity level 1 being the lowest.

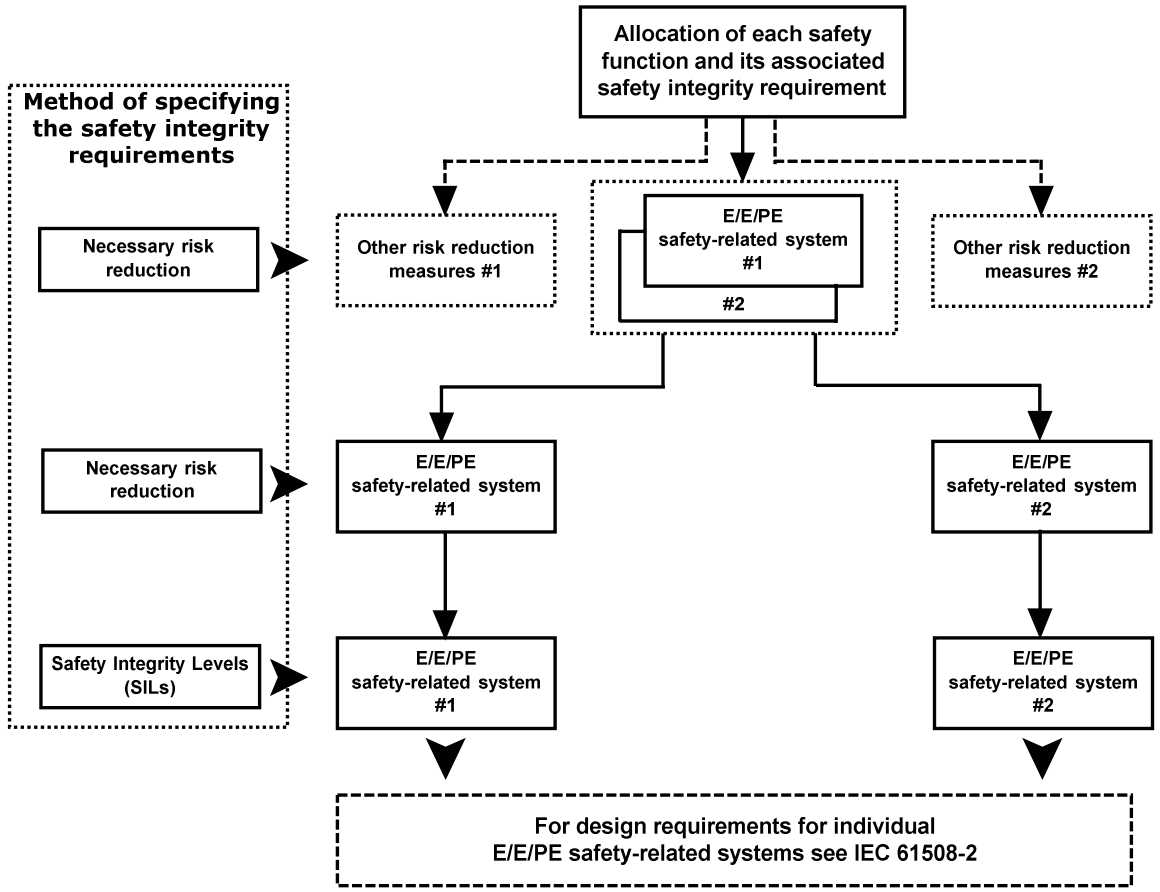
The safety integrity level target failure measures for the four safety integrity levels are specified in Tables 2 and 3 of IEC 61508-1. Two parameters are specified, one for safety-related systems operating in a low demand mode of operation and one for safety-related systems operating in a high demand or continuous mode of operation.

NOTE For safety-related systems operating in a low demand mode of operation, the safety integrity measure of interest is the probability of failure to perform its design function on demand. For safety-related systems operating in a high demand or continuous mode of operation, the safety integrity measure of interest is the average probability of a dangerous failure per hour (see 3.5.16 and 3.5.17 of IEC 61508-4).

A.8 Allocation of safety requirements

The allocation of safety requirements (both the safety functions and the safety integrity requirements) to the E/E/PE safety-related systems, other technology safety-related systems and other risk reduction measures is shown in Figure A.7 (this is identical to Figure 6 of IEC 61508-1). The requirements for the safety requirements allocation phase are given in 7.6 of IEC 61508-1.

The methods used to allocate the safety integrity requirements to the E/E/PE safety-related systems, other technology safety-related systems and other risk reduction measures depend, primarily, upon whether the necessary risk reduction is specified explicitly in a numerical manner or in a qualitative manner. These approaches are termed quantitative and qualitative methods respectively (see Annexes C, D, E, F and G).



NOTE 1 Safety integrity requirements are associated with each safety function before allocation (see 7.5.2.3 and 7.5.2.4 of IEC 61508-1).

NOTE 2 A safety function may be allocated across more than one safety-related system.

Figure A.7 – Allocation of safety requirements to the E/E/PE safety-related systems, and other risk reduction measures

A.9 Mitigation systems

Mitigation systems take action in the event of full or partial failure of other safety-related systems such as E/E/PE safety-systems. The objective is to reduce the consequences associated with a hazardous event rather than its frequency. Examples of mitigation systems include fire and gas systems (detection of fire/gas and subsequent action to put the fire out (e.g. by water deluge), and airbag systems in an automobile.

When determining the safety integrity requirements it should be recognised that when making judgments on the severity of the consequence, only the incremental consequences should be considered. That is, determine the increase in the severity of the consequence if the function did not operate over that when it does operate as intended. This can be done by first considering the consequences if the system fails to operate and then considering what difference will be made if the mitigation function operates correctly. In considering the consequences if the system fails to operate there will normally be a number of outcomes all with different probabilities. Event tree analysis (ETA) may be a useful tool for this.

NOTE Guidance on the determination of safety integrity levels for fire and gas and emergency shut down systems is included in Annex B of ISO 10418.

Annex B
(informative)

Selection of methods for determining safety integrity level requirements

B.1 General

This annex lists a number of techniques that can be used for determination of safety integrity levels. None of the methods are suitable for all applications and users will need to select the most suitable. In selecting the most appropriate method consideration should be given to the following factors:

1) the risk acceptance criteria that need to be met. Some of the techniques will not be suitable if it is required to demonstrate that risk has been reduced to as low as reasonably practicable;
2) the mode of operation of the safety function. Some methods are only suitable for low demand mode;
3) the knowledge and experience of the persons undertaking the SIL determination and what has been the traditional approach in the sector;
4) the confidence needed that the resulting residual risk meets the criteria specified by the user organisation. Some of the methods can be linked back to quantified targets but some approaches are qualitative only;
5) more than one method may be used. One method may be used for screening purposes followed by another more rigorous approach if the screening method shows the need for high safety integrity levels;
6) the severity of the consequences. More rigorous methods may be selected for consequences that include multiple fatalities;
7) whether common cause occurs between the E/E/PE safety related systems or between the E/E/PE safety related system and demand causes.

Whatever method is used all assumptions should be recorded for future safety management. All decisions should be recorded so that the SIL assessment can be verified and be subject to independent functional safety assessment.

B.2 The ALARP method

The ALARP principles may be used on its own or with other methods to determine the SIL requirements for a safety function. It can be used in a qualitative or quantitative way. When used in a qualitative way the SIL requirements for a specified safety function are increased until the frequency of occurance is reduced such that the conditions associated with Class II or Class III risk class are satisfied. When used in a quantitative way frequencies and consequences are specified numerically and the SIL requirements increased until it can be shown that the additional capital and operating cost associated with implementing a higher SIL would meet the condition associated with Class II or Class III risk class (see Figure C.1).

In using the ALARP method the boundary between the intolerable region and the ALARP region will need to be considered.

B.3 Quantitative method of SIL determination

The quantitative method is described in Annex D. It may be used together with the ALARP method described in Annex C.

The quantitative method can be used for both simple and complex applications. With complex applications, fault trees can be constructed to represent the hazard model. The top event will generally be one or more fatalities and logic constructed to represent demand causes and failures of the E/E/PE safety related systems that lead to the top event. Software tools are available to allow modeling of common cause if the same type of equipment is used for control and protection functions. In some complex applications, a single failure event may occur in more than one place in the fault tree and this will require a boolean reduction to be carried out. The tools also facilitate sensitivity analysis that shows the dominant factors that influence the frequency of the top event. SIL can be established by determining the required risk reduction to achieve the tolerable risk criteria.

The method is suitable for safety functions operating in continuous/high demand mode and low demand mode. The method normally results in low SILs because the risk model is specifically designed for each application and numeric values are used to represent each risk factor rather than the numeric ranges used in calibrated risk graphs. Quantitative methods however require the construction of a specific model for each hazardous event. Modeling requires skill, tools and knowledge of the application and can take considerable time to develop and verify.

The method facilitates demonstration that risk has been reduced to as low as reasonably practicable. This can be done by considering options for further risk reduction, integrating the additional facilities in the fault tree model and then determining the reduction in risk and comparing this with the cost of the option.

B.4 The risk graph method

The risk graph qualitative method is described in Annex E. The method enables the safety integrity level to be determined from knowledge of the risk factors associated with the EUC and the EUC control system. A number of parameters are introduced which together describe the nature of the hazardous situation when safety related systems fail or are not available. One parameter is chosen from each of four sets, and the selected parameters are then combined to decide the safety integrity level allocated to the safety functions. The method has been used extensively within the machinery sector, see ISO 14121-2 and Annex A of ISO 13849-1.

The method can be qualitative in which case the selection of the parameters is subjective and requires considerable judgment. The residual risk cannot be calculated from knowledge of the parameter values. It will not be suitable if an organisation requires confidence that residual risk is reduced to a specified quantitative value.

The parameters descriptions can include numeric values that are derived by calibrating the risk graph against numeric tolerability risk criteria. The residual risk can be calculated from numeric values used for each of the parameters. It will be suitable if an organisation requires confidence that residual risk is reduced to a specified quantitative value. Experience has shown that use of the calibrated risk graph method can result in high safety integrity levels. This is because calibration is usually carried out using worst case values of each parameter. Each parameter has a decade range so that for applications where all the parameters are average for the range, the SIL will be one higher than necessary for tolerable risk. The method is extensively used in the process and offshore sector.

The risk graph method does not take into account common cause failures between causes of demand and cause of the E/E/PE safety related system failure or common cause issues with other layers of protection.

B.5 Layer of protection analysis (LOPA)

The basic method is described in a number of books and the technique can be used in a number of different forms. A technique that can be used for SIL determination is described in Annex F.

The method is quantitative and the user will need to decide the tolerable frequencies for each consequence severity level. Numeric credit is given for protection layers that reduce the frequency of individual demand causes. Not all protection layers are relevant to all demand causes, so the technique can be used for more complex applications. The numeric values assigned to protection layers can be rounded up to the next significant figure or the next significant decade range. If numeric values of protection layers are rounded to the next significant figure, then the method on average gives lower requirements for risk reduction and lower SIL values than calibrated risk graphs.

Since numeric targets are assigned to specified consequence severity levels, the user can have confidence that residual risk meets corporate criteria.

The method as described is not suitable for functions that operate in continuous mode and does not take account of common cause failure between causes of demand and the E/E/PE safety related systems. The method can however be adjusted so as to be suitable for such cases.

B.6 Hazardous event severity matrix

The hazard event severity method is described in Annex G. An inherent assumption is that when a protection layer is added that an order of magnitude risk reduction is achieved. A further assumption is that protection layers are independent of demand cause and independent of each other. The method as described is not suitable for functions that operate in continuous mode. The method can be qualitative in which case the selection of the risk factors is subjective and requires considerable judgment. The residual risk cannot be calculated from knowledge of the risk factors selected. It will not be suitable if an organization requires confidence that residual risk is reduced to a specified quantitative value.

Annex C
(informative)

ALARP and tolerable risk concepts

C.1 General

This annex considers one particular approach to the achievement of a tolerable risk. The intention is not to provide a definitive account of the method but rather an illustration of the general principles. The approach includes a process of continuous improvement where all options that would reduce risk further are considered in terms of benefits and costs. Those intending to apply the methods indicated in this annex should consult the source material referenced (see reference [7] in the Bibliography).

C.2 ALARP model

C.2.1 Introduction

Clause C.2 outlines the main tests that are applied in regulating industrial risks and indicates that the activities involve determining whether:

a) the risk is so great that it shall be refused altogether; or

b) the risk is, or has been made, so small as to be insignificant; or

c) the risk falls between the two states specified in a) and b) above and has been reduced to the lowest practicable level, bearing in mind the benefits resulting from its acceptance and taking into account the costs of any further reduction.

With respect to c), the ALARP principle requires that any risk shall be reduced so far as is reasonably practicable, or to a level which is as low as reasonably practicable (these last 5 words form the abbreviation ALARP). If a risk falls between the two extremes (i.e. the unacceptable region and broadly acceptable region) and the ALARP principle has been applied, then the resulting risk is the tolerable risk for that specific application. This three zone approach is shown in Figure C.1.

Above a certain level, a risk is regarded as intolerable and cannot be justified in any ordinary circumstance.

Below that level, there is the tolerability region where an activity is allowed to take place provided the associated risks have been made as low as reasonably practicable. Tolerable here is different from acceptable: it indicates a willingness to live with a risk so as to secure certain benefits, at the same time expecting it to be kept under review and reduced as and when this can be done. Here a cost benefit assessment is required either explicitly or implicitly to weigh the cost and the need or otherwise for additional safety measures. The higher the risk, the more proportionately would be expected to be spent to reduce it. At the limit of tolerability, expenditure in gross disproportion to the benefit would be justified. Here the risk will by definition be substantial, and equity requires that a considerable effort is justified even to achieve a marginal reduction.

Where the risks are less significant, proportionately less needs to be spent in order to reduce them and at the lower end of the tolerability region, a balance between costs and benefits will suffice.

Below the tolerability region is the broadly acceptable region where the risks are small in comparison with the everyday risks we all experience. While in the broadly acceptable region, there is no need for a detailed working to demonstrate ALARP, it is, however, necessary to remain vigilant to ensure that the risk remains at this level.

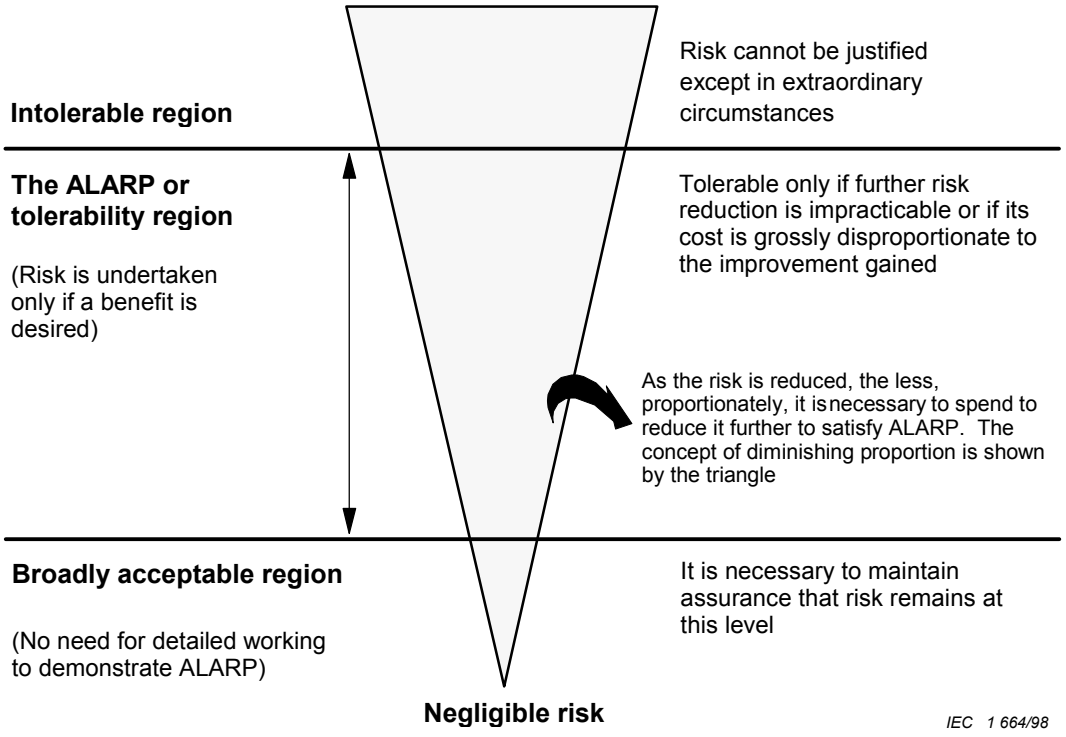


Figure C.1 – Tolerable risk and ALARP

The concept of ALARP can be used when qualitative or quantitative risk targets are adopted. Subclause C.2.2 outlines a method for quantitative risk targets. (Annex D and F outline quantitative methods and Annexes E and G outline qualitative methods for the determination of the necessary risk reduction for a specific hazard. The methods indicated could incorporate the concept of ALARP in the decision making.)

NOTE Further information on ALARP is given in reference [7] in the Bibliography.

C.2.2 Tolerable risk target

One way in which a tolerable risk target can be obtained is for a number of consequences to be determined and tolerable frequencies allocated to them. This matching of the consequences to the tolerable frequencies would take place by discussion and agreement between the interested parties (for example safety regulatory authorities, those producing the risks and those exposed to the risks).

To take into account ALARP concepts, the matching of a consequence with a tolerable frequency can be done through risk classes. Table C.1 is an example showing four risk classes (I, II, III, IV) for a number of consequences and frequencies. Table C.2 interprets each of the risk classes using the concept of ALARP. That is, the descriptions for each of the four risk classes are based on Figure C.1. The risks within these risk class definitions are the risks that are present when risk reduction measures have been put in place. With respect to Figure C.1, the risk classes are as follows:

– risk class I is in the intolerable region;
– risk classes II and III are in the ALARP region, risk class II being just inside the ALARP region;
– risk class IV is in the broadly acceptable region.

For each specific situation, or sector comparable industries, a table similar to Table C.1 would be developed taking into account a wide range of social, political and economic factors. Each consequence would be matched against a frequency and the table populated by the risk classes. For example, frequent in Table C.1 could denote an event that is likely to be

continually experienced, which could be specified as a frequency greater than 10 per year. A critical consequence could be a single death and/or multiple severe injuries or severe occupational illness.

Table C.1 – Example of risk classification of accidents

Frequency	Consequence			
	Catastrophic	**Critical**	**Marginal**	**Negligible**
Frequent	I	I	I	II
Probable	I	I	II	III
Occasional	I	II	III	III
Remote	II	III	III	IV
Improbable	III	III	IV	IV
Incredible	IV	IV	IV	IV
NOTE 1 The actual population with risk classes I, II, III and IV will be sector dependent and will also depend upon what the actual frequencies are for frequent, probable, etc. Therefore, this table should be seen as an example of how such a table could be populated, rather than as a specification for future use. NOTE 2 Determination of the safety integrity level from the frequencies in this table is outlined in Annex D.				

Table C.2 – Interpretation of risk classes

Risk class	Interpretation
Class I	Intolerable risk
Class II	Undesirable risk, and tolerable only if risk reduction is impracticable or if the costs are grossly disproportionate to the improvement gained
Class III	Tolerable risk if the cost of risk reduction would exceed the improvement gained
Class IV	Negligible risk

Annex D
(informative)

Determination of safety integrity levels – A quantitative method

D.1 General

This annex outlines how the safety integrity levels can be determined if a quantitative approach is adopted and illustrates how the information contained in tables such as Table C.1 can be used. A quantitative approach is of particular value when:

– the tolerable risk is to be specified in a numerical manner (for example that a specified consequence should not occur with a greater frequency than one in 10^4 years);

– numerical targets have been specified for the safety integrity levels for the safety-related systems. Such targets have been specified in this standard (see Tables 2 and 3 of IEC 61508-1).

This annex is not intended to be a definitive account of the method but is intended to illustrate the general principles. It is particularly applicable when the risk model is as indicated in Figures A.1 and A.2.

D.2 General method

The model used to illustrate the general principles is that shown in Figure A.1. The key steps in the method are as follows and will need to be done for each safety function to be implemented by the E/E/PE safety-related system:

– determine the tolerable risk from a table such as Table C.1;

– determine the EUC risk;

– determine the necessary risk reduction to meet the tolerable risk;

– allocate the necessary risk reduction to the E/E/PE safety-related systems, other technology safety-related systems and other risk reduction measures (see 7.6 of IEC 61508-1).

Table C.1 is populated with risk frequencies and allows a numerical tolerable risk target (F_t) to be specified.

The frequency associated with the risk that exists for the EUC, including the EUC control system and human factor issues (the EUC risk), without any protective features, can be estimated using quantitative risk assessment methods. This frequency with which a hazardous event could occur without protective features present (F_{np}) is one of two components of the EUC risk; the other component is the consequence of the hazardous event. F_{np} may be determined by:

– analysis of failure rates from comparable situations;

– data from relevant databases;

– calculation using appropriate predictive methods.

This standard places constraints on the minimum failure rates that can be claimed for the EUC control system (see 7.5.2.5 of IEC 61508-1). If it is to be claimed that the EUC control system has a failure rate less than these minimum failure rates, then the EUC control system shall be considered a safety-related system and shall be subject to all the requirements for safety-related systems in this standard.

D.3 Example calculation

Figure D.1 provides an example of how to calculate the target safety integrity for a single safety-related protection system. For such a situation

$$PFD_{avg} \leq F_t / F_{np}$$

where

PFD_{avg} is the average probability of failure on demand of the safety-related protection system, which is the target failure measure for safety-related protection systems operating in a low demand mode of operation (see Table 2 of IEC 61508-1 and 3.5.16 of IEC 61508-4);

F_t is the tolerable hazard frequency;

F_{np} is the demand rate on the safety-related protection system.

Also in Figure D.1:

– C is the consequence of the hazardous event;

– F_p is the risk frequency with the protective features in place.

It can be seen that determination of F_{np} for the EUC is important because of its relationship to PFD_{avg} and hence to the safety integrity level of the safety-related protection system.

The necessary steps in obtaining the safety integrity level (when the consequence C remains constant) are given below (as in Figure D.1), for the situation where the entire necessary risk reduction is achieved by a single safety-related protection system which must reduce the hazard rate, as a minimum, from F_{np} to F_t:

– determine the frequency element of the EUC risk without the addition of any protective features (F_{np});

– determine the consequence C without the addition of any protective features;

– determine, by use of Table C.1, whether for frequency F_{np} and consequence C a tolerable risk level is achieved. If, through the use of Table C.1, this leads to risk class I, then further risk reduction is required. Risk class IV or III would be tolerable risks. Risk class II would require further investigation;

NOTE Table C.1 is used to check whether or not further risk reduction measures are necessary, since it may be possible to achieve a tolerable risk without the addition of any protective features.

– determine the probability of failure on demand for the safety-related protection system (PFD_{avg}) to meet the necessary risk reduction (ΔR). For a constant consequence in the specific situation described, $PFD_{avg} = (F_p / F_{np}) = \Delta R$;

– for $PFD_{avg} = (F_p / F_{np})$, the safety integrity level can be obtained from Table 2 of IEC 61508-1 (for example, for $PFD_{avg} = 10^{-2} – 10^{-3}$, the safety integrity level = 2).

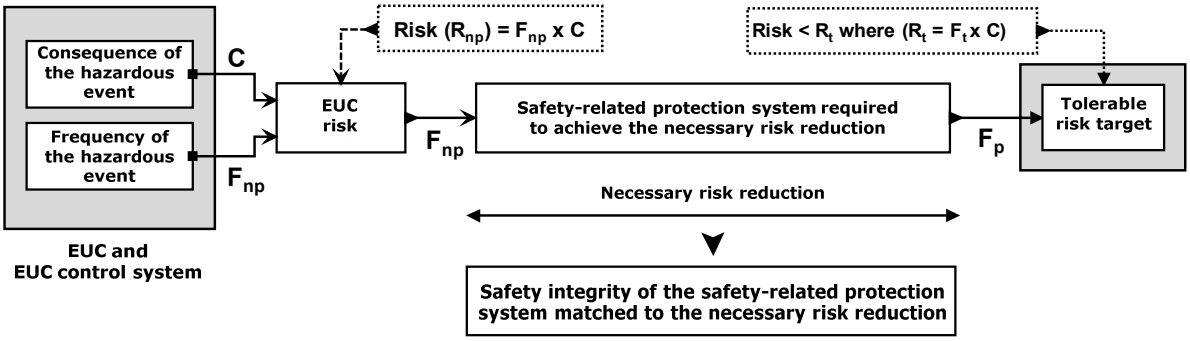


Figure D.1 – Safety integrity allocation – example for safety-related protection system

Annex E
(informative)

Determination of safety integrity levels – Risk graph methods

E.1 General

This annex describes the risk graph method, which is a method that enables the safety integrity level of a safety-related system to be determined from a knowledge of the risk factors associated with the EUC and the EUC control system. It is particularly applicable when the risk model is as indicated in Figures A.1 and A.2. The method can be used on a qualitative or quantitative basis.

Where this approach is adopted, in order to simplify matters a number of parameters are introduced which together describe the nature of the hazardous situation when safety-related systems fail or are not available. One parameter is chosen from each of four sets, and the selected parameters are then combined to decide the safety integrity level allocated to the safety functions. These parameters

– allow a meaningful graduation of the risks to be made; and
– contain the key risk assessment factors.

This annex is not intended to be a definitive account of the method but is intended to illustrate the general principles.

E.2 Risk graph synthesis

The following simplified procedure is based on the following equation:

$$R = (f) \text{ of a specified } (C)$$

where

R is the risk with no safety-related systems in place;

f is the frequency of the hazardous event with no safety-related systems in place;

C is the consequence of the hazardous event (the consequences could be related to harm associated with health and safety or harm from environmental damage).

The frequency of the hazardous event f is, in this case, considered to be made up of three influencing factors:

– frequency of, and exposure time in, the hazardous zone;
– the possibility of avoiding the hazardous event;
– the probability of the hazardous event taking place without the addition of any safety-related systems (but having in place other risk reduction facilities) – this is termed the probability of the unwanted occurrence.

This produces the following four risk parameters:

– consequence of the hazardous event (C);
– frequency of, and exposure time in, the hazardous zone (F);
– possibility of failing to avoid the hazardous event (P);
– probability of the unwanted occurrence (W).

The risk parameters may be decided on a qualitative basis as described in Table E.1 or on a quantitative basis as described in Table E.2. In deciding the numeric values associated with each parameter in Table E.2 a calibration process will be required.

E.3 Calibration

The objectives of the calibration process are as follows:

- to describe all parameters in such a way as to enable the SIL assessment team to make objective judgments based on the characteristics of the application;
- to ensure the SIL selected for an application is in accordance with corporate risk criteria and takes account of risks from other sources;
- to enable the parameter selection process to be verified.

Calibration of the risk graph is the process of assigning numerical values to risk graph parameters. This forms the basis for the assessment of the existing process risk and allows determination of the required integrity of the safety instrumented function under consideration. Each of the parameters is assigned a range of values such that when applied in combination a graded assessment of the risk which exists in the absence of the particular safety function is produced. Thus, a measure of the degree of reliance to be placed on the safety function is determined. The risk graph relates particular combinations of the risk parameters to safety integrity levels. The relationship between the combinations of risk parameters and safety integrity levels is established by considering the tolerable risk associated with specific hazards.

When considering the calibration of risk graphs, it is important to consider requirements relating to risk arising from both the owners' expectations and regulatory authority requirements. Risks to life can be considered in a number of ways as described in A.2 and Annex C.

If it is necessary to reduce the frequency of an individual fatality to a specified maximum then it cannot be assumed that all this risk reduction can be assigned to a single E/E/PE safety-related system. The exposed persons are subject to a wide range of risks arising from other sources (e.g., falls, fire and explosion risks). During calibration, the number of hazards that individuals are exposed to, and the total time at risk, will need to be considered.

When considering the extent of risk reduction required, an organization may have criteria relating to the incremental cost of averting a fatality. This can be calculated by dividing the annualised cost of the additional hardware and engineering associated with a higher level of integrity by the incremental risk reduction. An additional level of integrity is justified if the incremental cost of averting a fatality is less than a predetermined amount.

The above issues need to be considered before each of the parameter values can be specified. Most of the parameters are assigned a range (e.g., If the expected demand rate of a particular process falls between a specified decade range of demands per year then W3 may be used). Similarly, for demands in the lower decade range, W2 would apply and for demands in the next lower decade range, W1 applies. Giving each parameter a specified range assists the team in making decisions on which parameter value to select for a specific application. To calibrate the risk graph, values or value ranges are assigned to each parameter. The risk associated with each of the parameter combinations is then assessed against the defined risk criteria. Parameter descriptions are then modified so that for all combinations of all parameter values, the defined risk criteria is achieved. In the example calibration as shown in Table E.2 a "D" factor is introduced to enable the range of demands associated with each W factor to be modified so that tolerable risk is achieved. In some cases, the ranges associated with other risk factors may need to be modified to reflect the parameter values encountered in the spread of applications being considered. Calibration is an iterative process and continues until the specified risk acceptability criteria are satisfied for all combinations of parameter values.

The calibration activity does not need to be carried out each time the SIL for a specific application is to be determined. It is normally only necessary for organisations to undertake the work once, for similar hazards. Adjustment may be necessary for specific projects if the original assumptions made during the calibration are found to be invalid for any specific project.

When parameter assignments are made, information should be available as to how the values were derived.

It is important that this process of calibration is agreed at a senior level within the organization taking responsibility for safety. The decisions taken determine the overall safety achieved.

In general, it will be difficult for a risk graph to consider the possibility of dependent failure between the sources of demand and the equipment used within the E/E/PE safety related system. It can therefore lead to an over-estimation of the effectiveness of the E/E/PE safety related system. If risk graphs are calibrated to include demand rates higher than once per year, then the SIL requirements that results from use of the risk graph may be higher than necessary and the use of other techniques is recommended.

E.4 Other possible risk parameters

The risk parameters specified above are considered to be sufficiently generic to deal with a wide range of applications. There may, however, be applications which have aspects which require the introduction of additional risk parameters e.g. the use of new technologies in the EUC and the EUC control system. The purpose of the additional parameters would be to estimate more accurately the necessary risk reduction (see Figure A.1).

E.5 Risk graph implementation – general scheme

The combination of the risk parameters described above enables a risk graph such as that shown in Figure E.1 to be developed. With respect to Figure E.1:

$$C_A < C_B < C_C < C_D;\ F_A < F_B;\ P_A < P_B;\ W_1 < W_2 < W_3.$$

An explanation of this risk graph is as follows.

– Use of risk parameters C, F and P leads to a number of outputs X_1, X_2, X_3... X_n (the exact number being dependent upon the specific application area to be covered by the risk graph). Figure E.1 indicates the situation when no additional weighting is applied for the more serious consequences. Each one of these outputs is mapped onto one of three scales (W_1, W_2 and W_3). Each point on these scales is an indication of the necessary safety integrity that has to be met by the E/E/PE safety-related system under consideration. In practice, there will be situations when for specific consequences, a single E/E/PE safety-related system is not sufficient to give the necessary risk reduction;

– The mapping onto W1, W2 or W3 allows the contribution of other risk reduction measures to be made. The offset feature of the scales for W1, W2 and W3 is to allow for three different levels of risk reduction from other measures. That is, scale W3 provides the minimum risk reduction contributed by other measures (i.e. the highest probability of the unwanted occurrence taking place), scale W2 a medium contribution and scale W1 the maximum contribution. For a specific intermediate output of the risk graph (i.e. X1, X2... or X6) and for a specific W scale (i.e. W1, W2 or W3) the final output of the risk graph gives the safety integrity level of the E/E/PE safety-related system (i.e. 1, 2, 3 or 4) and is a measure of the required risk reduction for this system. This risk reduction, together with the risk reductions achieved by other measures (for example by other technology safety-related systems and other risk reduction measures) which are taken into account by the W scale mechanism, gives the necessary risk reduction for the specific situation.

The parameters indicated in Figure E.1 (C_A, C_B, C_C, C_D, F_A, F_B, P_A, P_B, W_1, W_2, W_3), and their weightings, would need to be accurately defined for each specific situation or sector comparable industries, and would also need to be defined in application sector international standards.

E.6 Risk graph example

An example of a risk graph implementation based on the example data in Table E.1 below is shown in Figure E.2. Use of the risk parameters C, F, and P lead to one of eight outputs. Each one of these outputs is mapped onto one of three scales (W1, W2 and W3). Each point on these scales (a, b, c, d, e, f, g and h) is an indication of the necessary risk reduction that has to be met by the safety-related system.

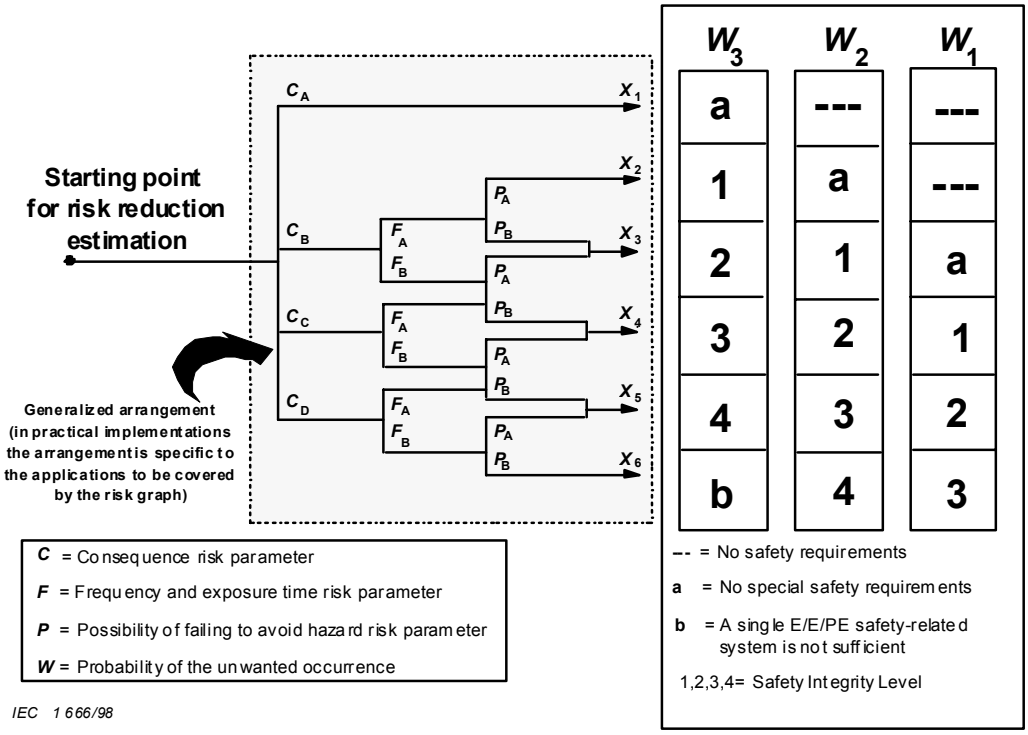
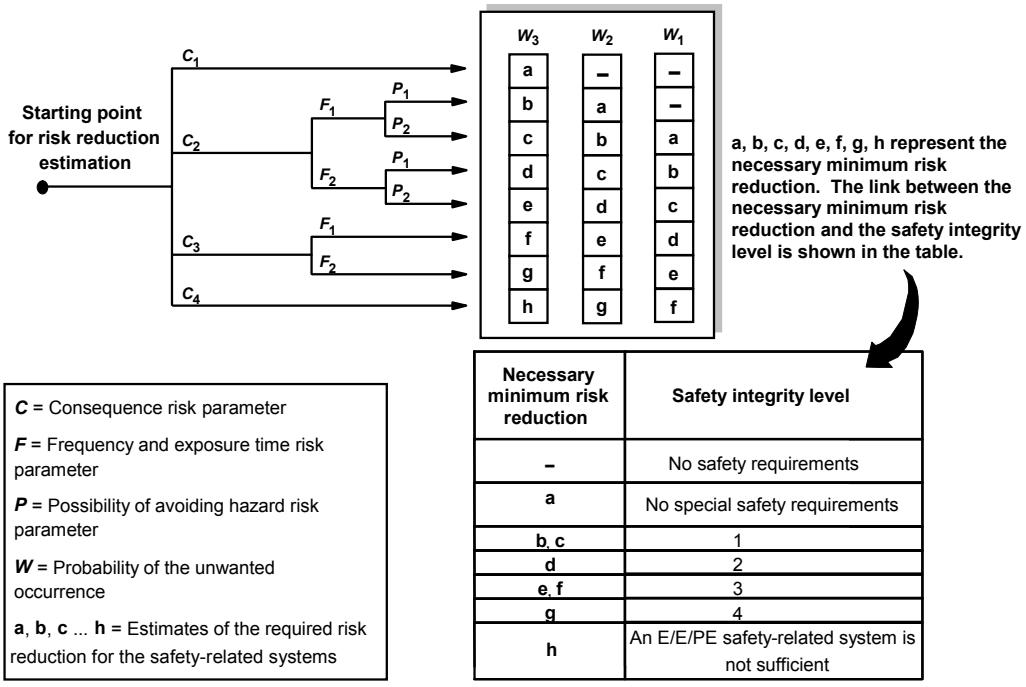


Figure E.1 – Risk Graph: general scheme



Necessary minimum risk reduction	Safety integrity level
–	No safety requirements
a	No special safety requirements
b, c	1
d	2
e, f	3
g	4
h	An E/E/PE safety-related system is not sufficient

Figure E.2 – Risk graph – example (illustrates general principles only)

Table E.1 – Example of data relating to risk graph (Figure E.2)

Risk parameter		Classification	Comments
Consequence (*C*)	C_1	Minor injury	1 The classification system has been developed to deal with injury and death to people. Other classification schemes would need to be developed for environmental or material damage
	C_2	Serious permanent injury to one or more persons; death to one person	
	C_3	Death to several people	2 For the interpretation of C_1, C_2, C_3 and C_4, the consequences of the accident and normal healing shall be taken into account
	C_4	Very many people killed	
Frequency of, and exposure time in, the hazardous zone (*F*)	F_1	Rare to more often exposure in the hazardous zone	3 See comment 1 above.
	F_2	Frequent to permanent exposure in the hazardous zone	
Possibility of avoiding the hazardous event (*P*)	P_1	Possible under certain conditions	4 This parameter takes into account – operation of a process (supervised (i.e. operated by skilled or unskilled persons) or unsupervised); – rate of development of the hazardous event (for example suddenly, quickly or slowly); – ease of recognition of danger (for example seen immediately, detected by technical measures or detected without technical measures); – avoidance of hazardous event (for example escape routes possible, not possible or possible under certain conditions); – actual safety experience (such experience may exist with an identical EUC or a similar EUC or may not exist).
	P_2	Almost impossible	
Probability of the unwanted occurrence (*W*)	W_1	A very slight probability that the unwanted occurrences will come to pass and only a few unwanted occurrences are likely	5 The purpose of the *W* factor is to estimate the frequency of the unwanted occurrence taking place without the addition of any safety-related systems (E/E/PE or other technology) but including any other risk reduction measures
	W_2	A slight probability that the unwanted occurrences will come to pass and few unwanted occurrences are likely	6 If little or no experience exists of the EUC, or the EUC control system, or of a similar EUC and EUC control system, the estimation of the *W* factor may be made by calculation. In such an event a worst case prediction shall be made
	W_3	A relatively high probability that the unwanted occurrences will come to pass and frequent unwanted occurrences are likely	

Table E.2 – Example of calibration of the general purpose risk graph

Risk parameter		Classification	Comments
Consequence (C) Number of fatalities This can be calculated by determining the numbers of people present when the area exposed to the hazard is occupied and multiplying by the vulnerability to the identified hazard The vulnerability is determined by the nature of the hazard being protected against. The following factors can be used: V=0,01 Small release of flammable or toxic material V=0,1 Large release of flammable or toxic material V=0,5 As above but also a high probability of catching fire or highly toxic material V=1 Rupture or explosion	C_A C_B C_C C_D	Minor injury Range 0,01 to 0,1 Range >0,1 to 1,0 Range > 1,0	1 The classification system has been developed to deal with injury and death to people 2 For the interpretation of C_A, C_B, C_C and C_D, the consequences of the accident and normal healing shall be taken into account
Occupancy (F) This is calculated by determining the proportional length of time the area exposed to the hazard is occupied during a normal working period NOTE 1 If the time in the hazardous area is different depending on the shift being operated then the maximum should be selected NOTE 2 It is only appropriate to use F_A where it can be shown that the demand rate is random and not related to when occupancy could be higher than normal. The latter is usually the case with demands which occur at equipment start-up or during the investigation of abnormalities	F_A F_B	Rare to more often exposure in the hazardous zone. Occupancy less than 0,1 Frequent to permanent exposure in the hazardous zone	3 See comment 1 above
Probability of avoiding the hazardous event (P) if the protection system fails to operate	P_A P_B	Adopted if all conditions in column 4 are satisfied Adopted if all the conditions are not satisfied	4 P_A should only be selected if all the following are true: – facilities are provided to alert the operator that the SIS has failed; – independent facilities are provided to shut down such that the hazard can be avoided or which enable all persons to escape to a safe area; – the time between the operator being alerted and a hazardous event occurring exceeds 1 h or is definitely sufficient for the necessary actions.

Table E.2 (*continued*)

Risk parameter		Classification	Comments
Demand rate (W) The number of times per year that the hazardous event would occur in absence of a the E/E/PE safety related system To determine the demand rate it is necessary to consider all sources of failure that can lead to one hazardous event. In determining the demand rate, limited credit can be allowed for control system performance and intervention. The performance which can be claimed if the control system is not to be designed and maintained according to IEC 61508, is limited to below the performance ranges associated with SIL 1	W_1 W_2 W_3	Demand rate less than 0,1 D per year Demand rate between 0,1 D and D per year Demand rate between D and 10 D per year For demand rates higher than 10 D per year higher integrity shall be needed	5 The purpose of the *W* factor is to estimate the frequency of the hazard taking place without the addition of the E/E/PE safety related systems If the demand rate is very high the SIL has to be determined by another method or the risk graph recalibrated. It should be noted that risk graph methods may not be the best approach in the case of applications operating in continuous mode (see 3.5.16 of IEC 61508-4). 6 The value of D should be determined from corporate criteria on tolerable risk taking into consideration other risks to exposed persons
NOTE This is an example to illustrate the application of the principles for the design of risk graphs. Risk graphs for particular applications and particular hazards will be agreed with those involved, taking into account tolerable risk, see Clauses E.1 to E.6.			

Annex F
(informative)

Semi-quantitative method using layer of protection analysis (LOPA)

F.1 General

F.1.1 Description

This annex describes a method called layer of protection analysis (LOPA). It is not intended to be a definitive account of the method, but is intended to illustrate the general principles.

F.1.2 Annex reference

This annex is based on a method described in more detail in an AIChE publication (see [8] in the Bibliography). This reference details many ways of using LOPA techniques.

In one approach, all relevant parameters are rounded to the higher decade range (for example, a probability of $5 \cdot 10^{-2}$ is rounded to 10^{-1}. This is a very conservative approach and can lead to significantly higher SIL levels. Data uncertainty should however be recognised by rounding all parameter values to the next highest significant figure (for example, $5{,}4 \cdot 10^{-2}$ should be rounded to $6 \cdot 10^{-2}$).

F.1.3 Method description

LOPA analyses hazards to determine if safety functions are required and if so, the required SIL of each safety function. The LOPA method needs to be adapted to meet the risk acceptance criteria to be applied. The method starts with data developed in the hazard identification and accounts for each identified hazard by documenting the initiating causes and the protection layers that prevent or mitigate the hazard. The total amount of risk reduction can then be determined and the need for more risk reduction analysed. If additional risk reduction is required and if it is to be provided in the form of an E/E/PE safety-related system, the LOPA methodology allows the determination of the appropriate SIL. For each hazard an appropriate SIL is determined to reduce risks to tolerable levels. Table F.1 hereinafter shows a typical LOPA format

F.2 Impact event

Using Table F.1, each Impact event description (consequence) determined from the hazard identification is entered in column 1 of Table F.1.

F.3 Severity level

The severity level of the event is entered in column 2 of Table F.1. The severity level will be derived from a table that specifies general descriptions of consequence levels e.g. minor, severe, catastrophic, with specified consequence ranges and maximum frequency for each severity level. In effect this table sets down the user tolerability criteria. Information will be needed to allow severity levels and maximum frequencies to be determined for events leading to safety and environmental consequences.

F.4 Initiating cause

All the initiating causes of the impact event are listed in column 3 of Table F.1. Impact events may have many initiating causes, and all should be listed.

F.5 Initiation likelihood

Likelihood values of each of the initiating causes listed in column 3 of Table F.1, in events per year, are entered into column 4 of Table F.1.

Initiation likelihood can be calculated from generic data on equipment failure rates and knowing proof test intervals, or from facility records. Low initiation likelihood should only be used where there is sufficient statistical basis for the data.

Table F.1 – LOPA report

[a]	1	2	3	4	5			6	7	8	9	10	11
					Protection layers (PLs)								
	Impact event description F.2	Severity level F.3	Initiating cause F.4	Initiation likelihood F.5	General design F.6.1	Control system F.6.2	Alarms, etc. F.6.3	Additional mitigation, restricted access F.7	Additional mitigation F.8	Intermediate event likelihood F.9	PFD_{avg} required for E/E/PES (and SIL) F.10	Tolerable Mitigated event likelihood F.11	Notes
1	Overspeed of rotor leading to fracture of casing	Loss of life of persons located adjacent to casing, fatalities will not exceed 2	Speed control system fails	0,1	1	1	1	0,1	0,1	10^{-3}	$5 \cdot 10^{-3}$ (SIL 2 with a minimum PFD_{avg} of $5 \cdot 10^{-3}$)	10^{-5}	
			Loss of load	1	1	0,1	1	0,1	0,1	10^{-3}			
			Clutch failure	0,1	1	0,1	1	0,1	0,1	10^{-4}			
						0,1 credit given to control system		Occupancy limited, persons not present 90 % of the time	Fatality will only occur if fragments contact persons	Total $2{,}1 \cdot 10^{-3}$		Tolerable frequency if fatalities do not exceed 5	
2	Repeat above case for environmental risk analysis												
3				Continued as required.									
.													
.													
N													

NOTE 1 Severity levels may be classified as C (catastrophic), E (extensive), S (serious) or M (minor). Tolerable mitigated event likelihood will depend on severity level.

NOTE 2 Units in columns 4, 8 and 10 are events per year.

NOTE 3 Units in columns 5 to 7 and 9 are dimensionless. The numbers between 0 and 1 are the factors by which event likelihood may be multiplied to represent the mitigating effect of the associated protection layer. Thus 1 means no mitigating effect and 0,1 means a factor of 10 risk reduction.

[a] Column and row numbers are given, as further descriptions of these are included in Annex F.

F.6 Protection layers (PLs)

F.6.1 General

Each PL consists of a grouping of equipment and/or administrative controls that function independently from other layers.

Design features that reduce the likelihood of an impact event from occurring when an initiating cause occurs are listed first in column 5 of Table F.1.

PLs should have the following important characteristics:

– Specificity: A PL is designed solely to prevent or to mitigate the consequences of one potentially hazardous event (for example, a runaway reaction, release of toxic material, a loss of containment, or a fire). Multiple causes may lead to the same hazardous event and therefore multiple event scenarios may initiate action of one PL.

– Effective: A PL must on its own be capable of preventing the outcome of concern when all other measures have completely failed

– Independence: A PL is independent of the other PLs associated with the identified hazardous event.

– Dependability: A PL can be counted on to do what it was designed to do. Both random and systematic failure modes are addressed in the design.

– Auditability: A PL is designed to facilitate regular validation of the protective functions. Proof testing and maintenance of the safety system are necessary.

F.6.2 Basic control system

The next item in column 5 of Table F.1 is the EUC control system. If a control function prevents the impact event from occurring when the initiating cause occurs, credit based on its PFD_{avg} is claimed. No credit should be claimed for a control function if failure of that function would cause a demand on the E/E/PE safety-related system. It should also be noted that the PFD_{avg} claimed from a control function should be limited to a minimum of 0,1 if the control function is not designed and operated as a safety system.

F.6.3 Alarms

The last item in column 5 of Table F.1 takes credit for alarms that alert the operator and utilize operator intervention. Credit for alarms should only be claimed under the following circumstances:

– Hardware and software used are separate and independent of that used for the control system (for example, input cards and processors should not be shared).

– The alarm is displayed with a high priority in a permanently manned location. Credit claimed for alarms should take into account the following:

  - the effectiveness of an alarm will depend on the complexity of the task that needs to be performed in the event of the alarm and the other tasks that need to be performed at the same time;.
  - the credit should be limited to a minimum PFD_{avg} of 0,1;
  - the operator needs to have sufficient time and independent facilities to be able to terminate the hazard. Normally, credit should not be claimed unless the time available between the alarm and the hazard exceeds 20 min.

F.7 and F.8 Additional mitigation

Mitigation layers are normally mechanical, structural, or procedural. Examples include:

– restricted access;
– reduction of ignition probability;
– any other factors that reduce the vulnerability of persons exposed to the hazard.

Mitigation layers may reduce the severity of the impact event, but not prevent the event from occurring. Examples include:

– deluge systems in the case of a fire;
– gas alarms;
– evacuation procedures that would reduce the probability of persons being exposed to an escalating event.

Under mitigation, the percentage occupancy of the most exposed person in the hazard zone can be taken account of. This percentage should be determined by establishing the number of hours in the hazardous zone per year and dividing by 8,760 h per year.

The appropriate PFD_{avg} or equivalent for all mitigation layers should be determined and listed in column 6 and 7 of Table F.1.

F.9 Intermediate event likelihood

The intermediate event likelihood for each cause is calculated by multiplying the following factors and the result in frequency per year entered in column 8 of Table F.1:

– vulnerability of the most exposed person;
– initiation likelihood (column 4);
– PFD_{avg} of the Protection Layers and mitigation layers (columns 5, 6 and 7).

The total intermediate event frequency should be calculated by adding intermediate event frequencies for each cause.

The total intermediate event frequency should be compared with the tolerable risk frequency for the associated severity level. If the total intermediate frequency exceeds the tolerable frequency, then risk reduction will be required. Inherently safer methods and solutions should be considered before additional PLs in the form of E/E/PE safety-related system are applied.

If the intermediate event likelihood figures cannot be reduced below the maximum frequency criteria then an E/E/PE safety-related system will be required.

F.10 Safety integrity levels (SILs)

If a safety function is needed, the required SIL can be determined as follows:

– Divide the maximum frequency for the associated severity level by the total intermediate event likelihood for to determine the PFD_{avg} required;
– The numeric target value of the PFD_{avg} can then be used in the safety requirement specification together with the associated SIL. The associated SIL can be obtained from Table 2 of IEC 61508-1;
– If the numeric value of PFD_{avg} is not to be in the process requirements specification and only the required SIL is to be stated, the SIL should be one level higher so that adequate risk reduction will be achieved with all values of PFD_{avg} associated with the specified SIL;

If the PFD_{avg} required for the tolerable risk is greater than or equal to 0,1 the function is allocated the classification “No special safety integrity requirements”.

F.11 Tolerable mitigated event likelihood

The tolerable mitigated event likelihood will depend on the severity level of the consequences. This will depend on the tolerable risk criteria adopted (see A.2 for tolerable risk criteria).

Annex G
(informative)

Determination of safety integrity levels – A qualitative method – hazardous event severity matrix

G.1 General

The numeric method described in Annex D is not applicable where the risk (or the frequency portion of it) cannot be quantified. This annex describes the hazardous event severity matrix method, which is a qualitative method that enables the safety integrity level of an E/E/PE safety-related system to be determined from knowledge of the risk factors associated with the EUC and the EUC control system. It is particularly applicable when the risk model is as indicated in Figures A.1 and A.2.

The scheme outlined in this annex assumes that each safety-related system and other risk reduction measure is independent.

This annex is not intended to be a definitive account of the method but is intended to illustrate the general principles of how such a matrix could be developed by those having a detailed knowledge of the specific parameters that are relevant to its construction. Those intending to apply the methods indicated in this annex should consult the source material referenced.

NOTE Further information on the hazardous event matrix is given in reference [4] in the Bibliography.

G.2 Hazardous event severity matrix

The following requirements underpin the matrix and each one is necessary for the method to be valid:

a) the E/E/PE safety-related systems and other risk reduction measures are independent;

b) each safety-related system (E/E/PE and other technology) and other risk reduction measures are considered as protection layers which provide, in their own right, partial risk reductions as indicated in Figure A.1;

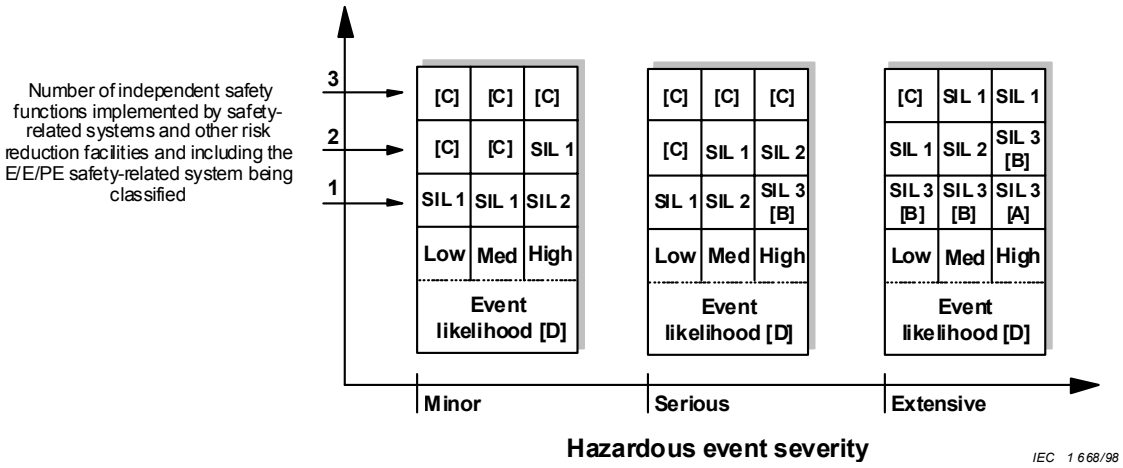
NOTE 1 This assumption is valid only if regular proof tests of the protection layers are carried out.

c) when one protection layer (see b) above) is added, then one order of magnitude improvement in safety integrity is achieved;

NOTE 2 This assumption is valid only if the safety-related systems and other risk reduction measures achieve an adequate level of independence.

d) only one E/E/PE safety-related system is used (but this may be in combination with an other technology safety-related system and/or other risk reduction measures), for which this method establishes the necessary safety integrity level;

e) The above considerations lead to the hazardous event severity matrix shown in Figure G.1. It should be noted that the matrix has been populated with example data to illustrate the general principles. For each specific situation, or sector comparable industries, a matrix similar to Figure G.1 would be developed and calibrated to the tolerable risk criteria applicable to the situation.



[A] One SIL 3 E/E/PE safety function does not provide sufficient risk reduction at this risk level. Additional risk reduction measures are required.

[B] One SIL 3 E/E/PE safety function may not provide sufficient risk reduction at this risk level. Hazard and risk analysis is required to determine whether additional risk reduction measures are necessary.

[C] An independent E/E/PE safety function is probably not required.

[D] Event likelihood is the likelihood that the hazardous event occurs without any safety function or other risk reduction measure.

[E] Event likelihood and the total number of independent protection layers are defined in relation to the specific application.

Figure G.1 – Hazardous event severity matrix – example (illustrates general principles only)

Bibliography

[1] IEC 61511 (all parts), *Functional safety – Safety instrumented systems for the process industry sector*

[2] IEC 62061, *Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems*

[3] IEC 61800-5-2, *Adjustable speed electrical power drive systems – Part 5-2: Safety requirements – Functional*

[4] ANSI/ISA S84:1996, *Application of safety Instrumented Systems for the Process Industries*

[5] Health and Safety Executive (UK) *publication, ISBN 011 886368 1, Tolerability of risk from nuclear power stations,* <www.hse.gov.uk/nuclear/tolerability.pdf>

[6] The Motor Industry Research Association, 1994, ISBN 09524156 0 7, *Development guidelines for vehicle based software*

[7] Health and Safety Executive (UK) publication, ISBN 0 7176 2151 0, *Reducing Risks, Protecting People*, <www.hse.gov.uk/risk/theory/r2p2.pdf>

[8] CCPS ISBN 0-8169-0811-7, *Layer of Protection Analysis – Simplified Process Risk Assessment*

[9] ISO/IEC 31010, *Risk management – Risk assessment techniques*[3]

[10] ISO 10418:2003, *Petroleum and natural gas industries – Offshore production installations – Basic surface process safety systems*

[11] ISO/TR 14121-2*, Safety of machinery – Risk assessment – Part 2: Practical guidance and examples of methods*

[12] ISO 13849-1:2006, *Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design*

[13] IEC 60601 (all parts), *Medical electrical equipment*

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

[15] IEC 61508-3, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements*

[16] IEC 61508-6, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3*

[17] IEC 61508-7, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures*

[18] IEC 61511-1, *Functional safety – Safety instrumented systems for the process industry sector – Part 1: Framework, definitions, system, hardware and software requirements*

_____

3 To be published.

IEC 61508-6

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Functional safety of electrical/electronic/programmable electronic safety-related systems –
Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité –
Partie 6: Lignes directrices pour l'application de la CEI 61508-2 et de la CEI 61508-3

IEC 61508-6:2010

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.
- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.
- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.
- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:
Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm
Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,…). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.
- Just Published CEI: www.iec.ch/online_news/justpub
Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.
- Electropedia: www.electropedia.org
Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.
- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:
Email: csc@iec.ch
Tél.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61508-6

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 6: Lignes directrices pour l'application de la CEI 61508-2 et de la CEI 61508-3

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XE

ICS 25.040.40

ISBN 978-2-88910-529-8

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

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

Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3

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) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61508-6 has been prepared by subcommittee 65A: System aspects, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 2000. This edition constitutes a technical revision.

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

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

FDIS	Report on voting
65A/553/FDIS	65A/577/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

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

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

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.

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, though 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.

– 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.

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

Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3

1 Scope

1.1 This part of IEC 61508 contains information and guidelines on IEC 61508-2 and IEC 61508-3.

- Annex A gives a brief overview of the requirements of IEC 61508-2 and IEC 61508-3 and sets out the functional steps in their application.
- Annex B gives an example technique for calculating the probabilities of hardware failure and should be read in conjunction with 7.4.3 and Annex C of IEC 61508-2 and Annex D.
- Annex C gives a worked example of calculating diagnostic coverage and should be read in conjunction with Annex C of IEC 61508-2.
- Annex D gives a methodology for quantifying the effect of hardware-related common cause failures on the probability of failure.
- Annex E gives worked examples of the application of the software safety integrity tables specified in Annex A of IEC 61508-3 for safety integrity levels 2 and 3.

1.2 IEC 61508-1, IEC 61508-2, 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 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 61508-6 plays in the achievement of functional safety for E/E/PE safety-related systems.

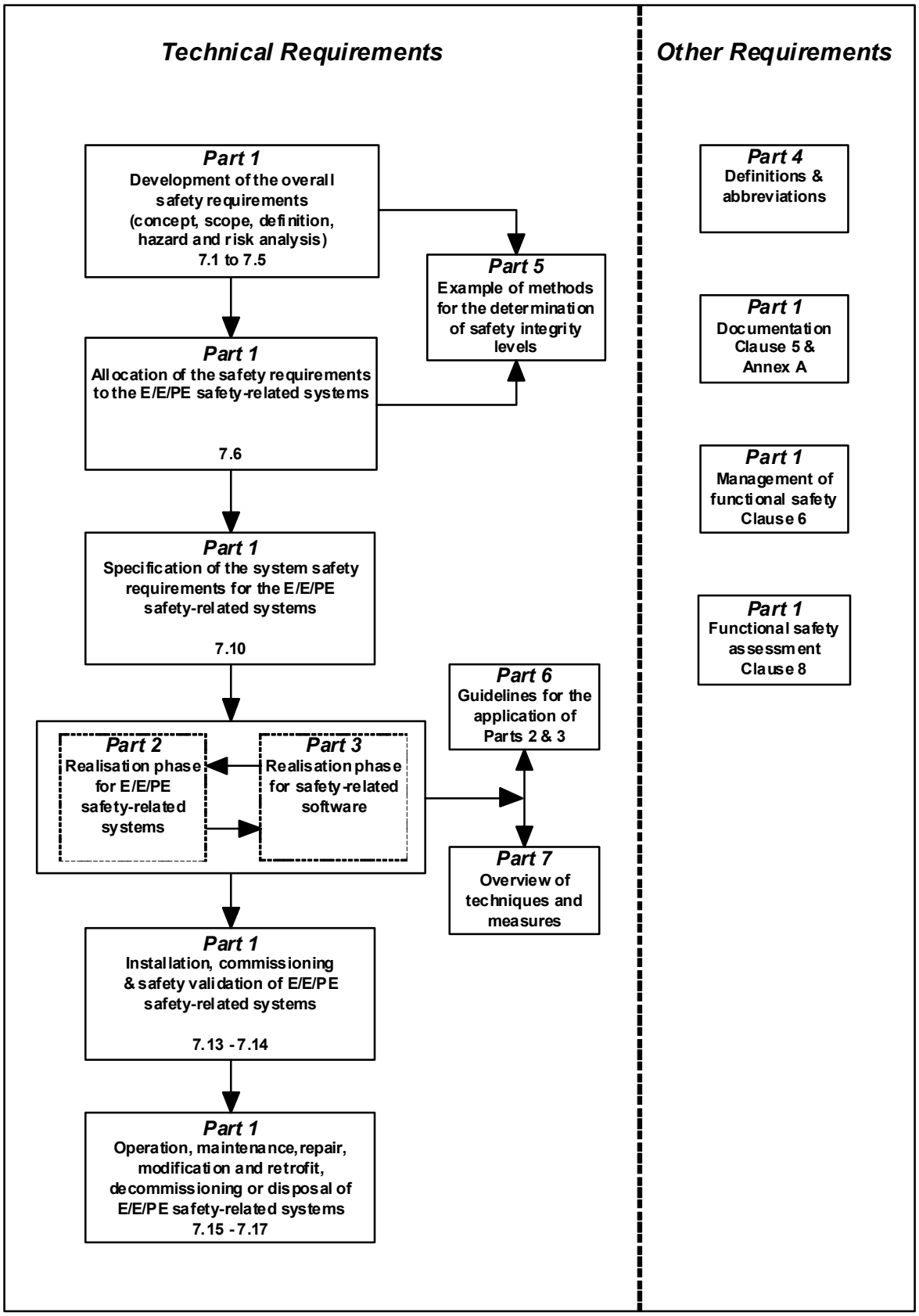


Figure 1 – Overall framework of the IEC 61508 series

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

IEC 61508-3:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements*

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

3 Definitions and abbreviations

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

Annex A
(informative)

Application of IEC 61508-2 and of IEC 61508-3

A.1 General

Machinery, process plant and other equipment may, in the case of malfunction (for example by failures of electrical, electronic and/or programmable electronic devices), present risks to people and the environment from hazardous events such as fires, explosions, radiation overdoses, machinery traps, etc. Failures can arise from either physical faults in the device (for example causing random hardware failures), or from systematic faults (for example human errors made in the specification and design of a system cause systematic failure under some particular combination of inputs), or from some environmental condition.

IEC 61508-1 provides an overall framework based on a risk approach for the prevention and/or control of failures in electro-mechanical, electronic, or programmable electronic devices.

The overall goal is to ensure that plant and equipment can be safely automated. A key objective of this standard is to prevent:

– failures of control systems triggering other events, which in turn could lead to danger (for example fire, release of toxic materials, repeat stroke of a machine, etc.); and

– undetected failures in protection systems (for example in an emergency shut-down system), making the systems unavailable when needed for a safety action.

IEC 61508-1 requires that a hazard and risk analysis at the process/machine level is carried out to determine the amount of risk reduction necessary to meet the risk criteria for the application. Risk is based on the assessment of both the consequence (or severity) and the frequency (or probability) of the hazardous event.

IEC 61508-1 further requires that the amount of risk reduction established by the risk analysis is used to determine if one or more safety-related systems[1] are required and what safety functions (each with a specified safety integrity)[2] they are needed for.

IEC 61508-2 and IEC 61508-3 take the safety functions and safety integrity requirements allocated to any system, designated as a E/E/PE safety-related system, by the application of IEC 61508-1 and establish requirements for safety lifecycle activities which:

– are to be applied during the specification, design and modification of the hardware and software; and

– focus on means for preventing and/or controlling random hardware and systematic failures (the E/E/PE system and software safety lifecycles)[3].

---

1 Systems necessary for functional safety and containing one or more electrical (electro-mechanical), electronic or programmable electronic (E/E/PE) devices are *designated* as E/E/PE safety-related systems and include all equipment necessary to carry out the required safety function (see 3.5.1 of IEC 61508-4).

2 Safety integrity is specified as one of four discrete levels. Safety integrity level 4 is the highest and safety integrity level 1 the lowest (see 3.5.4 and 3.5.8 of IEC 61508-4).

3 To enable the requirements of this standard to be clearly structured, a decision was made to order the requirements using a development process model in which each stage follows in a defined order with little iteration (sometimes referred to as a waterfall model). However, it is stressed that any lifecycle approach can be used provided a statement of equivalence is given in the safety plan for the project (see Clause 7 of IEC 61508-1).

IEC 61508-2 and IEC 61508-3 do not give guidance on which level of safety integrity is appropriate for a given required tolerable risk. This decision depends upon many factors, including the nature of the application, the extent to which other systems carry out safety functions and social and economic factors (see IEC 61508-1 and IEC 61508-5).

The requirements of IEC 61508-2 and IEC 61508-3 include:

– the application of measures and techniques[4], which are graded against the safety integrity level, for the avoidance of systematic failures[5] by preventative methods; and
– the control of systematic failures (including software failures) and random hardware failures by design features such as fault detection, redundancy and architectural features (for example diversity).

In IEC 61508-2, assurance that the safety integrity target has been satisfied for dangerous random hardware failures is based on:

– hardware fault tolerance requirements (see Tables 2 and 3 of IEC 61508-2); and
– the diagnostic coverage and frequency of proof tests of subsystems and components, by carrying out a reliability analysis using appropriate data.

In both IEC 61508-2 and IEC 61508-3, assurance that the safety integrity target has been satisfied for systematic failures is gained by:

– the correct application of safety management procedures;
– the use of competent staff;
– the application of the specified safety lifecycle activities, including the specified techniques and measures[6]; and
– an independent functional safety assessment[7].

The overall goal is to ensure that remaining systematic faults, commensurate with the safety integrity level, do not cause a failure of the E/E/PE safety-related system.

IEC 61508-2 has been developed to provide requirements for achieving safety integrity in the hardware[8] of the E/E/PE safety-related systems including sensors and final elements. Techniques and measures against both random hardware failures and systematic hardware failures are required. These involve an appropriate combination of fault avoidance and failure control measures as indicated above. Where manual action is needed for functional safety, requirements are given for the operator interface. Also diagnostic test techniques and measures, based on software and hardware (for example diversity), to detect random hardware failures are specified in IEC 61508-2.

IEC 61508-3 has been developed to provide requirements for achieving safety integrity for the software – both embedded (including diagnostic fault detection services) and application software. IEC 61508-3 requires a combination of fault avoidance (quality assurance) and fault tolerance approaches (software architecture), as there is no known way to prove the absence of faults in reasonably complex safety-related software, especially the absence of specification and design faults. IEC 61508-3 requires the adoption of such software

---

4 The required techniques and measures for each safety integrity level are shown in the tables in Annexes A and B of IEC 61508-2 and IEC 61508-3.

5 Systematic failures cannot usually be quantified. Causes include: specification and design faults in hardware and software; failure to take account of the environment (for example temperature); and operation-related faults (for example poor interface).

6 Alternative measures to those specified in the standard are acceptable provided justification is documented during safety planning (see Clause 6 of IEC 61508-1).

7 Independent assessment does not always imply third party assessment (see Clause 8 of IEC 61508-1).

8 Including fixed built-in software or software equivalents (also called firmware), such as application-specific integrated circuits.

engineering principles as: top down design; modularity; verification of each phase of the development lifecycle; verified software modules and software module libraries; and clear documentation to facilitate verification and validation. The different levels of software require different levels of assurance that these and related principles have been correctly applied.

The developer of the software may or may not be separate from the organization developing the whole E/E/PE system. In either case, close cooperation is needed, particularly in developing the architecture of the programmable electronics where trade-offs between hardware and software architectures need to be considered for their safety impact (see Figure 4 of IEC 61508-2).

A.2 Functional steps in the application of IEC 61508-2

The functional steps in the application of IEC 61508-2 are shown in Figures A.1 and A.2. The functional steps in the application of IEC 61508-3 are shown in Figure A.3.

Functional steps for IEC 61508-2 (see Figures A.1 and A.2) are as follows:

a) Obtain the allocation of safety requirements (see IEC 61508-1). Update the safety planning as appropriate during E/E/PE safety-related system development.

b) Determine the requirements for E/E/PE safety-related systems, including the safety integrity requirements, for each safety function (see 7.2 of IEC 61508-2). Allocate requirements to software and pass to software supplier and/or developer for the application of IEC 61508-3.

NOTE 1 The possibility of coincident failures in the EUC control system and E/E/PE safety-related system(s) needs to be considered at this stage (see A.5.4 of IEC 61508-5). These may result from failures of components having a common cause due to for example similar environmental influences. The existence of such failures could lead to a higher than expected residual risk unless properly addressed.

c) Start the phase of planning for E/E/PE safety-related system safety validation (see 7.3 of IEC 61508-2).

d) Specify the architecture (configuration) for the E/E/PE safety-related logic subsystem, sensors and final elements. Review with the software supplier/developer the hardware and software architecture and the safety implications of the trade-offs between the hardware and software (see Figure 4 of IEC 61508-2). Iterate if required.

e) Develop a model for the hardware architecture for the E/E/PE safety-related system. Develop this model by examining each safety function separately and determine the subsystem (component) to be used to carry out this function.

f) Establish the system parameters for each of the subsystems (components) used in the E/E/PE safety-related system. For each of the subsystems (elements), determine the following:

   – the proof test interval for failures which are not automatically revealed;
   – the mean time to restoration;
   – the diagnostic coverage (see Annex C of IEC 61508-2);
   – the probability of failure;
   – the required architectural constraints; for Route 1_H see 7.4.4.2 and Annex C of IEC 61508-2 and for Route 2_H see 7.4.4.3 of IEC 61508-2.

g) Create a reliability model for each of the safety functions that the E/E/PE safety-related system is required to carry out.

NOTE 2 A reliability model is a mathematical formula which shows the relationship between reliability and relevant parameters relating to equipment and conditions of use.

h) Calculate a reliability prediction for each safety function using an appropriate technique. Compare the result with the target failure measure determined in b) above and the requirements of Route 1_H (see 7.4.4.2 of IEC 61508-2) or Route 2_H (see 7.4.4.3 of

IEC 61508-2). If the predicted reliability does not meet the target failure measure and/or does not meet the requirements of Route 1_H or Route 2_H , then change

- where possible, one or more of the subsystem parameters (go back to f) above); and/or
- the hardware architecture (go back to d) above).

NOTE 3 A number of modelling methods are available and the analyst should choose which is the most appropriate (see Annex B for guidance on some methods that could be used).

i) Implement the design of the E/E/PE safety-related system. Select measures and techniques to control systematic hardware failures, failures caused by environmental influences and operational failures (see Annex A of IEC 61508-2).

j) Integrate the verified software (see IEC 61508-3) onto the target hardware (see 7.5 of IEC 61508-2 and Annex B of IEC 61508-2) and, in parallel, develop the procedures for users and maintenance staff to follow when operating the system (see 7.6 of IEC 61508-2 and Annex B of IEC 61508-2). Include software aspects (see A.3 f)).

k) Together with the software developer (see 7.7 of IEC 61508-3), validate the E/E/PE system (see 7.7 of IEC 61508-2 and Annex B of IEC 61508-2).

l) Hand over the hardware and results of the E/E/PE safety-related system safety validation to the system engineers for further integration into the overall system.

m) If maintenance/modification of the E/E/PE safety related system is required during operational life then re-activate IEC 61508-2 as appropriate (see 7.8 of IEC 61508-2).

A number of activities run across the E/E/PE safety related system safety lifecycle. These include verification (see 7.9 of IEC 61508-2) and functional safety assessment (see Clause 8 of IEC 61508-1).

In applying the above steps the E/E/PE safety related system safety techniques and measures appropriate to the required safety integrity level are selected. To aid in this selection, tables have been formulated, ranking the various techniques/measures against the four safety integrity levels (see Annex B of IEC 61508-2). Cross-referenced to the tables is an overview of each technique and measure with references to further sources of information (see Annexes A and B of IEC 61508-7).

Annex B provides one possible technique for calculating the probabilities of hardware failure for E/E/PE safety-related systems.

NOTE 4 In applying the above steps, alternative measures to those specified in the standard are acceptable provided justification is documented during safety planning (see Clause 6 of IEC 61508-1).

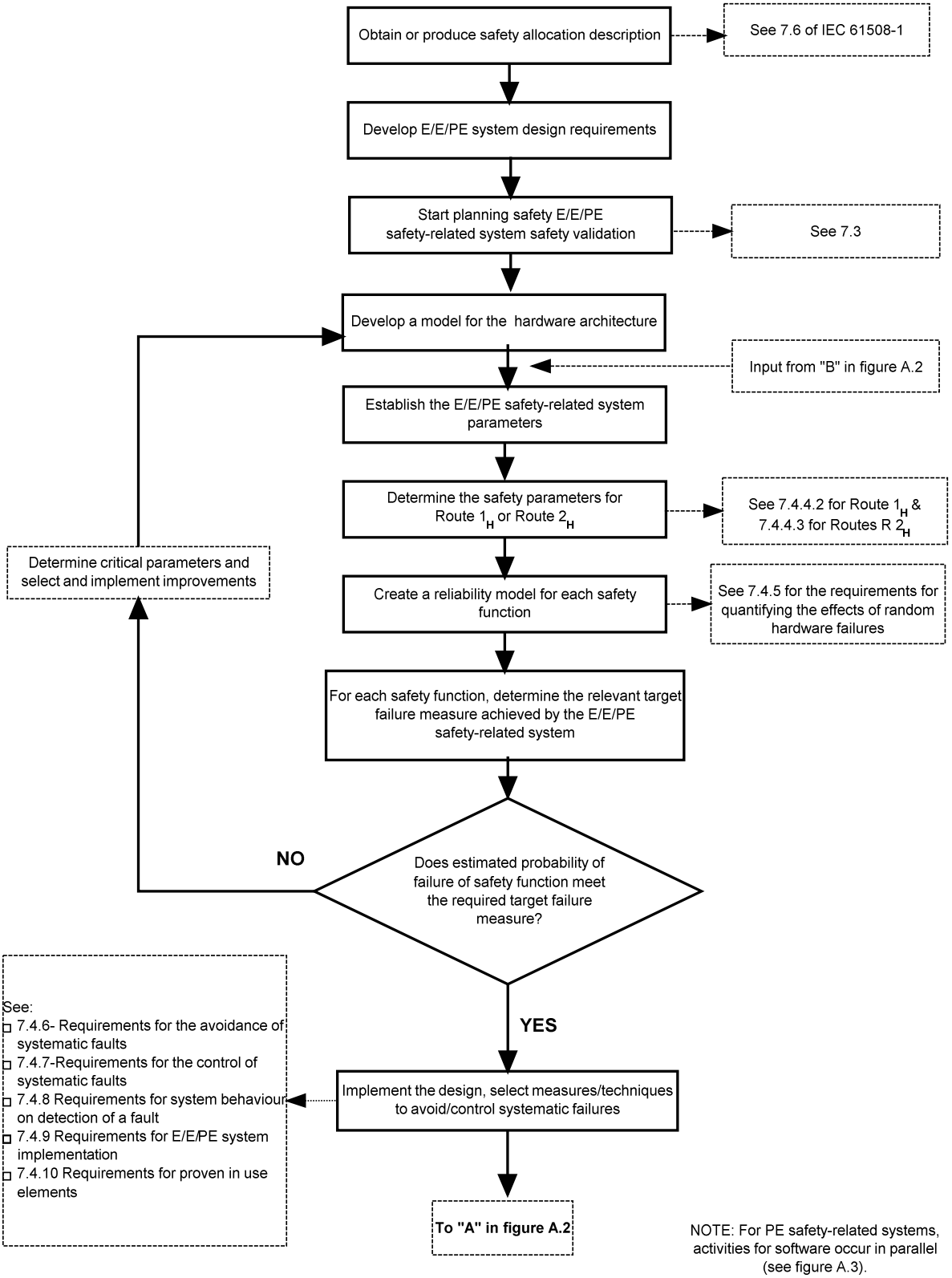
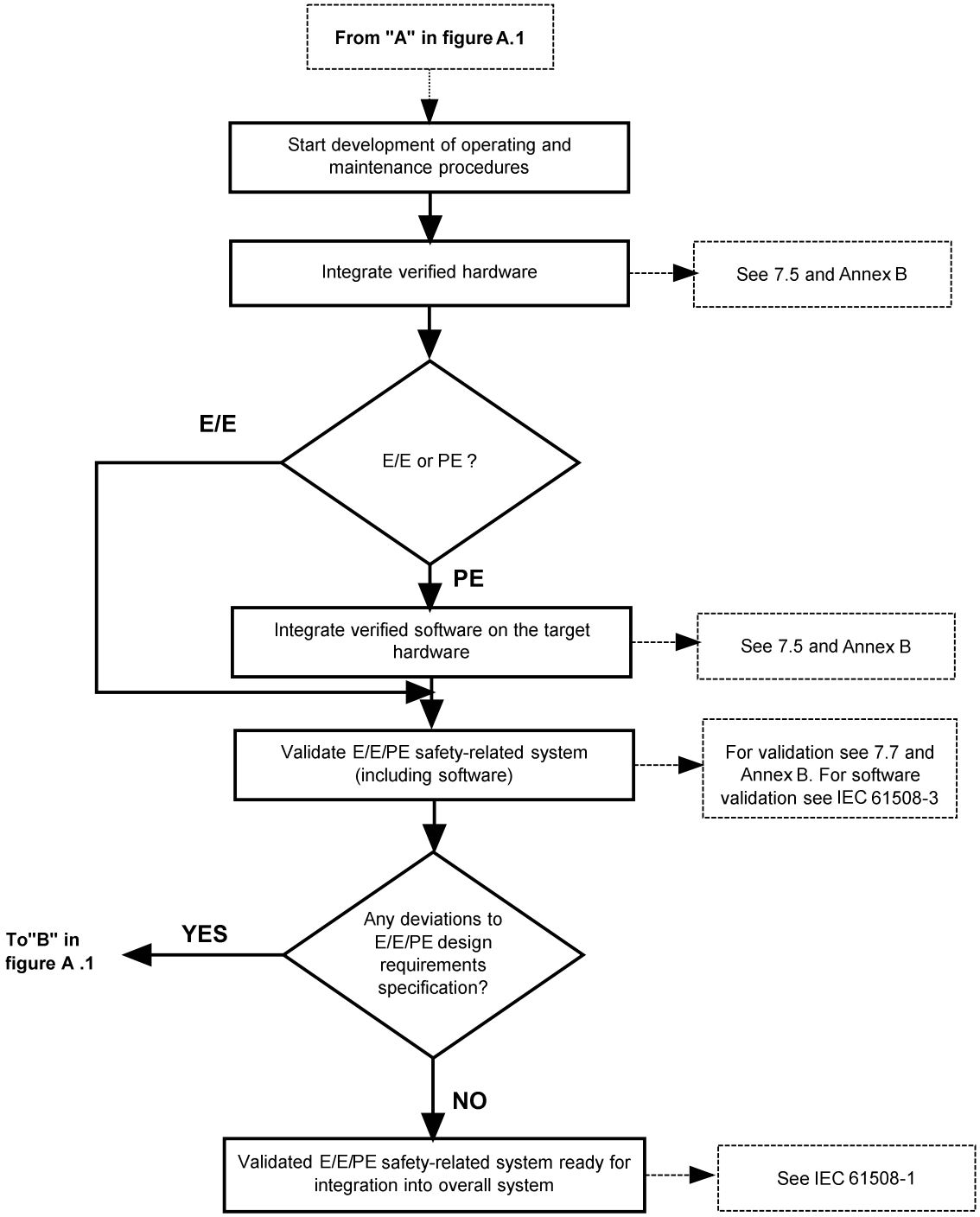


Figure A.1 – Application of IEC 61508-2



NOTE: For PE safety-related systems, activities for software occur in parallel (see figure A.3).

Figure A.2 – Application of IEC 61508-2 (Figure A.1 *continued*)

A.3 Functional steps in the application of IEC 61508-3

Functional steps for IEC 61508-3 (see Figure A.3) are as follows.

a) Obtain the requirements for the E/E/PE safety-related systems and relevant parts of the safety planning (see 7.3 of IEC 61508-2). Update the safety planning as appropriate during software development.

NOTE 1 Earlier lifecycle phases have already:

– specified the required safety functions and their associated safety integrity levels (see 7.4 and 7.5 of IEC 61508-1);

– allocated the safety functions to designated E/E/PE safety-related systems (see 7.6 of IEC 61508-1); and

– allocated functions to software within each E/E/PE safety-related system (see 7.2 of IEC 61508-2).

b) Determine the software architecture for all safety functions allocated to software (see 7.4 of IEC 61508-3 and Annex A of IEC 61508-3).

c) Review with the E/E/PE safety-related system's supplier/developer, the software and hardware architecture and the safety implications of the trade-offs between the software and hardware (see Figure 4 of IEC 61508-2). Iterate if required.

d) Start the planning for software safety verification and validation (see 7.3 and 7.9 of IEC 61508-3).

e) Design, develop and verify/test the software according to the:

– software safety planning;

– software safety integrity level; and

– software safety lifecycle.

f) Complete the final software verification activity and integrate the verified software onto the target hardware (see 7.5 of IEC 61508-3), and in parallel develop the software aspects of the procedures for users and maintenance staff to follow when operating the system (see 7.6 of IEC 61508-3, and A.2 k)).

g) Together with the hardware developer (see 7.7 of IEC 61508-2), validate the software in the integrated E/E/PE safety-related systems (see 7.7 of IEC 61508-3).

h) Hand over the results of the software safety validation to the system engineers for further integration into the overall system.

i) If modification of the E/E/PE safety-related system software is required during operational life then re-activate this IEC 61508-3 phase as appropriate (see 7.8 of IEC 61508-3).

A number of activities run across the software safety lifecycle. These include verification (see 7.9 of IEC 61508-3) and functional safety assessment (see Clause 8 of IEC 61508-3).

In applying the above steps, software safety techniques and measures appropriate to the required safety integrity are selected. To aid in this selection, tables have been formulated ranking the various techniques/measures against the four safety integrity levels (see Annex A of IEC 61508-3). Cross-referenced to the tables is an overview of each technique and measure with references to further sources of information (see Annex C of IEC 61508-7).

Worked examples in the application of the safety integrity tables are given in Annex E, and IEC 61508-7 includes a probabilistic approach to determining software safety integrity for pre-developed software (see Annex D of IEC 61508-7).

NOTE 2 In applying the above steps, alternative measures to those specified in the standard are acceptable provided justification is documented during safety planning (see Clause 6 of IEC 61508-1).

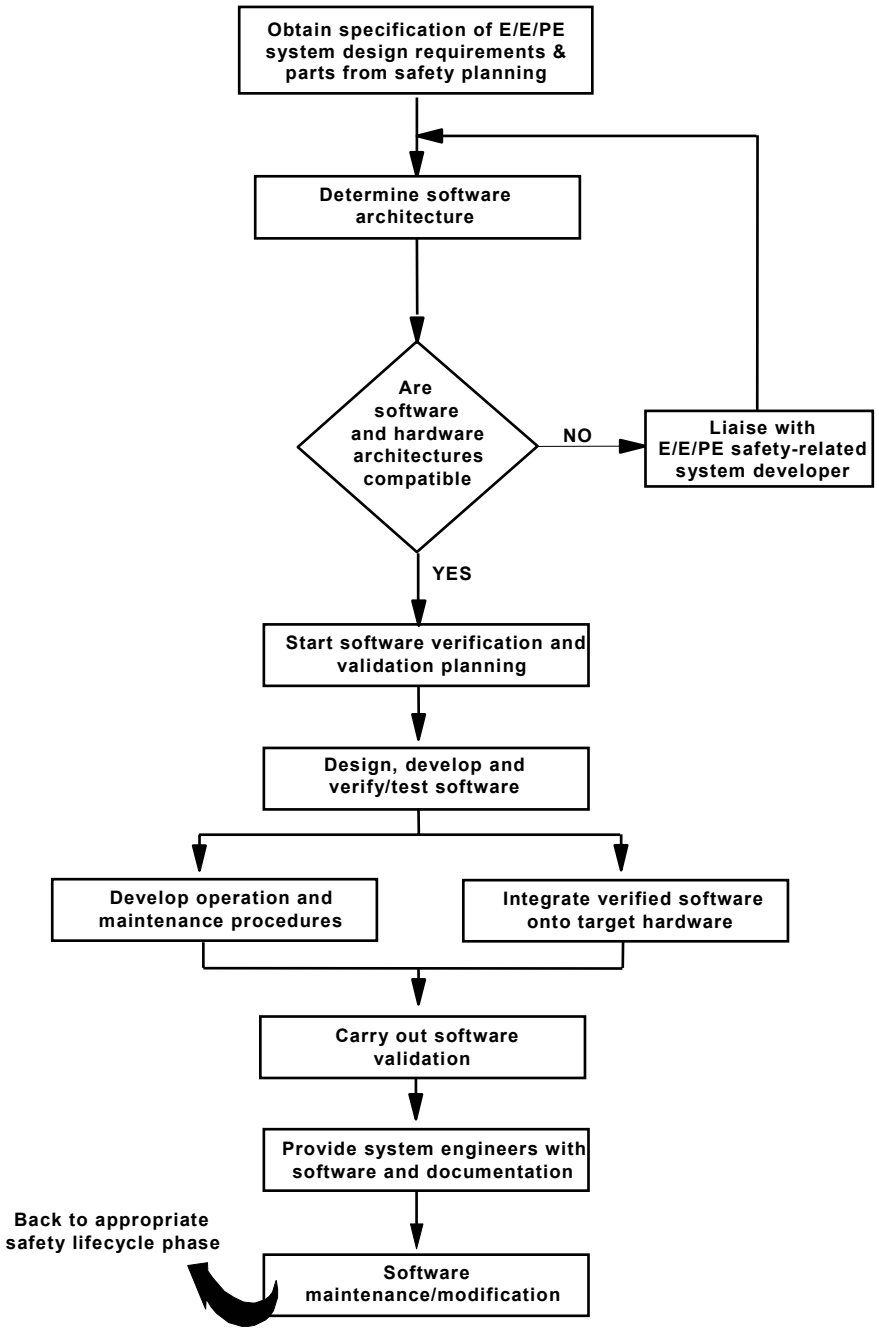


Figure A.3 – Application of IEC 61508-3

Annex B
(informative)

Example of technique for evaluating probabilities of hardware failure

B.1 General

This annex provides possible techniques for calculating the probabilities of hardware failure for E/E/PE safety-related systems installed in accordance with IEC 61508-1, IEC 61508-2 and IEC 61508-3. The information provided is informative in nature and should not be interpreted as the only evaluation techniques that might be used. It does, however, provide both a relatively simple approach for assessing the capability of E/E/PE safety-related systems and guidelines to use alternative techniques derived from the classical reliability calculations techniques.

NOTE 1 System architectures stated in this part are provided by way of examples and should not be considered exhaustive as there are many other architectures that may be used.

NOTE 2 See ISA-TR84.00.02-2002 [17] in the Bibliography.

A number of reliability techniques are more or less straightforwardly usable for the analysis of hardware safety integrity of E/E/PE safety-related systems. Classically, they are sorted according to the two following point of views:

– static (Boolean) versus dynamic (states/transitions) models;

– analytical versus Monte Carlo simulation calculations.

Boolean models encompass all models describing the static logical links between the elementary failures and the whole system failure. Reliability Block Diagrams (RBD) (see C.6.4 of IEC 61508-7 and IEC 61078 [4]) and Fault Trees (FT) (see B.6.6.5 and B.6.6.9 of IEC 61508-7) and IEC 61025 [18] belong to Boolean models.

States/transitions models encompass all models describing how the system behaves (jumps from states to states) according to arising events (failures, repairs, tests, etc.). Markovian (see B.6.6.6 of IEC 61508-7 and IEC 61165 [5]), Petri nets (see B.2.3.3 and B.6.6.10 of IEC 61508-7 and IEC 62551 [19]) and formal language models belong to states/transitions models. Two Markovian approaches are investigated: a simplified approach based on specific formulae (B.3) and a general approach allowing direct calculations on Markov graphs (B.5.2). For non Markovian safety systems, Monte Carlo simulations can be used instead. With present time personal computers this is achievable even for SIL 4 calculations. Subclauses B.5.3 and B.5.4 of this annex provides guidelines about handling Monte Carlo simulations (see B.6.6.8 of IEC 61508-7) on behavioural models based on Petri nets and formal languages modelling.

The simplified approach which is presented first is based on RBD graphical representations and specific Markovian formulae obtained from Taylor's developments and slightly conservative underlying hypotheses described in B.3.1.

All these methods can be used for the majority of safety related systems and, when deciding which technique to use on any particular application, it is very important that the user of a particular technique is competent in using the technique and this may be more important than the technique which is actually used. It is the responsibility of the analyst to verify that the underling hypotheses of any particular method are satisfied or any adjustments are required to obtain an adequate realist conservative result. In case of poor reliability data or dominant common cause failure, it may be sufficient to use the simplest model / techniques. Whether the loss of accuracy is significant can only be determined in the particular circumstances.

If software programmes are used to perform the calculations then the practitioner shall have an understanding of the formulae/techniques used by the software package to ensure its use is suitable for the specific application. The practitioner should also verify the software package by checking its output with some manual calculated test cases.

Where a failure of the EUC control system places a demand on the E/E/PE safety-related system, then the probability of a hazardous event occurring also depends on the probability of failure of the EUC control system. In that situation, it is necessary to consider the possibility of co-incident failure of components in the EUC control system and the E/E/PE safety-related system due to common cause failure mechanisms. The existence of such failures could lead to a higher than expected residual risk unless properly addressed.

B.2 Considerations about basic probabilistic calculations

B.2.1 Introduction

The reliability block diagram (RBD) on Figure B.1 is representing a safety loop made of three sensors (A, B, C), one logic solver (D), two final elements (E, F), and common cause failures (CCF).

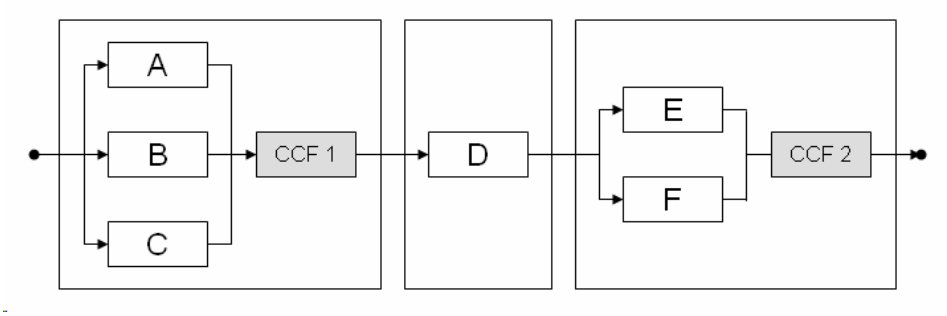


Figure B.1 – Reliability Block Diagram of a whole safety loop

This facilitates the identification of five failure combinations leading to the E/E/PE safety-related system failure. Each of them is a so-called minimal cut set:

– (A, B, C) is a triple failure;

– (E, F) is a double failure;

– (D) (CCF1) (CCF2) are single failures.

B.2.2 Low demand E/E/PE safety-related system

When a E/E/PE safety-related system is used in low demand mode, the standard requires that its PFD_{avg} (i.e. its average unavailability) be assessed. This is simply the ratio $MDT(T)/T$ where $MDT(T)$ is the mean down time over the period [0, T] of the E/E/PE safety-related system.

For safety system the probability of failure is, normally, very low and the probability to have two minimal cut sets at the same time is negligible. Therefore, the sum of the mean down times due to each cut sets gives a conservative estimate of the mean down time of the whole system. From Figure B.1 we find:

$$MDT \approx MDT^{ABC} + MDT^{D} + MDT^{EF}$$

Dividing by T gives:

$$PFD_{avg} \approx PFD_{avg}^{ABC} + PFD_{avg}^{D} + PFD_{avg}^{EF}$$

Therefore, for parts in series, PFD_{avg} calculations are very similar to those performed with ordinary probabilities when they are very small compared to 1.

However for parallel parts where multiple failure are required before the loss of the function like (E, F), it is clear that MDT^{EF} may not be calculated straightforwardly from MDT^{E} and MDT^{F}: The (E,F) system's *MDT* has to be calculated as

$$MDT^{EF} = \int_0^T PFD^E(t) PFD^F(t) dt$$

Therefore, ordinary probabilistic calculations (additions and multiplications) are no longer valid for PFD_{avg} calculations (integrals) of parts in parallel. PFD_{avg} has not the same properties as a genuine probability and its assimilation with a genuine probability is likely to lead to non conservative results. In particular, it is not possible to obtain the PFD_{avg} of an E/E/PE safety-related system just by combining in a conventional way the $PFD_{avg,i}$ of its components. As this is sometimes encouraged by commercial Boolean software packages, analysts should be vigilant to avoid such non conservative calculations which are undesirable when dealing with safety.

EXAMPLE For a redundant (1oo2) channel with a dangerous undetected failure rate λ_{DU} with a proof test interval τ, an incorrect probability model calculation could give $(\lambda_{DU}.\tau)^2/4$ when the actual result is $(\lambda_{DU}.\tau)^2/3$.

Calculations may be performed analytically or by using Monte Carlo simulation. This annex describes how to do that by using conventional reliability models based on Boolean (RBD or Fault-trees) or, states/transitions models (Markov, Petri nets, etc.).

B.2.3 Continuous or high demand mode E/E/PE safety-related system

B.2.3.1 General *PFH* formula

When an E/E/PE safety-related system is used in continuous or high demand mode, the standard requires the calculation of its *PFH* (i.e. its average frequency of dangerous failure). This is the average of the so called unconditional failure intensity (also called failure frequency) *w(t)* over the period of interest:

$$PFH(T) = \frac{1}{T}\int_0^T w(t) dt$$

Where the E/E/PE safety-related system is working in continuous mode and is the ultimate safety barrier, then the overall safety-related system failure will lead directly to a potentially hazardous situation. Hence for failures that cause the loss of the overall safety function no overall safety-related system repair can be considered in the calculations. However, if the failure of the overall safety-related system does not lead directly to the potential hazard due to some other safety barrier or equipment failure then it may be possible to consider the detection and repair of the safety-related system in its risk reduction calculation.

B.2.3.2 Un-reliability case (e.g. single barrier working in continuous mode)

This case is relevant when E/E/PE safety-related system working in continuous mode is the ultimate safety barrier. Therefore a potential hazard can occur as soon as it is failing. No overall system failures are acceptable over the period of interest.

In this case the *PFH* may be calculated by using the unreliability over the period of interest:

$$F(T):\ PFH(T) = \frac{1 - \exp\left[-\int_0^T \Lambda(t)dt\right]}{T} = \frac{F(T)}{T}$$

The overall system failure rate, $\Lambda(t)$, may be time dependent or constant.

When it is time dependant, we have: $PFH(T) = \frac{1 - \exp(-\Lambda_{avg}.T)}{T} \approx \Lambda_{avg}$

When the system is made of components completely and quickly repairable with constant failure and repair rates (e.g. dangerous detected failures), $\Lambda(t)$ reaches quickly an asymptotic constant value Λ_{as} and, when $PFH(T) << 1.$, we have:

$$PFH(T) = \frac{1 - \exp(-\Lambda_{as}.T)}{T} \approx \Lambda_{as} = \frac{1}{MTTF}$$

Λ_{as} exists only when the E/E/PE safety-related system working in continuous mode comprises only safe and DD failures (i.e. quickly detected and repaired). No repair of failures that can directly cause the overall failure of the safety function can be considered. For a redundant configuration where it is relevant to consider proof tests, then the asymptotic failure rate is not relevant and the previous equations have to be used. It is the job of the analyst to verify which case is relevant.

B.2.3.3 Unavailability case (e.g. multiple safety barriers)

When the E/E/PE safety-related system working in continuous mode is not the ultimate barrier, its failures only increase the demand frequency on other safety barriers, or when it is working in the high demand mode, such that it is possible to detect (automatically or manually) and repair a fault that could cause the direct loss of the safety function within the expected demand period. In this case its overall failures can be repaired and *PFH* may be calculated from the availability, *A(t),* and from the conditional failure intensity, $\Lambda_V(t)$, of the system.

Again, when the system is made of components completely and quickly repairable (i.e. when, in any degraded situation, there is an high probability to come back quickly to a good working state), $\Lambda_V(t)$ reaches quickly its asymptotic value, Λ_{Vas}, which, in addition, is also a good approximation of the true asymptotic overall system failure rate, $\Lambda_{as,}$ introduced in B.2.3.2.

This leads to the following approximations:

$$PFH = \frac{1}{MUT + MDT} = \frac{1}{MTBF} \approx \frac{1}{MUT} \approx \frac{1}{MTTF}$$

where

MUT is the acronym for Mean Up Time;

MDT is the acronym for Mean Down Time;

MTBF is the acronym for Mean Time Between Failure; and

MTTF is the acronym for Mean Time To Failure.

B.2.3.4 Failure rate considerations

Several formulae above use the overall system failure rate $\Lambda(t)$. Its evaluation is not so easy and some reminders are needed.

Series structures are very simple to handle as failure rates can be added. From Figure B.1, we can write $\Lambda(t) = \Lambda^{abc}(t) + \lambda^{CCF1}(t) + \lambda^{d}(t) + \Lambda^{ef}(t) + \lambda^{CCF2}(t)$ where $\Lambda(t)$ is the overall failure rate of the E/E/PE safety-related system and $\Lambda^{abc}(t)$, $\lambda^{CCF1}(t)$, $\lambda^{d}(t)$, $\Lambda^{ef}(t)$ and $\lambda^{CCF2}(t)$ the failure rates of the five minimal cut sets.

For parallel structures this is not so simple because there are no simple relationships with individual components failure rates. For example, let us consider cut set (E, F):

1) When E and F cannot be immediately be restored (e.g. DU failures), $\Lambda^{ef}(t)$ varies continuously from 0 to λ (failure rate of E or F). An asymptotic value is reached when one of the two components is likely to have failed. This is a very slow process as this arises when t becomes larger than $1/\lambda$. This asymptotic value will be never reached in actual life if E and F are periodically proof tested with a test interval $\tau << 1/\lambda$.
2) When E and F are restored in a relatively short period of time (e.g. DD failures), $\Lambda^{ef}(t)$ goes very fast to an asymptotic value $\Lambda^{ef}_{As} = 2\lambda^2/\mu$ which can be used as an equivalent constant failure rate. It is reached when t becomes greater than two or three times the larger *MTTR* of the components. It is a particular case of the completely and quickly repairable systems discussed above.

Therefore, in the general case, evaluating the overall system failure rates imply more complex calculations than the more simple series structure.

B.3 Reliability block diagram approach, assuming constant failure rate

B.3.1 Underlying hypothesis

The calculations are based on the following assumptions:

– the resulting average probability of failure on demand for the system is less than 10^{-1}, or the resultant average frequency of dangerous failure for the system is less than 10^{-5} h^{-1}

NOTE 1 This assumption means that the E/E/PE safety-related system is within the scope of IEC 61508 series and within the SIL 1 band (see Tables 2 and 3 of IEC 61508-1).

– component failure rates are constant over the life of the system;

– the sensor (input) subsystem comprises the actual sensor(s) and any other components and wiring, up to but not including the component(s) where the signals are first combined by voting or other processing (for example for two sensor channels, the configuration would be as shown in Figure B.2);

– the logic subsystem comprises the component(s) where the signals are first combined, and all other components up to and including where final signal(s) are presented to the final element subsystem;

– the final element (output) subsystem comprises all the components and wiring which process the final signal(s) from the logic subsystem including the final actuating component(s);

– the hardware failure rates used as inputs to the calculations and tables are for a single channel of the subsystem (for example, if 2oo3 sensors are used, the failure rate is for a single sensor and the effect of 2oo3 is calculated separately);

– the channels in a voted group all have the same failure rates and diagnostic coverage;

– the overall hardware failure rate of a channel of the subsystem is the sum of the dangerous failure rate and safe failure rate for that channel, which are assumed to be equal;

NOTE 2 This assumption affects the safe failure fraction (see Annex C of IEC 61508-2), but the safe failure fraction does not affect the calculated values for probability of failure given in this annex.

– for each safety function, there is perfect proof testing and repair (i.e. all failures that remain undetected are detected by the proof test), but for the effects of a non-perfect proof test see B.3.2.5;

– the proof test interval is at least an order of magnitude greater than the MRT;

– for each subsystem there is a single proof test interval and MRT;

– the expected interval between demands is at least an order of magnitude greater than the proof test interval;

– for all subsystems operating in low demand mode of operation, and for 1oo2, 1oo2D,1oo3 and 2oo3, voted groups operating in high demand or continuous mode of operation, the fraction of failures specified by the diagnostic coverage is both detected and repaired within the MTTR used to determine hardware safety integrity requirements;

EXAMPLE If a MTTR of 8 h is assumed, this includes the diagnostic test interval which is typically less than 1 h, the remainder being the MRT.

NOTE 3 For 1oo2, 1oo2D, 1oo3 and 2oo3 voted groups, it is assumed that any repair is on-line. Configuring an E/E/PE safety-related system, so that on any detected fault the EUC is put into a safe state, improves the average probability of failure on demand. The degree of improvement depends on the diagnostic coverage.

– for 1oo1 and 2oo2 voted groups operating in high demand or continuous mode of operation, the E/E/PE safety-related system always achieves a safe state after detecting a dangerous fault; to achieve this, the expected interval between demands is at least an order of magnitude greater than the diagnostic test intervals, or the sum of the diagnostic test intervals and the time to achieve a safe state is less than the process safety time;

NOTE 4 For process safety time see 3.6.20 of IEC 61508-4.

– when a power supply failure removes power from a de-energize-to-trip E/E/PE safety-related system and initiates a system trip to a safe state, the power supply does not affect the average probability of failure on demand of the E/E/PE safety-related system; if the system is energized to trip, or the power supply has failure modes that can cause unsafe operation of the E/E/PE safety-related system, the power supply should be included in the evaluation;

– where the term channel is used, it is limited to only that part of the system under discussion, which is usually either the sensor, logic or final element subsystem;

– the abbreviated terms are described in Table B.1.

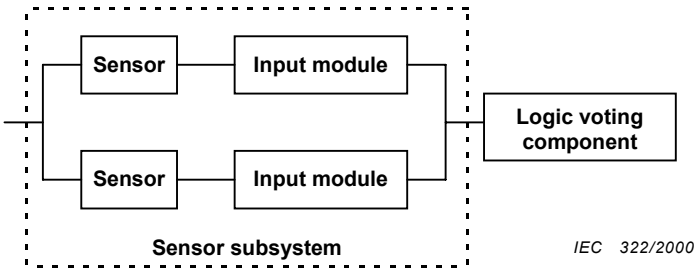


Figure B.2 – Example configuration for two sensor channels

Table B.1 – Terms and their ranges used in this annex
(applies to 1oo1, 1oo2, 2oo2, 1oo2D, 1oo3 and 2oo3)

Abbre-viation	Term (units)	Parameter ranges in Tables B.2 to B.5 and B.10 to B.13
T_1	Proof test interval (hour)	One month (730 h)[1] Three months (2 190 h)[1] Six months (4 380 h) One year (8 760 h) Two years (17 520 h)[2] Ten years (87 600 h)[2]
MTTR	Mean time to restoration (hour)	8 h NOTE MTTR = MRT = 8 hours is based on the assumption that the time to detect a dangerous failure, based on automatic detection, is << MRT.
MRT	Mean repair time (hour)	8 h NOTE MTTR = MRT = 8 hours is based on the assumption that the time to detect a dangerous failure, based on automatic detection, is << MRT
DC	Diagnostic coverage (expressed as a fraction in the equations and as a percentage elsewhere)	0 % 60 % 90 % 99 %
β	The fraction of undetected failures that have a common cause (expressed as a fraction in the equations and as a percentage elsewhere) (Tables B.2 to B.5 and B.10 to B.13 assume $\beta = 2 \times \beta_D$)	2 % 10 % 20 %
β_D	Of those failures that are detected by the diagnostic tests, the fraction that have a common cause (expressed as a fraction in the equations and as a percentage elsewhere) (Tables B.2 to B.5 and B.10 to B.13 assume $\beta = 2 \times \beta_D$)	1 % 5 % 10 %
λ_{DU}	Dangerous Undetected failure rate (per hour) of a channel in a subsystem	$0{,}05 \times 10^{-6}$ $0{,}25 \times 10^{-6}$ $0{,}5 \times 10^{-6}$ $2{,}5 \times 10^{-6}$ 5×10^{-6} 25×10^{-6}
PFD_G	Average probability of failure on demand for the group of voted channels (If the sensor, logic or final element subsystem comprises of only one voted group, then PFD_G is equivalent to PFD_S, PFD_L or PFD_{FE} respectively)	
PFD_S	Average probability of failure on demand for the sensor subsystem	
PFD_L	Average probability of failure on demand for the logic subsystem	

Abbre-viation	Term (units)	Parameter ranges in Tables B.2 to B.5 and B.10 to B.13
PFD_{FE}	Average probability of failure on demand for the final element subsystem	
PFD_{SYS}	Average probability of failure on demand of a safety function for the E/E/PE safety-related system	
PFH_G	Average frequency of dangerous failure for the group of voted channels (if the sensor, logic or final element subsystem comprises of only one voted group, then PFH_G is equivalent to PFH_S, PFH_L or PFH_{FE} respectively)	
PFH_S	Average frequency of dangerous failure for the sensor subsystem	
PFH_L	Average frequency of dangerous failure for the logic subsystem	
PFH_{FE}	Average frequency of dangerous failure for the final element subsystem	
PFH_{SYS}	Average frequency of dangerous failure of a safety function for the E/E/PE safety-related system	
λ	Total failure rate (per hour) of a channel in a subsystem	
λ_D	Dangerous failure rate (per hour) of a channel in a subsystem, equal to 0,5 λ (assumes 50 % dangerous failures and 50 % safe failures)	
λ_{DD}	Detected dangerous failure rate (per hour) of a channel in a subsystem (this is the sum of all the detected dangerous failure rates within the channel of the subsystem)	
λ_{DU}	Undetected dangerous failure rate (per hour) of a channel in a subsystem (this is the sum of all the undetected dangerous failure rates within the channel of the subsystem)	
λ_{SD}	Detected safe failure rate (per hour) of a channel in a subsystem (this is the sum of all the detected safe failure rates within the channel of the subsystem)	
t_{CE}	Channel equivalent mean down time (hour) for 1oo1, 1oo2, 2oo2 and 2oo3 architectures (this is the combined down time for all the components in the channel of the subsystem)	
t_{GE}	Voted group equivalent mean down time (hour) for 1oo2 and 2oo3 architectures (this is the combined down time for all the channels in the voted group)	
t_{CE}'	Channel equivalent mean down time (hour) for 1oo2D architecture (this is the combined down time for all the components in the channel of the subsystem)	
t_{GE}'	Voted group equivalent mean down time (hour) for 1oo2D architecture (this is the combined down time for all the channels in the voted group)	
T_2	Interval between demands (h)	
K	Fraction of the success of the autotest circuit in the 1oo2D system	
PTC	Proof Test Coverage	
[1] High demand or continuous mode only.		
[2] Low demand mode only.		

B.3.2 Average probability of failure on demand (for low demand mode of operation)

B.3.2.1 Procedure for calculations

The average probability of failure on demand of a safety function for the E/E/PE safety-related system is determined by calculating and combining the average probability of failure on demand for all the subsystems which together implement the safety function. Since in this annex the probabilities are small, this can be expressed by the following (see Figure B.3):

$$PFD_{SYS} = PFD_S + PFD_L + PFD_{FE}$$

where

PFD_{SYS} is the average probability of failure on demand of a safety function for the E/E/PE safety-related system;

PFD_S is the average probability of failure on demand for the sensor subsystem;

PFD_L is the average probability of failure on demand for the logic subsystem; and

PFD_{FE} is the average probability of failure on demand for the final element subsystem.

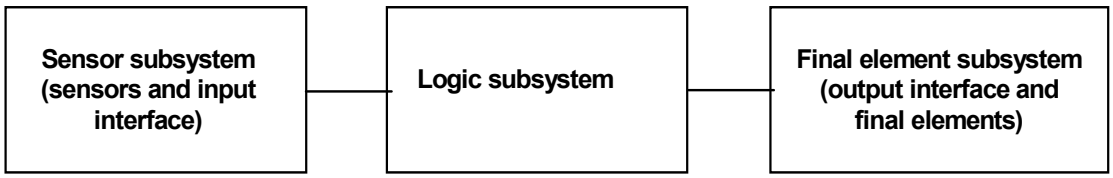


Figure B.3 – Subsystem structure

To determine the average probability of failure on demand for each of the subsystems, the following procedure should be adhered to for each subsystem in turn.

a) Draw the block diagram showing the sensor subsystem (input) components, logic subsystem components or final element subsystem (output) components. For example, sensor subsystem components may be sensors, barriers, input conditioning circuits; logic subsystem components may be processors and scanning devices; and final element subsystem components may be output conditioning circuits, barriers and actuators. Represent each subsystem as one or more 1oo1, 1oo2, 2oo2, 1oo2D, 1oo3 or 2oo3 voted groups.

b) Refer to the relevant table from Tables B.2 to B.5 which are for six-month, one-year, two-year and 10-year proof test intervals. These tables also assume an 8 h mean time to restoration for each failure once it has been revealed.

c) For each voted group in the subsystem, select from the relevant table of Tables B.2 to B.5:

   – architecture (for example, 2oo3);

   – diagnostic coverage of each channel (for example, 60 %);

   – the dangerous failure rate (per hour), λ_D, of each channel (for example, $2{,}5 \times 10^{-06}$);

   – the common cause failure β-factors, β and β_D, for the interaction between the channels in the voted group (for example, 2 % and 1 % respectively).

NOTE 1 It is assumed that every channel in the voted group has the same diagnostic coverage and failure rate (see Table B.1).

NOTE 2 It is assumed in Tables B.2 to B.5 (and in Tables B.10 to B.13) that the β-factor in the absence of diagnostic tests (also used for undetected dangerous failures in the presence of diagnostic tests), β, is 2 times the β-factor for failures detected by the diagnostic tests, β_D.

d) Obtain, from the relevant table from Tables B.2 to B.5, the average probability of failure on demand for the voted group.

e) If the safety function depends on more than one voted group of sensors or actuators, the combined average probability of failure on demand of the sensor or final element subsystem, PFD_S or PFD_{FE}, is given in the following equations, where PFD_{Gi} and PFD_{Gj} is the average probability of failure on demand for each voted group of sensors and final elements respectively:

$$PFD_S = \sum_i PFD_{Gi}$$

$$PFD_{FE} = \sum_j PFD_{Gj}$$

The formula used in all the equations for both *PFD* and system failure rate are all a function of component failure rate and mean down time (*MDT*). Where there are a number of elements in the system and it is required to calculate the combined elements overall *PFD* or system failure rate, then it is often necessary to use a single value for the *MDT* in the equations. However each element may have different failure detection mechanisms with different *MDT* and different elements may have different *MDT* values for the same failure mechanisms, in which case it is necessary to calculate a single value for the *MDT* which can represent all the elements in the path. This can be accomplished by considering the total paths overall failure rate then proportioning the individual *MDT's* equivalent to their failure rate contribution to the total failure rate under consideration.

As an example, if there are two elements in series but one with a proof test, T_1, and the other with a proof test, T_2, then the equivalent single value for the *MDT* is:

$$\lambda_T = \lambda_1 + \lambda_2$$

and

$$MDT_E = \frac{\lambda_1}{\lambda_T}\left(\frac{T_1}{2}\right) + \frac{\lambda_2}{\lambda_T}\left(\frac{T_2}{2}\right)$$

B.3.2.2 Architectures for low demand mode of operation

NOTE 1 This subclause should be read sequentially, since equations which are valid for several architectures are only stated where they are first used.

NOTE 2 The equations are based on the assumptions listed in B.3.1.

NOTE 3 The following examples are typical configurations and are not intended to be an exhaustive.

B.3.2.2.1 1oo1

This architecture consists of a single channel, where any dangerous failure leads to a failure of the safety function when a demand arises.

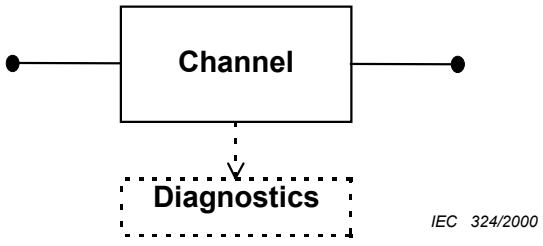


Figure B.4 – 1oo1 physical block diagram

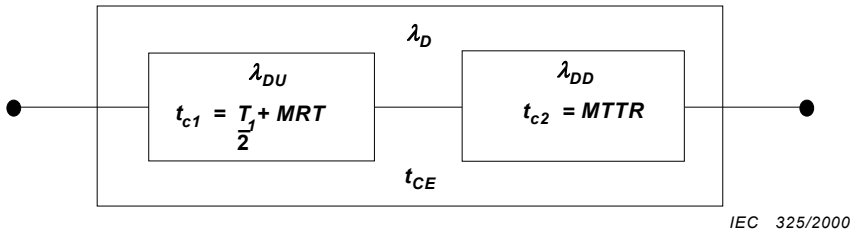


Figure B.5 – 1oo1 reliability block diagram

Figures B.4 and B.5 contain the relevant block diagrams. The dangerous failure rate for the channel is given by

$$\lambda_D = \lambda_{DU} + \lambda_{DD}$$

Figure B.5 shows that the channel can be considered to comprise of two components, one with a dangerous failure rate λ_{DU} resulting from undetected failures and the other with a dangerous failure rate λ_{DD} resulting from detected failures. It is possible to calculate the channel equivalent mean down time t_{CE}, adding the individual down times from both components, t_{c1} and t_{c2}, in direct proportion to each component's contribution to the probability of failure of the channel:

$$t_{CE} = \frac{\lambda_{DU}}{\lambda_D}\left(\frac{T_1}{2} + MRT\right) + \frac{\lambda_{DD}}{\lambda_D} MTTR$$

For every architecture, the detected dangerous failure rate and the undetected dangerous failure rate are given by

$$\lambda_{DU} = \lambda_D(1 - DC); \quad \lambda_{DD} = \lambda_D DC$$

For a channel with down time t_{CE} resulting from dangerous failures

$$PFD = 1 - e^{-\lambda_D t_{CE}}$$
$$\approx \lambda_D t_{CE} \qquad \text{since } \lambda_D t_{CE} << 1$$

Hence, for a 1oo1 architecture, the average probability of failure on demand is

$$PFD_G = (\lambda_{DU} + \lambda_{DD}) t_{CE}$$

B.3.2.2.2 1oo2

This architecture consists of two channels connected in parallel, such that either channel can process the safety function. Thus there would have to be a dangerous failure in both channels before a safety function failed on demand. It is assumed that any diagnostic testing would only report the faults found and would not change any output states or change the output voting.

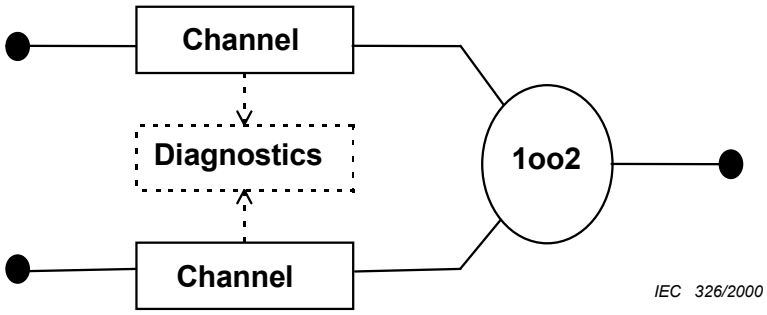


Figure B.6 – 1oo2 physical block diagram

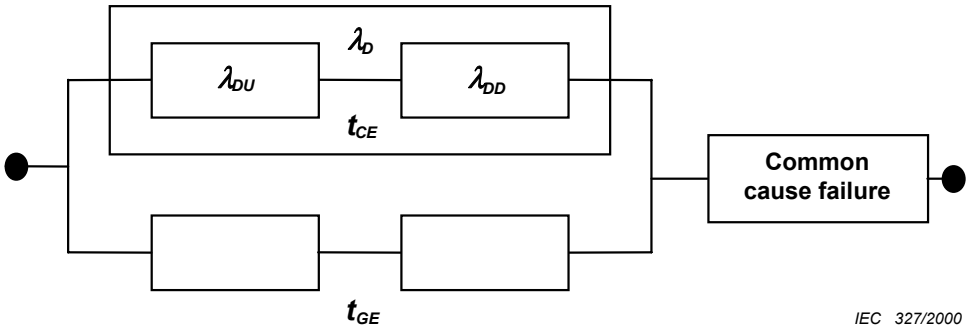


Figure B.7 – 1oo2 reliability block diagram

Figures B.6 and B.7 contain the relevant block diagrams. The value of t_{CE} is as given in B.3.2.2.1, but now it is necessary to also calculate the system equivalent down time t_{GE}, which is given by

$$t_{GE} = \frac{\lambda_{DU}}{\lambda_D}\left(\frac{T_1}{3} + MRT\right) + \frac{\lambda_{DD}}{\lambda_D} MTTR$$

The average probability of failure on demand for the architecture is

$$PFD_G = 2\left((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU}\right)^2 t_{CE} t_{GE} + \beta_D \lambda_{DD} MTTR + \beta\lambda_{DU}\left(\frac{T_1}{2} + MRT\right)$$

B.3.2.2.3 2oo2

This architecture consists of two channels connected in parallel so that both channels need to demand the safety function before it can take place. It is assumed that any diagnostic testing would only report the faults found and would not change any output states or change the output voting.

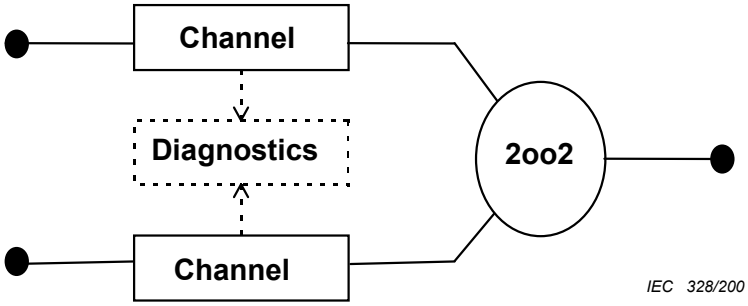


Figure B.8 – 2oo2 physical block diagram

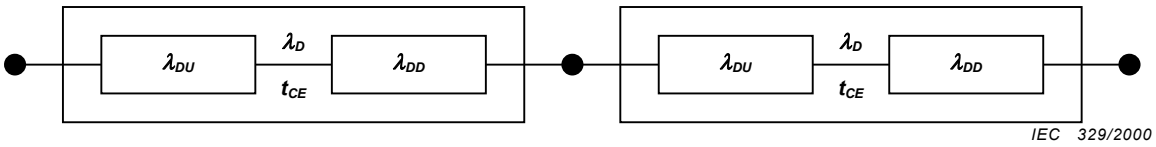


Figure B.9 – 2oo2 reliability block diagram

Figures B.8 and B.9 contain the relevant block diagrams. The value of t_{CE} is as given in B.3.2.2.1, and the average probability of failure on demand for the architecture is

$$PFD_G = 2\lambda_D t_{CE}$$

B.3.2.2.4 1oo2D

This architecture consists of two channels connected in parallel. During normal operation, both channels need to demand the safety function before it can take place. In addition, if the diagnostic tests in either channel detect a fault then the output voting is adapted so that the overall output state then follows that given by the other channel. If the diagnostic tests find faults in both channels or a discrepancy that cannot be allocated to either channel, then the output goes to the safe state. In order to detect a discrepancy between the channels, either channel can determine the state of the other channel via a means independent of the other channel. The channel comparison / switch over mechanism may not be 100 % efficient therefore K represents the efficiency of this inter-channel comparison / switch mechanism, i.e. the output may remain on the 2oo2 voting even with one channel detected as faulty.

NOTE The parameter K will need to be determined by an FMEA.

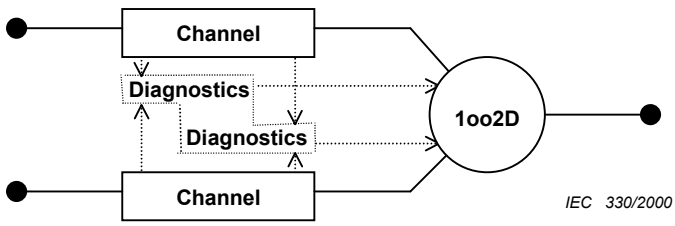


Figure B.10 – 1oo2D physical block diagram

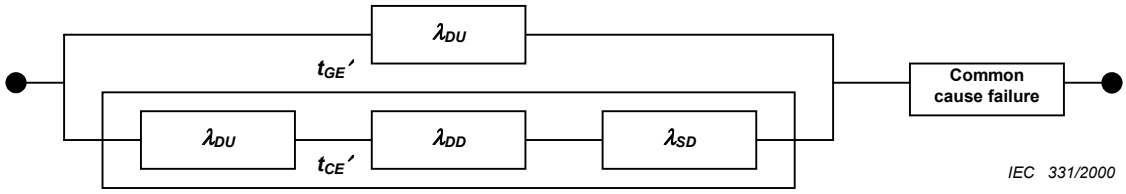


Figure B.11 – 1oo2D reliability block diagram

The detected safe failure rate for every channel is given by

$$\lambda_{SD} = \lambda_S DC$$

Figures B.10 and B.11 contain the relevant block diagrams. The values of the equivalent mean down times differ from those given for the other architectures in B.3.2.2 and hence are labelled t_{CE}' and t_{GE}'. Their values are given by

$$t_{CE}' = \frac{\lambda_{DU}\left(\frac{T_1}{2} + MRT\right) + (\lambda_{DD} + \lambda_{SD})MTTR}{\lambda_{DU} + (\lambda_{DD} + \lambda_{SD})}$$

$$t_{GE}' = \frac{T_1}{3} + MRT$$

The average probability of failure on demand for the architecture is

$$PFD_G = 2(1-\beta)\lambda_{DU}((1-\beta)\lambda_{DU} + (1-\beta_D)\lambda_{DD} + \lambda_{SD})t_{CE}'\,t_{GE}' + 2(1-K)\lambda_{DD}t_{CE}' + \beta\lambda_{DU}\left(\frac{T_1}{2} + MRT\right)$$

B.3.2.2.5 2oo3

This architecture consists of three channels connected in parallel with a majority voting arrangement for the output signals, such that the output state is not changed if only one channel gives a different result which disagrees with the other two channels.

It is assumed that any diagnostic testing would only report the faults found and would not change any output states or change the output voting.

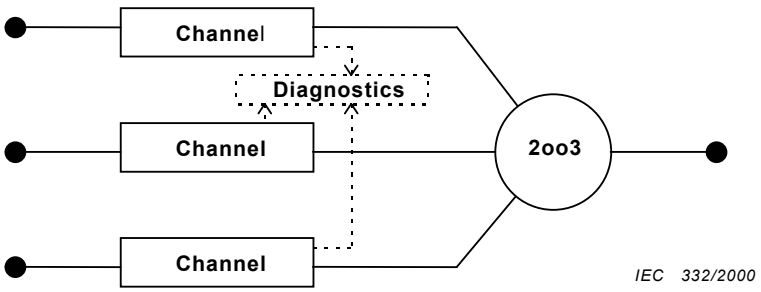


Figure B.12 – 2oo3 physical block diagram

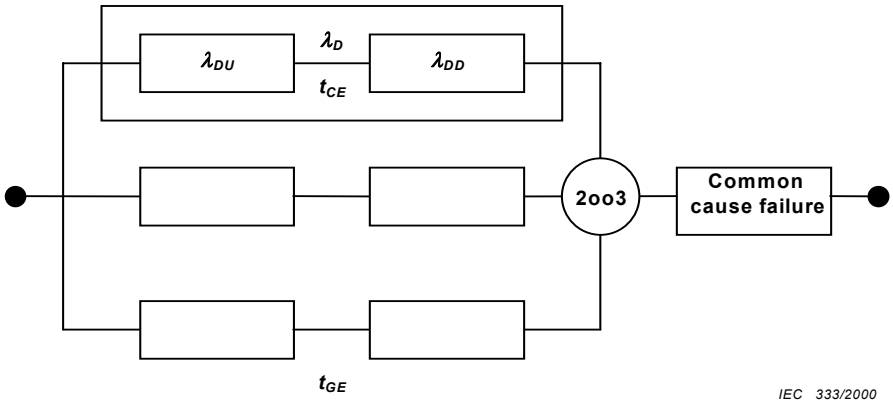


Figure B.13 – 2oo3 reliability block diagram

Figures B.12 and B.13 contain the relevant block diagrams. The value of t_{CE} is as given in B.3.2.2.1 and the value of t_{GE} is as given in B.3.2.2.2. The average probability of failure on demand for the architecture is

$$PFD_G = 6((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU})^2 t_{CE} t_{GE} + \beta_D \lambda_{DD} MTTR + \beta\lambda_{DU}\left(\frac{T_1}{2} + MRT\right)$$

B.3.2.2.6 1oo3

This architecture consists of three channels connected in parallel with a voting arrangement for the output signals, such that the output state follows 1oo3 voting.

It is assumed that any diagnostic testing would only report the faults found and would not change any output states or change the output voting.

The reliability diagram will be the same as for the 2oo3 case but with voting 1oo3. The value of t_{CE} is as given in B.3.2.2.1 and the value of t_{GE} is as given in B.3.2.2.2. The average probability of failure on demand for the architecture is

$$PFD_G = 6((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU})^3 t_{CE} t_{GE} t_{G2E} + \beta_D \lambda_{DD} MTTR + \beta\lambda_{DU}\left(\frac{T_1}{2} + MRT\right)$$

Where

IEC 332/2000

$$t_{G2E} = \frac{\lambda_{DU}}{\lambda_D}\left(\frac{T_1}{4} + MRT\right) + \frac{\lambda_{DD}}{\lambda_D} MTTR$$

B.3.2.3 Detailed tables for low demand mode of operation

Table B.2 – Average probability of failure on demand for a proof test interval of six months and a mean time to restoration of 8 h

Architecture	*DC*	λ_D = 0,5E-07			λ_D = 2,5E-07			λ_D = 0,5E-06		
		β = 2 %, β_D = 1 %	β = 10 %, β_D = 5 %	β = 20 %, β_D = 10 %	β = 2 %, β_D = 1 %	β = 10 %, β_D = 5 %	β = 20 %, β_D = 10 %	β = 2 %, β_D = 1 %	β = 10 %, β_D = 5 %	β = 20 %, β_D = 10 %
1oo1 (see Note 2)	**0 %**	1,1E-04			5,5E-04			1,1E-03		
	60 %	4,4E-05			2,2E-04			4,4E-04		
	90 %	1,1E-05			5,7E-05			1,1E-04		
	99 %	1,5E-06			7,5E-06			1,5E-05		
1oo2	**0 %**	2,2E-06	1,1E-05	2,2E-05	1,1E-05	5,5E-05	1,1E-04	2,4E-05	1,1E-04	2,2E-04
	60 %	8,8E-07	4,4E-06	8,8E-06	4,5E-06	2,2E-05	4,4E-05	9,1E-06	4,4E-05	8,8E-05
	90 %	2,2E-07	1,1E-06	2,2E-06	1,1E-06	5,6E-06	1,1E-05	2,3E-06	1,1E-05	2,2E-05
	99 %	2,6E-08	1,3E-07	2,6E-07	1,3E-07	6,5E-07	1,3E-06	2,6E-07	1,3E-06	2,6E-06
2oo2 (see Note 2)	**0 %**	2,2E-04			1,1E-03			2,2E-03		
	60 %	8,8E-05			4,4E-04			8,8E-04		
	90 %	2,3E-05			1,1E-04			2,3E-04		
	99 %	3,0E-06			1,5E-05			3,0E-05		
1oo2D (see Note 3)	**0 %**	2,2E-06	1,1E-05	2,2E-05	1,1E-05	5,5E-05	1,1E-04	2,4E-05	1,1E-04	2,2E-04
	60 %	1,4E-06	4,9E-06	9,3E-06	7,1E-06	2,5E-05	4,7E-05	1,4E-05	5,0E-05	9,3E-05
	90 %	4,3E-07	1,3E-06	2,4E-06	2,2E-06	6,6E-06	1,2E-05	4,3E-06	1,3E-05	2,4E-05
	99 %	6,0E-08	1,5E-07	2,6E-07	3,0E-07	7,4E-07	1,3E-06	6,0E-07	1,5E-06	2,6E-06
2oo3	**0 %**	2,2E-06	1,1E-05	2,2E-05	1,2E-05	5,6E-05	1,1E-04	2,7E-05	1,1E-04	2,2E-04
	60 %	8,9E-07	4,4E-06	8,8E-06	4,6E-06	2,2E-05	4,4E-05	9,6E-06	4,5E-05	8,9E-05
	90 %	2,2E-07	1,1E-06	2,2E-06	1,1E-06	5,6E-06	1,1E-05	2,3E-06	1,1E-05	2,2E-05
	99 %	2,6E-08	1,3E-07	2,6E-07	1,3E-07	6,5E-07	1,3E-06	2,6E-07	1,3E-06	2,6E-06
1oo3	**0 %**	2,2E-06	1,1E-05	2,2E-05	1,1E-05	5,5E-05	1,1E-04	2,2E-05	1,1E-04	2,2E-04
	60 %	8,8E-07	4,4E-06	8,8E-06	4,4E-06	2,2E-05	4,4E-05	8,8E-06	4,4E-05	8,8E-05
	90 %	2,2E-07	1,1E-06	2,2E-06	1,1E-06	5,6E-06	1,1E-05	2,2E-06	1,1E-05	2,2E-05
	99 %	2,6E-08	1,3E-07	2,6E-07	1,3E-07	6,5E-07	1,3E-06	2,6E-07	1,3E-06	2,6E-06

Architecture	*DC*	λ_D = 2,5E-06			λ_D = 0,5E-05			λ_D = 2,5E-05		
		β = 2 %, β_D = 1 %	β = 10 %, β_D = 5 %	β = 20 %, β_D = 10 %	β = 2 %, β_D = 1 %	β = 10 %, β_D = 5 %	β = 20 %, β_D = 10 %	β = 2 %, β_D = 1 %	β = 10 %, β_D = 5 %	β = 20 %, β_D = 10 %
1oo1 (see Note 2)	**0 %**	5,5E-03			1,1E-02			5,5E-02		
	60 %	2,2E-03			4,4E-03			2,2E-02		
	90 %	5,7E-04			1,1E-03			5,7E-03		
	99 %	7,5E-05			1,5E-04			7,5E-04		
1oo2	**0 %**	1,5E-04	5,8E-04	1,1E-03	3,7E-04	1,2E-03	2,3E-03	5,0E-03	8,8E-03	1,4E-02
	60 %	5,0E-05	2,3E-04	4,5E-04	1,1E-04	4,6E-04	9,0E-04	1,1E-03	2,8E-03	4,9E-03
	90 %	1,2E-05	5,6E-05	1,1E-04	2,4E-05	1,1E-04	2,2E-04	1,5E-04	6,0E-04	1,2E-03
	99 %	1,3E-06	6,5E-06	1,3E-05	2,6E-06	1,3E-05	2,6E-05	1,4E-05	6,6E-05	1,3E-04
2oo2 (see Note 2)	**0 %**	1,1E-02			2,2E-02			>1E-01		
	60 %	4,4E-03			8,8E-03			4,4E-02		
	90 %	1,1E-03			2,3E-03			1,1E-02		
	99 %	1,5E-04			3,0E-04			1,5E-03		
1oo2D (see Note 3)	**0 %**	1,5E-04	5,8E-04	1,1E-03	3,8E-04	1,2E-03	2,3E-03	5,0E-03	9,0E-03	1,4E-02
	60 %	7,7E-05	2,5E-04	4,7E-04	1,7E-04	5,2E-04	9,5E-04	1,3E-03	3,0E-03	5,1E-03
	90 %	2,2E-05	6,6E-05	1,2E-04	4,5E-05	1,3E-04	2,4E-04	2,6E-04	6,9E-04	1,2E-03
	99 %	3,0E-06	7,4E-06	1,3E-05	6,0E-06	1,5E-05	2,6E-05	3,0E-05	7,4E-05	1,3E-04
2oo3	**0 %**	2,3E-04	6,5E-04	1,2E-03	6,8E-04	1,5E-03	2,5E-03	1,3E-02	1,5E-02	1,9E-02
	60 %	6,3E-05	2,4E-04	4,6E-04	1,6E-04	5,1E-04	9,4E-04	2,3E-03	3,9E-03	5,9E-03
	90 %	1,2E-05	5,7E-05	1,1E-04	2,7E-05	1,2E-04	2,3E-04	2,4E-04	6,8E-04	1,2E-03
	99 %	1,3E-06	6,5E-06	1,3E-05	2,7E-06	1,3E-05	2,6E-05	1,5E-05	6,7E-05	1,3E-04
1oo3	**0 %**	1,1E-04	5,5E-04	1,1E-03	2,2E-04	1,1E-03	2,2E-03	1,4E-03	5,7E-03	1,1E-02
	60 %	4,4E-05	2,2E-04	4,4E-04	8,8E-05	4,4E-04	8,8E-04	4,6E-04	2,2E-03	4,4E-03
	90 %	1,1E-05	5,6E-05	1,1E-04	2,2E-05	1,1E-04	2,2E-04	1,1E-04	5,6E-04	1,1E-03
	99 %	1,3E-06	6,5E-06	1,3E-05	2,6E-06	1,3E-05	2,6E-05	1,3E-05	6,5E-05	1,3E-04

NOTE 1 This table gives example values of PFD_G, calculated using the equations in B.3.2 and depending on the assumptions listed in B.3.1. If the sensor, logic or final element subsystem comprises of only one group of voted channels, then PFD_G is equivalent to PFD_S, PFD_L or PFD_{FE} respectively (see B.3.2.1).

NOTE 2 The table assumes $\beta = 2 \times \beta_D$. For 1oo1 and 2oo2 architectures, the values of β and β_D do not affect the average probability of failure.

NOTE 3 The safe failure rate is assumed to be equal to the dangerous failure rate and K = 0,98.

Table B.3 – Average probability of failure on demand for a proof test interval of one year and mean time to restoration of 8 h

Architecture	*DC*	λ_D = 0,5E-07			λ_D = 2,5E-07			λ_D = 0,5E-06		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see Note 2)	**0 %**	2,2E-04			1,1E-03			2,2E-03		
	60 %	8,8E-05			4,4E-04			8,8E-04		
	90 %	2,2E-05			1,1E-04			2,2E-04		
	99 %	2,6E-06			1,3E-05			2,6E-05		
1oo2	**0 %**	4,4E-06	2,2E-05	4,4E-05	2,3E-05	1,1E-04	2,2E-04	5,0E-05	2,2E-04	4,4E-04
	60 %	1,8E-06	8,8E-06	1,8E-05	9,0E-06	4,4E-05	8,8E-05	1,9E-05	8,9E-05	1,8E-04
	90 %	4,4E-07	2,2E-06	4,4E-06	2,2E-06	1,1E-05	2,2E-05	4,5E-06	2,2E-05	4,4E-05
	99 %	4,8E-08	2,4E-07	4,8E-07	2,4E-07	1,2E-06	2,4E-06	4,8E-07	2,4E-06	4,8E-06
2oo2 (see Note 2)	**0 %**	4,4E-04			2,2E-03			4,4E-03		
	60 %	1,8E-04			8,8E-04			1,8E-03		
	90 %	4,5E-05			2,2E-04			4,5E-04		
	99 %	5,2E-06			2,6E-05			5,2E-05		
1oo2D (see Note 3)	**0 %**	4,5E-06	2,2E-05	4,4E-05	2,4E-05	1,1E-04	2,2E-04	5,0E-05	2,2E-04	4,4E-04
	60 %	2,8E-06	9,8E-06	1,9E-05	1,4E-05	4,9E-05	9,3E-05	2,9E-05	9,9E-05	1,9E-04
	90 %	8,5E-07	2,6E-06	4,8E-06	4,3E-06	1,3E-05	2,4E-05	8,5E-06	2,6E-05	4,8E-05
	99 %	1,0E-07	2,8E-07	5,0E-07	5,2E-07	1,4E-06	2,5E-06	1,0E-06	2,8E-06	5,0E-06
2oo3	**0 %**	4,6E-06	2,2E-05	4,4E-05	2,7E-05	1,1E-04	2,2E-04	6,2E-05	2,4E-04	4,5E-04
	60 %	1,8E-06	8,8E-06	1,8E-05	9,5E-06	4,5E-05	8,8E-05	2,1E-05	9,1E-05	1,8E-04
	90 %	4,4E-07	2,2E-06	4,4E-06	2,3E-06	1,1E-05	2,2E-05	4,6E-06	2,2E-05	4,4E-05
	99 %	4,8E-08	2,4E-07	4,8E-07	2,4E-07	1,2E-06	2,4E-06	4,8E-07	2,4E-06	4,8E-06
1oo3	**0 %**	4,4E-06	2,2E-05	4,4E-05	2,2E-05	1,1E-04	2,2E-04	4,4E-05	2,2E-04	4,4E-04
	60 %	1,8E-06	8,8E-06	1,8E-05	8,8E-06	4,4E-05	8,8E-05	1,8E-05	8,8E-05	1,8E-04
	90 %	4,4E-07	2,2E-06	4,4E-06	2,2E-06	1,1E-05	2,2E-05	4,4E-06	2,2E-05	4,4E-05
	99 %	4,8E-08	2,4E-07	4,8E-07	2,4E-07	1,2E-06	2,4E-06	4,8E-07	2,4E-06	4,8E-06

Architecture	*DC*	λ_D = 2,5E-06			λ_D = 0,5E-05			λ_D = 2,5E-05		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see Note 2)	**0 %**	1,1E-02			2,2E-02			>1E-01		
	60 %	4,4E-03			8,8E-03			4,4E-02		
	90 %	1,1E-03			2,2E-03			1,1E-02		
	99 %	1,3E-04			2,6E-04			1,3E-03		
1oo2	**0 %**	3,7E-04	1,2E-03	2,3E-03	1,1E-03	2,7E-03	4,8E-03	1,8E-02	2,4E-02	3,2E-02
	60 %	1,1E-04	4,6E-04	9,0E-04	2,8E-04	9,7E-04	1,8E-03	3,4E-03	6,6E-03	1,1E-02
	90 %	2,4E-05	1,1E-04	2,2E-04	5,1E-05	2,3E-04	4,5E-04	3,8E-04	1,3E-03	2,3E-03
	99 %	2,4E-06	1,2E-05	2,4E-05	4,9E-06	2,4E-05	4,8E-05	2,6E-05	1,2E-04	2,4E-04
2oo2 (see Note 2)	**0 %**	2,2E-02			4,4E-02			>1E-01		
	60 %	8,8E-03			1,8E-02			8,8E-02		
	90 %	2,2E-03			4,5E-03			2,2E-02		
	99 %	2,6E-04			5,2E-04			2,6E-03		
1oo2D (see Note 3)	**0 %**	3,8E-04	1,2E-03	2,3E-03	1,1E-03	2,7E-03	4,9E-03	1,8E-02	2,5E-02	3,4E-02
	60 %	1,7E-04	5,1E-04	9,5E-04	3,8E-04	1,1E-03	1,9E-03	3,9E-03	7,1E-03	1,1E-02
	90 %	4,4E-05	1,3E-04	2,4E-04	9,1E-05	2,7E-04	4,8E-04	5,8E-04	1,4E-03	2,5E-03
	99 %	5,2E-06	1,4E-05	2,5E-05	1,0E-05	2,8E-05	5,0E-05	5,4E-05	1,4E-04	2,5E-04
2oo3	**0 %**	6,8E-04	1,5E-03	2,5E-03	2,3E-03	3,8E-03	5,6E-03	4,8E-02	5,0E-02	5,3E-02
	60 %	1,6E-04	5,1E-04	9,4E-04	4,8E-04	1,1E-03	2,0E-03	8,4E-03	1,1E-02	1,5E-02
	90 %	2,7E-05	1,2E-04	2,3E-04	6,4E-05	2,4E-04	4,6E-04	7,1E-04	1,6E-03	2,6E-03
	99 %	2,5E-06	1,2E-05	2,4E-05	5,1E-06	2,4E-05	4,8E-05	3,1E-05	1,3E-04	2,5E-04
1oo3	**0 %**	2,2E-04	1,1E-03	2,2E-03	4,6E-04	2,2E-03	4,4E-03	4,7E-03	1,3E-02	2,3E-02
	60 %	8,8E-05	4,4E-04	8,8E-04	1,8E-04	8,8E-04	1,8E-03	1,0E-03	4,5E-03	8,9E-03
	90 %	2,2E-05	1,1E-04	2,2E-04	4,4E-05	2,2E-04	4,4E-04	2,2E-04	1,1E-03	2,2E-03
	99 %	2,4E-06	1,2E-05	2,4E-05	4,8E-06	2,4E-05	4,8E-05	2,4E-05	1,2E-04	2,4E-04

NOTE 1 This table gives example values of PFD_G, calculated using the equations in B.3.2 and depending on the assumptions listed in B.3.1. If the sensor, logic or final element subsystem comprises of only one group of voted channels, then PFD_G is equivalent to PFD_S, PFD_L or PFD_{FE} respectively (see B.3.2.1).

NOTE 2 The table assumes $\beta = 2 \times \beta_D$. For 1oo1 and 2oo2 architectures, the values of β and β_D do not affect the average probability of failure.

NOTE 3 The safe failure rate is assumed to be equal to the dangerous failure rate and K = 0,98.

Table B.4 – Average probability of failure on demand for a proof test interval of two years and a mean time to restoration of 8 h

Architecture	*DC*	λ_D = 0,5E-07			λ_D = 2,5E-07			λ_D = 0,5E-06		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see Note 2)	**0 %**	4,4E-04			2,2E-03			4,4E-03		
	60 %	1,8E-04			8,8E-04			1,8E-03		
	90 %	4,4E-05			2,2E-04			4,4E-04		
	99 %	4,8E-06			2,4E-05			4,8E-05		
1oo2	**0 %**	9,0E-06	4,4E-05	8,8E-05	5,0E-05	2,2E-04	4,4E-04	1,1E-04	4,6E-04	8,9E-04
	60 %	3,5E-06	1,8E-05	3,5E-05	1,9E-05	8,9E-05	1,8E-04	3,9E-05	1,8E-04	3,5E-04
	90 %	8,8E-07	4,4E-06	8,8E-06	4,5E-06	2,2E-05	4,4E-05	9,1E-06	4,4E-05	8,8E-05
	99 %	9,2E-08	4,6E-07	9,2E-07	4,6E-07	2,3E-06	4,6E-06	9,2E-07	4,6E-06	9,2E-06
2oo2 (see Note 2)	**0 %**	8,8E-04			4,4E-03			8,8E-03		
	60 %	3,5E-04			1,8E-03			3,5E-03		
	90 %	8,8E-05			4,4E-04			8,8E-04		
	99 %	9,6E-06			4,8E-05			9,6E-05		
1oo2D (see Note 3)	**0 %**	9,0E-06	4,4E-05	8,8E-05	5,0E-05	2,2E-04	4,4E-04	1,1E-04	4,6E-04	9,0E-04
	60 %	5,7E-06	2,0E-05	3,7E-05	2,9E-05	9,9E-05	1,9E-04	6,0E-05	2,0E-04	3,7E-04
	90 %	1,7E-06	5,2E-06	9,6E-06	8,5E-06	2,6E-05	4,8E-05	1,7E-05	5,2E-05	9,6E-05
	99 %	1,9E-07	5,4E-07	9,8E-07	9,5E-07	2,7E-06	4,9E-06	1,9E-06	5,4E-06	9,8E-06
2oo3	**0 %**	9,5E-06	4,4E-05	8,8E-05	6,2E-05	2,3E-04	4,5E-04	1,6E-04	5,0E-04	9,3E-04
	60 %	3,6E-06	1,8E-05	3,5E-05	2,1E-05	9,0E-05	1,8E-04	4,7E-05	1,9E-04	3,6E-04
	90 %	8,9E-07	4,4E-06	8,8E-06	4,6E-06	2,2E-05	4,4E-05	9,6E-06	4,5E-05	8,9E-05
	99 %	9,2E-08	4,6E-07	9,2E-07	4,6E-07	2,3E-06	4,6E-06	9,3E-07	4,6E-06	9,2E-06
1oo3	**0 %**	8,8E-06	4,4E-05	8,8E-05	4,4E-05	2,2E-04	4,4E-04	8,8E-05	4,4E-04	8,8E-04
	60 %	3,5E-06	1,8E-05	3,5E-05	1,8E-05	8,8E-05	1,8E-04	3,5E-05	1,8E-04	3,5E-04
	90 %	8,8E-07	4,4E-06	8,8E-06	4,4E-06	2,2E-05	4,4E-05	8,8E-06	4,4E-05	8,8E-05
	99 %	9,2E-08	4,6E-07	9,2E-07	4,6E-07	2,3E-06	4,6E-06	9,2E-07	4,6E-06	9,2E-06

Architecture	*DC*	λ_D =2,5E-06			λ_D = 0,5E-05			λ_D = 2,5E-05		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see Note 2)	**0 %**	2,2E-02			4,4E-02			>1E-01		
	60 %	8,8E-03			1,8E-02			8,8E-02		
	90 %	2,2E-03			4,4E-03			2,2E-02		
	99 %	2,4E-04			4,8E-04			2,4E-03		
1oo2	**0 %**	1,1E-03	2,7E-03	4,8E-03	3,3E-03	6,5E-03	1,0E-02	6,6E-02	7,4E-02	8,5E-02
	60 %	2,8E-04	9,7E-04	1,8E-03	7,5E-04	2,1E-03	3,8E-03	1,2E-02	1,8E-02	2,5E-02
	90 %	5,0E-05	2,3E-04	4,5E-04	1,1E-04	4,6E-04	9,0E-04	1,1E-03	2,8E-03	4,9E-03
	99 %	4,7E-06	2,3E-05	4,6E-05	9,5E-06	4,6E-05	9,2E-05	5,4E-05	2,4E-04	4,6E-04
2oo2 (see Note 2)	**0 %**	4,4E-02			8,8E-02			>1E-01		
	60 %	1,8E-02			3,5E-02			>1E-01		
	90 %	4,4E-03			8,8E-03			4,4E-02		
	99 %	4,8E-04			9,6E-04			4,8E-03		
1oo2D (see Note 3)	**0 %**	1,1E-03	2,7E-03	4,8E-03	3,4E-03	6,6E-03	1,1E-02	6,7E-02	7,7E-02	9,0E-02
	60 %	3,8E-04	1,1E-03	1,9E-03	9,6E-04	2,3E-03	4,0E-03	1,3E-02	1,9E-02	2,6E-02
	90 %	9,0E-05	2,6E-04	4,8E-04	1,9E-04	5,4E-04	9,8E-04	1,5E-03	3,2E-03	5,3E-03
	99 %	9,6E-06	2,7E-05	4,9E-05	1,9E-05	5,4E-05	9,8E-05	1,0E-04	2,8E-04	5,0E-04
2oo3	**0 %**	2,3E-03	3,7E-03	5,6E-03	8,3E-03	1,1E-02	1,4E-02	1,9E-01	1,8E-01	1,7E-01
	60 %	4,8E-04	1,1E-03	2,0E-03	1,6E-03	2,8E-03	4,4E-03	3,2E-02	3,5E-02	4,0E-02
	90 %	6,3E-05	2,4E-04	4,6E-04	1,6E-04	5,1E-04	9,4E-04	2,4E-03	4,0E-03	6,0E-03
	99 %	4,8E-06	2,3E-05	4,6E-05	1,0E-05	4,7E-05	9,2E-05	6,9E-05	2,5E-04	4,8E-04
1oo3	**0 %**	4,6E-04	2,2E-03	4,4E-03	1,0E-03	4,5E-03	8,9E-03	2,4E-02	3,7E-02	5,5E-02
	60 %	1,8E-04	8,8E-04	1,8E-03	3,6E-04	1,8E-03	3,5E-03	3,1E-03	9,9E-03	1,8E-02
	90 %	4,4E-05	2,2E-04	4,4E-04	8,8E-05	4,4E-04	8,8E-04	4,6E-04	2,2E-03	4,4E-03
	99 %	4,6E-06	2,3E-05	4,6E-05	9,2E-06	4,6E-05	9,2E-05	4,6E-05	2,3E-04	4,6E-04

NOTE 1 This table gives example values of PFD_G, calculated using the equations in B.3.2 and depending on the assumptions listed in B.3.1. If the sensor, logic or final element subsystem comprises of only one group of voted channels, then PFD_G is equivalent to PFD_S, PFD_L or PFD_{FE} respectively (see B.3.2.1).

NOTE 2 The table assumes $\beta = 2 \times \beta_D$. For 1oo1 and 2oo2 architectures, the values of β and β_D do not affect the average probability of failure.

NOTE 3 The safe failure rate is assumed to be equal to the dangerous failure rate and K = 0,98.

Table B.5 – Average probability of failure on demand for a proof test interval of ten years and a mean time to restoration of 8 h

Architecture	*DC*	λ_D = 0,5E-07			λ_D = 2,5E-07			λ_D = 0,5E-06		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see Note 2)	**0 %**	2,2E-03			1,1E-02			2,2E-02		
	60 %	8,8E-04			4,4E-03			8,8E-03		
	90 %	2,2E-04			1,1E-03			2,2E-03		
	99 %	2,2E-05			1,1E-04			2,2E-04		
1oo2	**0 %**	5,0E-05	2,2E-04	4,4E-04	3,7E-04	1,2E-03	2,3E-03	1,1E-03	2,7E-03	4,8E-03
	60 %	1,9E-05	8,9E-05	1,8E-04	1,1E-04	4,6E-04	9,0E-04	2,7E-04	9,6E-04	1,8E-03
	90 %	4,4E-06	2,2E-05	4,4E-05	2,3E-05	1,1E-04	2,2E-04	5,0E-05	2,2E-04	4,4E-04
	99 %	4,4E-07	2,2E-06	4,4E-06	2,2E-06	1,1E-05	2,2E-05	4,5E-06	2,2E-05	4,4E-05
2oo2 (see Note 2)	**0 %**	4,4E-03			2,2E-02			4,4E-02		
	60 %	1,8E-03			8,8E-03			1,8E-02		
	90 %	4,4E-04			2,2E-03			4,4E-03		
	99 %	4,5E-05			2,2E-04			4,5E-04		
1oo2D (see Note 3)	**0 %**	5,0E-05	2,2E-04	4,4E-04	3,7E-04	1,2E-03	2,3E-03	1,1E-03	2,7E-03	4,8E-03
	60 %	2,9E-05	9,9E-05	1,9E-04	1,7E-04	5,1E-04	9,5E-04	3,8E-04	1,1E-03	1,9E-03
	90 %	8,4E-06	2,6E-05	4,8E-05	4,3E-05	1,3E-04	2,4E-04	9,0E-05	2,6E-04	4,8E-04
	99 %	8,9E-07	2,6E-06	4,8E-06	4,5E-06	1,3E-05	2,4E-05	8,9E-06	2,6E-05	4,8E-05
2oo3	**0 %**	6,2E-05	2,3E-04	4,5E-04	6,8E-04	1,5E-03	2,5E-03	2,3E-03	3,7E-03	5,6E-03
	60 %	2,1E-05	9,0E-05	1,8E-04	1,6E-04	5,0E-04	9,3E-04	4,7E-04	1,1E-03	2,0E-03
	90 %	4,6E-06	2,2E-05	4,4E-05	2,7E-05	1,1E-04	2,2E-04	6,3E-05	2,4E-04	4,5E-04
	99 %	4,4E-07	2,2E-06	4,4E-06	2,3E-06	1,1E-05	2,2E-05	4,6E-06	2,2E-05	4,4E-05
1oo3	**0 %**	4,4E-05	2,2E-04	4,4E-04	2,2E-04	1,1E-03	2,2E-03	4,6E-04	2,2E-03	4,4E-03
	60 %	1,8E-05	8,8E-05	1,8E-04	8,8E-05	4,4E-04	8,8E-04	1,8E-04	8,8E-04	1,8E-03
	90 %	4,4E-06	2,2E-05	4,4E-05	2,2E-05	1,1E-04	2,2E-04	4,4E-05	2,2E-04	4,4E-04
	99 %	4,4E-07	2,2E-06	4,4E-06	2,2E-06	1,1E-05	2,2E-05	4,4E-06	2,2E-05	4,4E-05

Architecture	*DC*	λ_D =2,5E-06			λ_D = 0,5E-05			λ_D = 2,5E-05		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see Note 2)	**0 %**	>1E-01			>1E-01			>1E-01		
	60 %	4,4E-02			8,8E-02			>1E-01		
	90 %	1,1E-02			2,2E-02			>1E-01		
	99 %	1,1E-03			2,2E-03			1,1E-02		
1oo2	**0 %**	1,8E-02	2,4E-02	3,2E-02	6,6E-02	7,4E-02	8,5E-02	>1E-01	>1E-01	>1E-01
	60 %	3,4E-03	6,6E-03	1,1E-02	1,2E-02	1,8E-02	2,5E-02	>1E-01	>1E-01	>1E-01
	90 %	3,8E-04	1,2E-03	2,3E-03	1,1E-03	2,8E-03	4,9E-03	1,8E-02	2,5E-02	3,5E-02
	99 %	2,4E-05	1,1E-04	2,2E-04	5,1E-05	2,3E-04	4,5E-04	3,8E-04	1,3E-03	2,3E-03
2oo2 (see Note 2)	**0 %**	>1E-01			>1E-01			>1E-01		
	60 %	8,8E-02			>1E-01			>1E-01		
	90 %	2,2E-02			4,4E-02			>1E-01		
	99 %	2,2E-03			4,5E-03			2,2E-02		
1oo2D (see Note 3)	**0 %**	1,8E-02	2,5E-02	3,3E-02	6,6E-02	7,7E-02	9,0E-02	1,6E+00	1,5E+00	1,4E+00
	60 %	3,9E-03	7,1E-03	1,1E-02	1,3E-02	1,9E-02	2,6E-02	2,6E-01	2,7E-01	2,8E-01
	90 %	5,7E-04	1,4E-03	2,5E-03	1,5E-03	3,1E-03	5,2E-03	2,0E-02	2,7E-02	3,5E-02
	99 %	4,6E-05	1,3E-04	2,4E-04	9,5E-05	2,7E-04	4,9E-04	6,0E-04	1,5E-03	2,5E-03
2oo3	**0 %**	4,8E-02	5,0E-02	5,3E-02	1,9E-01	1,8E-01	1,7E-01	4,6E+00	4,0E+00	3,3E+00
	60 %	8,3E-03	1,1E-02	1,4E-02	3,2E-02	3,5E-02	4,0E-02	7,6E-01	7,1E-01	6,6E-01
	90 %	6,9E-04	1,5E-03	2,6E-03	2,3E-03	3,9E-03	5,9E-03	4,9E-02	5,4E-02	6,0E-02
	99 %	2,7E-05	1,2E-04	2,3E-04	6,4E-05	2,4E-04	4,6E-04	7,1E-04	1,6E-03	2,6E-03
1oo3	**0 %**	4,7E-03	1,3E-02	2,3E-02	2,4E-02	3,7E-02	5,5E-02	2,5E+00	2,0E+00	1,6E+00
	60 %	1,0E-03	4,5E-03	8,9E-03	3,0E-03	9,8E-03	1,8E-02	1,7E-01	1,8E-01	1,9E-01
	90 %	2,2E-04	1,1E-03	2,2E-03	4,6E-04	2,2E-03	4,4E-03	4,8E-03	1,3E-02	2,4E-02
	99 %	2,2E-05	1,1E-04	2,2E-04	4,4E-05	2,2E-04	4,4E-04	2,2E-04	1,1E-03	2,2E-03

NOTE 1 This table gives example values of PFD_G, calculated using the equations in B.3.2 and depending on the assumptions listed in B.3.1. If the sensor, logic or final element subsystem comprises of only one group of voted channels, then PFD_G is equivalent to PFD_S, PFD_L or PFD_{FE} respectively (see B.3.2.1).

NOTE 2 The table assumes $\beta = 2 \times \beta_D$. For 1oo1 and 2oo2 architectures, the values of β and β_D do not affect the average probability of failure.

NOTE 3 The safe failure rate is assumed to be equal to the dangerous failure rate and K = 0,98.

B.3.2.4 Example for low demand mode of operation

Consider a safety function requiring a SIL 2 system. Suppose that the initial assessment for the system architecture, based on previous practice, is for one group of three analogue pressure sensors, voting 2oo3. The logic subsystem is a redundant 1oo2D configured PE system driving a single shut-down valve plus a single vent valve. Both the shut-down and vent valves need to operate in order to achieve the safety function. The architecture is shown in Figure B.14. For the initial assessment, a proof test period of one year is assumed.

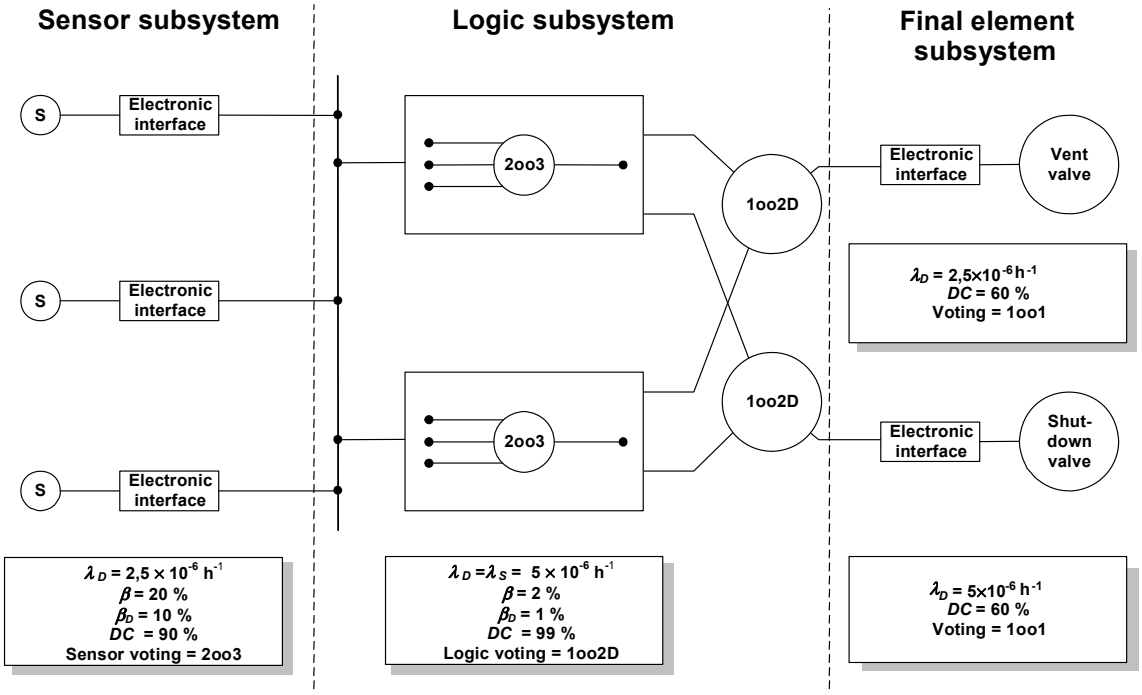


Figure B.14 – Architecture of an example for low demand mode of operation

Table B.6 – Average probability of failure on demand for the sensor subsystem in the example for low demand mode of operation (one year proof test interval and 8 h *MTTR*)

Architecture	*DC*	λ_D = 2,5E-06		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
2oo3	0 %	6,8E-04	1,5E-03	2,5E-03
	60 %	1,6E-04	5,1E-04	9,4E-04
	90 %	2,7E-05	1,2E-04	2,3E-04
	99 %	2,5E-06	1,2E-05	2,4E-05
NOTE This table is abstracted from Table B.3.				

Table B.7 – Average probability of failure on demand for the logic subsystem in the example for low demand mode of operation (one year proof test interval and 8 h *MTTR*)

Architecture	*DC*	λ_D = 0,5E-05		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo2D	0 %	1,1E-03	2,7E-03	4,8E-03
	60 %	2,0E-04	9,0E-04	1,8E-03
	90 %	4,5E-05	2,2E-04	4,4E-04
	99 %	4,8E-06	2,4E-05	4,8E-05
NOTE This table is abstracted from Table B.3.				

Table B.8 – Average probability of failure on demand for the final element subsystem in the example for low demand mode of operation (one year proof test interval and 8 h *MTTR*)

Architecture	*DC*	λ_D = 2,5E-06	λ_D = 0,5E-05
1oo1	0 %	1,1E-02	2,2E-02
	60 %	4,4E-03	8,8E-03
	90 %	1,1E-03	2,2E-03
	99 %	1,3E-04	2,6E-04
NOTE This table is abstracted from Table B.3.			

From Tables B.6 to B.8 the following values are derived.

For the sensor subsystem,

$PFD_S = 2{,}3 \times 10^{-4}$

For the logic subsystem,

$PFD_L = 4{,}8 \times 10^{-6}$

For the final element subsystem,

$PFD_{FE} = 4{,}4 \times 10^{-3} + 8{,}8 \times 10^{-3}$

$= 1{,}3 \times 10^{-2}$

Therefore, for the safety function,

$PFD_{SYS} = 2{,}3 \times 10^{-4} + 4{,}8 \times 10^{-6} + 1{,}3 \times 10^{-2}$

$= 1{,}3 \times 10^{-2}$

$\equiv$ **safety integrity level 1**

To improve the system to meet safety integrity level 2, one of the following could be done:

a) change the proof test interval to six months

$PFD_S = 1{,}1 \times 10^{-4}$

$PFD_L = 2{,}6 \times 10^{-6}$

$PFD_{FE} = 2{,}2 \times 10^{-3} + 4{,}4 \times 10^{-3}$

$= 6{,}6 \times 10^{-3}$

$PFD_{SYS} = 6{,}7 \times 10^{-3}$

≡ **safety integrity level 2**

b) change the 1oo1 shutdown valve (which is the output device with the lower reliability) to 1oo2 (assuming β = 10 % and β_D = 5 %)

PFD_S = $2{,}3 \times 10^{-4}$

PFD_L = $4{,}8 \times 10^{-6}$

PFD_{FE} = $4{,}4 \times 10^{-3} + 9{,}7 \times 10^{-4}$

= $5{,}4 \times 10^{-3}$

PFD_{SYS} = $5{,}6 \times 10^{-3}$

≡ **safety integrity level 2**

B.3.2.5 Effects of a non-perfect proof test

Faults in the safety system that are not detected by either diagnostic tests or proof tests may be found by other methods arising from events such as a hazardous event requiring operation of the safety function or during an overhaul of the equipment. If the faults are not detected by such methods it should be assumed that the faults will remain for the life of the equipment. Consider a normal proof test period of T_1 where the fraction of faults detected when a proof test is performed is designated as PTC (proof test coverage) and the fraction of the faults not detected when a proof test is performed is designated as (1-PCT). These latter faults which are not detected at the proof test will only be revealed when a demand is made on the safety-related system at demand period T_2. Therefore, the proof test period (T_1) and the demand period (T_2) govern the effective down time..

An example of this is given below for a 1oo2 architecture. T_2 is the time between demands on the system:

$$t_{CE} = \frac{\lambda_{DU}(PTC)}{\lambda_D}\left(\frac{T_1}{2} + MRT\right) + \frac{\lambda_{DU}(1-PTC)}{\lambda_D}\left(\frac{T_2}{2} + MRT\right) + \frac{\lambda_{DD}}{\lambda_D}MTTR$$

$$t_{GE} = \frac{\lambda_{DU}(PTC)}{\lambda_D}\left(\frac{T_1}{3} + MRT\right) + \frac{\lambda_{DU}(1-PTC)}{\lambda_D}\left(\frac{T_2}{3} + MRT\right) + \frac{\lambda_{DD}}{\lambda_D}MTTR$$

$$PFD_G = 2\left((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU}\right)^2 t_{CE} t_{GE} + \beta_D \lambda_{DD} MTTR + \beta\lambda_{DU}(PTC)\left(\frac{T_1}{2} + MRT\right) +$$

$$\beta\lambda_{DU}(1-PTC)\left(\frac{T_2}{2} + MRT\right)$$

Table B.9 below gives the numeric results of a 1oo2 system with a 100 % one-year proof test (T_1 = 1 year) compared against a 90 % proof test where the demand period T_2 is assumed to be 10 years. This example has been calculated assuming a failure rate of $0{,}5 \times 10^{-5}$ per hour, a β value of 10 % and a β_D value of 5 %.

Table B.9 – Example for a non-perfect proof test

Architecture	*DC*	λ_D = 0,5E-05	
		100 % proof test	90 % proof test
		β = 10 % β_D = 5 %	β = 10 % β_D = 5 %
1oo2	0 %	2,7E-03	6,0E-03
	60 %	9,7E-04	2,0E-03
	90 %	2,3E-04	4,4E-04
	99 %	2,4E-05	4,4E-05

B.3.3 Average frequency of dangerous failure (for high demand or continuous mode of operation)

B.3.3.1 Procedure for calculations

The method for calculating the probability of failure of a safety function for an E/E/PE safety-related system operating in high demand or continuous mode of operation is identical with that for calculating for a low demand mode of operation (see B.2.1), except that average probability of failure on demand (PFD_{SYS}) is replaced with average frequency of dangerous failure (PFH_{SYS}).

The overall probability of a dangerous failure of a safety function for the E/E/PE safety-related system, PFH_{SYS}, is determined by calculating the dangerous failure rates for all the sub-systems which together provide the safety function and adding together these individual values. Since in this annex the probabilities are small, this can be expressed by the following:

$$PFH_{SYS} = PFH_S + PFH_L + PFH_{FE}$$

where

PFH_{SYS} is the average frequency of dangerous failure of a safety function for the E/E/PE safety-related system;

PFH_S is the average frequency of dangerous failure for the sensor subsystem;

PFH_L is the average frequency of dangerous failure for the logic subsystem; and

PFH_{FE} is the average frequency of dangerous failure for the final element subsystem.

B.3.3.2 Architectures for high demand or continuous mode of operation

NOTE 1 This subclause should be read sequentially, since equations which are valid for several architectures are only stated where they are first used. See also B.3.2.2.

NOTE 2 The calculations are based on the assumptions listed in B.3.1.

B.3.3.2.1 1oo1

Figures B.4 and B.5 show the relevant block diagrams.

$$\lambda_D = \lambda_{DU} + \lambda_{DD}$$

$$t_{CE} = \frac{\lambda_{DU}}{\lambda_D}\left(\frac{T_1}{2} + MRT\right) + \frac{\lambda_{DD}}{\lambda_D} MTTR$$

$$\lambda_{DU} = \lambda_D(1 - DC); \qquad \lambda_{DD} = \lambda_D DC$$

If it is assumed that the safety system puts the EUC into a safe state on detection of any failure, for a 1oo1 architecture the following is obtained

$$PFH_G = \lambda_{DU}$$

B.3.3.2.2 1oo2

Figures B.6 and B.7 show the relevant block diagrams. The value of t_{CE} is as given in B.3.3.2.1. If it is assumed that the safety system puts the EUC into a safe state once there is detection of a failure in both channels and taking a conservative approach, the following is obtained

$$PFH_G = 2((1 - \beta_D)\lambda_{DD} + (1 - \beta)\lambda_{DU})(1 - \beta)\lambda_{DU} t_{CE} + \beta\lambda_{DU}$$

B.3.3.2.3 2oo2

Figures B.8 and B.9 show the relevant block diagrams. If it is assumed that each channel is put into a safe state on detection of any fault, for a 2oo2 architecture, the following is obtained

$$PFH_G = 2\lambda_{DU}$$

B.3.3.2.4 1oo2D

Figures B.10 and B.11 show the relevant block diagrams.

$$\lambda_{SD} = \frac{\lambda}{2} DC$$

$$t_{CE}' = \frac{\lambda_{DU}\left(\frac{T_1}{2} + MRT\right) + (\lambda_{DD} + \lambda_{SD})MTTR}{\lambda_{DU} + \lambda_{DD} + \lambda_{SD}}$$

$$PFH_G = 2(1-\beta)\lambda_{DU}((1-\beta)\lambda_{DU} + (1-\beta_D)\lambda_{DD} + \lambda_{SD})t_{CE}' + 2(1-K)\lambda_{DD} + \beta\lambda_{DU}$$

B.3.3.2.5 2oo3

Figures B.12 and B.13 show the relevant block diagrams. The value of t_{CE} is as given in B.3.3.2.1. If it is assumed that the safety system puts the EUC into a safe state once there is detection of a failure in any two channels and taking a conservative approach, the following is obtained

$$PFH_G = 6((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU})(1-\beta)\lambda_{DU}t_{CE} + \beta\lambda_{DU}$$

B.3.3.2.6 1oo3

Figures B.12 and B.13 show the relevant block diagrams. The value of t_{CE} and t_{GE} is as given in B.3.3.2.1. and B.3.2.2.2. If it is assumed that the safety system puts the EUC into a safe state once there is detection of a failure in the tree channels and taking a conservative approach, the following is obtained

$$PFH_G = 6((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU})^2(1-\beta)\lambda_{DU}t_{CE}t_{GE} + \beta\lambda_{DU}$$

B.3.3.3 Detailed tables for high demand or continuous mode of operation

Table B.10 – Average frequency of a dangerous failure (in high demand or continuous mode of operation) for a proof test interval of one month and a mean time to restoration of 8 h

Architecture	DC	λ_D = 0,5E-07			λ_D = 2,5E-07			λ_D = 0,5E-06		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see note 2)	**0 %**	5.0E-08			2.5E-07			5.0E-07		
	60 %	2.0E-08			1.0E-07			2.0E-07		
	90 %	5.0E-09			2.5E-08			5.0E-08		
	99 %	5.0E-10			2.5E-09			5.0E-09		
1oo2	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.0E-09	1.0E-08	2.0E-08	4.0E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09
2oo2 (see note 2)	**0 %**	1.0E-07			5.0E-07			1.0E-06		
	60 %	4.0E-08			2.0E-07			4.0E-07		
	90 %	1.0E-08			5.0E-08			1.0E-07		
	99 %	1.0E-09			5.0E-09			1.0E-08		
1oo2D (see note 3)	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07
	60 %	1.6E-09	3.2E-09	5.2E-09	8.0E-09	1.6E-08	2.6E-08	1.6E-08	3.2E-08	5.2E-08
	90 %	1.9E-09	2.3E-09	2.8E-09	9.5E-09	1.2E-08	1.4E-08	1.9E-08	2.3E-08	2.8E-08
	99 %	2.0E-09	2.0E-09	2.1E-09	1.0E-08	1.0E-08	1.0E-08	2.0E-08	2.0E-08	2.1E-08
2oo3	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.1E-09	2.5E-08	5.0E-08	1.1E-08	5.0E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.0E-09	1.0E-08	2.0E-08	4.1E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09
1oo3	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.0E-09	1.0E-08	2.0E-08	4.0E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09

Architecture	DC	λ_D =2,5E-06			λ_D = 0,5E-05			λ_D = 2,5E-05		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see note 2)	**0 %**	2.5E-06			5.0E-06			2.5E-05		
	60 %	1.0E-06			2.0E-06			1.0E-05		
	90 %	2.5E-07			5.0E-07			2.5E-06		
	99 %	2.5E-08			5.0E-08			2.5E-07		
1oo2	**0 %**	5.4E-08	2.5E-07	5.0E-07	1.2E-07	5.2E-07	1.0E-06	9.5E-07	2.9E-06	5.3E-06
	60 %	2.1E-08	1.0E-07	2.0E-07	4.3E-08	2.0E-07	4.0E-07	2.7E-07	1.1E-06	2.1E-06
	90 %	5.1E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07	5.5E-08	2.5E-07	5.0E-07
	99 %	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.1E-09	2.5E-08	5.0E-08
2oo2 (see note 2)	**0 %**	5.0E-06			1.0E-05			5.0E-05		
	60 %	2.0E-06			4.0E-06			2.0E-05		
	90 %	5.0E-07			1.0E-06			5.0E-06		
	99 %	5.0E-08			1.0E-07			5.0E-07		
1oo2D (see note 3)	**0 %**	5.4E-08	2.5E-07	5.0E-07	1.2E-07	5.2E-07	1.0E-06	9.5E-07	2.9E-06	5.3E-06
	60 %	8.1E-08	1.6E-07	2.6E-07	1.6E-07	3.2E-07	5.2E-07	8.7E-07	1.7E-06	2.7E-06
	90 %	9.5E-08	1.2E-07	1.4E-07	1.9E-07	2.3E-07	2.8E-07	9.6E-07	1.2E-06	1.4E-06
	99 %	1.0E-07	1.0E-07	1.0E-07	2.0E-07	2.0E-07	2.1E-07	1.0E-06	1.0E-06	1.0E-06
2oo3	**0 %**	6.3E-08	2.6E-07	5.1E-07	1.5E-07	5.5E-07	1.0E-06	1.8E-06	3.6E-06	5.9E-06
	60 %	2.2E-08	1.0E-07	2.0E-07	4.9E-08	2.1E-07	4.1E-07	4.2E-07	1.2E-06	2.2E-06
	90 %	5.2E-09	2.5E-08	5.0E-08	1.1E-08	5.1E-08	1.0E-07	6.6E-08	2.6E-07	5.1E-07
	99 %	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.4E-09	2.5E-08	5.0E-08
1oo3	**0 %**	5.0E-08	2.5E-07	5.0E-07	1.0E-07	5.0E-07	1.0E-06	5.1E-07	2.5E-06	5.0E-06
	60 %	2.0E-08	1.0E-07	2.0E-07	4.0E-08	2.0E-07	4.0E-07	2.0E-07	1.0E-06	2.0E-06
	90 %	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07	5.0E-08	2.5E-07	5.0E-07
	99 %	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08

NOTE 1 This table gives example values of PFH_G, calculated using the equations in B.3.3 and depending on the assumptions listed in B.3.1. If the sensor, logic or final element subsystem comprises of only one group of voted channels, then PFH_G is equivalent to PFH_S, PFH_L or PFH_{FE} respectively (see B.3.3.1).

NOTE 2 The table assumes $\beta = 2 \times \beta_D$. For 1oo1 and 2oo2 architectures, the values of β and β_D do not affect the average frequency of a dangerous failure.

NOTE 3 The safe failure rate is assumed to be equal to the dangerous failure rate and K = 0,98.

Table B.11 – Average frequency of a dangerous failure (in high demand or continuous mode of operation) for a proof test interval of three month and a mean time to restoration of 8 h

Architecture	DC	λ_D = 0,5E-07			λ_D = 2,5E-07			λ_D = 0,5E-06		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see note 2)	**0 %**	5.0E-08			2.5E-07			5.0E-07		
	60 %	2.0E-08			1.0E-07			2.0E-07		
	90 %	5.0E-09			2.5E-08			5.0E-08		
	99 %	5.0E-10			2.5E-09			5.0E-09		
1oo2	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.1E-09	2.5E-08	5.0E-08	1.1E-08	5.0E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.0E-09	1.0E-08	2.0E-08	4.1E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09
2oo2 (see note 2)	**0 %**	1.0E-07			5.0E-07			1.0E-06		
	60 %	4.0E-08			2.0E-07			4.0E-07		
	90 %	1.0E-08			5.0E-08			1.0E-07		
	99 %	1.0E-09			5.0E-09			1.0E-08		
1oo2D (see note 3)	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.1E-09	2.5E-08	5.0E-08	1.1E-08	5.0E-08	1.0E-07
	60 %	1.6E-09	3.2E-09	5.2E-09	8.0E-09	1.6E-08	2.6E-08	1.6E-08	3.2E-08	5.2E-08
	90 %	1.9E-09	2.3E-09	2.8E-09	9.5E-09	1.2E-08	1.4E-08	1.9E-08	2.3E-08	2.8E-08
	99 %	2.0E-09	2.0E-09	2.1E-09	1.0E-08	1.0E-08	1.0E-08	2.0E-08	2.0E-08	2.1E-08
2oo3	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.4E-09	2.5E-08	5.0E-08	1.2E-08	5.1E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.1E-09	1.0E-08	2.0E-08	4.3E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09
1oo3	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.0E-09	1.0E-08	2.0E-08	4.0E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09

Architecture	DC	λ_D = 2,5E-06			λ_D = 0,5E-05			λ_D = 2,5E-05		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see note 2)	**0 %**	2.5E-06			5.0E-06			2.5E-05		
	60 %	1.0E-06			2.0E-06			1.0E-05		
	90 %	2.5E-07			5.0E-07			2.5E-06		
	99 %	2.5E-08			5.0E-08			2.5E-07		
1oo2	**0 %**	6.3E-08	2.6E-07	5.1E-07	1.5E-07	5.4E-07	1.0E-06	1.8E-06	3.6E-06	5.9E-06
	60 %	2.2E-08	1.0E-07	2.0E-07	4.9E-08	2.1E-07	4.1E-07	4.2E-07	1.2E-06	2.2E-06
	90 %	5.1E-09	2.5E-08	5.0E-08	1.1E-08	5.0E-08	1.0E-07	6.4E-08	2.6E-07	5.1E-07
	99 %	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.2E-09	2.5E-08	5.0E-08
2oo2 (see note 2)	**0 %**	5.0E-06			1.0E-05			5.0E-05		
	60 %	2.0E-06			4.0E-06			2.0E-05		
	90 %	5.0E-07			1.0E-06			5.0E-06		
	99 %	5.0E-08			1.0E-07			5.0E-07		
1oo2D (see note 3)	**0 %**	6.3E-08	2.6E-07	5.1E-07	1.5E-07	5.4E-07	1.0E-06	1.8E-06	3.6E-06	5.9E-06
	60 %	8.2E-08	1.6E-07	2.6E-07	1.7E-07	3.3E-07	5.3E-07	1.0E-06	1.8E-06	2.8E-06
	90 %	9.5E-08	1.2E-07	1.4E-07	1.9E-07	2.3E-07	2.8E-07	9.6E-07	1.2E-06	1.4E-06
	99 %	1.0E-07	1.0E-07	1.0E-07	2.0E-07	2.0E-07	2.1E-07	1.0E-06	1.0E-06	1.0E-06
2oo3	**0 %**	9.0E-08	2.8E-07	5.3E-07	2.6E-07	6.3E-07	1.1E-06	4.5E-06	5.9E-06	7.6E-06
	60 %	2.6E-08	1.1E-07	2.0E-07	6.6E-08	2.2E-07	4.2E-07	8.5E-07	1.6E-06	2.5E-06
	90 %	5.4E-09	2.5E-08	5.0E-08	1.2E-08	5.1E-08	1.0E-07	9.3E-08	2.9E-07	5.3E-07
	99 %	5.1E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.7E-09	2.6E-08	5.1E-08
1oo3	**0 %**	5.0E-08	2.5E-07	5.0E-07	1.0E-07	5.0E-07	1.0E-06	5.5E-07	2.5E-06	5.0E-06
	60 %	2.0E-08	1.0E-07	2.0E-07	4.0E-08	2.0E-07	4.0E-07	2.0E-07	1.0E-06	2.0E-06
	90 %	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07	5.0E-08	2.5E-07	5.0E-07
	99 %	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08

NOTE 1 This table gives example values of PFH_G, calculated using the equations in B.3.3 and depending on the assumptions listed in B.3.1. If the sensor, logic or final element subsystem comprises of only one group of voted channels, then PFH_G is equivalent to PFH_S, PFH_L or PFH_{FE} respectively (see B.3.3.1).

NOTE 2 The table assumes $\beta = 2 \times \beta_D$. For 1oo1 and 2oo2 architectures, the values of β and β_D do not affect the average frequency of a dangerous failure.

NOTE 3 The safe failure rate is assumed to be equal to the dangerous failure rate and K = 0,98.

Table B.12 – Average frequency of a dangerous failure (in high demand or continuous mode of operation) for a proof test interval of six month and a mean time to restoration of 8 h

Architecture	DC	λ_D = 0,5E-07			λ_D = 2,5E-07			λ_D = 0,5E-06		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see note 2)	**0 %**	5.0E-08			2.5E-07			5.0E-07		
	60 %	2.0E-08			1.0E-07			2.0E-07		
	90 %	5.0E-09			2.5E-08			5.0E-08		
	99 %	5.0E-10			2.5E-09			5.0E-09		
1oo2	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.3E-09	2.5E-08	5.0E-08	1.1E-08	5.1E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.0E-09	1.0E-08	2.0E-08	4.2E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09
2oo2 (see note 2)	**0 %**	1.0E-07			5.0E-07			1.0E-06		
	60 %	4.0E-08			2.0E-07			4.0E-07		
	90 %	1.0E-08			5.0E-08			1.0E-07		
	99 %	1.0E-09			5.0E-09			1.0E-08		
1oo2D (see note 3)	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.3E-09	2.5E-08	5.0E-08	1.1E-08	5.1E-08	1.0E-07
	60 %	1.6E-09	3.2E-09	5.2E-09	8.0E-09	1.6E-08	2.6E-08	1.6E-08	3.2E-08	5.2E-08
	90 %	1.9E-09	2.3E-09	2.8E-09	9.5E-09	1.2E-08	1.4E-08	1.9E-08	2.3E-08	2.8E-08
	99 %	2.0E-09	2.0E-09	2.1E-09	1.0E-08	1.0E-08	1.0E-08	2.0E-08	2.0E-08	2.1E-08
2oo3	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.8E-09	2.6E-08	5.1E-08	1.3E-08	5.3E-08	1.0E-07
	60 %	4.1E-10	2.0E-09	4.0E-09	2.1E-09	1.0E-08	2.0E-08	4.5E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.1E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09
1oo3	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.0E-09	1.0E-08	2.0E-08	4.0E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09

Architecture	DC	λ_D = 2,5E-06			λ_D = 0,5E-05			λ_D = 2,5E-05		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see note 2)	**0 %**	2.5E-06			5.0E-06			2.5E-05		
	60 %	1.0E-06			2.0E-06			1.0E-05		
	90 %	2.5E-07			5.0E-07			2.5E-06		
	99 %	2.5E-08			5.0E-08			2.5E-07		
1oo2	**0 %**	7.6E-08	2.7E-07	5.2E-07	2.1E-07	5.9E-07	1.1E-06	3.1E-06	4.7E-06	6.8E-06
	60 %	2.4E-08	1.0E-07	2.0E-07	5.7E-08	2.1E-07	4.1E-07	6.3E-07	1.4E-06	2.3E-06
	90 %	5.3E-09	2.5E-08	5.0E-08	1.1E-08	5.1E-08	1.0E-07	7.8E-08	2.7E-07	5.2E-07
	99 %	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.4E-09	2.5E-08	5.0E-08
2oo2 (see note 2)	**0 %**	5.0E-06			1.0E-05			5.0E-05		
	60 %	2.0E-06			4.0E-06			2.0E-05		
	90 %	5.0E-07			1.0E-06			5.0E-06		
	99 %	5.0E-08			1.0E-07			5.0E-07		
1oo2D (see note 3)	**0 %**	7.6E-08	2.7E-07	5.2E-07	2.1E-07	5.9E-07	1.1E-06	3.1E-06	4.7E-06	6.8E-06
	60 %	8.4E-08	1.6E-07	2.6E-07	1.8E-07	3.3E-07	5.3E-07	1.2E-06	2.0E-06	2.9E-06
	90 %	9.5E-08	1.2E-07	1.4E-07	1.9E-07	2.3E-07	2.8E-07	9.8E-07	1.2E-06	1.4E-06
	99 %	1.0E-07	1.0E-07	1.0E-07	2.0E-07	2.0E-07	2.1E-07	1.0E-06	1.0E-06	1.0E-06
2oo3	**0 %**	1.3E-07	3.2E-07	5.5E-07	4.2E-07	7.7E-07	1.2E-06	8.4E-06	9.2E-06	1.0E-05
	60 %	3.3E-08	1.1E-07	2.1E-07	9.1E-08	2.4E-07	4.4E-07	1.5E-06	2.1E-06	2.9E-06
	90 %	5.8E-09	2.6E-08	5.1E-08	1.3E-08	5.3E-08	1.0E-07	1.3E-07	3.2E-07	5.6E-07
	99 %	5.1E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	6.1E-09	2.6E-08	5.1E-08
1oo3	**0 %**	5.0E-08	2.5E-07	5.0E-07	1.0E-07	5.0E-07	1.0E-06	7.1E-07	2.7E-06	5.1E-06
	60 %	2.0E-08	1.0E-07	2.0E-07	4.0E-08	2.0E-07	4.0E-07	2.1E-07	1.0E-06	2.0E-06
	90 %	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07	5.0E-08	2.5E-07	5.0E-07
	99 %	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08

NOTE 1 This table gives example values of PFH_G, calculated using the equations in B.3.3 and depending on the assumptions listed in B.3.1. If the sensor, logic or final element subsystem comprises of only one group of voted channels, then PFH_G is equivalent to PFH_S, PFH_L or PFH_{FE} respectively (see B.3.3.1).

NOTE 2 The table assumes $\beta = 2 \times \beta_D$. For 1oo1 and 2oo2 architectures, the values of β and β_D do not affect the average frequency of a dangerous failure.

NOTE 3 The safe failure rate is assumed to be equal to the dangerous failure rate and K = 0,98.

Table B.13 – Average frequency of a dangerous failure (in high demand or continuous mode of operation) for a proof test interval of one year and a mean time to restoration of 8 h

Architecture	DC	λ_D = 0,5E-07			λ_D = 2,5E-07			λ_D = 0,5E-06		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see note 2)	**0 %**	5.0E-08			2.5E-07			5.0E-07		
	60 %	2.0E-08			1.0E-07			2.0E-07		
	90 %	5.0E-09			2.5E-08			5.0E-08		
	99 %	5.0E-10			2.5E-09			5.0E-09		
1oo2	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.5E-09	2.5E-08	5.0E-08	1.2E-08	5.2E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.1E-09	1.0E-08	2.0E-08	4.3E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.1E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09
2oo2 (see note 2)	**0 %**	1.0E-07			5.0E-07			1.0E-06		
	60 %	4.0E-08			2.0E-07			4.0E-07		
	90 %	1.0E-08			5.0E-08			1.0E-07		
	99 %	1.0E-09			5.0E-09			1.0E-08		
1oo2D (see note 3)	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.5E-09	2.5E-08	5.0E-08	1.2E-08	5.2E-08	1.0E-07
	60 %	1.6E-09	3.2E-09	5.2E-09	8.1E-09	1.6E-08	2.6E-08	1.6E-08	3.2E-08	5.2E-08
	90 %	1.9E-09	2.3E-09	2.8E-09	9.5E-09	1.2E-08	1.4E-08	1.9E-08	2.3E-08	2.8E-08
	99 %	2.0E-09	2.0E-09	2.1E-09	1.0E-08	1.0E-08	1.0E-08	2.0E-08	2.0E-08	2.1E-08
2oo3	**0 %**	1.1E-09	5.1E-09	1.0E-08	6.6E-09	2.6E-08	5.1E-08	1.6E-08	5.5E-08	1.0E-07
	60 %	4.1E-10	2.0E-09	4.0E-09	2.3E-09	1.0E-08	2.0E-08	5.0E-09	2.1E-08	4.1E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.2E-10	2.5E-09	5.0E-09	1.1E-09	5.1E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09
1oo3	**0 %**	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07
	60 %	4.0E-10	2.0E-09	4.0E-09	2.0E-09	1.0E-08	2.0E-08	4.0E-09	2.0E-08	4.0E-08
	90 %	1.0E-10	5.0E-10	1.0E-09	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08
	99 %	1.0E-11	5.0E-11	1.0E-10	5.0E-11	2.5E-10	5.0E-10	1.0E-10	5.0E-10	1.0E-09

Architecture	DC	λ_D = 2,5E-06			λ_D = 0,5E-05			λ_D = 2,5E-05		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %	β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo1 (see note 2)	**0 %**	2.5E-06			5.0E-06			2.5E-05		
	60 %	1.0E-06			2.0E-06			1.0E-05		
	90 %	2.5E-07			5.0E-07			2.5E-06		
	99 %	2.5E-08			5.0E-08			2.5E-07		
1oo2	**0 %**	1.0E-07	2.9E-07	5.4E-07	3.1E-07	6.8E-07	1.1E-06	5.8E-06	6.9E-06	8.5E-06
	60 %	2.9E-08	1.1E-07	2.1E-07	7.4E-08	2.3E-07	4.2E-07	1.1E-06	1.7E-06	2.6E-06
	90 %	5.5E-09	2.5E-08	5.0E-08	1.2E-08	5.2E-08	1.0E-07	1.0E-07	3.0E-07	5.4E-07
	99 %	5.1E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.6E-09	2.6E-08	5.0E-08
2oo2 (see note 2)	**0 %**	5.0E-06			1.0E-05			5.0E-05		
	60 %	2.0E-06			4.0E-06			2.0E-05		
	90 %	5.0E-07			1.0E-06			5.0E-06		
	99 %	5.0E-08			1.0E-07			5.0E-07		
1oo2D (see note 3)	**0 %**	1.0E-07	2.9E-07	5.4E-07	3.1E-07	6.8E-07	1.1E-06	5.8E-06	6.9E-06	8.5E-06
	60 %	8.9E-08	1.7E-07	2.7E-07	1.9E-07	3.5E-07	5.4E-07	1.7E-06	2.3E-06	3.2E-06
	90 %	9.6E-08	1.2E-07	1.4E-07	1.9E-07	2.3E-07	2.8E-07	1.0E-06	1.2E-06	1.4E-06
	99 %	1.0E-07	1.0E-07	1.0E-07	2.0E-07	2.0E-07	2.1E-07	1.0E-06	1.0E-06	1.0E-06
2oo3	**0 %**	2.1E-07	3.8E-07	6.1E-07	7.3E-07	1.0E-06	1.4E-06	1.6E-05	1.6E-05	1.6E-05
	60 %	4.6E-08	1.2E-07	2.2E-07	1.4E-07	2.9E-07	4.7E-07	2.8E-06	3.2E-06	3.8E-06
	90 %	6.6E-09	2.6E-08	5.1E-08	1.6E-08	5.6E-08	1.0E-07	2.1E-07	3.9E-07	6.2E-07
	99 %	5.2E-10	2.5E-09	5.0E-09	1.1E-09	5.1E-09	1.0E-08	6.9E-09	2.7E-08	5.1E-08
1oo3	**0 %**	5.1E-08	2.5E-07	5.0E-07	1.1E-07	5.1E-07	1.0E-06	1.4E-06	3.2E-06	5.5E-06
	60 %	2.0E-08	1.0E-07	2.0E-07	4.0E-08	2.0E-07	4.0E-07	2.6E-07	1.0E-06	2.0E-06
	90 %	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07	5.1E-08	2.5E-07	5.0E-07
	99 %	5.0E-10	2.5E-09	5.0E-09	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08

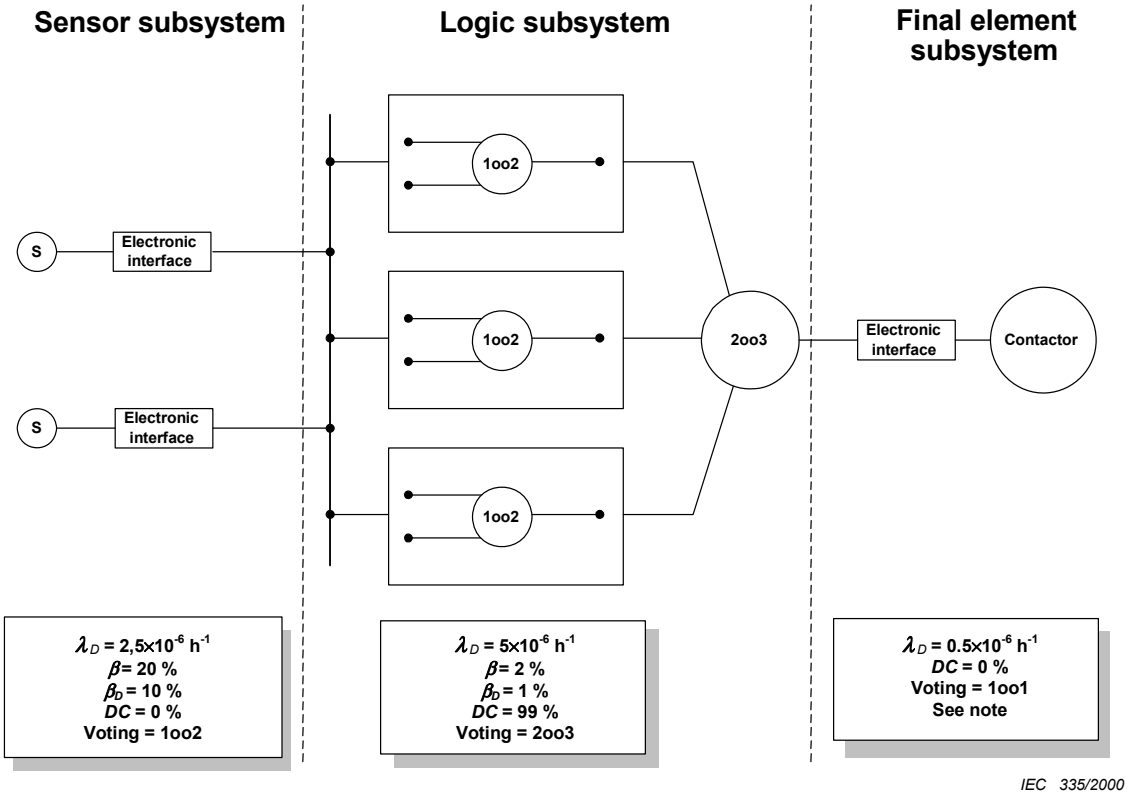
NOTE 1 This table gives example values of PFH_G, calculated using the equations in B.3.3 and depending on the assumptions listed in B.3.1. If the sensor, logic or final element subsystem comprises of only one group of voted channels, then PFH_G is equivalent to PFH_S, PFH_L or PFH_{FE} respectively (see B.3.3.1).

NOTE 2 The table assumes $\beta = 2 \times \beta_D$. For 1oo1 and 2oo2 architectures, the values of β and β_D do not affect the average frequency of a dangerous failure.

NOTE 3 The safe failure rate is assumed to be equal to the dangerous failure rate and K = 0,98.

B.3.3.4 Example for high demand or continuous mode of operation

Consider a safety function requiring a SIL 2 system. Suppose that the initial assessment for the system architecture, based on previous practice, is for one group of two sensors, voting 1oo2. The logic subsystem is a redundant 2oo3 configured PE system driving a single shutdown contactor. This is shown in Figure B.15 For the initial assessment, a proof test period of six months is assumed.



NOTE The final element subsystem has an overall safe failure fraction greater than 60 %.

Figure B.15 – Architecture of an example for high demand or continuous mode of operation

Table B.14 – Average frequency of a dangerous failure for the sensor subsystem in the example for high demand or continuous mode of operation (six month proof test interval and 8 h *MTTR*)

Architecture	*DC*	λ_D =2,5E-06		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
1oo2	0 %	7,6E-08	2,7E-07	5,2E-07
	60 %	2,4E-08	1,0E-07	2,0E-07
	90 %	5,3E-09	2,5E-08	5,0E-08
	99 %	5,0E-10	2,5E-09	5,0E-09
NOTE This table is abstracted from Table B.12.				

Table B.15 – Average frequency of a dangerous failure for the logic subsystem in the example for high demand or continuous mode of operation (six month proof test interval and 8 h *MTTR*)

Architecture	*DC*	λ_D = 0,5E-05		
		β = 2 % β_D = 1 %	β = 10 % β_D = 5 %	β = 20 % β_D = 10 %
2oo3	0 %	4,2E-07	7,7E-07	1,2E-06
	60 %	9,1E-08	2,4E-07	4,4E-07
	90 %	1,3E-08	5,3E-08	1,0E-07
	99 %	1,0E-09	5,0E-09	1,0E-08
NOTE This table is abstracted from Table B.12.				

Table B.16 – Average frequency of a dangerous failure for the final element subsystem in the example for high demand or continuous mode of operation (six month proof test interval and 8 h *MTTR*)

Architecture	*DC*	λ_D = 0,5E-06
1oo1	0 %	5,0E-07
	60 %	2,0E-07
	90 %	5,0E-08
	99 %	5,0E-09
NOTE This table is abstracted from Table B.12.		

From Tables B.14 to B.16 the following values are derived.

For the sensor subsystem,

$PFH_S = 5{,}2 \times 10^{-7}$ / h

For the logic subsystem,

$PFH_L = 1{,}0 \times 10^{-9}$/ h

For the final element subsystem,

$PFH_{FE} = 5{,}0 \times 10^{-7}$ / h

Therefore, for the safety function,

$PFH_{SYS} = 5{,}2 \times 10^{-7} + 1{,}0 \times 10^{-9} + 5{,}0 \times 10^{-7}$

$= 1{,}02 \times 10^{-6}$ / h

≡ **safety integrity level 1**

To improve the system to meet safety integrity level 2, one of the following could be done:

a) change the input sensor type and mounting to improve the defences against common cause failure, thus improving β from 20 % to 10 % and β_D from 10 % to 5 %;

$PFH_S = 2{,}7 \times 10^{-7}$ / h

$PFH_L = 1{,}0 \times 10^{-9}$ / h

$PFH_{FE} = 5{,}0 \times 10^{-7}$ / h

$PFH_{SYS} = 7{,}7 \times 10^{-7}$ / h

≡ **safety integrity level 2**

b) change the single output device to two devices in 1oo2 (β = 10 % and β_D = 5 %).

PFH_S = $5{,}2 \times 10^{-7}$ / h

PFH_L = $1{,}0 \times 10^{-9}$ / h

PFH_{FE} = $5{,}1 \times 10^{-8}$ / h

PFH_{SYS} = $5{,}7 \times 10^{-7}$ / h

≡ **safety integrity level 2**

B.4 Boolean approach

B.4.1 General

The Boolean approach encompasses the techniques representing the logical function linking the individual component failures to the overall system failure. The main Boolean models used in the reliability field are Reliability Block Diagrams (RBD), Fault Trees (FT), Event Trees and Cause Consequence Diagrams. Only the first two methods are considered here. The aim of all these methods is to represent the logical structure of the system. However, its behaviour over time is not included in these model techniques. Therefore, care has to be taken when considering behavioural features (e.g. time dependent features such as periodic proof tests) when undertaking the calculations. The first approach for using Boolean models is to split the graphical representation from the calculations. This has been described in the previous section where RBD are used to model the structure and Markovian calculations used to assess *PFD* or *PFH*. Further considerations are now discussed for probabilistic calculations on RBD and FT.

This approach is limited to components behaving reasonably independently from each other.

B.4.2 Reliability block diagram model

A lot of examples of RBD have been previously given and Figure B.1 represents, for example, a whole safety loop made of three sensors (A, B, C) working in 1oo3, one logic solver (D) and two terminal elements (E, F) working in 1oo2.

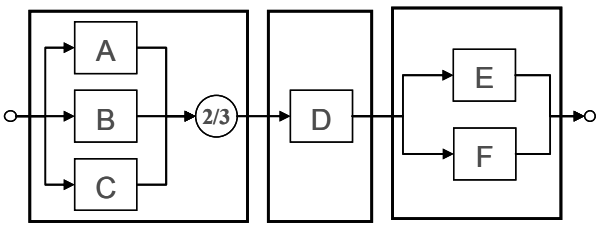


Figure B.16 – Reliability block diagram of a simple whole loop with sensors organised into 2oo3 logic

Figure B.16 shows a similar loop with sensors working in 2oo3. The main interest of such graphical representation is threefold: it remains very close to the physical structure of the system under study, it is widely used by engineers and it is a good support for discussion.

The main shortcoming is that RBD is more a method of representation than a method of analysis in itself.

For more details on RBD see C.6.4 of IEC 61508-7 and IEC 61078.

B.4.3 Fault tree model

Fault trees have exactly the same properties as RBD but in addition they constitute an effective deductive (top-down) method of analysis helping reliability engineers to develop

models step by step from the top event (unwanted or undesirable event) to the individual components failures.

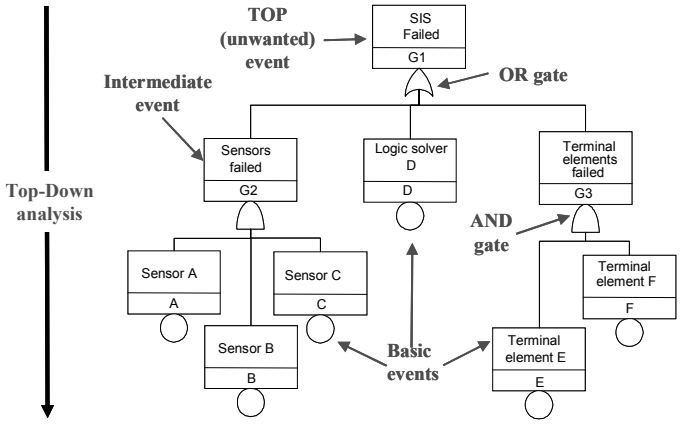


Figure B.17 – Simple fault tree equivalent to the reliability block diagram presented on Figure B.1

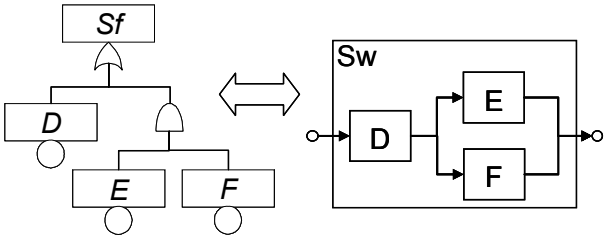
Figure B.17 shows a fault tree which is perfectly equivalent to the RBD presented on Figure B.1 but where the steps of the top-down analysis are identified (for example: E/E/PE safety-related system failed => Sensor failed => sensor A failed). In FT, the elements in series are linked by "OR gates" and element in parallel (i.e. redundant) are linked by "AND gates".

For more details on FT see B.6.6.5 and B.6.6.9 of IEC 61508-7 and IEC 61025.

B.4.4 *PFD* calculations

B.4.4.1 General overview

RBD and FT representing exactly the same things, the calculations may be handled exactly in the same way. Figure B.18 shows small equivalent FT and RBD which will be used to show the main principles of the calculations.



NOTE In this figure, italic letters are used for failed items and non-italic letters for working items.

Figure B.18 – Equivalence fault tree / reliability block diagram

The small FT represents the logical function $Sf = D \cup (E \cap F)$ where *Sf* is the failure of the system and *D, E, F* the failures of the individual components. The small RBD represents the logical function $Sw = D \cap (E \cup F)$ where Sw is the good functioning of the system and D, E, F the good functioning of the individual components. Then *Sf* = NOT Sw, and *Sf* and Sw represent exactly the same information (i.e. the logical function and its dual).

The primary use of FT and RBD is identifying the combinations of the various component failures leading to the overall system failure. They are the minimal cut sets so-called because they indicate where to cut the RBD in order that a signal sent at the input does not reach the output. In this case, there are two cut sets: the single failure (*D*) and the double failure (*E, F*).

Applying basic probabilistic mathematics on logical functions lead straightforwardly to the probability of failure P_{Sf} of the system and we obtain

$$P_{Sf} = P(D) + P(E \cap F) - P(D \cap E \cap F)$$

If the components are independent this formula becomes:

$$P_{Sf} = P_D + P_E P_F - P_D P_E P_F$$

where

P_i is the probability that component i is failed.

This formula is time independent and reflects only the logical structure of the system.

Therefore both RDB and FT are basically static, i.e. time independent models.

Nevertheless, if the probability of failure of each individual component at time *t* is independent of what happens on the other component over *[0, t]* the above formula remains valid at any time and we can write:

$$P_{Sf}(t) = P_D(t) + P_E(t)P_F(t) - P_D(t)P_E(t)P_F(t)$$

The analyst should verify if the required approximations are acceptable or not and finally, the instantaneous unavailability $U_{Sf}(t)$ of the system is obtained:

$$U_{Sf}(t) = U_D(t) + U_E(t)U_F(t) - U_D(t)U_E(t)U_F(t)$$

The conclusion is that fault trees or reliability block diagrams allow calculating directly the instantaneous unavailability $U_{Sf}(t)$ of E/E/PE safety-related systems and that additional calculations are needed and according to B.2.2:

$$PFD_{avg}(T) = \frac{1}{T} MDT(T) = \frac{1}{T}\int_0^T U_{Sf}(t)dt$$

This principle may be applied to minimal cut sets:

– single failure (*D*): $PFD^D(\tau) = \frac{1}{\tau}\int_0^\tau \lambda_D t.dt = \lambda_D \tau / 2$

– double failure (*E, F*): $PFD^{EF}(\tau) = \frac{1}{\tau}\int_0^\tau \lambda_E.\lambda_F.t^2 dt = \lambda_E \lambda_F \tau^2 / 3$

B.4.4.2 Calculations based on fault trees or reliability block diagram tools

The formula $U_{Sf}(t) = U_D(t) + U_E(t)U_F(t) - U_D(t)U_E(t)U_F(t)$ described above is only a particular case of the so-called Poincaré formula. More generally if $Sf = \bigcup_i C_i$ where (C_i) represents the minimal cut sets of the system:

$$P(\bigcup_{i-1}^{n} C_i) = \sum_{j=1}^{n} P(C_j) - \sum_{j=1}^{n}\sum_{i=1}^{j-1} P(C_j \bigcap C_i) + \sum_{j=3}^{n}\sum_{i=2}^{j-1}\sum_{k=1}^{j-1} P(C_j \bigcap C_i \bigcap C_k) - \ldots .$$

The number of minimal cut sets increases exponentially when the number of individual components increases. Then the Poincaré formula leads to a combinatory explosion of terms very quickly intractable by hand. Fortunately this problem has been analysed over the last forty years and numerous algorithms have been developed to manage such calculations. At the present time the last developments and the most powerful ones are based on the so-called Binary Decision Diagrams (BDD) which are derived from a sophisticated Shannon decomposition of the logical function.

A lot of commercial software packages mainly based on fault tree models are used daily by reliability engineers in various industry fields (nuclear, petroleum, aeronautics, automotive, etc.). They can be used for PFD_{avg} calculation but analysts have to be very cautious because some of them implement wrong PFD_{avg} calculations. The main mistake encountered is the conventional combination of the $PFD_{avg,i}$ of individual components (generally obtained simply by $\lambda_i \tau/2$) to produce a result which is supposed to be the whole system PFD_{avg}. As shown above this is wrong and non conservative.

Anyway, fault tree software packages may be used to calculate the instantaneous system unavailability $U_{Sf}(t)$ from the instantaneous unavailabilities of its components $U_i(t)$. Then the average of $U_{Sf}(t)$ may be done over the period of interest to evaluate the PFD_{avg}. Depending on the software in use this can be done by the software itself or by side calculations.

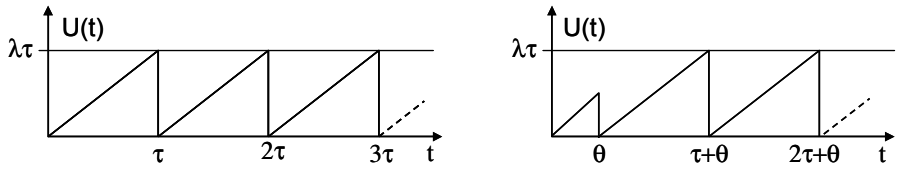


Figure B.19 – Instantaneous unavailability *U(t)* of single periodically tested components

The ideal case previously described is presented on the left of Figure B.19:

$$U_i(t) = \lambda \zeta \text{ and } \zeta = t \text{ modulo } \tau.$$

This is a so called saw-tooth curve increasing linearly from 0 to $\lambda\tau$ and restarting from 0 after a test or a repair (which are considered to be instantaneous as the EUC is stopped during them).

When several components are used in redundant structures, the tests may be staggered as shown on the right of Figure B.19 where the first test interval is different from the others. That has no impact on the PFD_{avg} or on maximum value which are equal to $\lambda\tau/2$ and $\lambda\tau$ in both cases.

Of course in less ideal cases these curves may be more complicated than that. Guidelines will be given in B.5.2 to design more accurate saw tooth curves but for the purpose of this chapter the curves presented on Figure B.19 are sufficient.

This can be applied to the small fault tree presented on Figure B.18 as illustrated in Figure B.20 (where DU means Dangerous Undetected and CCF means Common Cause Failure) . We have considered that the system was made of two redundant components (E and F) and that (D) is a common cause failure on these components. The calculation has been achieved with the following figures:

$$\lambda_{DU} = 3{,}5 \times 10^{-6}/\text{h},\ \tau = 4\ 380 \text{ h and } \beta = 1\ \%$$

A small β factor has been chosen to ensure that CCF does not dominate the result at the top and to gain a better understanding on how that works.

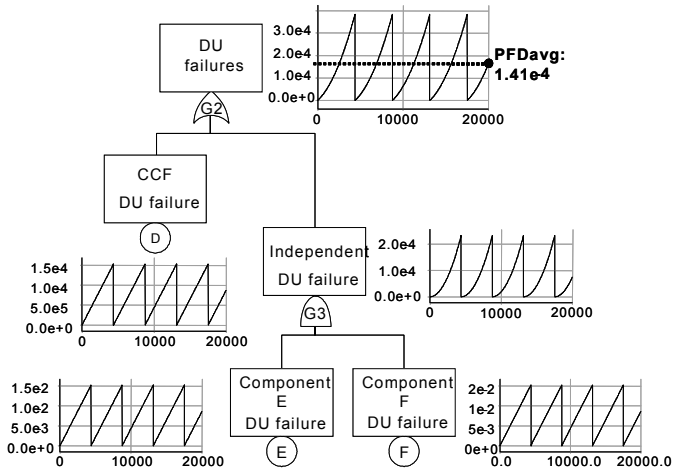


Figure B.20 – Principle of PFD_{avg} calculations when using fault trees

It is easy to recognize the kind of saw-tooth curves presented on the left hand side of Figure B.19 as inputs for *D, E* and *F*. The CCF (*D*) is tested each time *E* or *F* are tested. Then, as *E* and *F* are tested at the same time every 6 months, the CCF (*D*) is also tested every 6 months.

By using one of the algorithms developed for fault tree calculations, it is easy to draw the saw-tooth curves at outputs of all logical gates. The PFD_{avg} is calculated by averaging the result obtained for the top event. This can be performed by Fault Tree software itself or by manual calculations. $PFD_{avg} = 1{,}4 \times 10^{-4}$ is obtained and according to this standard, this meets the target failure measure for SIL 3 for a low demand mode of operation.

As shown on Figure B.20, the graphs are smooth between tests. Therefore there are no difficulties to evaluate the average, provided the instant of tests are identified and taken under consideration.

It is interesting to notice that as soon as redundancy is implemented, the saw-tooth curves at top event level are no longer linear between tests (i.e. the overall system failure rate is no longer constant).

It is also interesting to measure the impact on the PFD_{avg} of staggering the tests of the redundant components instead of performing them at the same time. This is illustrated in Figure B.21 where the tests of the component F have been staggered from those of component E by 3 months.

This has several important effects:

– CCF are now tested every 3 months (i.e. each time E is tested and each time F is tested). This proof test frequency is the double as the previous case.

– the top event saw-tooth curve has also a proof test frequency double than the one applied before.

– the saw tooth curve is less spread around its average than in the previous case.

– the PFD_{avg} has decreased to $8{,}3 \times 10^{-5}$: with this new test policy the system is SIL 4.

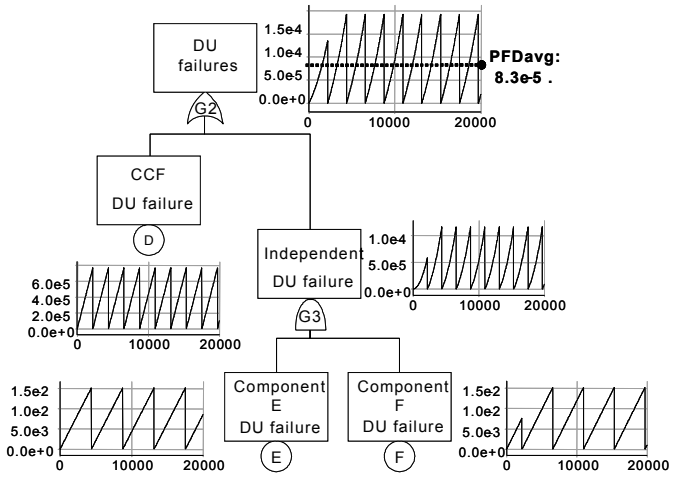


Figure B.21 – Effect of staggering the tests

If the tests are staggered and adequate procedures implemented this will increase the likelihood of detecting CCFs and is an effective method of reducing the CCF for systems operating in a low demand mode of operation. Here it has been improved from SIL 3 to SIL 4 (from hardware failures point of view and if other requirements of the IEC 61508 series are met).

Figure B.22 represents the saw-tooth curve obtained when adding a component G (λ_{DU} = 7 × 10^{-9}/h and never tested) and a component H (λ_{DU} = 4 × 10^{-8}/h tested every 2 years) in series with the system modelled on Figure B.20.

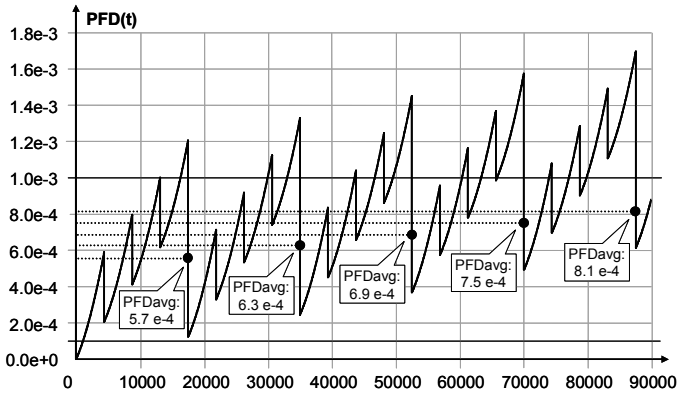


Figure B.22 – Example of complex testing pattern

The impact of the component G never tested is twofold: *PFD(t)* never goes back to zero after the test performed every two years and PFD_{avg} increases continuously (black circles corresponding to the PFD_{avg} over the period covered by the related dotted line).

Even if the elementary saw-tooth curves (like those presented on Figure B.19) are very simple, the results at the top event level may be rather complicated but this does not raise particular difficulties.

This clause aims only to illustrate the principle of the calculation by using Boolean models. Subclause B.5.2 related to the Markovian approaches, will give some guidelines to design more sophisticated input saw-tooth curves for elementary components.

It can be concluded that, when the individual components are reasonably independent there is no problem to handle PFD_{avg} calculations for E/E/PE safety-related systems by using

classical Boolean techniques. This is not so simple from theoretical point of view and the analyst doing the study should have acquired a sound knowledge of the probabilistic calculations to identify and discard the wrong PFD_{avg} implementations sometimes encountered. Provided these precautions are taken, any fault tree software package may be used for above calculations.

Boolean techniques may be also used for *PFH* calculations but the theoretical developments are beyond the purpose of this informative annex.

B.5 States/transition approaches

B.5.1 General

Boolean models are basically time-independent and the introduction of time is possible only in specific particular cases. This is rather artificial and a good knowledge of probabilistic calculations is needed to avoid mistakes. Therefore other probabilistic models, dynamic in nature may be used instead. In the reliability field they are fundamentally based on the next approach in two steps:

– identification of all the states of the system under study;

– analysis of the jumps (transitions) of the system from states to states, according to events arising and along its life.

This is why they are gathered in the category of states/transitions models.

The general approach actually consists in building a kind of automaton behaving like the system under study when events (failures, repairs, tests, etc.) are arising. As per this standard, E/E/PE safety-related systems have only discrete states, this is equivalent to building a so-called finite state automaton. Those models are dynamic in nature and may be implemented in various manners: graphic representations, specific formal languages or common programming languages. This annex presents two of them which are very different but complementary:

– Markov model which has been developed at the very beginning of the last century. It is rather well known and handled analytically;

– Petri net model which has been developed in the sixties. It is less well known (but more and more used because of its flexibility) and handled by Monte Carlo simulation.

Both are based on graphical drawings very helpful for users. Other techniques based on formal languages modelling will be very quickly analysed at the end of this clause.

B.5.2 Markovian approach

B.5.2.1 Principle of modelling

The Markovian approach is the elder of all the dynamic approaches used in the reliability field. Markov processes are split between those which are "amnesic" (homogeneous Markov processes where all transition rates are constant) and the others (semi Markov processes). As the future of a homogeneous Markov process does not depend on its past, analytical calculations are relatively straightforward. This is more difficult for semi Markov processes for which Monte Carlo simulation can be used. In this part of the IEC 61508 series, only homogeneous Markov processes are considered and the term "Markov processes" is used for the sake of simplicity (see C.6.4 of IEC 61508-7 and IEC 61165).

The fundamental basic formula of Markov processes is the following:

$$P_i(t+dt) = \sum_{k \neq i} P_k(t)\lambda_{ki}dt + P_i(t)(1 - \sum_{k \neq i} \lambda_{ik}dt)$$

In this formula, λ_{ki} is the transition rate (e.g. failure or repair rate) from state *i* to state *k*. It is self explaining: the probability to be in state *i* at *t+dt* is the probability to jump toward *i* (when in another state *k*) or to remain in state *i* (if already in this state) between *t* and *t* + *dt*.

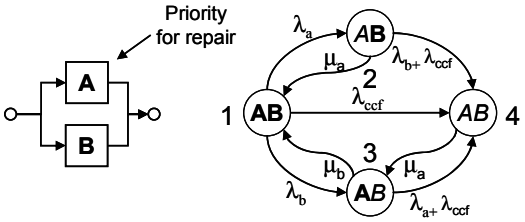


Figure B.23 – Markov graph modelling the behaviour of a two component system

There is a straightforward relationship between the above equation and a graphical representation like Figure B.23 which models a system made of two components with a single repair team (component A having priority to be repaired) and a common cause failure. In this figure **A** indicates that A is working and *A* that it has failed. As the detection times must be considered μ_a and μ_b in Figure B.23 are the restoration rates of the components (i.e. μ_a=*1/MTTR*$_a$ and μ_b=*1/MTTR*$_b$).

For example the probability to be in state 4 is simply calculated as follows:

$$P_4(t+dt) = [P_1(t)\lambda_{ccf} + P_2(t)(\lambda_b + \lambda_{ccf}) + P_3(t)(\lambda_a + \lambda_{ccf})]dt + P_4(t)(1-\mu_a dt)$$

This leads to a vectorial differential equation,

$d\vec{P}(t)/dt = [M]\vec{P}(t)$, which is conventionally solved by:

$$\vec{P}(t) = e^{t[M]}\vec{P}(0)$$

where

[*M*] is the Markovian matrix containing the transition rates and $\vec{P}(0)$ the vector of the initial conditions (generally a column vector with 1 for the perfect state an 0 for the others).

Even if an exponential of a matrix has not exactly the same properties of an ordinary exponential, it is possible to write:

$$\vec{P}(t) = e^{(t-t1)[M]}e^{t1[M]}\vec{P}(0) = e^{(t-t1)[M]}\vec{P}(t1)$$

This demonstrates the basic property of Markov processes: the knowledge of the probabilities of the states at a given instant *t1* summarizes all the past and is enough to calculate how the system evolves in the future from *t1*. This is very useful for *PFD* calculations.

Efficient algorithms have been developed and implemented in software packages a long time ago in order to solve above equations. Then, when using this approach, the analyst can focus only on the building of the models and not on the underlying mathematics even if, anyway, he has to understand at least what is described in this appendix.

Figure B.24 shows the principle of *PFD* calculations:

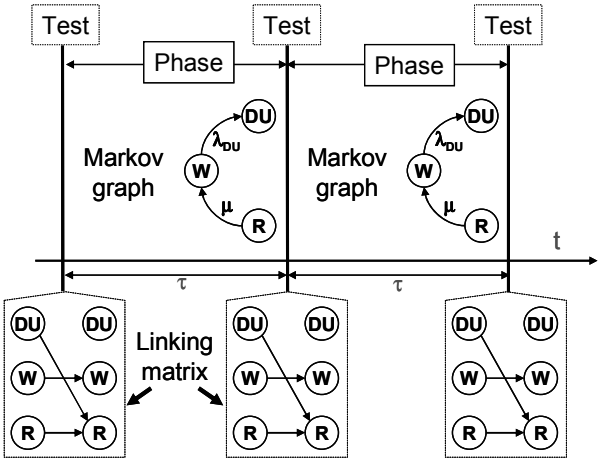


Figure B.24 – Principle of the multiphase Markovian modelling

PFD calculations are related to E/E/PE safety-related systems working in low demand mode and periodically (proof) tested. For such systems repairs are initiated only when tests are performed. The tests are singular points along the time but this is not a problem as a multi phase Markovian approach may be used to deal with.

For example, a simple system made of one periodically tested single component has three states as shown on Figure B.24: working (W), dangerous failure undetected (DU) and under repair (R).

Between tests its behaviour is modelled by the Markovian process on the upper part of Figure B.24: it can fail (W→ DU) or under repair (R→ W). As no repair may be started within a test interval, there is no transition from DU to R. Because the diagnostic of the failure has been performed before entering state R, μ is the repair rate of the component (i.e. μ=1/*MRT*) in Figure B.24.

When a test is performed (see linking matrix on Figure B.14), a repair is started if a failure has occurred (DU→ R), the component remains working if it was in a good functioning state (W→ W) and in the very hypothetical case that a repair started at the previous test is not finished, remains under repair (R→ R). A linking matrix [*L*] may by used to calculate the initial conditions at the beginning of state *i+1* from the probabilities of the states at the end of test *i*. This gives the following equation:

$$\begin{bmatrix} P_{DU}(0) \\ P_w(0) \\ P_R(0) \end{bmatrix}_{i+1} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{DU}(\tau) \\ P_w(\tau) \\ P_R(\tau) \end{bmatrix}_i \equiv \vec{P}_{i+1}(0) = [L]\vec{P}_i(\tau)$$

Replacing $\vec{P}_i(\tau)$ by its value, leads to an equation of recurrence which allows to calculate the initial conditions at the beginning of each test intervals:

$$\vec{P}_{i+1}(0) = [L]e^{t[M]}\vec{P}_i(0)$$

This can be used to calculate the probabilities at any time $t = i\tau + \zeta$. For example, within test interval *i,* the following is obtained:

$$\vec{P}(t) = \vec{P}_i(\zeta) = e^{\zeta[M]}\vec{P}_i(0), \quad (i-1)\tau \le t < i\tau, \quad \zeta = t \bmod \tau$$

Obtaining the instantaneous unavailability is straightforward by summing the probabilities of the states where the system is unavailable. A line vector (q_k) is helpful to express that:

$$U(t) = \sum_{k=1}^{n} q_k P_k(t)$$

where $q_k = 1$ if the system is unavailable in state k, and $q_k = 0$ otherwise.

For the simple model, $PFD(t) = U(t) = P_{DU}(t) + P_R(t)$ is obtained and the look of the saw-tooth curve obtained with this model is illustrated on Figure B.25.

The PFD_{avg} is calculated in the way previously described through the *MDT* which in turn is easy to calculate from the Mean Cumulated Times spent in the states:

$$\vec{MCT}(T) = \int_0^T \vec{P}(t)dt$$

Like for $\vec{P}(t)$, efficient algorithms are well known to perform this calculation over *[0, T]* to finally obtain: $PFD_{avg}(T) = \frac{1}{T}\sum_{k=1}^{n} q_k MCT_k(T)$

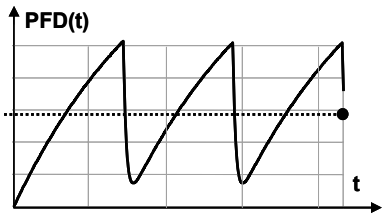


Figure B.25 – Saw-tooth curve obtained by multiphase Markovian approach

Applying this formula on the model presented on Figure B.24 leads to:

$$PFD_{avg}(T) = \frac{1}{T}\left[MCT_{DU}(T) + MCT_R(T)\right]$$

This may be reduced to the first term if the EUC is shut down during the repair.

The black circle on Figure B.25 is the PFD_{avg} of the saw-tooth curve over the whole period of calculation.

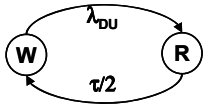


Figure B.26 – Approximated Markovian model

Note that the above calculations are often performed by using the approximated model presented on Figure B.26 where the state DU and R have been merged and where $\tau/2$ (i.e. the mean time to detect the failure) has been used as equivalent restoration time. This is valid only if the Markovian equations have been previously solved by another way in order to find this equivalence. This approximation is only applicable if the repair time is negligible. Also, the method may be very difficult for large complex systems.

The simple model of Figure B.24 may be easily improved for more realistic components. On Figure B.27, the linking matrix models a component which has both a probability, γ,to be failed due to the test (i.e. a genuine on demand failure) and a probability, σ, that the failure was not discovered by the test (by human error).

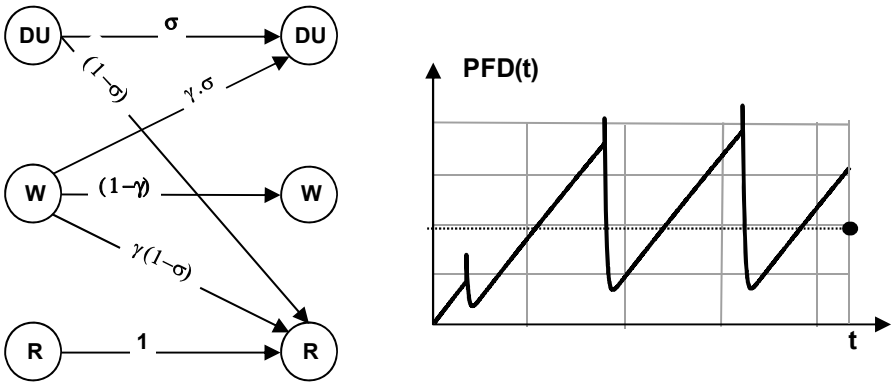


Figure B.27 – Impact of failures due to the demand itself

The look of the saw-tooth curve has changed and the jumps observed for each test correspond to the probability of on demand failure γ. Again the black circle presents the PFD_{avg}.

When a (redundant) component is disconnected to be tested, it becomes unavailable during the whole performance of the test and this contributes to its PFD_{avg}. Therefore, the test duration, π,shall be considered and an additional phase introduced between test intervals. This is shown on Figure B.28 where states R and W are modelled in this phase only for the sake of the completeness.

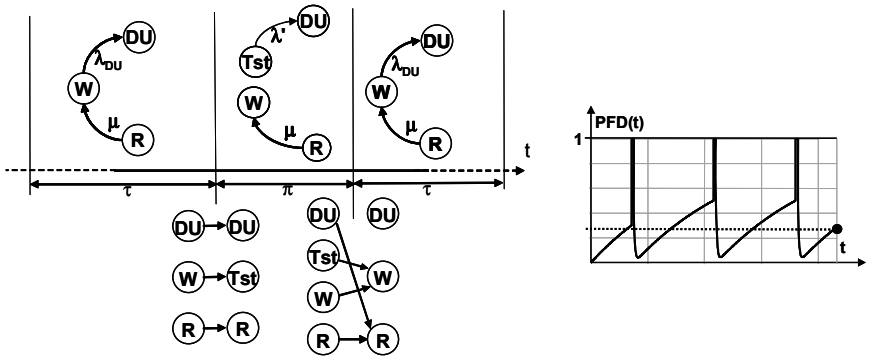


Figure B.28 – Modelling of the impact of test duration

In this Markov model, the system is unavailable in states R, DU and Tst. This is more complicated than before but the principle of calculation remains exactly the same. The behaviour of the saw-tooth curve is shown on the right. The system is unavailable during the test durations and this may be the top contribution to PFD_{avg}.

On the previous Markov graphs, only the dangerous undetected failures have been considered but the dangerous detected failure may be represented as well. The difference is that repair starts at once as it is represented on Figure B.29. Therefore μ_{DD} is the restoration rate of the component (μ_{DD} =*1/MTTR*)when μ_{DU} is its repair rate (μ_{DU}=1*/MRT)*.

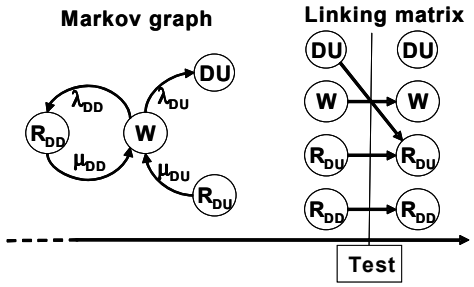


Figure B.29 – Multiphase Markovian model with both DD and DU failures

Safe failures should be represented if needed but the Markov graphs presented here are chosen to be as simple as possible.

The main problem with Markov graphs is that the number of states increases exponentially when the number of components of the system under study increases. Therefore building Markov graphs and performing above calculations without drastic approximations becomes very quickly intractable by hand.

Using an efficient Markovian software package helps to cope with calculation difficulties. There are a lot of such packages available even if they are not necessarily usable directly for PFD_{avg} calculation purpose: most of them perform instantaneous unavailability calculations but only some of them calculate the mean cumulated times spent in the states and only a few allow multiphase modelling. Anyway, there are no real difficulties to adapt them to PFD_{avg} calculations.

Concerning the modelling itself and when dependencies between components are light, Markovian and Boolean approaches can be mixed:

– Markov models are used to establish the instantaneous unavailabilities of each of the components;

– fault trees or reliability block diagrams are used to combine the individual unavailabilities to calculate the instantaneous unavailability *PFD(t)* of the whole system;

– PFD_{avg} is obtained by averaging *PFD(t)*.

This mixed approach has been described in Clause B.4 and saw-tooth curves like those on Figures B.25, B.27 and B.28 may be used as input to fault trees.

When dependencies between components cannot be neglected, some tools are available to build automatically the Markov graphs. They are based on models of a higher level than Markov models (e.g. Petri nets, formal language). Due to the combinatory explosion of the number of states, can still lead to difficulties.

This combined approach is very efficient for modelling complex systems.

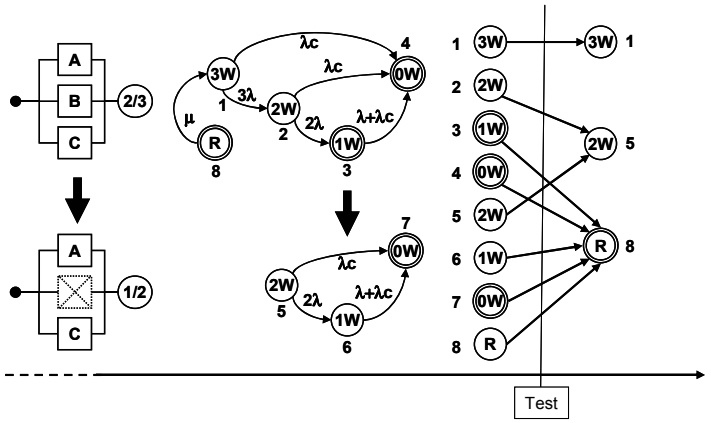


Figure B.30 – Changing logic (2oo3 to 1oo2) instead of repairing first failure

The system modelled in Figure B.30 is made of three components tested at the same time and working in 2oo3. When a failure is detected, the logic is changed from 2oo3 to 1oo2 as a 1oo2 logic is better than a 2oo3 logic from safety point of view (but worse from spurious failure point of view). It is only when a second failure is detected that the repair occurs and this consists in replacing all the three elements by three new ones. This introduces systemic constraints and it is not possible to build the behaviour of the whole system just by combining the independent behaviours of its components.

B.5.2.2 Principle of *PFH* calculations

For *PFH* calculations, the same type of multiphase Markovian modelling can be used for DU failures detected by proof tests. In order to simplify, only the principle of *PFH* calculations for DD failures which only need conventional (monophase) Markov models is shown. Of course, for E/E/PE safety-related systems working in continuous mode and having DU failures detected by periodical proof tests, the multiphase Markovian approach should be used. This does not change the principle considered hereafter.

Figure B.31 presents two Markov graphs modelling the same system made of two redundant components with a common cause failure. On the left hand side, the components (A and B) are repairable. On the right hand side, they are not repairable.

On both graphs state 4 (*AB*) is absorbing. The system remains failed after an overall failure and *P(t) = P1(t) + P2(t) + P3(t)* is the probability that no failure has occurred over *[0, t]*. Then, *R(t) = P(t)* is the reliability of the system and *F(t) = 1 - R(t) = P4(t)* is its unreliability.

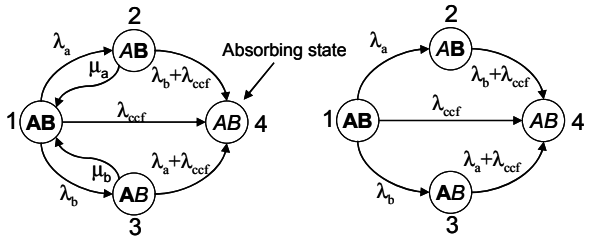


Figure B.31 – "Reliability" Markov graphs with an absorbing state

As discussed in B.2.3., this reliability model is adequate for dealing with the situation when the failure of the E/E/PE safety-related system leads immediately to a dangerous situation. Again μ_a and μ_b are the restoration rates of the components (i.e. μ_a=1/*MTTR*$_a$ and μ_b=1/*MTTR*$_b$)

Such a reliability Markov graph provides the *PFH* directly by *PFH* = *F(T)/T*. For example from Figure B.31 $PFH(T) = P4(T)/T$ (provided *P4(T)* << 1), is directly obtained.

Such a reliability Markov graph also allows the *MTTF* of the system to be calculated by the following formula:

$$MTTF = \lim_{t \to \infty} \sum_{k=1}^{n} a_k MCT_k(t)$$

In this formula $MCT_k(t)$ is the mean cumulated time spent in state *k,* and a_k = 1 if *k* is a good working state and a_k = 0 otherwise.

An upper bound is obtained by:

$$PFH \approx 1/MTTF$$

Efficient algorithms have been developed and almost all the Markovian software packages may be used for *F(T)* and *MTTF* calculations.

The above *PFH* estimations are valid in any case even if there is no overall system constant failure rate (as for the graph on the right hand side of Figure B.31). The only constraint is to use a reliability Markov graph with one (or several) absorbing state(s). Of course this holds when using a multiphase model.

When all the states are completely and quickly repairable, the overall system failure rate $\Lambda(t)$ converge quickly toward an asymptotic value Λ_{as} = 1/*MTTF*. In such graphs, except the perfect and the absorbing states, all states are quasi instantaneous (because the *MTTRs* of the components are short compared to their *MTTF*). This allows evaluating directly the overall system constant failure rates of each scenario starting from the perfect state and leading to the absorbing state. The Markov graph on the left hand side of Figure B.31 models such a completely and quickly repairable system. That is:

– $1 \rightarrow 4$: $\Lambda_{14} = \lambda_{ccf}$

– $1 \rightarrow 2 \rightarrow 4$: $\Lambda_{124} = \lambda_a . (\lambda_b + \lambda_{ccf})/[(\lambda_b + \lambda_{ccf}) + \mu_a] \approx \lambda_a . (\lambda_b + \lambda_{ccf})/\mu_a$

– $1 \rightarrow 3 \rightarrow 4$: $\Lambda_{134} = \lambda_b . (\lambda_a + \lambda_{ccf})/[(\lambda_a + \lambda_{ccf}) + \mu_b] \approx \lambda_b . (\lambda_a + \lambda_{ccf})/\mu_b$

For the scenario $1 \rightarrow 3 \rightarrow 4$ in above formulae, λ_b is the transition rate governing the jump out of the perfect state, and $(\lambda_a + \lambda_{ccf})/\mu_b$ is the probability to jump to 4 rather to come back to 1 when in state 3.

Finally: $\Lambda_{as} = \Lambda_{12} + \Lambda_{124} + \Lambda_{134} = 1/\text{MTTF}$

This can be easily generalized to complex Markov graphs but this is valid only for completely and quickly repairable systems, i.e. DD failures.

The Markov graph on the right hand side of Figure B.31 is not completely and quickly repairable. Therefore applying above calculation would lead to wrong results.

When the E/E/PE safety-related system working in continuous mode is used in conjunction with other safety barriers, its availability shall be considered. This is what is represented on both graphs of Figure B.32 below: there is no absorbing state and the system is repaired after an overall failure. *P(t) = P1(t) + P2(t) + P3(t)* is the probability that the system is working at *t*. Then, *A(t) = P(t)* is its availability and *U(t)* = 1 - *A(t) = P4(t)* its unavailability.

This case is very different from the example represented in Figure B.31 and *R(t)* and *A(t)* should be used correctly, as should *U(T)* and *F(T)* if correct results are to be obtained.

In case of DD failures, the simplest way to handle this problem is to calculate the upper bound of *PFH* through *MDT* and *MUT* as explained in B.2.3.

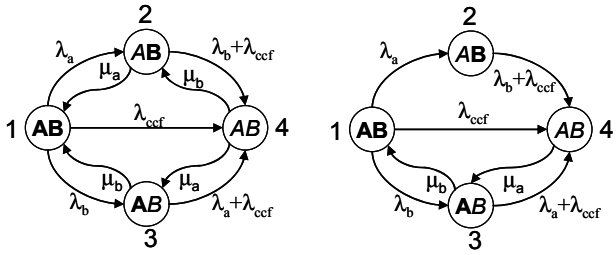


Figure B.32 – "Availability" Markov graphs without absorbing states

An interesting property of availability Markov graphs is that they reach an asymptotic equilibrium when the probability to enter a given state is equal to the probability to get out of it. Noted by:

– $P_{i,as} = \lim_{t\to\infty} P_i(t)$ the asymptotic value of $P_i(t)$

– $\lambda_i = \sum_{j\neq i} \lambda_{ij}$ the transition rate from *i* to any other state

Each time the system visits state *i*, the mean time in this state is $Mst_i = 1/\lambda_i$.

This allows calculating $MUT = \sum_i (1-q_i) P_{i,as} Mst_i$ and $MDT = \sum_i q_i P_{i,as} Mst_i$

where

q_i = 0, if *i* is a working state, and 1 otherwise.

Finally the following is obtained: $PFH = 1/(MUT + MDT) = 1/\sum_i P_{i,as} Mst_i = 1/\sum_i \frac{P_{i,as}}{\lambda_i}$

It should be noted that the number of failures observed over *[0, T]* is given by: $n = T/\sum_i \frac{P_{i,as}}{\lambda_i}$.

As most of the Markovian software packages are able to find the asymptotic probabilities there are no particular difficulties to achieve above calculations.

When the period under interest is too short for allowing the convergence of the Markov process, *PFH* may be calculated with: $w(t) = \sum_{i\neq f} \lambda_{if} P_i(t)$.

This gives: $PFH(T) = \sum_{i\neq f} \lambda_{if} \frac{\int_0^T P_i(t)dt}{T} = \frac{\sum_{i\neq f} \lambda_{if} MCT_i(T)}{T}$

There is again no difficulty to perform these calculations with a Markov software package providing the cumulated time in each of the states.

In the case of completely and quickly repairable systems (DD failures) the Vesely rate $\Lambda_v(t)$ converges very quickly toward an asymptotic value, Λ_{as}, which is a good approximation of the overall system constant failure rate of the system. Therefore in this case, the *PFH* may be calculated in the same way as in the reliability case.

In case of DU failures, this is more complicated due to the multiphase modelling. The above formula may be generalized to: $PFH(T) = \dfrac{\sum_{\varphi=1}^{n} \sum_{i \neq f} \lambda_{if} MCT_i(T_\varphi)}{\sum_{\varphi=1}^{n} T_\varphi}$

In this formula, T_φ is the duration of phase φ.

Multiphase Markovian processes generally reach equilibrium when the probability to jump out off a given state is equal to the probability to enter in it. The asymptotic values have nothing to do with those which are described above but they can be used in the above formula.

In conclusion, it can be said that the Markovian approach provides a lot of possibilities to calculate the *PFH* of an E/E/PE safety-related system working in continuous mode. Nevertheless, a good understanding of the underlying mathematics is needed to use them properly.

B.5.3 Petri nets and Monte Carlo simulation approach

B.5.3.1 Principle of modelling

An efficient way of modelling dynamic systems is to build a finite state automaton behaving as close as possible as the E/E/PE safety-related systems under study. Petri nets (see B.2.3.3 and B.6.6.10 of IEC 61508-7) have been proven to be very efficient for this purpose for the following reasons:

– they are easy to handle graphically;

– the size of the models increases linearly according to the number of components to be modelled;

– they are very flexible and allow modelling almost all type of constraints;

– they are a perfect support for Monte Carlo simulation (see B.6.6.8 of IEC 61508-7).

Originally developed in the 1960's for the formal proof on automata, they have been quickly hijacked in two steps by reliability engineers, in the seventies, for the automatic building of big Markov graphs and in the 1980's, for Monte Carlo simulation purpose.

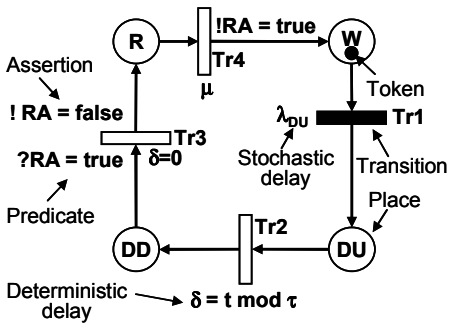


Figure B.33 – Petri net for modelling a single periodically tested component

The typical sub-Petri net for a simple periodically tested component is made of three parts:

1) the static part (i.e. a drawing):
   a) places (circles) corresponding to potential states;
   b) transitions (rectangles) corresponding to potential events;
   c) upstream arrows (from places to transitions) validating transitions;
   d) downstream arrows (from transitions to places) indicating what happens when transitions are fired.
2) the scheduling part:
   a) stochastic delays representing random delays elapsing before events occur;
   b) deterministic delays representing known delays elapsing before events occur.
3) the dynamic part:
   a) tokens (small black circles) moving when events occur to indicate which of the potential states are actually achieved;
   b) predicates (any formula which may be true or false) validating transitions;
   c) assertions (any equation) updating some variables when a transition is fired.

In addition some rules allow validating and firing a transition:

4) validation of a transition (i.e. conditions for the corresponding event may arise):
   a) all upstream places have at least one token;
   b) all predicates must be "true".
5) firing of a transition(i.e. what happens when the corresponding event is arising):
   a) one token is removed from upstream places;
   b) one token is added in downstream places;
   c) assertions are updated.

Most of the notions in relationship with Petri nets are introduced above and the remaining ones will be introduced when needed.

B.5.3.2 Monte Carlo simulation principle

Monte Carlo simulation consists of the animation of behavioural models by using random numbers to evaluate how many times the system remains in states governed either by random or deterministic delays (see also B.6.6.8 of IEC 61508-7).

This can be explained by using the Petri net presented on Figure B.33:

– At the beginning the token is in place *W* and the component is in good working order.
– Only one event may arise from this state - a dangerous undetected failure- (transition *Tr1* is valid and drawn in black).
– The time spent in this state is stochastic and governed by an exponential distribution of parameter λ_{DU}. The Monte Carlo simulation consists of firing a random number (see below) to calculate the delay *d1* before the failure is going to occur (i.e. *Tr1* is going to be fired).
– When *d1* is elapsed, *Tr1* is fired and the token moves to place *DU* (more precisely one token is removed from place *W* and one token is added in place *DU*).
– The component reaches the dangerous undetected state and the transition *Tr2* becomes valid.
– The detection of the dangerous failure occurs after a deterministic delay *d2* (*d*2 = *t modulo* τ when *t* is the current time and τ the test interval). This simulates the test interval.

– When *t*2 is elapsed, i.e. the dangerous failure is detected, the token enters in place *DD.* The component is now waiting to be repaired and *Tr*3 becomes valid.

– The delay *d*3 for firing *Tr*3 (start of repair) does not depend on the component itself but on the availability of the repair resources represented by the message *RA*. This is governed by events arising in another part of the whole Petri net not represented on Figure B.33.

– The repair starts as soon as the repair team becomes available (i.e. the predicate *?RA = true* becomes true) and the token enters in place *R*. Repair resources become immediately unavailable for another intervention and the assertion *!RA = false* is used to update the value of *RA*. This prevents any other repair at the same time.

– The stochastic transition *Tr*4 (i.e. the end of the repair) becomes valid and the delay *d4* may be calculated by firing a random number according to the repair rate μ.

– When *d*4 is elapsed, *Tr*4 is fired and the component is coming back in its good working state (the token enters in place *W*). The repair resources are again available and *RA* is updated through the assertion *!RA = true*.

– And so on... as long as the firing of next valid transition belongs to the period under interest *[0, T]*.

When the next firing is no longer inside *[0, T]*, the simulation is stopped and one history of the component is achieved. All along the progress of the history, relevant parameters may be recorded as the mean marking of the places (i.e. the ratio of the time with one token in the place over the duration T), the transition firing frequencies, the time to the first occurrence of a given event, etc.

The principle of Monte Carlo simulation is to realize a great number of such histories and to perform classical statistics on the results in order to assess the relevant parameters.

Contrary to analytical calculations, Monte Carlo simulation allows to mix easily deterministic and random delays which may be simulated from their cumulated probability distribution *F(d)* and random numbers z_i uniformly distributed over *[0, 1]*. Those random numbers are available in almost any programming languages and powerful algorithms are available to do that.

Then, a sample (d_i), distributed according to *F(d),* is obtained from a sample (z_i) by the operation: $d_i = F^{-1}(z_i)$. This is very easy when the analytical form of $F^{-1}(z)$ is available as, for example, exponentially distributed delays: $d_i = -\frac{1}{\lambda_{DU}} Log(z_i)$.

In respect of the accuracy of the calculations, concerning a given simulated parameter *X*, the basic statistics allow the calculation of the average, variance, standard deviation and confidence of the sample (X_i) which has been simulated:

– average : $\overline{X} = \frac{\sum_i x_i}{N}$

– variance: $\sigma^2 = \frac{\sum_i (x_i - \overline{X})^2}{N}$ and standard deviation : σ

– 90 % confidence interval around $\overline{X}$: $Conf = 1{,}64 \frac{\sigma}{\sqrt{N}}$

Therefore, when using Monte Carlo simulation, the accuracy of the results can always be estimated. For example, considering 90 % of chance that the true result $\hat{X}$ belongs to the interval $\left[\overline{X} - 1{,}64\sigma / \sqrt{N},\quad \overline{X} + 1{,}64\sigma / \sqrt{N}\right]$.

This interval is reducing when the number of histories increases and when the frequency of occurrence of *X* increases.

With present time personal computers, there are no real difficulties to achieve calculations even for SIL 4 E/E/PE safety-related systems.

B.5.3.3 Principle of *PFD* calculations

The sub-Petri-net on Figure B.33 may be used directly to evaluate the PFD_{avg} of the component because the mean marking of place *W* which is equal to the ratio of the time spent in *W* (i.e. with *W* marked by the token) to the duration *T*, is in fact the average availability *A* of the component. We have $PFD_{avg} = 1 - A$.

The accuracy of the calculation may be estimated as explained above.

More complex behaviours may be represented by using specific sub-Petri Nets. Figure B.34 gives an idea of what can be done for modelling periodically tested components, common cause failures (CCF) and repair resources.

On the left hand side is modelled a periodically tested component which jumps across the states working (*W*), dangerously failed undetected (*DU*), under testing (DUT), dangerously failed detected (*DD*), ready for repair (*RR*) and under repair (*R*).

When it fails (*DU*), the message *!-Ci* (equivalent to *!Ci = false*) is emitted to inform that the component is failed. Then it waits until a periodical test is started (DUT). The periodic test interval is τ and the staggering θ. After what the test is performed for a duration equal to π and the state *DD* is reached. If a spare part is available (at least one token in *SP*), the component becomes ready for repair (*RR*) and the variable *NbR* is increased by 1 to inform the repair resources of the number of components needing to be repaired. When the repair resources are on location (one token in *OL*) the repair starts (*R*) and the token is removed from *OL*. When it is achieved the component comes back in good functioning state, the message *!Ci* is emitted (i.e. !*Ci* = *true*), *NbR* is decreased by 1 and the token is given back in *OL* to allow further repairs. And so on.

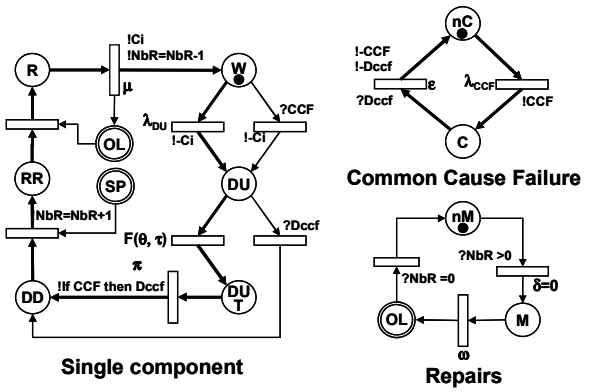


Figure B.34 – Petri net to model common cause failure and repair resources

The variable *NbR* is used by the sub-Petri net devoted to repairs. When it becomes positive, the mobilisation of the resources is started (*M*) and after a certain delay they are ready to work on location (*OL*). The token in *OL* is used to validate the starting of the repair of one failed component. Therefore, only one repair may be done at the same time. When all repairs have been performed (i.e. *NbR* = 0) the repair resources are demobilized.

On Figure B.34 a common cause failure (CCF) has also been modelled. When it occurs (λ_{DCC}), the message *!CCF* becomes true and is used to put all the affected components in their *DU* states. The relevant messages C_i become false and the components are repaired

independently from each other. When the test of a component is finished, the assertion *!IF CCF then Dccf* allows to inform all the other components that a CCF has been detected. This message is used to put them immediately in their *DD* states. This message is also used to reset the CCF sub-Petri net but this is done after a while (ε) to insure that all components have been put in their *DD* state before resetting.

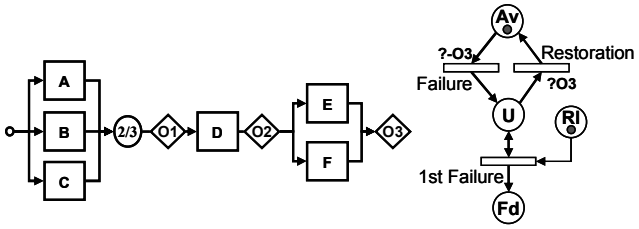


Figure B.35 – Using reliability block diagrams to build Petri net and auxiliary Petri net for *PFD* and *PFH* calculations

The sub-Petri Nets on Figure B.34 are intended to be used as parts of more complex models. One way to use them is illustrated on Figure B.35 where the reliability block diagram of Figure B.16 which has been slightly adapted by introducing the intermediary outputs *Oi*.

The components A, B, C, D, E, F may be modelled by a set of sub- Petri nets like those on Figure B.34 with for example a CCF for (A, B, C) and another one for (E, F) and the same repair resources for all components. The remaining problem is only to link the component together according to the logic of the reliability block diagram and to calculate the PFD_{avg} under interest.

Linking the components is very easy by using the messages C_i and building the following assertions:

– $O_1 = C_a.C_b + C_a.C_c + C_b.C_c$
– $O_2 = O_1.C_d$
– $O_3 = O_2.(C_e + C_f)$

Therefore when O_3 is true, the whole E/E/PE safety-related system is working well and it is unavailable otherwise. This message is used in the sub-Petri net on the right hand side in order to model the various states of the E/E/PE safety-related systems: available (*Av*), unavailable (*U*), reliable(*Rl*) and unreliable (*Fd*).

For *PFD* calculation the focus is only on *Av* and *U*: when O_3 becomes false, the system fails and becomes unavailable and when O_3 becomes true, the system is restored and becomes available again. This is very simple and the mean marking of *Av* is the average availability of the system and the mean marking of *U* its average unavailability, i.e. its PFD_{avg}.

Therefore, the Monte Carlo simulation performs automatically the integral of the instantaneous unavailability and it is not necessary to calculate it except if the saw tooth curve is wanted. This would be easily done by evaluating the mean marking of *U* at given instant rather over the whole *[0, T]* period.

What is exposed above only intends to illustrate the broad lines of using Petri nets for SIL calculations purposes but the potential modelling possibilities are virtually endless.

B.5.3.4 Principle of *PFH* calculations

For the *PFH* calculation, the principles remain exactly the same as above and the same sub models can be used for DU failures. Figure B.36 illustrates a sub-Petri net modelling a DD failure which is revealed and repaired as soon as it has been detected.

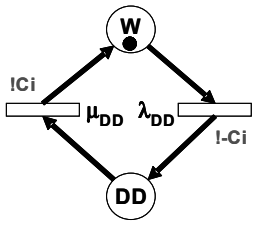


Figure B.36 – Simple Petri net for a single component with revealed failures and repairs

Such components' models may be used as explained above in relationship with a reliability block diagram representing the whole system like on Figure B.35.

When the E/E/PE safety-related system working in continuous mode is the ultimate safety barrier, an accident occurs as soon as it is failing and the *PFH* evaluation must be achieved through the reliability of the system. This is done by the lower part of the sub PN presented on the right hand side of Figure B.35: the average frequency of the first failure of the system over *[0, T]* is its unreliability *F(T)*. Then, when *F(T)* is small compared to 1, and according to the *PFH* definition, we have *PFH = F(T)/T*.

Due to the token in *RI*, the first failure is a one shot transition. Provided that all histories lead to a failure (i.e. T is long enough) the mean time spent with a token in *RI* is the *MTTF* of the system. Then *PFH* ≈ 1/*MTTF* provides an upper bound of *PFH*.

When the E/E/PE safety-related system working in continuous mode is not the ultimate safety barrier it does not provoke directly an accident when it fails. Then it is repaired after an overall failure and its *PFH* shall be calculated through the unavailability of the system. It is obtained directly from the frequency *Nbf* of the transition failure. This provides the number of times the system has failed over a given period and therefore we have *PFH(T) = Nbf/T*.

It is interesting to note that when *T* is long enough, the *MUT* may be calculated through the mean cumulated time MCT_{Av} in state *Av* and the *MDT* through the mean cumulated time MCT_U in state *U*. The mean cumulated times MCT_A and MCT_U are very easy to calculate within the Monte Carlo simulation just by cumulating the times when a token is present in *Av* or *U*. We have $MUT = MCT_A / Nbf$ and $MUT = MCT_u / Nbf$. This may be used for evaluating *PFH* = 1/*(MUT + MDT)* = 1/*MTBF = Nbf/T.*

All these results are obtained directly because the Monte Carlo simulation provides naturally the average values. What is exposed above only intends to illustrate the broad lines of using Petri nets for SIL calculations purposes but the potential modelling possibilities are virtually endless.

B.5.4 Other approaches

The relationship between the size of the models and the number of components of the system under study varies dramatically according to the type of approach in use. It is linear for fault trees and Petri nets but exponential for Markov processes. Therefore fault tree and Petri net approaches make it potentially easier to handle much larger systems than Markovian approach. This is why Petri nets are sometimes used to produce large Markov graphs.

The underlying formal languages behind the graphical representations described here above produce flat models: each component is described separately, at the same level. This makes large models sometimes hard to master and to maintain. A way to overcome this problem is to use structured languages providing compact hierarchical models. Several such formal languages have been developed recently and some software packages are available. As an example, we can consider, the AltaRica Data Flow language published in 2 000 in order to be freely used by the reliability community and designed to accurately model the functional and dysfunctional properties of industrial systems (see references in B.7).

Figure B.37 is equivalent to the reliability block diagram presented Figure B.1. This model is hierarchical because single modules are modelled only once and then reused as many times as needed at the system level (i.e. in the node main). This allows achieving very compact models.

In order to simplify the presentation, only two transitions have been represented for the components: failure and repair (i.e. DD failures revealed and repaired as soon as they arise).

Logical operators (*or*, *and*) are used to describe the logic of the system. This is done in direct relationship with the reliability block diagram and the flow Out models the state of the system: it is working when *Out* is *true* and failed when *Out* is *false*.

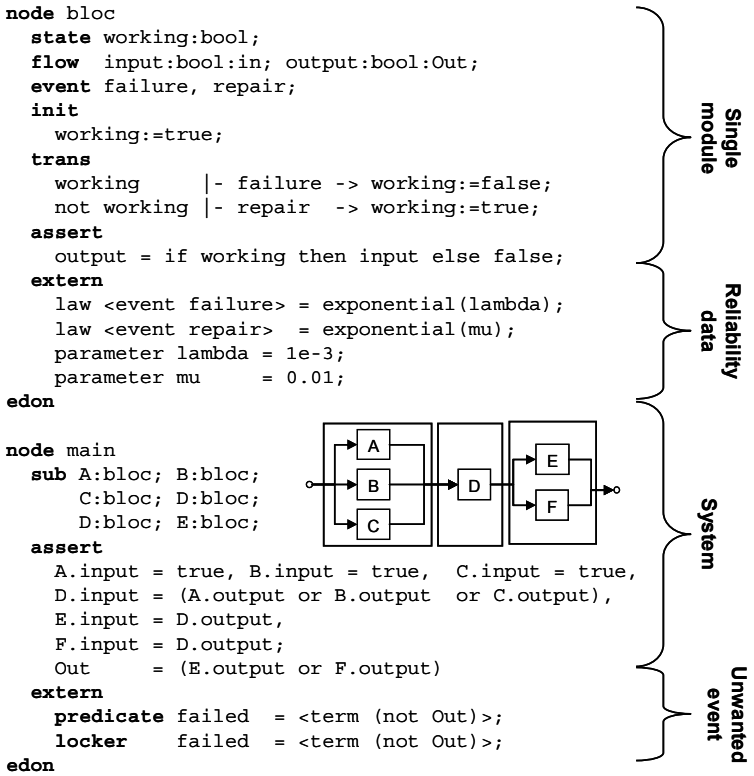


Figure B.37 – Example of functional and dysfunctional modelling with a formal language

This provides good behavioural models for efficient Monte Carlo simulation and all what has been described above for Petri net remains valid here for PFD_{avg} or *PFH* calculations. Therefore we are not developing this part further.

This formal language possesses similar mathematical properties as Petri nets and therefore it is possible to compile one model towards the other without difficulty. But it also generalizes the properties of the underlying languages behind Fault trees or Markov processes. Therefore, providing that the description has been restricted to the properties of Markov processes or Fault trees, it is possible to compile the model into equivalent Markov graph or fault trees. It is the purpose of the key words “predicate” and “locker” at the end of the model to provide directives for fault trees or Markov model generation or for direct Monte Carlo simulation.

Using a formal language designed to model properly the system behaviour both from functional and dysfunctional points of view allows:

– Monte Carlo simulations to be performed directly on the models;

– Markov graphs to be generated and analytical calculations to be performed as previously shown (when the language has been restricted to Markov properties);

– Equivalent Fault trees to be generated and analytical calculations to be performed as previously shown (when the language has been restricted to Boolean properties).

Such functional and dysfunctional formal languages are general purpose languages. There are no difficulties to use them in the particular case of E/E/PE safety-related systems. They provide an efficient way to master PFD_{avg} and *PFH* calculations of E/E/PE safety-related systems when several protection layers, various types of failure modes, complex proof test patterns, components dependencies, maintenance resources, etc., i.e. when the other methods have shown that they have reached their limits.

B.6 Handling uncertainties

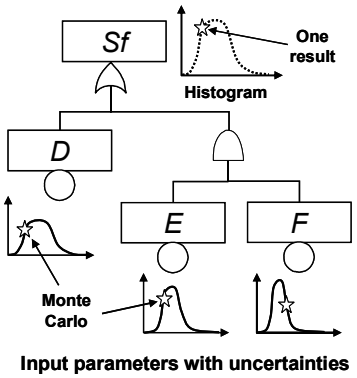


Figure B.38 – Uncertainty propagation principle

One of the main problems encountered when performing probabilistic calculations is linked to the uncertainties on the reliability parameters. Therefore, it is useful when performing *PFD* or *PFH* calculations to evaluate what the corresponding impact is on the uncertainty of the results.

Care needs to be taken when dealing with this issue and using Monte Carlo simulation provides an efficient way for this purpose as it is illustrated on Figure B.38.

On this figure the input reliability parameters (e.g. the dangerous undetected failure rates) are no longer certain and they are replaced by random variables. The density of probability of such random variable is more or less sharp or flat according to the degree of uncertainty: the probability density of F is sharper than the one of E or D. This means that there is less uncertainty on F than on E or D for example.

The principle of calculation is the following:

1) generating one set of input parameters by using random numbers according to the probabilistic distributions of those parameters (similar to what has been explained in B.3.2) ;
2) performing one calculation by using the above generated set of input parameters;
3) recording the output result (this constitutes one value used in step 4);
4) performing steps 1 to 3 again and again until a sufficient number of values (for example 100 or 1 000) is obtained in order to constitute an histogram (dotted line on Figure B.38);
5) analysing statistically the histogram to obtain the average and the standard deviation of the output result.

The average of this histogram is the PFD_{avg} or the *PFH* according to the type of calculation performed and the standard deviation measures the uncertainty on this results. The smaller the standard deviation, the more accurate is the PFD_{avg} or the *PFH* calculation.

The principle illustrated here on fault tree is very general and may be applied on any of the calculation methods depicted in this annex: simplified formulae, Markov processes and even Petri nets or formal languages approaches. When the calculation is already performed by Monte Carlo simulation, a two step Monte Carlo simulation shall be used.

The probabilistic distribution for a given input reliability parameter shall be chosen according to the knowledge gathered about it. This may be:

– a uniform distribution between lower and upper bounds;

– a triangular distribution with a most probable value;

– a log-normal law with a given error factor;

– a Chi square law, etc.

The first ones may be assessed by engineering judgement when there is not very much field feedback available. When there is more field feedback available, the last ones may be used because the field feedback provides average parameter values as well as confidence intervals on these average values.

For example if n failures have been observed over a cumulated observation time T, we have:

– $\hat{\lambda} = n/T$ is the maximum likelihood estimator of the failure rate

– $\lambda_{Inf,\alpha} = \frac{1}{2T}\chi^2_{(1-\alpha),2n}$ lower bound with a probability α % to be lower than $\lambda_{\text{Inf},\alpha}$

– $\lambda_{Sup,\alpha} = \frac{1}{2T}\chi^2_{\alpha,2(n+1)}$ upper bound with a probability α % to be higher than $\lambda_{\text{Sup},\alpha}$

When α = 5 %, then the true value of λ has 90 chances over 100 to belong to the interval $[\lambda_{\text{Inf},\alpha}, \lambda_{\text{Sup},\alpha}]$. The smaller this interval, the more accurate the value of λ. Normally, good reliability data bases provide this information. Analysts should consider reliability data provided without confidence intervals (or information allowing to calculate them) very cautiously.

$\hat{\lambda}$, $\lambda_{\text{Inf},\alpha}$ and $\lambda_{\text{Sup},\alpha}$ can be used to build a relevant distribution to model the failure rate λ of a given failure mode and its uncertainties. This is clear for χ^2 distribution but the log-normal law has also shown this to be very efficient for that purpose. Such a log-normal law is defined by its average $\hat{\lambda}$ or its median $\lambda_{50\ \%}$ and its so-called error factor.

This law has a very interesting property: $\lambda_{\text{Inf},\alpha} = \lambda_{50\ \%}/ef_\alpha$ and $\lambda_{\text{Sup},\alpha} = \lambda_{50\ \%}.ef_\alpha$.

Then it is defined with only two parameters: $\hat{\lambda}$ *and* $ef_\alpha \approx \sqrt{\frac{\lambda_{Sup,\alpha}}{\lambda_{Inf,\alpha}}}$

When ef_α = 1 there is no uncertainties, when ef_α = 3,3 there is a factor of about 10 between the lower and upper confidence bounds, etc.

These laws may be used in turn with Monte Carlo simulations in order to take under consideration both the impacts of average values and uncertainties on PFD_{avg} or *PFH*. Therefore it is always possible to master the uncertainties within probabilistic calculations. Some software packages implement directly such calculations.

When analysing redundant systems the analysis should not only consider the uncertainty of the basic element failure rate but also the accuracy of the CCF failure rate. Even if there is good field feed back data for each of the elements there is rarely good CCF field data and hence this will be the most uncertain.

B.7 References

References [4] to [9] and [22] to [24] in the Bibliography give further details on evaluating probabilities of failure.

Annex C
(informative)

Calculation of diagnostic coverage and safe failure fraction – worked example

A method for calculating diagnostic coverage and safe failure fraction is given in Annex C of IEC 61508-2. This annex briefly describes the use of this method to calculate the diagnostic coverage. It is assumed that all of the information specified in IEC 61508-2 is available and has been used where required in obtaining the values shown in Table C.1. Table C.2 gives limitations on diagnostic coverage that can be claimed for certain E/E/PE safety-related elements. The values in Table C.2 are based on engineering judgement.

To understand all the values in Table C.1, a detailed hardware schematic would be required, from which the effect of all failure modes could be determined. These values are only examples, for instance some components in Table C.1 assume no diagnostic coverage because it is practically impossible to detect all failure modes of all components.

Table C.1 has been derived as follows:

a) A failure mode and effect analysis has been carried out to determine the effect of each failure mode for every component on the behaviour of the system without diagnostic tests. The fractions of the overall failure rate associated with each failure mode are shown for each component, divided between safe (S) and dangerous (D) failures. The division between safe and dangerous failures may be deterministic for simple components but is otherwise based on engineering judgement. For complex components, where a detailed analysis of each failure mode is not possible, a division of failures into 50 % safe, 50 % dangerous is generally accepted. For this table, the failure modes given in reference a) have been used, although other divisions between failure modes are possible and may be preferable.

b) The diagnostic coverage for each specific diagnostic test on each component is given (in the column labelled "DC_{comp}"). Specific diagnostic coverages are given for the detection of both safe and dangerous failures. Although open-circuit or short-circuit failures for simple components (for example resistors, capacitors and transistors) are shown to be detected with a specific diagnostic coverage of 100 %, the use of table C.2 has limited the diagnostic coverage with respect to item U16, a complex type B component, to 90 %.

c) Columns (1) and (2) give the safe and dangerous failure rates, in the absence of diagnostic tests, for each component (λ_S and $\lambda_{DD} + \lambda_{Du}$ respectively).

d) We can consider a detected dangerous failure to be effectively a safe failure, so we now find the division between effectively safe failures (i.e. either detected safe, undetected safe or detected dangerous failures) and undetected dangerous failures. The effective safe failure rate is found by multiplying the dangerous failure rate by the specific diagnostic coverage for dangerous failures and adding the result to the safe failure rate (see column (3)). Likewise, the undetected dangerous failure rate is found by subtracting the specific diagnostic coverage for dangerous failures from one and multiplying the result by the dangerous failure rate (see column (4)).

e) Column (5) gives the detected safe failure rate and column (6) gives the detected dangerous failure rate, found by multiplying the specific diagnostic coverage by the safe and dangerous failure rates respectively.

f) The table yields the following results:

   – total safe failure rate $\sum \lambda_S + \sum \lambda_{Dd} = 9{,}9 \times 10^{-7}$
     (including detected dangerous failures)

   – total undetected dangerous failure rate $\sum \lambda_{Du} = 5{,}1 \times 10^{-8}$

- total failure rate $\sum \lambda_S + \sum \lambda_{Dd} + \sum \lambda_{Du}$ = 1,0 × 10^{-6}
- total undetected safe failure rate $\sum \lambda_{Su}$ = 2,7 × 10^{-8}
- diagnostic coverage for safe failures $\frac{\sum \lambda_{Sd}}{\sum \lambda_S} = \frac{3{,}38}{3{,}65}$ = 93 %
- diagnostic coverage for dangerous failures
  $\frac{\sum \lambda_{Dd}}{\sum \lambda_{Dd} + \sum \lambda_{Du}} = \frac{6{,}21}{6{,}72}$ = 92 % (normally termed simply “diagnostic coverage”)
- safe failure fraction $\frac{\sum \lambda_S + \sum \lambda_{Dd}}{\sum \lambda_S + \sum \lambda_{Dd} + \sum \lambda_{Du}} = \frac{986}{365 + 672}$ = 95 %

g) The division of the failure rate without diagnostic tests is 35 % safe failures and 65 % dangerous failures.

Table C.1 – Example calculations for diagnostic coverage and safe failure fraction

Item	No	Type	Division of safe and dangerous failures for each failure mode								Division of safe and dangerous failures for diagnostic coverage and calculated failure rates ($\times 10^{-9}$)						
			OC		SC		Drift		Function		DC_{comp}		(1)	(2)	(3)	(4)	(5)	(6)
			S	D	S	D	S	D	S	D	S	D	λ_S	$\lambda_{DD}+\lambda_{Du}$	$\lambda_S+\lambda_{DD}$	λ_{DU}	λ_{SD}	λ_{DD}
Print	1	Print	0,5	0,5	0,5	0,5	0	0	0	0	0,99	0,99	11,0	11,0	21,9	0,1	10,9	10,9
CN1	1	Con96pin	0,5	0,5	0,5	0,5					0,99	0,99	11,5	11,5	22,9	0,1	11,4	11,4
C1	1	100nF	1	0	1	0	0	0	0	0	1	0	3,2	0,0	3,2	0,0	3,2	0,0
C2	1	10µF	0	0	1	0	0	0	0	0	1	0	0,8	0,0	0,8	0,0	0,8	0,0
R4	1	1M	0,5	0,5	0,5	0,5					1	1	1,7	1,7	3,3	0,0	1,7	1,7
R6	1	100k									0	0	0,0	0,0	0,0	0,0	0,0	0,0
OSC1	1	OSC24 MHz	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	1	1	16,0	16,0	32,0	0,0	16,0	16,0
U8	1	74HCT85	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,99	0,99	22,8	22,8	45,4	0,2	22,6	22,6
U16	1	MC68000-12	0	1	0	1	0,5	0,5	0,5	0,5	0,90	0,90	260,4	483,6	695,6	48,4	234,4	435,2
U26	1	74HCT74	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,99	0,99	22,8	22,8	45,4	0,2	22,6	22,6
U27	1	74F74	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,99	0,99	14,4	14,4	28,7	0,1	14,3	14,3
U28	1	PAL16L8A	0	1	0	1	0	1	0	1	0,98	0,98	0,0	88,0	86,2	1,8	0,0	86,2
T1	1	BC817	0	0	0	0,67	0	0,5	0	0	1	1	0,0	0,2	0,4	0,0	0,0	0,2
Total													365	672	986	50,9	338	621

NOTE None of the failure modes of item R6 are detected, but a failure does not affect either safety or availability.

Key

S	Safe failure
D	Dangerous failure
OC	Open circuit
SC	Short circuit
Drift	Change of value
Function	Functional failures
DC_{comp}	Specific diagnostic coverage for the component

See also Table B.1, although in this table failure rates are for the individual components in question rather than every component in a channel.

Table C.2 – Diagnostic coverage and effectiveness for different elements

Component	Low diagnostic coverage	Medium diagnostic coverage	High diagnostic coverage
CPU (see Note 3)	total less than 70 %	total less than 90 %	
register, internal RAM	50 % - 70 %	85 % - 90 %	99 % - 99,99 %
coding and execution including flag register	50 % - 60 %	75 % - 95 %	-
(see Note 3)	50 % - 70 %	85 % - 98 %	-
address calculation (see Note 3)	50 % - 60 %	60 % - 90 %	85 % - 98 %
program counter, stack pointer	50 % - 70 %		
	40 % - 60 %		
Bus			
memory management unit	50 %	70 %	90 % - 99 %
bus-arbitration	50 %	70 %	90 % - 99 %
Interrupt handling	40 % - 60 %	60 % - 90 %	85 % - 98 %
Clock (quartz) (see Note 4)	50 %	-	95 % - 99 %
Program flow monitoring			
temporal (see Note 3)	40 % - 60 %	60 % - 80 %	-
logical (see Note 3)	40 % - 60 %	60 % - 90 %	-
temporal and logical (see Note 5)	-	65 % - 90 %	90 % - 98 %
Invariable memory	50 % - 70 %	99 %	99,99 %
Variable memory	50 % - 70 %	85 % - 90 %	99 % - 99,99 %
Discrete hardware			
digital I/O	70 %	90 %	99 %
analogue I/O	50 % - 60 %	70 % - 85 %	99 %
power supply	50 % - 60 %	70 % - 85 %	99 %
Communication and mass storage	90 %	99,9 %	99,99 %
Electromechanical devices	90 %	99 %	99,9 %
Sensors	50 % - 70 %	70 % - 85 %	99 %
Final elements	50 % - 70 %	70 % - 85 %	99 %

NOTE 1 This table should be read in conjunction with Table A.1 of IEC 61508-2 which provides the failure modes to be considered.

NOTE 2 When a range is given for diagnostic coverage, the upper interval boundaries may be set only for narrowly tolerated monitoring means, or for test measures that stress the function to be tested in a highly dynamic manner.

NOTE 3 For techniques where there is no high diagnostic coverage figure, at present no measures and techniques of high effectiveness are known.

NOTE 4 At present no measures and techniques of medium effectiveness are known for quartz clocks.

NOTE 5 The minimum diagnostic coverage for a combination of temporal and logical program flow monitoring is medium.

See references [10] to [12] in the Bibliography.

Annex D
(informative)

A methodology for quantifying the effect of hardware-related common cause failures in E/E/PE systems

D.1 General

D.1.1 Introduction

This standard incorporates a number of measures which deal with systematic failures. However, no matter how well these measures are applied, there is a residual probability of systematic failures occurring. Although this does not significantly affect the reliability calculations for single-channel systems, the potential for failures which may affect more than one channel in a multi-channel system (or several components in a redundant safety system), i.e. common cause failures, results in substantial errors when reliability calculations are applied to multi-channel or redundant systems.

This informative annex describes two methodologies which allow common cause failures to be taken into account in the safety assessment of multi-channel or redundant E/E/PE systems. Using these methodologies gives a more accurate estimation of the integrity of such a system than ignoring the potential for common cause failures.

The first methodology is used to calculate a value for β, factor frequently used in the modelling of common cause failures. This can be used to estimate the rate of common cause failures applicable to two or more systems operating in parallel from the random hardware failure rate of one of those systems (see D.5). It is generally accepted that the random hardware failure figures that are collected will include a number of failures that were caused by systematic failures.

Alternative methodologies may be preferred in some cases, for example, where a more accurate β-factor can be proven as a result of the availability of data on common cause failures or when the number of impacted elements is higher than four. The second methodology, i.e. the binomial failure rate (also called shock model) method, can be used

D.1.2 Brief overview

The failures of a system are considered to arise from two dissimilar sources:

– random hardware failures; and

– systematic failures.

The former are assumed to occur randomly in time for any component and to result in a failure of a channel within a system of which the component forms part when the latter appears immediately and in a deterministic way when the system reach the situation for which the underlying systematic error is existing.

There is a finite probability that independent random hardware failures could occur in all channels of a multi-channel system so that all of the channels were simultaneously in a failed state. Because random hardware failures are assumed to occur randomly with time, the probability of such failures concurrently affecting parallel channels is low compared to the probability of a single channel failing. This probability can be calculated using well-established techniques but the result may be very optimistic when the failures are not fully independent from each other.

Dependent failures are traditionally split between (see reference [18] in the Bibliography):

– Common Cause Failure (CCF) causing multiple failures from a single shared cause. The multiple failures may occur simultaneous or over a period of time;

– Common Mode Failures (CMF) which are a particular case of CCF in which multiple equipment items fail in the same mode;

– Cascade failures which are propagating failures.

The term CCF is often used to cover all kind of dependant failures as it is done in this annex. They are also split between

– Dependent failures due to clear deterministic causes;

– Residual potential multiple failure events not explicitly considered in the analysis because of not enough accuracy, no clear deterministic causes or impossibility to gather reliability data.

The first one should be analysed, modelled and quantified in a conventional way and only the second ones should be handled as shown in this informative Annex D. Nevertheless, the systematic failures -which are perfect dependent failures not identified during the safety analysis (otherwise they would have been removed)- are handled in particular manner in this standard and this annex applies mainly for hardware random dependent failures.

Therefore, common cause failures which result from a single cause, may affect more than one channel or more than one component. These may result from a systematic fault (for example, a design or specification mistake) or an external stress leading to an early random hardware failure (for example, an excessive temperature resulting from the random hardware failure of a common cooling fan, which accelerates the life of the components or takes them outside their specified operating environment) or, possibly, a combination of both. Because common cause failures are likely to affect more than one channel in a multi-channel system, the probability of common cause failure is likely to be the dominant factor in determining the overall probability of failure of a multi-channel system and if this is not taken into account a realistic estimate of the safety integrity level of the combined system is unlikely to be obtained.

D.1.3 Defence against common cause failures

Although common cause failures result from a single cause, they do not all manifest themselves necessarily simultaneously in all channels. For example, if a cooling fan fails, all of the channels of a multi-channel E/E/PE system could fail, leading to a common cause failure. However, all of the channels are unlikely to warm at the same rate or to have the same critical temperature. Therefore, failures occur at different times in the different channels.

The architecture of programmable systems allows them to carry out internal diagnostic testing functions during their on-line operation. These can be employed in a number of ways, for example

– a single channel PE system can continuously be checking its internal operation together with the functionality of the input and output devices. If designed from the outset, a test coverage in the region of 99 % is achievable (see [13] in the Bibliography). If 99 % of internal faults are revealed before they can result in a failure, the probability of single-channel faults which can ultimately contribute to common cause failures is significantly reduced.

– in addition to internal testing, each channel in a PE system can monitor the outputs of other channels in a multi-channel PE system (or each PE device can monitor another PE device in a multi-PE system). Therefore, if a failure occurs in one channel, this can be detected and a safe shut-down initiated by the one or more remaining channels that have not failed and are executing the cross-monitoring test. (It should be noted that cross-monitoring is effective only when the state of the control system is continuously changing, for example the interlock of a frequently used guard in a cyclic machine, or when brief changes can be introduced without affecting the controlled function.) This cross-

monitoring can be carried out at a high rate, so that, just before a non-simultaneous common cause failure, a cross-monitoring test is likely to detect the failure of the first channel to fail and is able to put the system into a safe state before a second channel is affected.

In the case of the cooling fan example, the rate of temperature rise and the susceptibility of each channel are slightly different, resulting in the second channel failing possibly several tens of minutes after the first. This allows the diagnostic testing to initiate a safe shutdown before the second channel succumbs to the common cause fault.

As a result of the above

– PE-based systems have the potential to incorporate defences against common cause failures and, therefore, be less susceptible to them when compared to other technologies;

– a different β-factor may be applicable to PE-based systems when compared to other technologies. Therefore, β-factor estimates based on historic data are likely to be invalid. (None of the existing investigated models used for estimating the probability of common cause failure allow for the effect of automatic cross-monitoring.)

– because common cause failures that are distributed in time may be revealed by the diagnostic tests before they affect all channels, such failures may not be recognized or reported as being common cause failures.

There are three avenues that can be taken to reduce the probability of potentially dangerous common cause failures.

a) Reduce the number of random hardware and systematic failures overall. (This reduces the areas of the ellipses in Figure D.1 leading to a reduction in the area of overlap.)

b) Maximize the independence of the channels (separation and diversity). (This reduces the amount of overlap between the ellipses in Figure D.1 whilst maintaining their area.)

c) Reveal non-simultaneous common cause failures while only one, and before a second, channel has been affected, i.e. use diagnostic tests or proof test staggering.

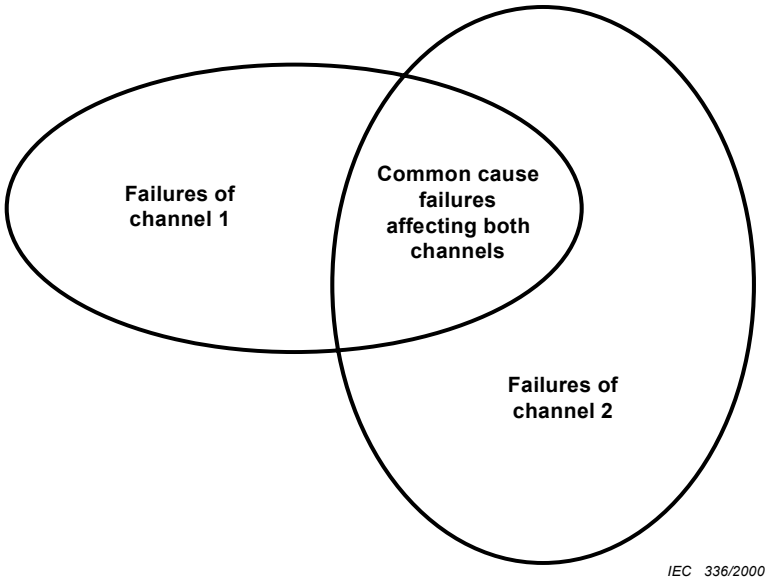


Figure D.1 – Relationship of common cause failures to the failures of individual channels

With systems with more than two channels, the common cause failure may affect all channels or only multiple channels but not all depending on the source of the common mode. Thus the approach taken in this annex, according to the first method, is to calculate the β value based on a duplex system voting 1oo2 and then use a multiplying factor to the derived β depending on the total number of channels and the voting requirements. (see Table D.5).

D.1.4 Approach adopted in the IEC 61508 series

The IEC 61508 series is based on these three avenues and requires a threefold approach:

a) Apply the techniques specified in IEC 61508-2/3 to reduce the overall probability of systematic failure to a level commensurate with the probability of random hardware failure.

b) Quantify those factors that can be quantified, i.e. take into account the probability of random hardware failure, as specified in IEC 61508-2.

c) Derive, by what is considered at the present time to be the best practicable means, a factor relating the probability of common cause failure of the hardware to the probability of random hardware failure. The methodology described in this annex relates to the derivation of this factor.

Most methodologies for estimating the probability of common cause failures attempt to make their predictions from the probability of random hardware failure. Clearly, the justification for any direct relationship between these probabilities is tenuous, nevertheless, such a correlation has been found in practice and probably results from second-order effects. For example, the higher the probability of random hardware failure of a system

– the higher the amount of maintenance required by the system. The probability of a systematic fault being introduced during maintenance depends on the number of times maintenance is carried out, and this also affects the rate of human errors leading to common cause failures. This leads to a relationship between the probability of random hardware failure and the probability of common cause failure. For example:
  - a repair, followed by testing and, possibly, recalibration is required each time a random hardware failure occurs;
  - for a given safety integrity level, a system with a higher probability of random hardware failure requires proof tests to be carried out more frequently and with greater depth/complexity, leading to additional human interference.
– the more complex the system. The probability of random hardware failure depends on the number of components, and, hence, the complexity of a system. A complex system is less easily understood, so is more prone to the introduction of systematic faults. In addition, the complexity makes it difficult to detect the faults, by either analysis or test, and can lead to parts of the logic of a system not being exercised except in infrequent circumstances. Again, this leads to a relationship between the probability of random hardware failure and the probability of common cause failure.

Several approaches are currently in use to handle CCF (β factor, multiple Greek letters, α factor, binomial failure rate ...) [20]. Two of the current models are proposed in this informative annex for the third part of the threefold approach already described. Despite the limitations, it is believed that they represent the best way forward at the present time to handle the probability of common cause failure:

– the well-established β-factor model which is widely used and realistic to deal with multi-channel system typically up to four dependent elements.
– the binomial failure rate [21] (also known as the shock model) which can be used when the number of dependent elements is greater than four.

The following two difficulties are faced when using the β-factor or the shock models on a E/E/PE system.

– What value should be chosen for the parameters? Many sources (for example reference [13]) suggest ranges within which the value of the β-factor is likely to occur but no actual value is given, leaving the user to make a subjective choice. To overcome this problem, the β-factor methodology in this annex is based on the system originally described in reference [14] and recently redefined in reference [15].

– Neither the β-factor nor the shock models take into account the sophisticated diagnostic testing capabilities of modern PE systems, which can be used to detect a non-simultaneous common cause failure before it has had sufficient time to manifest itself fully. To overcome this deficiency, the approach described in references [14] and [15] has been modified to reflect the effect of diagnostic tests in the estimation of the likely value of β.

The diagnostic testing functions running within a PE system are continuously comparing the operation of the PE system with predefined states. These states can be predefined in software or in hardware (for example, by a watchdog timer). Looked on in this way, the diagnostic testing functions may be thought of as an additional, and partially diverse, channel running in parallel with the PE system.

Cross-monitoring between channels also can be carried out. For many years, this technique has been used in dual-channel interlocking systems based solely on relays. However, with relay technology, it is usually possible to carry out the cross-checks only when the channels change state, making such tests inappropriate for revealing non-simultaneous common cause failures where systems remain in the same (for example, ON) state for long periods. With PE system technology, cross-monitoring may be carried out with a high repetition frequency.

D.2 Scope of the methodology

The scope of the methodology is limited to common cause failures within hardware. The reasons for this include the following:

– the β-factor and shock models relate the probability of common cause failure to the probability of random hardware failure. The probability of common cause failures which involve the system as a whole depends on the complexity of the system (possibly dominated by the user software) and not on the hardware alone. Clearly, any calculations based on the probability of random hardware failure cannot take into account the complexity of the software;

– reporting of common cause failures is generally limited to hardware failures, the area of most concern to the manufacturers of the hardware;

– it is not considered practicable to model systematic failures (for example software failures);

– the measures specified in IEC 61508-3 are intended to reduce the probability of software-related common cause failure to an acceptable level for the target safety integrity level.

Therefore, the estimate of the probability of common cause failure derived by this methodology relates to only those failures associated with the hardware. It should NOT be assumed that the methodology can be used to obtain an overall failure rate which takes the probability of software-related failure into account.

D.3 Points taken into account in the methodology

Because sensors, logic subsystem and final elements are subject to, for example, different environmental conditions and diagnostic tests with varying levels of capability, the methodology should be applied to each of these subsystems separately. For example, the logic subsystem is more likely to be in a controlled environment, whereas the sensors may be mounted outside on pipework that is exposed to the elements.

Programmable electronic channels have the potential for carrying out sophisticated diagnostic testing functions. These can

– have a high diagnostic coverage within the channels;
– monitor additional redundancy channels;
– have a high repetition rate; and
– in an increasing number of cases, also monitor sensors and/or final elements.

A large fraction of common cause failures do not occur concurrently in all of the affected channels. Therefore, if the repetition frequency of the diagnostic tests is sufficiently high, a large fraction of common cause failures can be revealed and, hence, avoided before they affect all available channels.

Not all features of a multi-channel system, that has a bearing on its immunity to common cause failures, can be evaluated by diagnostic tests. However, those features relating to diversity or independence are made more effective. Any feature which is likely to increase the time between channel failures in a non-simultaneous common cause failure (or reduce the fraction of simultaneous common cause failures) increases the probability of the diagnostic tests detecting the failure and putting the plant into a safe state. Therefore, the features relating to immunity to common cause failures are divided into those whose effect is thought to be increased by the use of diagnostic tests and those whose effect is not. This leads to the two columns, X and Y respectively, in Table D.1.

Although, for a three-channel system, the probability of common cause failures which affect all three channels is likely to be slightly lower than the probability of failures which affect two channels, it is assumed, in order to simplify the β-factor methodology, that the probability is independent of the number of affected channels, i.e. it is assumed that if a common cause failure occurs it affects all channels. An alternative method is the shock model.

There is no known data on hardware-related common cause failures available for the calibration of the methodology. Therefore, the tables in this annex are based on engineering judgement.

Diagnostic test routines are sometimes not regarded as having a direct safety role so may not receive the same level of quality assurance as the routines providing the main control functions. The methodology was developed on the presumption that the diagnostic tests have an integrity commensurate with the target safety integrity level. Therefore, any software-based diagnostic test routines should be developed using techniques appropriate to the target safety integrity level.

D.4 Using the β-factor to calculate the probability of failure in an E/E/PE safety-related system due to common cause failures

Consider the effect of common cause failures on a multi-channel system with diagnostic tests running within each of its channels.

Using the β-factor model, the common cause dangerous failure rate is:

$$\lambda_D \beta$$

where

λ_D is the random hardware dangerous failure rate for each individual channel and β is the β-factor in the absence of diagnostic tests, i.e. the fraction of single-channel failures that affect all channels.

We now assume that common cause failures affect all channels, and that the span of time between the first channel and all channels being affected is small compared to the time interval between successive common cause failures.

Suppose that there are diagnostic tests running in each channel which detect and reveal a fraction of the failures. We can divide all failures into two categories: those that lie outside the coverage of the diagnostic tests (and so can never be detected) and those that lie within the coverage (so would eventually be detected by the diagnostic tests).

The overall failure rate due to dangerous common cause failures is then given by:

$$\lambda_{DU}\beta + \lambda_{DD}\beta_D$$

where

- λ_{DU} is the undetected dangerous failure rate of a single channel, i.e. the failure rate of the failures which lie outside the coverage of the diagnostic tests; clearly, any reduction in the β-factor resulting from the repetition rate of the diagnostic tests cannot affect this fraction of the failures;
- β is the common cause failure factor for undetectable dangerous faults, which is equal to the overall β-factor that would be applicable in the absence of diagnostic testing;
- λ_{DD} is the detected dangerous failure rate of a single channel, i.e. the failure rate of the failures of a single channel that lie within the coverage of the diagnostic tests. If the repetition rate of the diagnostic tests is high, a fraction of the failures are revealed leading to a reduction in the value of β, i.e. β_D;
- β_D is the common cause failure factor for detectable dangerous faults. As the repetition rate of the diagnostic testing is increased, the value of β_D falls increasingly below β;
- β is obtained from Table D.5 which uses the results of D.4, using a score, $S = X + Y$ (see D.5);
- β_D is obtained from Table D.5 which uses the results of D.4, using a score, $S_D = X(Z+1)+Y$.

D.5 Using the tables to estimate β

The β-factor should be calculated for the sensors, the logic subsystem and the final elements separately.

In order to minimize the probability of occurrence of common cause failures, one should first establish which measures lead to an efficient defence against their occurrence. The implementation of the appropriate measures in the system lead to a reduction in the value of the β-factor used in estimating the probability of failure due to common cause failures.

Table D.1 lists the measures and contains associated values, based on engineering judgement, which represent the contribution each measure makes in the reduction of common cause failures. Because sensors and final elements are treated differently to the programmable electronics, separate columns are used in the table for scoring the programmable electronics and the sensors or final elements.

Extensive diagnostic tests may be incorporated into programmable electronic systems which allow the detection of non-simultaneous common cause failures. To allow diagnostic tests to be taken into account in the estimation of the β-factor, the overall contribution of each measure in Table D.1 is divided, using engineering judgement, into two sets of values, X and Y. For each measure, the X:Y ratio represents the extent to which the measure's contribution against common clause failures can be improved by diagnostic testing.

The user of Table D.1 should ascertain which measures apply to the system in question, and sum the corresponding values shown in each of columns X_{LS} and Y_{LS} for the logic subsystem,

or X_{SF} and Y_{SF} for the sensors or final elements, the sums being referred to as X and Y, respectively.

Tables D.2 and D.3 may be used to determine a factor Z from the frequency and coverage of the diagnostic tests, taking into account the important Note 4 which limits when a non-zero value of Z should be used. The score S is then calculated using the following equations, as appropriate (see previous clause):

– $S = X + Y$ to obtain the value of β_{int} (the β-factor for undetected failures); and

– $S_D = X(Z + 1) + Y$ to obtain the value of $\beta_{D\ int}$ (the β-factor for detected failures).

Here S or S_D is a score which is used in Table D.4 to determine the appropriate β_{int} -factor.

β_{int} and $\beta_{D\ int}$ are the values of the common cause failure prior to considering the effect of different degrees of redundancy.

Table D.1 – Scoring programmable electronics or sensors/final elements

Item	Logic subsystem		Sensors and final elements	
	X_{LS}	Y_{LS}	X_{SF}	Y_{SF}
Separation/segregation				
Are all signal cables for the channels routed separately at all positions?	1,5	1,5	1,0	2,0
Are the logic subsystem channels on separate printed-circuit boards?	3,0	1,0		
Are the logic subsystems physically separated in an effective manner? For example, in separate cabinets.	2,5	0,5		
If the sensors/final elements have dedicated control electronics, is the electronics for each channel on separate printed-circuit boards?			2,5	1,5
If the sensors/final elements have dedicated control electronics, is the electronics for each channel indoors and in separate cabinets?			2,5	0,5
Diversity/redundancy				
Do the channels employ different electrical technologies for example, one electronic or programmable electronic and the other relay?	8,0			
Do the channels employ different electronic technologies for example, one electronic, the other programmable electronic?	6,0			
Do the devices employ different physical principles for the sensing elements for example, pressure and temperature, vane anemometer and Doppler transducer, etc?			9,0	
Do the devices employ different electrical principles/designs for example, digital and analogue, different manufacturer (not re-badged) or different technology?			6,5	
Is low diversity used, for example hardware diagnostic tests using the same technology?	2,0	1,0		
Is medium diversity used, for example hardware diagnostic tests using different technology?	3,0	2,0		
Were the channels designed by different designers with no communication between them during the design activities?	1,5	1,5		
Are separate test methods and people used for each channel during commissioning?	1,0	0,5	1,0	2,0
Is maintenance on each channel carried out by different people at different times?	3,0		3,0	
Complexity/design/application/maturity/experience				
Does cross-connection between channels preclude the exchange of any information other than that used for diagnostic testing or voting purposes?	0,5	0,5	0,5	0,5
Is the design based on techniques used in equipment that has been used successfully in the field for > 5 years?	0,5	1,0	1,0	1,0
Is there more than 5 years experience with the same hardware used in similar environments?	1,0	1,5	1,5	1,5
Is the system simple, for example no more than 10 inputs or outputs per channel?		1,0		
Are inputs and outputs protected from potential levels of over-voltage and over-current?	1,5	0,5	1,5	0,5
Are all devices/components conservatively rated (for example, by a factor of 2 or more)?	2,0		2,0	
Assessment/analysis and feedback of data				
Have the results of the failure modes and effects analysis or fault-tree analysis been examined to establish sources of common cause failure and have predetermined sources of common cause failure been eliminated by design?		3,0		3,0
Were common cause failures considered in design reviews with the results fed back into the design? (Documentary evidence of the design review activity is required.)		3,0		3,0
Are all field failures fully analyzed with feedback into the design? (Documentary evidence of the procedure is required.)	0,5	3,5	0,5	3,5
Procedures/human interface				
Is there a written system of work to ensure that all component failures (or degradations) are detected, the root causes established and other similar items inspected for similar potential causes of failure?		1,5	0,5	1,5
Are procedures in place to ensure that: maintenance (including adjustment or calibration) of any part of the independent channels is staggered, and, in addition to the manual checks carried out following maintenance, the diagnostic tests are allowed to run satisfactorily between the completion of maintenance on one channel and the start of maintenance on another?	1,5	0,5	2,0	1,0
Do the documented maintenance procedures specify that all parts of redundant systems (for example, cables, etc.) intended to be independent of each other, are not to be relocated?	0,5	0,5	0,5	0,5
Is all maintenance of printed-circuit boards, etc. carried out off-site at a qualified repair centre and have all the repaired items gone through a full pre-installation testing?	0,5	1,0	0,5	1,5
Does the system have low diagnostic coverage (60 % to 90 %) and report failures to the level of a field-replaceable module?	0,5			
Does the system have medium diagnostics coverage (90 % to 99 %) and report failures to the level of a field-replaceable module?	1,5	1,0		
Does the system have high diagnostics coverage (>99 %) and report failures to the level of a field-replaceable module?	2,5	1,5		
Do the system diagnostic tests report failures to the level of a field-replaceable module?			1,0	1,0

Table D.1 *(continued)*

Item	Logic subsystem		Sensors and final elements	
	X_{LS}	Y_{LS}	X_{SF}	Y_{SF}
Competence/training/safety culture				
Have designers been trained (with training documentation) to understand the causes and consequences of common cause failures?	2,0	3,0	2,0	3,0
Have maintainers been trained (with training documentation) to understand the causes and consequences of common cause failures?	0,5	4,5	0,5	4,5
Environmental control				
Is personnel access limited (for example locked cabinets, inaccessible position)?	0,5	2,5	0,5	2,5
Is the system likely to operate always within the range of temperature, humidity, corrosion, dust, vibration, etc., over which it has been tested, without the use of external environmental control?	3,0	1,0	3,0	1,0
Are all signal and power cables separate at all positions?	2,0	1,0	2,0	1,0
Environmental testing				
Has the system been tested for immunity to all relevant environmental influences (for example EMC, temperature, vibration, shock, humidity) to an appropriate level as specified in recognized standards?	10,0	10,0	10,0	10,0

NOTE 1 A number of the items relate to the operation of the system, which may be difficult to predict at design time. In these cases, the designers should make reasonable assumptions and subsequently ensure that the eventual user of the system is made aware of, for example, the procedures to be put in place in order to achieve the designed level of safety integrity. This could be by including the necessary information in the accompanying documentation.

NOTE 2 The values in the *X* and *Y* columns are based on engineering judgement and take into account the indirect as well as the direct effects of the items in column 1. For example, the use of field-replaceable modules leads to

– repairs being carried out by the manufacturer under controlled conditions instead of (possibly incorrect) repairs being made under less appropriate conditions in the field. This leads to a contribution in the *Y* column because the potential for systematic (and, hence, common cause) failures is reduced;

– a reduction in the need for on-site manual interaction and the ability quickly to replace faulty modules, possibly on-line, so increasing the efficacy of the diagnostics for identifying failures before they become common-cause failures. This leads to a strong entry in the *X* column.

Table D.2 – Value of *Z* – programmable electronics

Diagnostic coverage	Diagnostic test interval		
	Less than 1 min	Between 1 min and 5 min	Greater than 5 min
≥ 99 %	2,0	1,0	0
≥ 90 %	1,5	0,5	0
≥ 60 %	1,0	0	0

Table D.3 – Value of *Z* – sensors or final elements

Diagnostic coverage	Diagnostic test interval			
	Less than 2 h	Between 2 h and two days	Between two days and one week	Greater than one week
≥ 99 %	2,0	1,5	1,0	0
≥ 90 %	1,5	1,0	0,5	0
≥ 60 %	1,0	0,5	0	0

NOTE 1 The methodology is most effective if account is taken uniformly across the list of the categories in Table D.1. Therefore, it is strongly recommended that the total score in the *X* and *Y* columns for each category should be not less than the total score in the *X* and *Y* columns divided by 20. For example, if the total score ($X + Y$) is 80, none of the categories (for example, procedures/human interface) should have a total score ($X + Y$) of less than four.

NOTE 2 When using Table D.1, take account of the scores for all items that apply. The scoring has been designed to allow for items which are not mutually exclusive. For example, a system with logic subsystem channels in separate racks is entitled to both the score for "Are the logic subsystem channels in separate cabinets?" and that for "Are the logic subsystem channels on separate printed-circuit boards?".

NOTE 3 If sensors or final elements are PE-based, they should be treated as part of the logic subsystem if they are enclosed within the same building (or vehicle) as the device that constitutes the major part of the logic subsystem, and as sensors or final elements if they are not so enclosed.

NOTE 4 For a non-zero value of *Z* to be used, it should be ensured that the equipment under control is put into a safe state before a non-simultaneous common cause failure can affect all the channels. The time taken to assure this safe state should be less than the claimed diagnostic test interval. A non-zero value for *Z* can be used only if:

– the system initiates an automatic shut-down on detection of a fault; or

– a safe shut-down is not initiated after a first fault [9], but the diagnostic tests:

  – determine the locality of the fault and are capable of localizing the fault; and

  – continue to be capable of placing the EUC in a safe state after the detection of any subsequent faults; or

– a formal system of work is in place to ensure that the cause of any revealed fault is fully investigated within the claimed diagnostic test interval; and

  – if the fault has the potential for leading to a common cause failure, the plant is immediately shut-down; or

  – the faulty channel is repaired within the claimed diagnostic test interval.

NOTE 5 In the process industries, it is unlikely to be feasible to shut down the EUC when a fault is detected within the diagnostic test interval as described in Table D.2. This methodology should not be interpreted as a requirement for process plants to be shut down when such faults are detected. However, if a shut-down is not implemented, no reduction in the β-factor can be gained by the use of diagnostic tests for the programmable electronics. In some industries, a shut-down may be feasible within the described time. In these cases, a non-zero value of *Z* may be used.

NOTE 6 Where diagnostic tests are carried out in a modular way, the repetition time used in Tables D.2 or D.3 is the time between the successive completions of the full set of diagnostic testing modules. The diagnostic coverage is the total coverage provided by all of the modules.

Table D.4 – Calculation of β_{int} or $\beta_{D\ int}$

Score (*S* or S_D)	Corresponding value of β_{int} or $\beta_{D\ int}$ for the:	
	Logic subsystem	**Sensors or final elements**
120 or above	0,5 %	1 %
70 to 120	1 %	2 %
45 to 70	2 %	5 %
Less than 45	5 %	10 %
NOTE 1 The maximum levels of $\beta_{D\ int}$ shown in this table are lower than would normally be used, reflecting the use of the techniques specified elsewhere in this standard for the reduction in the probability of systematic failures as a whole, and of common cause failures as a result of this.		
NOTE 2 Values of $\beta_{D\ int}$ lower than 0,5 % for the logic subsystem and 1 % for the sensors would be difficult to justify.		

The β_{int} derived from Table D.4 is the common cause failure associated with a 1oo2 system. For other levels of redundancy (MooN) this β_{int} value will change as given in Table D.5 to yield the final value of β.

Table D.5 can also be used to determine the final value of β_D but where there is β_{int} this can be substituted for $\beta_{D\ int}$.

NOTE 7 For related relevant information (on PDS method) see [25] in Bibliography

9) The operation of the system on the identification of a fault should be taken into account. For example, a simple 2oo3 system should be shut down (or repaired) within the times quoted in Tables D.2 or D.3, following the identification of a single failure. If this is not done, a failure of a second channel could result in the two failed channels outvoting the remaining (good) channel. A system which automatically reconfigures itself to 1oo2 voting when one channel fails, and which automatically shuts down on the occurrence of a second failure, has an increased probability of revealing the fault in the second channel and so a non-zero value for *Z* may be claimed.

Table D.5 – Calculation of β for systems with levels of redundancy greater than 1oo2

MooN		N			
		2	3	4	5
M	1	β_{int}	0,5 β_{int}	0,3 β_{int}	0,2 β_{int}
	2	-	1,5 β_{int}	0,6 β_{int}	0,4 β_{int}
	3	-	-	1,75 β_{int}	0,8 β_{int}
	4	-	-	-	2 β_{int}

D.6 Examples of the use of the β-factor methodology

In order to demonstrate the effect of using the β-factor methodology, some simple examples have been worked through in Table D.6 for the programmable electronics.

For categories not relating to diversity nor redundancy, typical values for X and Y were used. These were obtained by halving the maximum score for the category.

In the diverse system examples, the values for the diversity/redundancy category are derived from the following properties considered in Table D.1:

– one system is electronic, the other uses relay technology;
– the hardware diagnostic tests use different technologies;
– the different designers did not communicate during the design process;
– different test methods and test personnel were used to commission the systems; and
– maintenance is carried out by different people at different times.

In the redundancy system examples, the values for the diversity/redundancy category are derived from the property that the hardware diagnostics are carried out by an independent system, which uses the same technology as the redundancy systems.

For both the diverse and redundancy systems, a maximum and minimum value was used for *Z*, leading to four example systems in total.

Table D.6 – Example values for programmable electronics

Category		Diverse system with good diagnostic testing	Diverse system with poor diagnostic testing	Non Diverse system with good diagnostic testing	Non Diverse system with poor diagnostic testing
Separation/segregation	*X*	3,50	3,50	3,50	3,50
	Y	1,50	1,50	1,50	1,50
Diversity/redundancy	*X*	14,50	14,50	2,00	2,00
	Y	3,00	3,00	1,00	1,00
Complexity/design/.....	*X*	2,75	2,75	2,75	2,75
	Y	2,25	2,25	2,25	2,25
Assessment/analysis/....	*X*	0,25	0,25	0,25	0,25
	Y	4,75	4,75	4,75	4,75
Procedures/human interface	*X*	3,50	3,50	3,50	3,50
	Y	3,00	3,00	3,00	3,00
Competence/training/...	*X*	1,25	1,25	1,25	1,25
	Y	3,75	3,75	3,75	3,75
Environmental control	*X*	2,75	2,75	2,75	2,75
	Y	2,25	2,25	2,25	2,25
Environmental test	*X*	5,00	5,00	5,00	5,00
	Y	5,00	5,00	5,00	5,00
Diagnostic coverage	*Z*	2,00	0,00	2,00	0,00
Total *X*		33,5	33,5	21	21
Total *Y*		25,5	25,5	23,5	23,5
Score *S*		59	59	44,5	44,5
β		2 %	2 %	5 %	5 %
Score S_D		126	59	86,5	44,5
β_D		0,5 %	2 %	1 %	5 %
Diverse system 1002 (Table D.5) Non Diverse system is triplex with 2003 voting (Table D.5)		0,5 %	2 %	1,5 %	7,5 %

D.7 Binomial failure rate (Shock model) – CCF approach

The common cause failure (CCF) field feedback shows that if numerous double failures, a few triple failures and perhaps one quadruple failure have been observed, no multiple failures beyond order four have ever been observed from an explicit single cause which could not have been identified during the safety analysis. Consequently, the probability of multiple dependent failures decreases when the order of the CCF increases. Therefore, if the β-factor model is realistic for double failures and slightly pessimistic for triple failures it becomes much too conservative for quadruple failures and beyond. Let us consider the typical example of a safety instrumented system which closes the n production wells (e.g. n=150) of an oil field when a blocked outlet is occurring. Of course 2, 3 or even 4 wells may fail to close due to a non explicit CCF but not the n wells as modelled by the β-factor (otherwise the CCF would be explicit and should be analysed as an individual failure). Another typical example occurs when dealing with several safety layers at the same time. Considering, for example, the potential CCF between sensors of two protection layers may imply to consider the CCFs between six sensors (i.e. 3 sensors for each layer).

Several models have been proposed [18] to deal with this difficulty but most of them require so many reliability parameters (e.g. multiple Greek letters or α-models) that they become unrealistic. Among them, the binomial failure rate (Shock model) introduced by Vesely in 1977 and improved by Atwood in 1986 provides a pragmatic solution [18, 19]. The principle is that when a CCF occurs, it is similar to a shock on the related components. This shock may be lethal (i.e. same impact as in the β-factor model) or non-lethal and in this case there is only a certain probability that a given component fails due to the shock. Then the probability of having *k* failures due to the non-lethal shock is binomially distributed.

This model needs only 3 parameters to be implemented:

– ω lethal shock rate;
– ρ non-lethal shock rate;
– γ conditional probability of failure of a component given a non-lethal shock.

Figure D.2 gives an example on implementing this method when using a fault tree.

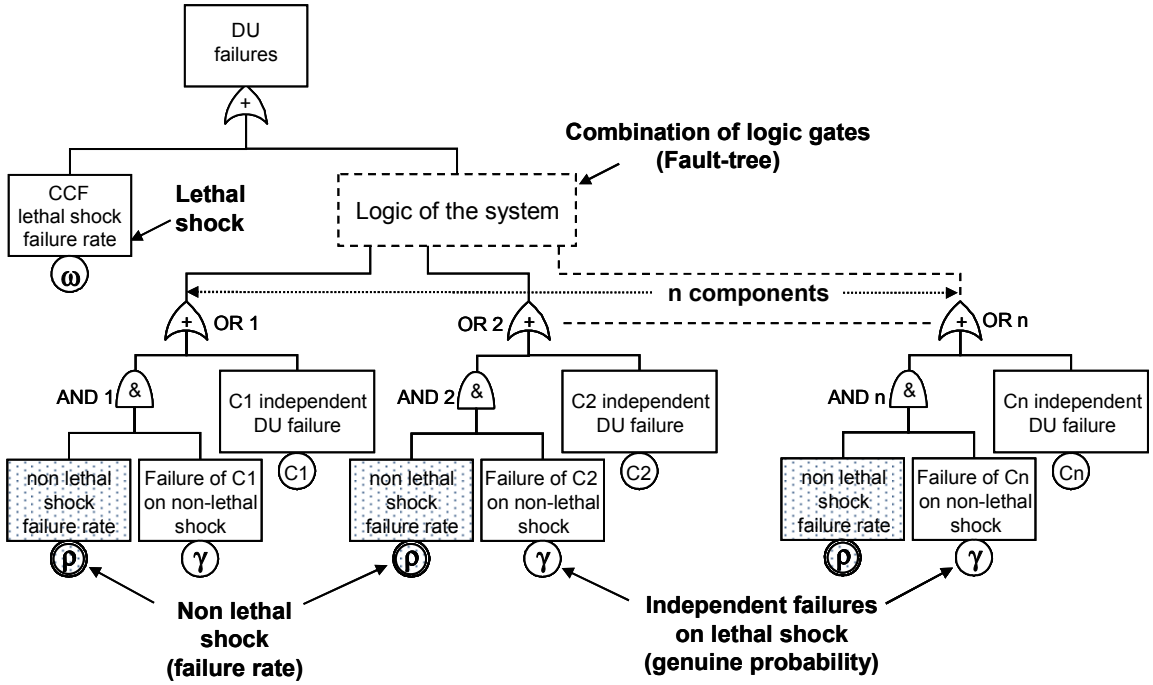


Figure D.2 – Implementing shock model with fault trees

Identical components can be linked to the *β*-factor model by splitting β in two parts β_L and β_{NL}:

– $\beta = \beta_L + \beta_{NL}$
– failure rate due to lethal shock: $\lambda_{DU} \times \beta_L$
– failure rate due to non lethal shock: $\lambda_{DU} \times \beta_{NL}$
– independent failure rate: $\lambda_{DU}[1 - (\beta_L + \beta_{NL})]$

In the fault tree represented in Figure D.2, this becomes:

– lethal shock rate: $\omega = \lambda_{DU} \times \beta_L$
– non-lethal shock rate : $\rho = \lambda_{DU} \times \beta_{NL} / \gamma$

As usual, the main problem is to evaluate the values of the three parameters (ω, ρ, γ) or (β_L, β_{NL}, $\tilde{\gamma}$) Reference [19] gives some indications and provides other references about the statistical treatments allowing to evaluate(ω, ρ, γ) from field feedback.

If no data are available the engineering judgement can be used through pragmatic approaches. For example the following procedure may be used with fault tree modelling when there are more than 3 similar items:

1) estimate β as in the β-factor method;
2) consider that β_L is negligible ($\beta_{NL} = \beta$);

3) estimate γ in order to be sure to obtain conservative results. Considered that double failures have at least an impact 10 times higher than the quadruple failure (hypothesis which is certainly conservative), the following formula may be used:

$$\gamma = \sqrt{\frac{C_N^2}{10.C_N^4}}$$

where

N is the number of similar items;

C_N^2 is the number of potential double failures; and

C_N^4 is the number of potential quadruple failures

4) calculate ρ in function of the number N of similar items:

$$\rho = \frac{\beta\lambda_{DU}}{C_N^2\gamma^2 + C_N^3\gamma^3}$$

In this method of working, the top contributors are double and triple failures and the results are conservative compared to the results obtained with the β factor method with only 3 components. The CCF double and triple failures are taken under consideration properly and unrealistic multiple failures are not completely neglected.

This model is very easily implemented into fault tree calculation models like those presented in Annex B, e.g. fault trees in B.4.3. This allows handling safety systems comprising a lot of similar components in a very simple and easy way.

D.8 References

References [13] to [15] and [20] and [21] in the Bibliography provide useful information relating to common cause failures.

Annex E
(informative)

Example applications of software safety integrity tables of IEC 61508-3

E.1 General

This annex gives two worked examples in the application of the software safety integrity tables specified in Annex A of IEC 61508-3:

a) safety integrity level 2: a programmable electronic safety-related system required for a process within a chemical plant;

b) safety integrity level 3: a shut-down application based on a high-level language.

These examples illustrate how software development techniques might be selected in particular circumstances from the tables of Annexes A and B of IEC 61508-3.

It should be emphasized that these illustrations are not definitive applications of the standard to these examples. IEC 61508-3 states clearly at several points that given the large number of factors that affect software systematic capability, it is not possible to give an algorithm for combining the techniques and measures that will be correct for any given application.

For a real system, all the entries in the tables should be supported by documented justification that the comments made are correct and that they represent an appropriate response for the particular system and application. This justification is likely to be assisted by referring to the guidance of IEC 61508-3 Annex C which discusses the desirable properties which, if achieved in the appropriate lifecycle phase, may convincingly justify confidence that the eventual software has sufficient systematic safety integrity.

E.2 Example for safety integrity level 2

This example is a safety integrity level 2 programmable electronic safety-related system required for a process within a chemical plant. The programmable electronic safety-related system utilizes ladder logic for the application program, and is an illustration of limited variability language application programming.

The application consists of several reactor vessels linked by intermediate storage vessels which are filled with inert gas at certain points in the reaction cycle to suppress ignition and explosions. The programmable electronic safety-related system functions include: receiving inputs from the sensors; energizing and interlocking the valves, pumps and actuators; detecting dangerous situations and activating the alarm; interfacing to a distributed control system, as required by the safety requirements specification.

Assumptions:

– the programmable electronic safety-related system controller is a PLC;

– the hazard and risk analysis has established that a programmable electronic safety-related system is required, and that safety integrity level 2 is required in this application (by the application of IEC 61508-1 and IEC 61508-2);

– although the controller operates in real time, only a relatively slow response is needed;

– there are interfaces to a human operator and to a distributed control system;

– the source code of the system software and the design of the programmable electronics of the PLC is not available for examination, but has been qualified against IEC 61508-3 to safety integrity level 2;

– the language used for application programming is ladder logic, produced using the PLC supplier's development system;

– the application code is required to run on only a single type of PLC;

– the whole of the software development was reviewed by a person independent of the software team;

– a person independent of the software team witnessed and approved the validation testing;

– modifications (if needed) require authorization by a person independent of the software team.

NOTE 1 For the definition of an independent person, see IEC 61508-4.

NOTE 2 See the notes to 7.4.2, 7.4.3, 7.4.4 and 7.4.5 of IEC 61508-3 for information on the division of responsibility between the PLC supplier and user when limited variability programming is used.

The following tables show how Annex A of IEC 61508-3 may be interpreted for this application.

Table E.1 – Software safety requirements specification

(See 7.2 of IEC 61508-3)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1a	Semi-formal methods	Table B.7	R	Cause-effect diagrams, sequence diagrams, function blocks. Typically used for PLC application software requirements specification
1b	Formal methods	B.2.2, C.2.4	R	Not used for limited variability programming
2	Forward traceability between the system safety requirements and the software safety requirements	C.2.11	R	Check completeness: review to ensure that all system safety requirements are addressed by software safety requirements
3	Backward traceability between the safety requirements and the perceived safety needs	C.2.11	R	Minimise complexity and functionality: review to ensure that all software safety requirements are actually needed to address system safety requirements
4	Computer-aided specification tools to support appropriate techniques/measures above	B.2.4	R	Development tools supplied by the PLC manufacturer

NOTE 1 In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.

NOTE 2 The software safety requirements were specified in natural language.

Table E.2 – Software design and development – software architecture design

(see 7.4.3 of IEC 61508-3)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1	Fault detection	C.3.1	R	Checking of data range, watch-dog timer, I/O, communication. Raise an alarm if errors (see 3a)
2	Error detecting codes	C.3.2	R	Embedded with user options - careful selection required
3a	Failure assertion programming	C.3.3	R	Dedicate some PLC program ladder logic to test certain essential safety conditions (see 1)
3b	Diverse monitor techniques (with independence between the monitor and the monitored function in the same computer)	C.3.4	R	Not preferred: increased software complexity to guarantee independence.
3c	Diverse monitor techniques (with separation between the monitor computer and the monitored computer)	C.3.4	R	Check legal I/O combinations in an independent hardware safety monitor
3d	Diverse redundancy, implementing the same software safety requirements specification	C.3.5	---	Not preferred: insufficient increased safety benefit over 3c.
3e	Functionally diverse redundancy, implementing different software safety requirements specification	C.3.5	---	Not preferred: substantially achieved by 3c.
3f	Backward recovery	C.3.6	R	Embedded with user options – careful selection required
3g	Stateless software design (or limited state design)	C.2.12	---	Not used. Process control needs states to memorise plant condition.
4a	Re-try fault recovery mechanisms	C.3.7	R	Used as required by the application (see 2 and 3c)
4b	Graceful degradation	C.3.8	R	Not used for limited variability programming
5	Artificial intelligence - fault correction	C.3.9	NR	Not used for limited variability programming
6	Dynamic reconfiguration	C.3.10	NR	Not used for limited variability programming
7	Modular approach	Table B.9	HR	
8	Use of trusted/verified software elements (if available)	C.2.10	HR	Pre-existing code from earlier projects
9	Forward traceability between the software safety requirements specification and software architecture	C.2.11	R	Check completeness: review to ensure that all software safety requirements are addressed by the software architecture
10	Backward traceability between the software safety requirements specification and software architecture	C.2.11	R	Minimise complexity and functionality: review to ensure that all architecture safety requirements are actually needed to address software safety requirements
11a	Structured diagrammatic methods	C.2.1	HR	Data flow methods and data logic tables may be used for representing at least the design architecture
11b	Semi-formal methods	Table B.7	R	May be used for DCS interface
11c	Formal design and refinement methods	B.2.2, C.2.4	R	Rarely used for limited variability programming
11d	Automatic software generation	C.4.6	R	Not used for limited variability programming
12	Computer-aided specification and design tools	B.2.4	R	Development tools supplied by the PLC manufacturer

Technique/Measure		Ref.	SIL 2	Interpretation in this application
13a	Cyclic behaviour, with guaranteed maximum cycle time	C.3.11	HR	Not used. Hardware monitoring of PLC cycle time.
13b	Time-triggered architecture	C.3.11	HR	Not used. Hardware monitoring of PLC cycle time.
13c	Event-driven, with guaranteed maximum response time	C.3.11	HR	Not used. Hardware monitoring of PLC cycle time.
14	Static resource allocation	C.2.6.3	R	Not used. Dynamic resources issues do not arise in limited variability programming
15	Static synchronisation of access to shared resources	C.2.6.3	---	Not used. Dynamic resources issues do not arise in limited variability programming
NOTE 1 In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B. NOTE 2 It is impractical to implement some of the above techniques in limited variability programming.				

Table E.3 – Software design and development – support tools and programming language

(See 7.4.4 of IEC 61508-3)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1	Suitable programming language	C.4.5	HR	Usually ladder, and often the proprietary variety of the PLC supplier
2	Strongly typed programming language	C.4.1	HR	Not used. Use PLC-oriented structured text (see [16] in the Bibliography)
3	Language subset	C.4.2	---	Beware of complex "macro" instructions, interrupts which alter PLC scan cycle, etc.
4a	Certified tools and certified translators	C.4.3	HR	Available from some PLC manufacturers
4b	Tools and translators: increased confidence from use	C.4.4	HR	PLC supplier's development kit; in-house tools developed over several projects
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.4 – Software design and development – detailed design

(See 7.4.5 and 7.4.6 of IEC 61508-3)
(Includes software system design, software module design and coding)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1a	Structured methods	C.2.1	HR	Not used for limited variability programming
1b	Semi-formal methods	Table B.7	HR	Cause-effect diagrams, sequence diagrams, function blocks. Typical for limited variability programming
1c	Formal design and refinement methods	B.2.2, C.2.4	R	Not used for limited variability programming
2	Computer-aided design tools	B.3.5	R	Development tools supplied by the PLC manufacturer
3	Defensive programming	C.2.5	R	Included in the system software
4	Modular approach	Table B.9	HR	Order and group the PLC program ladder logic to maximize its modularity with respect to the functions required
5	Design and coding standards	C.2.6 Table B.1	HR	In-house conventions for documentation and maintainability
6	Structured programming	C.2.7	HR	Similar to modularity in this context
7	Use of trusted/verified software elements (if available)	C.2.10	HR	Function blocks, part programs
8	Forward traceability between the software safety requirements specification and software design	C.2.11	R	Check completeness: review to ensure that all software safety requirements are addressed by the software design

NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.

Table E.5 – Software design and development – software module testing and integration

(See 7.4.7 and 7.4.8 of IEC 61508-3)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1	Probabilistic testing	C.5.1	R	Not used for limited variability programming
2	Dynamic analysis and testing	B.6.5 Table B.2	HR	Used
3	Data recording and analysis	C.5.2	HR	Records of test cases and results
4	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, boundary value analysis, and input partitioning
5	Performance testing	Table B.6	R	Not used for limited variability programming
6	Model based testing	C.5.27	R	Not used for limited variability programming
7	Interface testing	C.5.3	R	Included in functional and black-box testing
8	Test management and automation tools	C.4.7	HR	Development tools supplied by the PLC manufacturer
9	Forward traceability between the software design specification and the module and integration test specifications	C.2.11	R	Check completeness: review to ensure that an adequate test is planned to examine the functionality of all modules and their integration with appropriately related modules.
10	Formal verification	C.5.12	---	Not used for limited variability programming
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.6 – Programmable electronics integration (hardware and software)

(See 7.5 of IEC 61508-3)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, boundary value analysis, and input partitioning
2	Performance testing	Table B.6	R	When the PLC system is assembled for factory acceptance test
3	Forward traceability between the system and software design requirements for hardware/software integration and the hardware/software integration test specifications	C.2.11	R	Review to ensure that the hardware/software integration tests are adequate
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.7 – Software aspects of system safety validation

(See 7.7 of IEC 61508-3)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1	Probabilistic testing	C.5.1	R	Not used for limited variability programming
2	Process simulation	C.5.18	R	Not used for limited variability programming, but becoming more commonly used in PLC systems development
3	Modelling	Table B.5	R	Not used for limited variability programming, but becoming more commonly used in PLC systems development
4	Functional and black-box testing	B.5.1 B.5.2 Table B.3	HR	Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, boundary value analysis, and input partitioning
5	Forward traceability between the software safety requirements specification and the software safety validation plan	C.2.11	R	Check completeness: review to ensure that adequate software validation tests are planned to address the software safety requirements
6	Backward traceability between the software safety validation plan and the software safety requirements specification	C.2.11	R	Minimise complexity: review to ensure that all validation tests are relevant.
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.8 – Software modification

(See 7.8 of IEC 61508-3)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1	Impact analysis	C.5.23	HR	An impact analysis is carried out to consider how the effect of the proposed changes is limited by the modularity of the overall system
2	Reverify changed software module	C.5.23	HR	Repeat earlier tests
3	Reverify affected software modules	C.5.23	HR	Repeat earlier tests
4a	Revalidate complete system	Table A.7	R	Impact analysis showed that the modification is necessary, so revalidation is done as required
4b	Regression validation	C.5.25	HR	
5	Software configuration management	C.5.24	HR	Baselines, records of changes, impact on other system requirements
6	Data recording and analysis	C.5.2	HR	Records of test cases and results
7	Forward traceability between the Software safety requirements specification and the software modification plan (including reverification and revalidation)	C.2.11	R	Adequate modification procedures to achieve the software safety requirements
8	Backward traceability between the software modification plan (including reverification and revalidation)and the Software safety requirements specification	C.2.11	R	Adequate modification procedures to achieve the software safety requirements
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.9 – Software verification

(See 7.9 of IEC 61508-3)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1	Formal proof	C.5.12	R	Not used for limited variability programming
2	Animation of specification and design	C.5.26	R	
3	Static analysis	B.6.4 Table B.8	HR	Clerical cross-referencing of usage of variables, conditions, etc.
4	Dynamic analysis and testing	B.6.5 Table B.2	HR	Automatic test harness to facilitate regression testing
5	Forward traceability between the software design specification and the software verification (including data verification) plan	C.2.11	R	Check completeness: review to ensure adequate test of functionality.
6	Backward traceability between the software verification (including data verification) plan and the software design specification	C.2.11	R	Minimise complexity: review to ensure that all verification tests are relevant.
7	Offline numerical analysis	C.2.13	R	Not used. The numerical stability of calculations is not a major concern here
Software module testing and integration		See Table E.5 of this standard		
Programmable electronics integration testing		See Table E.6 of this standard		
Software system testing (validation)		See Table E.7 of this standard		
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.10 – Functional safety assessment

(see Clause 8 of IEC 61508-3)

Technique/Measure		Ref.	SIL 2	Interpretation in this application
1	Checklists	B.2.5	R	Used
2	Decision/truth tables	C.6.1	R	Used to a limited degree
3	Failure analysis	Table B.4	R	Cause-consequence diagrams at system level, but otherwise, failure analysis is not used for limited variability programming
4	Common cause failure analysis of diverse software (if diverse software is actually used)	C.6.3	R	Not used for limited variability programming
5	Reliability block diagram	C.6.4	R	Not used for limited variability programming
6	Forward traceability between the requirements of Clause 8 and the plan for software functional safety assessment	C.2.11	R	Check completeness of coverage of the functional safety assessment
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

E.3 Example for safety integrity level 3

This second example is a shut-down application based on a high-level language, of safety integrity level 3.

The software system is relatively large in terms of safety systems; more than 30 000 lines of source code are developed specifically for the system. Also, the usual intrinsic functions are used – at least two diverse operating systems and pre-existing code from earlier projects (proven in use). In total, the system constitutes more than 100 000 lines of source code, if it were available as such.

The whole hardware (including sensors and actuators) is a dual-channel system with its outputs to the final elements connected as a logical AND.

Assumptions:

– although fast response is not required a maximum response time is guaranteed;
– there are interfaces to sensors, actuators and annunciators to human operators;
– the source code of the operating systems, graphic routines and commercial mathematical routines is not available;
– the system is very likely to be subject to later changes;
– the specifically developed software uses one of the common procedural languages;
– it is partially object oriented;
– all parts for which source code is not available are implemented diversely, with the software components being taken from different suppliers and their object code generated by diverse translators;
– the software runs on several commercially available processors that fulfil the requirements of IEC 61508-2;
– all requirements of IEC 61508-2 for control and avoidance of hardware faults are fulfilled by the embedded system; and
– the software development was assessed by an independent organization.

NOTE For the definition of an independent organization, see IEC 61508-4.

The following tables show how the annex tables of IEC 61508-3 may be interpreted for this application.

Table E.11 – Software safety requirements specification

(See 7.2 of IEC 61508-3)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1a	Semi-formal methods	Table B.7	HR	Block diagrams, sequence diagrams, state transition diagrams
1b	Formal methods	B.2.2, C.2.4	R	Only exceptionally
2	Forward traceability between the system safety requirements and the software safety requirements	C.2.11	HR	Check completeness: review to ensure that all system safety requirements are addressed by software safety requirements
3	Backward traceability between the safety requirements and the perceived safety needs	C.2.11	HR	Minimise complexity and functionality: review to ensure that all software safety requirements are actually needed to address system safety requirements
4	Computer-aided specification tools to support appropriate techniques/measures above	B.2.4	HR	Tools supporting the chosen methods
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.12 – Software design and development – software architecture design

(see 7.4.3 of IEC 61508-3)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1	Fault detection	C.3.1	HR	Used as far as dealing with sensor, actuator and data transmission failures and which are not covered by the measures within the embedded system according to the requirements of IEC 61508-2
2	Error detecting codes	C.3.2	R	Only for external data transmissions
3a	Failure assertion programming	C.3.3	R	Results of the application functions are checked for validity
3b	Diverse monitor techniques (with independence between the monitor and the monitored function in the same computer)	C.3.4	R	Not preferred: increased software complexity to guarantee independence.
3c	Diverse monitor techniques (with separation between the monitor computer and the monitored computer)	C.3.4	R	Used for some safety related functions where 3a is not used
3d	Diverse redundancy, implementing the same software safety requirements specification	C.3.5	---	Used for some functions where source code is not available
3e	Functionally diverse redundancy, implementing different software safety requirements specification	C.3.5	R	Not preferred: substantially achieved by 3c.
3f	Backward recovery	C.3.6	---	Not used
3g	Stateless software design (or limited state design)	C.2.12	R	Not used. A controlled shutdown needs states to memorise plant condition.
4a	Re-try fault recovery mechanisms	C.3.7	---	Not used
4b	Graceful degradation	C.3.8	HR	Yes, because of the nature of the technical process

Technique/Measure		Ref.	SIL 3	Interpretation in this application
5	Artificial intelligence - fault correction	C.3.9	NR	Not used
6	Dynamic reconfiguration	C.3.10	NR	Not used
7	Modular approach	Table B.9	HR	Needed because of the size of the system
8	Use of trusted/verified software elements (if available)	C.2.10	HR	pre-existing code from earlier projects
9	Forward traceability between the software safety requirements specification and software architecture	C.2.11	HR	Review to ensure that all software safety requirements are addressed by the software architecture
10	Backward traceability between the software safety requirements specification and software architecture	C.2.11	HR	Minimise complexity and functionality: review to ensure that all architecture safety requirements are actually needed to address software safety requirements
11a	Structured diagrammatic methods	C.2.1	HR	Needed because of the size of the system
11b	Semi-formal methods	Table B.7	HR	Block diagrams, sequence diagrams, state transition diagrams
11c	Formal design and refinement methods	B.2.2, C.2.4	R	Not used
11d	Automatic software generation	C.4.6	R	Not used. Avoid translator/generator uncertainty.
12	Computer-aided specification and design tools	B.2.4	HR	Tools supporting the chosen method
13a	Cyclic behaviour, with guaranteed maximum cycle time	C.3.11	HR	Not used
13b	Time-triggered architecture	C.3.11	HR	Not used
13c	Event-driven, with guaranteed maximum response time	C.3.11	HR	Not used
14	Static resource allocation	C.2.6.3	HR	Not used. Choose programming language to avoid dynamic resources issues
15	Static synchronisation of access to shared resources	C.2.6.3	R	Not used. Choose programming language to avoid dynamic resources issues
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.13 – Software design and development – support tools and programming language

(See 7.4.4 of IEC 61508-3)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1	Suitable programming language	C.4.5	HR	Full variability high-level language selected
2	Strongly typed programming language	C.4.1	HR	Used
3	Language subset	C.4.2	HR	Defined subset for the selected language
4a	Certified tools and certified translators	C.4.3	HR	Not available
4b	Tools and translators: increased confidence from use	C.4.4	HR	Available, and used
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.14 – Software design and development – detailed design

(See 7.4.5 and 7.4.6 of IEC 61508-3)
(Includes software system design, software module design and coding)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1a	Structured methods	C.2.1	HR	Widely used. In particular, SADT and JSD
1b	Semi-formal methods	Table B.7	HR	Finite state machines/state transition diagrams, block diagrams, sequence diagrams
1c	Formal design and refinement methods	B.2.2, C.2.4	R	Only exceptionally, for some very basic components only
2	Computer-aided design tools	B.3.5	HR	Used for the selected methods
3	Defensive programming	C.2.5	HR	All measures except those which are automatically inserted by the compiler are explicitly used in application software where they are effective
4	Modular approach	Table B.9	HR	Software module size limit, information hiding/encapsulation, one entry/one exit point in subroutines and functions, fully defined interface, …
5	Design and coding standards	C.2.6 Table B.1	HR	Use of coding standard, no dynamic objects, no dynamic variables, limited use of interrupts, limited use of pointers, limited use of recursion, no unconditional jumps, …
6	Structured programming	C.2.7	HR	Used
7	Use of trusted/verified software elements (if available)	C.2.10	HR	Available, and used
8	Forward traceability between the software safety requirements specification and software design	C.2.11	HR	Review to ensure that all software safety requirements are addressed by the software design

NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.

Table E.15 – Software design and development – software module testing and integration

(See 7.4.7 and 7.4.8 of IEC 61508-3)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1	Probabilistic testing	C.5.1	R	Used for software modules where no source code available and the definition of boundary values and equivalence classes for test data is difficult
2	Dynamic analysis and testing	B.6.5 Table B.2	HR	Used for software modules where source code is available. Test cases from boundary value analysis, performance modelling, equivalence classes and input partitioning, structure-based testing
3	Data recording and analysis	C.5.2	HR	Records of test cases and results
4	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	Used for software module testing where no source code is available and for integration testing. Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, prototyping, boundary value analysis, equivalence classes and input partitioning

Technique/Measure		Ref.	SIL 3	Interpretation in this application
5	Performance testing	Table B.6	HR	Used during integration testing on the target hardware
6	Model based testing	C.5.27	HR	Not used
7	Interface testing	C.5.3	HR	Included in functional and black-box testing
8	Test management and automation tools	C.4.7	HR	Used where available
9	Forward traceability between the software design specification and the module and integration test specifications	C.2.11	HR	Review to ensure that the integration tests are sufficient
10	Formal verification	C.5.12	R	Not used
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.16 – Programmable electronics integration (hardware and software)

(See 7.5 of IEC 61508-3)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	Used as additional tests to software integration testing (see Table E.15 above) Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, prototyping, boundary value analysis, equivalence classes and input partitioning
2	Performance testing	Table B.6	HR	Extensively used
3	Forward traceability between the system and software design requirements for hardware/software integration and the hardware/software integration test specifications	C.2.11	HR	Review to ensure that the integration tests are sufficient
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.17 – Software aspects of system safety validation

(See 7.7 of IEC 61508-3)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1	Probabilistic testing	C.5.1	R	Not used for validation
2	Process simulation	C.5.18	HR	Finite state machines, performance modelling, prototyping and animation
3	Modelling	Table B.5	HR	Not used for validation
4	Functional and black-box testing	B.5.1 B.5.2 Table B.3	HR	Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, boundary value analysis, and input partitioning
5	Forward traceability between the software safety requirements specification and the software safety validation plan	C.2.11	HR	Check completeness: review to ensure that all software safety requirements are addressed by the validation plan
6	Backward traceability between the software safety validation plan and the software safety requirements specification	C.2.11	HR	Minimise complexity: review to ensure that all validation tests are relevant
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.18 – Modification

(See 7.8 of IEC 61508-3)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1	Impact analysis	C.5.23	HR	Used
2	Reverify changed software module	C.5.23	HR	Used
3	Reverify affected software modules	C.5.23	HR	Used
4a	Revalidate complete system	Table A.7	HR	Depends on the result of the impact analysis
4b	Regression validation	C.5.25	HR	Used
5	Software configuration management	C.5.24	HR	Used
6	Data recording and analysis	C.5.2	HR	Used
7	Forward traceability between the Software safety requirements specification and the software modification plan (including reverification and revalidation)	C.2.11	HR	Check completeness: review to ensure that the modification procedures are adequate to achieve the software safety requirements
8	Backward traceability between the software modification plan (including reverification and revalidation)and the software safety requirements specification	C.2.11	HR	Minimise complexity: review to ensure that all modification procedures are necessary
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.19 – Software verification

(See 7.9 of IEC 61508-3)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1	Formal proof	C.5.12	R	Only exceptionally, for some very basic classes only
2	Animation of specification and design	C.5.26	R	Not used
3	Static analysis	B.6.4 Table B.8 C.5.14	HR	For all newly developed code. Boundary value analysis, checklists, control flow analysis, data flow analysis, Fagan inspections, design reviews
4	Dynamic analysis and testing	B.6.5 Table B.2	HR	For all newly developed code
5	Forward traceability between the software design specification and the software verification (including data verification) plan	C.2.11	HR	Check completeness: review to ensure that the modification procedures are adequate for the software safety requirements
6	Backward traceability between the software verification (including data verification) plan and the software design specification	C.2.11	HR	Minimise complexity: review to ensure that all modification procedures are necessary
7	Offline numerical analysis	C.2.13	HR	Not used. The numerical stability of calculations is not a major concern here
Software module testing and integration		See Table E.15 of this standard		
Programmable electronics integration testing		See Table E.16 of this standard		
Software system testing (validation)		See Table E.17 of this standard		
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Table E.20 – Functional safety assessment

(see Clause 8 of IEC 61508-3)

Technique/Measure		Ref.	SIL 3	Interpretation in this application
1	Checklists	B.2.5	R	Used
2	Decision/truth tables	C.6.1	R	Used, to a limited degree
3	Failure analysis	Table B.4	HR	Fault-tree analysis is extensively used, and cause consequence diagrams are used to a limited degree
4	Common cause failure analysis of diverse software (if diverse software is actually used)	C.6.3	HR	Used
5	Reliability block diagram	C.6.4	R	Used
6	Forward traceability between the requirements of Clause 8 and the plan for software functional safety assessment	C.2.11	HR	Check completeness of coverage of the functional safety assessment
NOTE In the reference columns (entitled Ref), the informative references “B.x.x.x”, “C.x.x.x” refer to descriptions of techniques in IEC 61508-7 Annexes B and C, while “Table A.x”, “Table B.x” refer to tables of techniques in IEC 61508-3 Annexes A and B.				

Bibliography

[1] IEC 61511 (all parts), *Functional safety – Safety instrumented systems for the process industry sector*

[2] IEC 62061, *Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems*

[3] IEC 61800-5-2, *Adjustable speed electrical power drive systems – Part 5-2: Safety requirements – Functional*

The following references give further details on evaluating probabilities of failure (see Annex B).

[4] IEC 61078:2006, *Analysis techniques for dependability – Reliability block diagram and boolean methods*

[5] IEC 61165:2006, *Application of Markov techniques*

[6] BS 5760, *Reliability of system equipment and components – Part 2: Guide to assessment of reliability*

[7] D. J. SMITH, *Reliability, maintainability and risk – Practical methods for engineers*, Butterworth-Heinemann, 5th edition, 1997, ISBN 0-7506-3752-8

[8] R. BILLINGTON and R. N. ALLAN, *Reliability evaluation of engineering systems*, Plenum, 1992, ISBN 0-306-44063-6

[9] W. M. GOBLE, *Evaluating control system reliability – Techniques and applications*, Instrument Society of America, 1992, ISBN 1-55617-128-5

Useful references for the calculation of diagnostic coverage (see Annex C) include the following.

[10] Reliability Analysis Center (RAC), *Failure Mode/Mechanism Distributions*, 1991, Department of Defense, United States of America, PO Box 4700, 201 Mill Street, Rome, NY 13440-8200, Organization report number: FMD-91, NSN 7540-01-280-5500

[11] ALLESSANDRO BIROLINI*, Qualität und Zuverlassigkeit technischer Systeme, Theorie, Praxis, Management*, Dritte Auflage,1991, Springer-Verlag, Berlin Heidelberg New York, ISBN 3-540-54067-9, 3 Aufl., ISBN 0-387-54067-9 3 ed. (available in German only)

[12] MIL-HDBK-217F, *Military Handbook Reliability prediction of electronic equipment*, 2 December 1991, Department of Defense, United States of America

The following references provide useful information relating to common cause failures (see Annex D).

[13] *Health and Safety Executive Books*, email hsebooks@prolog.uk.com

[14] R. HUMPHREYS, A., PROC., *Assigning a numerical value to the beta factor common-cause evaluation,* Reliability 1987

[15] UPM3.1, *A pragmatic approach to dependent failures assessment for standard systems*, AEA Technology, Report SRDA-R-13, ISBN 085 356 4337, 1996

The following standard is referred to in Table E.3.

[16] IEC 61131-3:2003, *Programmable controllers – Part 3: Programming languages*

[17] ISA-TR84.00.02-2002 – Parts 1-5, Safety Instrumented Functions (SIF) Safety Integrity Level (SIL) Evaluation Techniques Package.

[18] IEC 61025:2006, *Fault tree analysis (FTA)*

[19] IEC 62551, *Analysis techniques for dependability – Petri Net technique*[10]

[20] ANIELLO AMENDOLA, kluwer academic publisher, ISPRA 16-19 November 1987, *Advanced seminar on Common Cause Failure Analysis in Probabilistic Safety Assessment,* ISBN 0-7923-0268-0

[21] CORWIN L. ATWOOD, *The Binomial Failure Rate Common Cause Model,* Technometrics May 1986 Vol 28 n°2

[22] A. ARNOLD, A. GRIFFAULT, G. POINT, AND A. RAUZY. *The altarica language and its semantics*. *Fundamenta Informaticae*, 34,pp.109–124, 2000

[23] M. BOITEAU, Y. DUTUIT, A. RAUZY AND J.-P. SIGNORET, The AltaRica *Data-Flow Language in Use: Assessment of Production Availability of a MultiStates System*, *Reliability Engineering and System Safety*, Elsevier, Vol. 91, pp 747-755

[24] A. RAUZY. *Mode automata and their compilation into fault trees. Reliability Engineering and System Safety*, Elsevier 2002, Volume 78, Issue 1, pp 1-12

[25] For PDS method; see <www.sintef.no/pds>; and further background material in:Hokstad, Per; Corneliussen, Kjell Source: Reliability Engineering and System Safety, v 83, n 1, p 111-120, January 2004

[26] IEC 60601 (all parts), *Medical electrical equipment*

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

[28] IEC 61508-5:2010 *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 5: Examples of methods for the determination of safety integrity levels*

[29] IEC 61508-7:2010 *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures*

———————

10 Under consideration.

IEC 61508-7

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 7: Présentation de techniques et mesures

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.
- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.
- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.
- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:
Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm
Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,…). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.
- Just Published CEI: www.iec.ch/online_news/justpub
Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.
- Electropedia: www.electropedia.org
Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.
- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:
Email: csc@iec.ch
Tél.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61508-7

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 7: Présentation de techniques et mesures

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE XG
CODE PRIX

ICS 25.040.40; 35.240.50

ISBN 978-2-88910-530-4

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

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

Part 7: Overview of techniques and measures

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) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61508-7 has been prepared by subcommittee 65A: System aspects, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 2000. This edition constitutes a technical revision.

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

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

FDIS	Report on voting
65A/554/FDIS	65A/578/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

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

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

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 [21], [22] and [37]).

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.

– 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.

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

Part 7: Overview of techniques and measures

1 Scope

1.1 This part of IEC 61508 contains an overview of various safety techniques and measures relevant to IEC 61508-2 and IEC 61508-3.

The references should be considered as basic references to methods and tools or as examples, and may not represent the state of the art.

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 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 for parts 1 to 7 of IEC 61508 and indicates the role that IEC 61508-7 plays in the achievement of functional safety for E/E/PE safety-related systems.

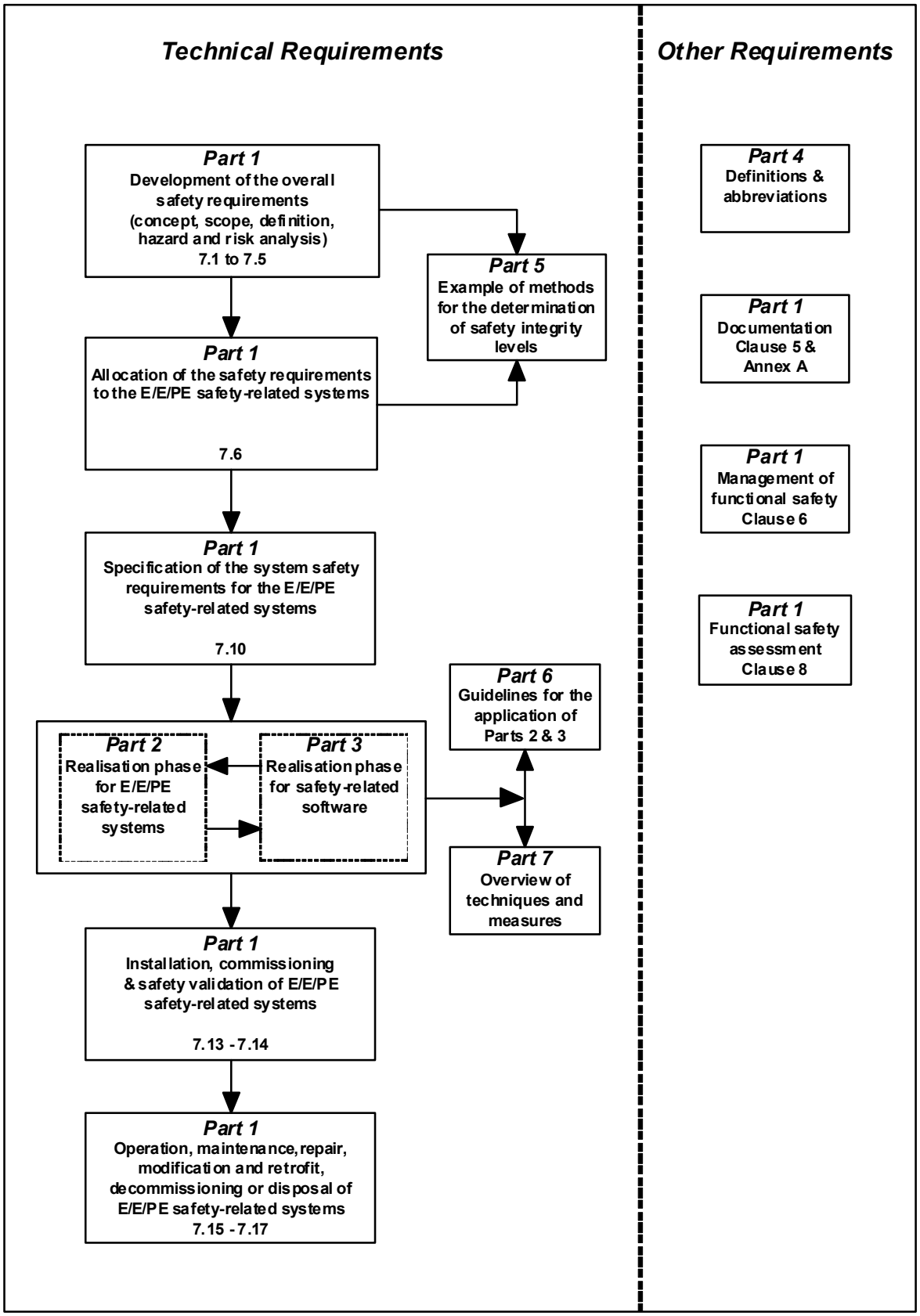


Figure 1 – Overall framework of IEC 61508

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

3 Definitions and abbreviations

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

Annex A
(informative)

Overview of techniques and measures for E/E/PE safety-related systems: control of random hardware failures (see IEC 61508-2)

A.1 Electric

Global objective: To control failures in electromechanical components.

A.1.1 Failure detection by on-line monitoring

NOTE This technique/measure is referenced in Tables A.2, A.3, A.7 and A.13 to A.18 of IEC 61508-2.

Aim: To detect failures by monitoring the behaviour of the E/E/PE safety-related system in response to the normal (on-line) operation of the equipment under control (EUC).

Description: Under certain conditions, failures can be detected using information about (for example) the time behaviour of the EUC. For example, if a switch, which is part of the E/E/PE safety-related system, is normally actuated by the EUC, then if the switch does not change state at the expected time, a failure will have been detected. It is not usually possible to localise the failure.

A.1.2 Monitoring of relay contacts

NOTE This technique/measure is referenced in Tables A.2 and A.14 of IEC 61508-2.

Aim: To detect failures (for example welding) of relay contacts.

Description: Forced contact (or positively guided contact) relays are designed so that their contacts are rigidly linked together. Assuming there are two sets of changeover contacts, *a* and *b*, if the normally open contact, *a*, welds, the normally closed contact, *b*, cannot close when the relay coil is next de-energised. Therefore, the monitoring of the closure of the normally closed contact *b* when the relay coil is de-energised may be used to prove that the normally open contact *a* has opened. Failure of normally closed contact *b* to close indicates a failure of contact *a*, so the monitoring circuit should ensure a safe shut-down, or ensure that shut-down is continued, for any machinery controlled by contact *a*.

References:

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Verriegelungseinrichtungen. F. Kreutzkampf, W. Hertel, Sicherheitstechnisches Informations- und Arbeitsblatt 330212, BIA-Handbuch. 17. Lfg. X/91, Erich Schmidt Verlag, Bielefeld.
www.BGIA-HANDBUCHdigital.de/330212

A.1.3 Comparator

NOTE This technique/measure is referenced in Tables A.2, A.3, A.4 of IEC 61508-2.

Aim: To detect, as early as possible, (non-simultaneous) failures in an independent processing unit or in the comparator.

Description: The signals of independent processing units are compared cyclically or continuously by a hardware comparator. The comparator may itself be externally tested, or it may use self-monitoring technology. Detected differences in the behaviour of the processors lead to a failure message.

A.1.4 Majority voter

NOTE This technique/measure is referenced in Tables A.2, A.3 and A.4 of IEC 61508-2.

Aim: To detect and mask failures in one of at least three hardware channels.

Description: A voting unit using the majority principle (2 out of 3, 3 out of 3, or *m* out of *n*) is used to detect and mask failures. The voter may itself be externally tested, or it may use self-monitoring technology.

References:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

A.1.5 Idle current principle (de-energised to trip)

NOTE This technique/measure is referenced in Table A.16 of IEC 61508-2.

Aim: To execute the safety function if power is cut or lost.

Description: The safety function is executed if the contacts are open and no current flows. For example, if brakes are used to stop a dangerous movement of a motor, the brakes are opened by closing contacts in the safety-related system and are closed by opening the contacts in the safety-related system.

Reference:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

A.2 Electronic

Global objective: To control failure in solid-state components.

A.2.1 Tests by redundant hardware

NOTE This technique/measure is referenced in Tables A.3, A.15, A.16 and A.18 of IEC 61508-2.

Aim: To detect failures using hardware redundancy, i.e. using additional hardware not required to implement the process functions.

Description: Redundant hardware can be used to test at an appropriate frequency the specified safety functions. This approach is normally necessary for realising A.1.1 or A.2.2.

A.2.2 Dynamic principles

NOTE This technique/measure is referenced in Table A.3 of IEC 61508-2.

Aim: To detect static failures by dynamic signal processing.

Description: A forced change of otherwise static signals (internally or externally generated) helps to detect static failures in components. This technique is often associated with electromechanical components.

Reference:

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993. http://www.bgia-handbuchdigital.de/330220

A.2.3 Standard test access port and boundary-scan architecture

NOTE This technique/measure is referenced in Tables A.3, A.15 and A.18 of IEC 61508-2.

Aim: To control and observe what happens at each pin of an IC.

Description: Boundary-scan test is an IC design technique which increases the testability of the IC by resolving the problem of how to gain access to the circuit test points within it. In a typical boundary-scan IC, comprised of core logic and input and output buffers, a shift-register stage is placed between the core logic and the input and output buffers adjacent to each IC pin. Each shift-register stage is contained in a boundary-scan cell. The boundary-scan cell can control and observe what happens at each input and output pin of an IC, via the standard test access port. Internal testing of the IC core logic is accomplished by isolating the on-chip core logic from stimuli received from surrounding components, and then performing an internal self-test. These tests can be used to detect failures in the IC.

Reference:

IEEE 1149-1:2001, *IEEE standard test access port and boundary-scan architecture, IEEE Computer Society*, 2001, ISBN: 0-7381-2944-5

A.2.4 (Not used)

A.2.5 Monitored redundancy

NOTE This technique/measure is referenced in Table A.3 of IEC 61508-2.

Aim: To detect failure, by providing several functional units, by monitoring the behaviour of each of these to detect failures, and by initiating a transition to a safe condition if any discrepancy in behaviour is detected.

Description: The safety function is executed by at least two hardware channels. The outputs of these channels are monitored and a safe condition is initiated if a fault is detected (i.e. if the output signals from all channels are not identical).

References:

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993. http://www.bgia-handbuchdigital.de/330220

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0

A.2.6 Electrical/electronic components with automatic check

NOTE This technique/measure is referenced in Table A.3 of IEC 61508-2.

Aim: To detect faults by periodic checking of the safety functions.

Description: The hardware is tested before starting the process, and is tested repeatedly at suitable intervals. The EUC continues to operate only if each test is successful.

References:

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993. http://www.bgia-handbuchdigital.de/330220

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0

A.2.7 Analogue signal monitoring

NOTE This technique/measure is referenced in Tables A.3 and A.13 of IEC 61508-2.

Aim: To improve confidence in measured signals.

Description: Wherever there is a choice, analogue signals are used in preference to digital on/off states. For example, trip or safe states are represented by analogue signal levels, usually with signal level tolerance monitoring. The technique provides continuity monitoring and a higher level of confidence in the transmitter, reducing the necessary proof-test frequency of the transmitter sensing function. External interfaces, for example impulse lines, will also require testing.

A.2.8 De-rating

NOTE This technique/measure is referenced in 7.4.2.13 of IEC 61508-2.

Aim: To increase the reliability of hardware components.

Description: Hardware components are operated at levels which are guaranteed by the design of the system to be well below the maximum specification ratings. De-rating is the practice of ensuring that under all normal operating circumstances, components are operated well below their maximum stress levels.

A.3 Processing units

Global objective: To recognise failures which lead to incorrect results in processing units.

A.3.1 Self-test by software: limited number of patterns (one-channel)

NOTE This technique/measure is referenced in Table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the processing unit.

Description: The hardware is built using standard techniques which do not take any special safety requirements into account. The failure detection is realised entirely by additional software functions which perform self-tests using at least two complementary data patterns (for example 55hex and AAhex).

A.3.2 Self-test by software: walking bit (one-channel)

NOTE This technique/measure is referenced in Table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the physical storage (for example registers) and instruction decoder of the processing unit.

Description: The failure detection is realised entirely by additional software functions which perform self-tests using a data pattern (for example walking-bit pattern) which tests the physical storage (data and address registers) and the instruction decoder. However, the diagnostic coverage is only 90 %.

A.3.3 Self-test supported by hardware (one-channel)

NOTE This technique/measure is referenced in Table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the processing unit, using special hardware that increases the speed and extends the scope of failure detection.

Description: Additional special hardware facilities support self-test functions to detect failure. For example, this could be a hardware unit which cyclically monitors the output of a certain bit pattern according to the watch-dog principle.

A.3.4 Coded processing (one-channel)

NOTE This technique/measure is referenced in Table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the processing unit.

Description: Processing units can be designed with special failure-recognising or failure-correcting circuit techniques. So far, these techniques have been applied only to relatively simple circuits and are not widespread; however, future developments should not be excluded.

References:

Le processeur codé: un nouveau concept appliqué à la sécurité des systèmes de transports. Gabriel, Martin, Wartski, Revue Générale des chemins de fer, No. 6, June 1990

Vital Coded Microprocessor Principles and Application for Various Transit Systems. P. Forin, IFAC Control Computers Communications in Transportation, 79-84, 1989

A.3.5 Reciprocal comparison by software

NOTE This technique/measure is referenced in Table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the processing unit, by dynamic software comparison.

Description: Two processing units exchange data (including results, intermediate results and test data) reciprocally. A comparison of the data is carried out using software in each unit and detected differences lead to a failure message.

A.4 Invariable memory ranges

Global objective: The detection of information modifications in the invariable memory.

A.4.1 Word-saving multi-bit redundancy (for example ROM monitoring with a modified Hamming code)

NOTE 1 This technique/measure is referenced in Table A.5 of IEC 61508-2.

NOTE 2 See also A.5.6 “RAM monitoring with a modified Hamming code, or detection of data failures with error-detection-correction codes (EDC)” and C.3.2 “Error detecting and correcting codes”.

Aim: To detect all single-bit failures, all two-bit failures, some three-bit failures, and some all-bit failures in a 16-bit word.

Description: Every word of memory is extended by several redundant bits to produce a modified Hamming code with a Hamming distance of at least 4. Every time a word is read, checking of the redundant bits can determine whether or not a corruption has taken place. If a difference is found, a failure message is produced. The procedure can also be used to detect

addressing failures, by calculating the redundant bits for the concatenation of the data word and its address.

References:

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967

Error detecting and error correcting codes. R. W. Hamming, The Bell System Technical Journal 29 (2), 147-160, 1950

A.4.2 Modified checksum

NOTE This technique/measure is referenced in Table A.5 of IEC 61508-2.

Aim: To detect all odd-bit failures, i.e. approximately 50 % of all possible bit failures.

Description: A checksum is created by a suitable algorithm which uses all the words in a block of memory. The checksum may be stored as an additional word in ROM, or an additional word may be added to the memory block to ensure that the checksum algorithm produces a predetermined value. In a later memory test, a checksum is created again using the same algorithm, and the result is compared with the stored or defined value. If a difference is found, a failure message is produced.

A.4.3 Signature of one word (8-bit)

NOTE This technique/measure is referenced in Table A.5 of IEC 61508-2.

Aim: To detect all one-bit failures and all multi-bit failures within a word, as well as approximately 99,6 % of all possible bit failures.

Description: The contents of a memory block is compressed (using either hardware or software) using a cyclic redundancy check (CRC) algorithm into one memory word. A typical CRC algorithm treats the whole contents of the block as byte-serial or bit-serial data flow, on which a continued polynomial division is carried out using a polynomial generator. The remainder of the division represents the compressed memory contents – it is the "signature" of the memory – and is stored. The signature is computed once again in later tests and compared with one already stored. A failure message is produced if there is a difference.

A.4.4 Signature of a double word (16-bit)

NOTE This technique/measure is referenced in Table A.5 of IEC 61508-2.

Aim: To detect all one-bit failures and all multi-bit failures within a word, as well as approximately 99,998 % of all possible bit failures.

Description: This procedure calculates a signature using a cyclic redundancy check (CRC) algorithm, but the resulting value is at least two words in size. The extended signature is stored, recalculated and compared as in the single-word case. A failure message is produced if there is a difference between the stored and recalculated signatures.

A.4.5 Block replication (for example double ROM with hardware or software comparison)

NOTE This technique/measure is referenced in Table A.5 of IEC 61508-2.

Aim: To detect all bit failures.

Description: The address space is duplicated in two memories. The first memory is operated in the normal manner. The second memory contains the same information and is accessed in parallel to the first. The outputs are compared and a failure message is produced if a

difference is detected. In order to detect certain kinds of bit errors, the data must be stored inversely in one of the two memories and inverted once again when read.

A.5 Variable memory ranges

Global objective: Detecting failures during addressing, writing, storing and reading.

NOTE Soft-errors are listed in Table A.1, IEC 61508-2 as faults to be detected during operation or to be analysed in the derivation of the safe failure fraction. Causes of soft errors are: (1) Alpha particles from package decay, (2) Neutrons, (3) external EMI noise, (4) Internal cross-talk. External EMI noise is covered by other requirements of this international standard.

The effect of Alpha particles and Neutrons should be mastered by safety integrity measures at runtime. Safety integrity measures effective for hard errors may not be effective for soft errors, e.g. RAM tests, such as walk-path, galpat, etc. are not effective, whereas monitoring techniques such as Parity and ECC with recurring read of the memory cells are.

A soft error occurs when a radiation event causes enough of a charge disturbance to reverse or flip the data state of a low energized semiconductor memory cell, register, latch, or flip-flop. The error is called “soft” because the circuit itself is not permanently damaged by the radiation. Soft-errors are classified in Single Bit Upsets (SBU) or Single Event Upsets (SEU) and Multi-Bit Upsets (MBU).

If the disturbed circuit is a storage element like memory cell or flip-flop, the state is stored until the next (intended) write operation. The new data will be stored correctly. In a combinatory circuit the effect is rather a glitch because there is a continuous energy flow from the component driving this node. On connecting wires and communication lines the effect could also be a glitch. However due to the larger capacitance the effect by Alpha particles and Neutrons is considered negligible.

Soft-errors may be relevant to variable memory of any kind, i.e., to DRAM, SRAM, register banks in µP, cache, pipelines, configuration registers of devices such as ADC, DMA, MMU, Interrupt controller, complex timers. Sensitivity to alpha and neutron particles is a function of both core voltage and geometry. Smaller geometries at 2,5 V core voltage and especially below 1,8 V would require more evaluation and more effective protective measures.

The soft error rate has been reported (see a) and i) below) to be in a range of 700 Fit/MBit to 1 200 Fit/MBit for (embedded) memories. This is a reference value to be compared with data coming from the silicon process with which the device is implemented. Until recently SBU were considered to be dominant, but the latest forecast (see a) below) reports a growing percentage of MBU of the overall soft-error rate (SER) for technologies from 65 nm down.

The following literature and sources give details about soft-errors:

a) Altitude SEE Test European Platform (ASTEP) and First Results in CMOS 130 nm SRAM. J-L. Autran, P. Roche, C. Sudre et al. Nuclear Science, IEEE Transactions on Volume 54, Issue 4, Aug. 2007 Page(s):1002 - 1009

b) Radiation-Induced Soft Errors in Advanced Semiconductor Technologies, Robert C. Baumann, Fellow, IEEE, IEEE TRANSACTIONS ON DEVICE AND MATERIALS RELIABILITY, VOL. 5, NO. 3, SEPTEMBER 2005

c) Soft errors' impact on system reliability, Ritesh Mastipuram and Edwin C Wee, Cypress Semiconductor, 2004

d) Trends And Challenges In VLSI Circuit Reliability, C. Costantinescu, Intel, 2003, IEEE Computer Society

e) Basic mechanisms and modeling of single-event upset in digital microelectronics, P. E. Dodd and L. W. Massengill, IEEE Trans. Nucl. Sci., vol. 50, no. 3, pp. 583–602, Jun. 2003.

f) Destructive single-event effects in semiconductor devices and ICs, F. W. Sexton, IEEE Trans. Nucl. Sci., vol. 50, no. 3, pp. 603–621, Jun. 2003.

g) Coming Challenges in Microarchitecture and Architecture, Ronen, Mendelson, Proceedings of the IEEE, Volume 89, Issue 3, Mar 2001 Page(s):325 – 340

h) Scaling and Technology Issues for Soft Error Rates, A Johnston, 4th Annual Research Conference on Reliability Stanford University, October 2000

i) International Technology Roadmap for Semiconductors (ITRS), several papers.

A.5.1 RAM test "checkerboard" or "march"

NOTE This technique/measure is referenced in Table A.6 of IEC 61508-2.

Aim: To detect predominantly static bit failures.

Description: A checker-board type pattern of 0 s and 1 s is written into the cells of a bit-oriented memory. The cells are then inspected in pairs to ensure that the contents are the same and correct. The address of the first cell of such a pair is variable and the address of the second cell of the pair is formed by inverting bitwise the first address. In the first run, the address range of the memory is run towards higher addresses from the variable address, and in a second run towards lower addresses. Both runs are then repeated with an inverted pre-assignment. A failure message is produced if any difference occurs.

In a RAM test "march", the cells of a bit-oriented memory are initialised by a uniform bit stream. In the first run, the cells are inspected in ascending order: each cell is checked for the correct contents and its contents are inverted. The background, which is created in the first run, is treated in a second run in descending order and in the same manner. Both first runs are repeated with an inverted pre-assignment in a third or fourth run. A failure message is produced if a difference occurs.

A.5.2 RAM test "walkpath"

NOTE This technique/measure is referenced in Table A.6 of IEC 61508-2.

Aim: To detect static and dynamic bit failures and cross-talk between memory cells.

Description: The memory range to be tested is initialised by a uniform bit stream. The first cell is then inverted and the remaining memory area is inspected to ensure that the background is correct. After this, the first cell is re-inverted to return it to its original value, and the whole procedure is repeated for the next cell. A second run of the "wandering bit model" is carried out with an inverse background pre-assignment. A failure message is produced if a difference occurs.

A.5.3 RAM test "galpat" or "transparent galpat"

NOTE This technique/measure is referenced in Table A.6 of IEC 61508-2.

Aim: To detect static bit failures and a large proportion of dynamic couplings.

Description: In the RAM test "galpat", the chosen range of memory is first initialised uniformly (i.e. all 0 s or all 1 s). The first memory cell to be tested is then inverted and all the remaining cells are inspected to ensure that their contents are correct. After every read access to one of the remaining cells, the inverted cell is also checked. This procedure is repeated for each cell in the chosen memory range. A second run is carried out with the opposite initialisation. Any difference produces a failure message.

The "transparent galpat" test is a variation on the above procedure: instead of initialising all cells in the chosen memory range, the existing contents are left unchanged and signatures are used to compare the contents of sets of cells. The first cell to be tested in the chosen range is selected, and the signature S1 of all remaining cells in the range is calculated and stored. The cell to be tested is then inverted and the signature S2 of all the remaining cells is recalculated. (After every read access to one of the remaining cells, the inverted cell is also checked.) S2 is compared with S1, and any difference produces a failure message. The cell under test is re-inverted to re-establish the original contents, and the signature S3 of all the remaining cells is recalculated and compared with S1. Any difference produces a failure message. All memory cells in the chosen range are tested in the same manner.

A.5.4 RAM test "Abraham"

NOTE This technique/measure is referenced in Table A.6 of IEC 61508-2.

Aim: To detect all stuck-at and coupling failures between memory cells.

Description: The proportion of faults detected exceeds that of the RAM test "galpat". The number of operations required to perform the entire memory test is about 30 n, where n is the

number of cells in the memory. The test can be made transparent for use during the operating cycle by partitioning the memory and testing each partition in different time segments.

Reference:

Efficient Algorithms for Testing Semiconductor Random-Access Memories. R. Nair, S. M. Thatte, J. A. Abraham, IEEE Trans. Comput. C-27 (6), 572-576, 1978

A.5.5 One-bit redundancy (for example RAM monitoring with a parity bit)

NOTE This technique/measure is referenced in Table A.6 of IEC 61508-2.

Aim: To detect 50 % of all possible bit failures in the memory range tested.

Description: Every word of the memory is extended by one bit (the parity bit) which completes each word to an even or odd number of logical 1 s. The parity of the data word is checked each time it is read. If the wrong number of 1 s is found, a failure message is produced. The choice of even or odd parity should be made such that, whichever of the zero word (nothing but 0 s) and the one word (nothing but 1 s) is the more unfavourable in the event of a failure, then that word is not a valid code. Parity can also be used to detect addressing failures, when the parity is calculated for the concatenation of the data word and its address.

A.5.6 RAM monitoring with a modified Hamming code, or detection of data failures with error-detection-correction codes (EDC)

NOTE 1 This technique/measure is referenced in Table A.6 of IEC 61508-2.

NOTE 2 See also A.4.1 “Word-saving multi-bit redundancy (for example ROM monitoring with a modified Hamming code)”and C.3.2 “Error detecting and correcting codes”.

Aim: To detect all odd-bit failures, all two-bit failures, some three-bit and some multi-bit failures.

Description: Every access to memory is extended by several redundant bits to produce a modified Hamming code with a Hamming distance of at least 4. Every time data is read, one can determine whether a corruption has taken place by checking the redundant bits. If a difference is found, a failure message is produced. The procedure can also be used to detect addressing failure, when the redundant bits are calculated for the concatenation of the data and its address.

References:

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967

Error detecting and error correcting codes. R. W. Hamming, The Bell System Technical Journal 29 (2), 147-160, 1950

A.5.7 Double RAM with hardware or software comparison and read/write test

NOTE This technique/measure is referenced in Table A.6 of IEC 61508-2.

Aim: To detect all bit failures.

Description: The address space is duplicated in two memories. The first memory is operated in the normal manner. The second memory contains the same information and is accessed in parallel to the first. The outputs are compared and a failure message is produced if a difference is detected. In order to detect certain kinds of bit errors, the data must be stored inversely in one of the two memories and inverted once again when read.

A.6 I/O-units and interfaces (external communication)

Global objective: To detect failures in input and output units (digital, analogue, serial or parallel) and to prevent the sending of inadmissible outputs to the process.

A.6.1 Test pattern

NOTE This technique/measure is referenced in Tables A.7, A.13 and A.14 of IEC 61508-2.

Aim: To detect static failures (stuck-at failures) and cross-talk.

Description: This is a dataflow-independent cyclical test of input and output units. It uses a defined test pattern to compare observations with the corresponding expected values. The test pattern information, the test pattern reception, and test pattern evaluation must all be independent of each other. The EUC should not be inadmissibly influenced by the test pattern.

A.6.2 Code protection

NOTE This technique/measure is referenced in Tables A.7, A.15, A.16 and A.18 of IEC 61508-2.

Aim: To detect random hardware and systematic failures in the input/output dataflow.

Description: This procedure protects the input and output information from both systematic and random hardware failures. Code protection provides dataflow-dependent failure detection of the input and output units, based on information redundancy and/or time redundancy. Typically, redundant information is superimposed on input and/or output data. This then provides a means to monitor the correct operation of the input or output circuits. Many techniques are possible, for example a carrier frequency signal may be superimposed on the output signal of a sensor. The logic unit may then check for the presence of the carrier frequency or redundant code bits may be added to an output channel to allow the monitoring of the validity of a signal passing between the logic unit and final actuator.

A.6.3 Multi-channel parallel output

NOTE This technique/measure is referenced in Table A.7 of IEC 61508-2.

Aim: To detect random hardware failures (stuck-at failures), failures caused by external influences, timing failures, addressing failures, drift failures and transient failures.

Description: This is a dataflow-dependent multi-channel parallel output with independent outputs for the detection of random hardware failures. Failure detection is carried out via external comparators. If a failure occurs, the EUC is switched off directly. This measure is only effective if the dataflow changes during the diagnostic test interval.

A.6.4 Monitored outputs

NOTE This technique/measure is referenced in Table A.7 of IEC 61508-2.

Aim: To detect individual failures, failures caused by external influences, timing failures, addressing failures, drift failures (for analogue signals) and transient failures.

Description: This is a dataflow-dependent comparison of outputs with independent inputs to ensure compliance with a defined tolerance range (time, value). A detected failure cannot always be related to the defective output. This measure is only effective if the dataflow changes during the diagnostic test interval.

A.6.5 Input comparison/voting

NOTE This technique/measure is referenced in Tables A.7 and A.13 of IEC 61508-2.

Aim: To detect individual failures, failures caused by external influences, timing failures, addressing failures, drift failures (for analogue signals) and transient failures.

Description: This is a dataflow-dependent comparison of independent inputs to ensure compliance with a defined tolerance range (time, value). There will be 1 out of 2, 2 out of 3 or better redundancy. This measure is only effective if the dataflow changes during the diagnostic test interval.

A.7 Data paths (internal communication)

Global objective: To detect failures caused by a defect in the information transfer.

A.7.1 One-bit hardware redundancy

NOTE This technique/measure is referenced in Table A.8 of IEC 61508-2.

Aim: To detect all odd-bit failures, i.e. 50 % of all the possible bit failures in the data stream.

Description: The bus is extended by one line (bit) and this additional line (bit) is used to detect failures by parity checking.

A.7.2 Multi-bit hardware redundancy

NOTE This technique/measure is referenced in Table A.8 of IEC 61508-2.

Aim: To detect failures during the communication on the bus and in serial transmission links.

Description: The bus is extended by two or more lines (bits) and these additional lines (bits) are used in order to detect failures by Hamming code techniques.

A.7.3 Complete hardware redundancy

NOTE This technique/measure is referenced in Table A.8 of IEC 61508-2.

Aim: To detect failures during the communication by comparing the signals on two buses.

Description: The bus is doubled and the additional lines (bits) are used in order to detect failures.

A.7.4 Inspection using test patterns

NOTE This technique/measure is referenced in Table A.8 of IEC 61508-2.

Aim: To detect static failures (stuck-at failure) and cross-talk.

Description: This is a dataflow-independent cyclical test of data paths. It uses a defined test pattern to compare observations with the corresponding expected values.

The test pattern information, the test pattern reception, and test pattern evaluation must all be independent of each other. The EUC should not be inadmissibly influenced by the test pattern.

A.7.5 Transmission redundancy

NOTE This technique/measure is referenced in Table A.8 of IEC 61508-2.

Aim: To detect transient failures in bus communication.

Description: The information is transferred several times in sequence. The repetition is effective only against transient failures.

A.7.6 Information redundancy

NOTE This technique/measure is referenced in Table A.8 of IEC 61508-2.

Aim: To detect failures in bus communication.

Description: Data is transmitted in blocks, together with a calculated checksum for each block. The receiver then re-calculates the checksum of the received data and compares the result with the received checksum.

A.8 Power supply

Global objective: To detect or tolerate failures caused by a defect in the power supply.

A.8.1 Overvoltage protection with safety shut-off

NOTE This technique/measure is referenced in Table A.9 of IEC 61508-2.

Aim: To protect the safety-related system against overvoltage.

Description: Overvoltage is detected early enough that all outputs can be switched to a safe condition by the power-down routine or there is a switch-over to a second power unit.

Reference:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

A.8.2 Voltage control (secondary)

NOTE This technique/measure is referenced in Table A.9 of IEC 61508-2.

Aim: To monitor the secondary voltages and initiate a safe condition if the voltage is not in its specified range.

Description: The secondary voltage is monitored and a power-down is initiated, or there is a switch-over to a second power unit, if it is not in its specified range.

A.8.3 Power-down with safety shut-off

NOTE This technique/measure is referenced in Table A.9 of IEC 61508-2.

Aim: To shut off the power with all safety critical information stored.

Description: Overvoltage or undervoltage is detected early enough so that the internal state can be saved in non-volatile memory (if necessary), and so that all outputs can be set to a safe condition by the power-down routine, or that all outputs can be switched to a safe condition by the power-down routine, or there is a switch-over to a second power unit.

A.9 Temporal and logical program sequence monitoring

NOTE This group of techniques and measures is referenced in Tables A.15, A.16 and A.18 of IEC 61508-2.

Global objective: To detect a defective program sequence. A defective program sequence exists if the individual elements of a program (for example software modules, subprograms or

commands) are processed in the wrong sequence or period of time, or if the clock of the processor is faulty.

A.9.1 Watch-dog with separate time base without time-window

NOTE This technique/measure is referenced in Tables A.10 and A.11 of IEC 61508-2.

Aim: To monitor the behaviour and the plausibility of the program sequence.

Description: External timing elements with a separate time base (for example watch-dog timers) are periodically triggered to monitor the computer's behaviour and the plausibility of the program sequence. It is important that the triggering points are correctly placed in the program. The watch-dog is not triggered at a fixed period, but a maximum interval is specified.

A.9.2 Watch-dog with separate time base and time-window

NOTE This technique/measure is referenced in Tables A.10 and A.11 of IEC 61508-2.

Aim: To monitor the behaviour and the plausibility of the program sequence.

Description: External timing elements with a separate time base (for example watch-dog timers) are periodically triggered to monitor the computer's behaviour and the plausibility of the program sequence. It is important that the triggering points are correctly placed in the program. A lower and upper limit is given for the watch-dog timer. If the program sequence takes a longer or shorter time than expected, emergency action is taken.

A.9.3 Logical monitoring of program sequence

NOTE This technique/measure is referenced in Tables A.10 and A.11 of IEC 61508-2.

Aim: To monitor the correct sequence of the individual program sections.

Description: The correct sequence of the individual program sections is monitored using software (counting procedure, key procedure) or using external monitoring facilities. It is important that the checking points are placed in the program correctly.

A.9.4 Combination of temporal and logical monitoring of program sequences

NOTE This technique/measure is referenced in Tables A.10 and A.11 of IEC 61508-2.

Aim: To monitor the behaviour and the correct sequence of the individual program sections.

Description: A temporal facility (for example a watch-dog timer) monitoring the program sequence is retriggered only if the sequence of the program sections is also executed correctly.

A.9.5 Temporal monitoring with on-line check

NOTE This technique/measure is referenced in Tables A.10 and A.11 of IEC 61508-2.

Aim: To detect faults in the temporal monitoring.

Description: The temporal monitoring is checked at start-up, and a start is only possible if the temporal monitoring operates correctly. For example, a heat sensor could be checked by a heated resistor at start-up.

A.10 Ventilation and heating

NOTE This group of techniques and measures is referenced in Tables A.16 and A.18 of IEC 61508-2.

Global objective: To control failures in the ventilation or heating, and/or their monitoring, if this is safety-related.

A.10.1 Temperature sensor

Aim: To detect over- or under-temperature before the system begins to operate outside specification.

Description: A temperature sensor monitors temperature at the most critical points of the E/E/PE safety-related system. Before the temperature leaves the specified range, emergency action is taken.

A.10.2 Fan control

Aim: To detect incorrect operation of the fans.

Description: The fans are monitored for correct operation. If a fan is not working properly, maintenance (or ultimately, emergency) action is taken.

A.10.3 Actuation of the safety shut-off via thermal fuse

Aim: To shut off the safety-related system before the system works outside of its thermal specification.

Description: A thermal fuse is used to shut off the safety-related system. For a PES, the shut-off is introduced by a power-down routine which stores all information necessary for emergency action.

A.10.4 Staggered message from thermo-sensors and conditional alarm

Aim: To indicate that the safety-related system is working outside its thermal specification.

Description: The temperature is monitored and an alarm is raised if the temperature is outside of a specified range.

A.10.5 Connection of forced-air cooling and status indication

Aim: To prevent overheating by forced-air cooling.

Description: The temperature is monitored and forced-air cooling is introduced if the temperature is higher than a specified limit. The user is informed of the status.

A.11 Communication and mass-storage

Global objective: To control failures during communication with external sources and mass-storage.

A.11.1 Separation of electrical energy lines from information lines

NOTE This technique/measure is referenced in Table A.16 of IEC 61508-2.

Aim: To minimise cross-talk induced by high currents in the information lines.

Description: Electrical energy supply lines are separated from the lines carrying the information. The electrical field which could induce voltage spikes on the information lines decreases with distance.

A.11.2 Spatial separation of multiple lines

NOTE This technique/measure is referenced in Tables A.16 of IEC 61508-2.

Aim: To minimise cross-talk induced by high currents in multiple lines.

Description: Lines carrying duplicated signals are separated from each other. The electrical field which could induce voltage spikes on the multiple lines decreases with the distance. This measure also reduces common cause failures.

A.11.3 Increase of interference immunity

NOTE This technique/measure is referenced in Tables A.16 and A.18 of IEC 61508-2.

Aim: To minimise electromagnetic interference on the safety-related system.

Description: Design techniques such as shielding and filtering are used to increase the interference immunity of the safety-related system to electromagnetic disturbances which may be radiated or conducted on power or signal lines, or result from electrostatic discharges.

NOTE See [16] and [17] for immunity requirements for safety-related systems and for equipment intended to perform safety-related functions (functional safety) in industrial applications.

References:

IEC/TR 61000-5-2:1997, *Electromagnetic compatibility (EMC) – Part 5: Installation and mitigation guidelines – Section 2: Earthing and cabling*

Principles and Techniques of Electromagnetic Compatibility, Second Edition, C. Christopoulos, CRC Press, 2007, ISBN-10: 0849370353, ISBN-13: 978-0849370359

Noise Reduction Techniques in Electronic Systems. H. W. Ott, John Wiley Interscience, 2nd Edition, 1988

EMC for Product Designers. T. Williams, Newnes, 2007, ISBN 0750681705

Grounding and Shielding Techniques in Instrumentation, 3rd edition, R. Morrison . Wiley-Interscience, New York, 1986, ISBN-10: 0471838055, ISBN-13: 978-0471838050

A.11.4 Antivalent signal transmission

NOTE This technique/measure is referenced in Tables A.7 and A.16 of IEC 61508-2.

Aim: To detect the same induced voltages in multiple signal transmission lines.

Description: All duplicated information is transmitted with antivalent signals (for example logic 1 and 0). Common cause failures (for example by electromagnetic emission) can be detected by an antivalent comparator.

Reference:

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch. 20. Lfg. V/93, Erich Schmidt Verlag, Bielefeld. http://www.bgia-handbuchdigital.de/330220

A.12 Sensors

Global objective: To control failures in the sensors of the safety-related system.

A.12.1 Reference sensor

NOTE This technique/measure is referenced in Table A.13 of IEC 61508-2.

Aim: To detect the incorrect operation of a sensor.

Description: An independent reference sensor is used to monitor the operation of a process sensor. All input signals are checked at suitable time intervals by the reference sensor to detect failures of the process sensor.

Reference:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

A.12.2 Positive-activated switch

NOTE This technique/measure is referenced in Table A.13 of IEC 61508-2.

Aim: To open a contact by a direct mechanical connection between switch cam and contact.

Description: A positive-activated switch opens its normally closed contacts by a direct mechanical connection between switch cam and contact. This ensures that whenever the switch cam is in the operated position, the switch contacts must be open.

Reference:

Verriegelung beweglicher Schutzeinrichtungen. F. Kreutzkampf, K. Becker, Sicherheitstechnisches Informations- und Arbeitsblatt 330210, BIA-Handbuch. 1. Lfg. IX/85, Erich Schmidt Verlag, Bielefeld

A.13 Final elements (actuators)

Global objective: To control failures in the final elements in the safety-related system.

A.13.1 Monitoring

NOTE This technique/measure is referenced in Table A.14 of IEC 61508-2.

Aim: To detect the incorrect operation of an actuator.

Description: The operation of the actuator is monitored (for example by the positively activated contacts of a relay, see monitoring of relay contacts in A.1.2). The redundancy introduced by this monitoring can be used to trigger emergency action.

References:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Verriegelungseinrichtungen. F. Kreutzkampf, W. Hertel, Sicherheitstechnisches Informations- und Arbeitsblatt 330212, BIA-Handbuch. 17. Lfg. X/91, Erich Schmidt Verlag, Bielefeld

A.13.2 Cross-monitoring of multiple actuators

NOTE This technique/measure is referenced in Table A.14 of IEC 61508-2.

Aim: To detect faults in actuators by comparing the results.

Description: Each multiple actuator is monitored by a different hardware channel. If a discrepancy occurs, emergency action is taken.

A.14 Measures against the physical environment

NOTE This technique/measure is referenced in Tables A.16 and A.18 of IEC 61508-2

Aim: To prevent influences of the physical environment (water, dust, corrosive substances) causing failures.

Description: The enclosure of the equipment is designed to withstand the expected environment.

Reference:

IEC 60529:1989, *Degrees of protection provided by enclosures (IP Code)*

Annex B
(informative)

Overview of techniques and measures for E/E/PE safety related systems: avoidance of systematic failures (see IEC 61508-2 and IEC 61508-3)

NOTE Many techniques in this annex are applicable to software but have not been duplicated in Annex C.

B.1 General measures and techniques

B.1.1 Project management

NOTE This technique/measure is referenced in Tables B.1 to B.6 of IEC 61508-2.

Aim: To avoid failures by adoption of an organisational model and rules and measures for development and testing of safety-related systems.

Description: The most important and best measures are

- the creation of an organisational model, especially for quality assurance which is set down in a quality assurance handbook; and
- the establishment of regulations and measures for the creation and validation of safety-related systems in cross-project and project-specific guidelines.

A number of important basic principles are set down in the following:

- definition of a design organisation:
  - tasks and responsibilities of the organisational units,
  - authority of the quality assurance departments,
  - independence of quality assurance (internal inspection) from development;
- definition of a sequence plan (activity models):
  - determination of all activities which are relevant during execution of the project including internal inspections and their scheduling,
  - project update;
- definition of a standardised sequence for an internal inspection:
  - planning, execution and checking of the inspection (inspection theory),
  - releasing mechanisms for subproducts,
  - the safekeeping of repeat inspections;
- configuration management:
  - administration and checking of versions,
  - detection of the effects of modifications,
  - consistency inspections after modifications;
- introduction of a quantitative assessment of quality assurance measures:
  - requirement acquisition,
  - failure statistics;
- introduction of computer-aided universal methods, tools and training of personnel.

References:

ISO 9001:2008, *Quality management systems – Requirements*

ISO/IEC 15504 (all parts), *Information technology – Process assessment*

CMMI: Guidelines for Process Integration and Product Improvement, 2nd Edition. M. B. Chrissis, M. Konrad, S. Shrum, Addison-Wesley Professional, 2006, ISBN-10: 0-3212-7967-0, ISBN-13: 978-0-3212-7967-5

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0

B.1.2 Documentation

NOTE 1 This technique/measure is referenced in Tables B.1 to B.6 of IEC 61508-2.

NOTE 2 See also Clause 5 and Annex A of IEC 61508-1.

Aim: To avoid failures and facilitate assessment of system safety, by documenting each step during development.

Description: The operational capacity and safety, as well as the care taken in development by all parties involved, has to be demonstrated during assessment. In order to be able to show the development care, and in order to guarantee the verification of the evidence of safety at any time, special importance is given to the documentation.

Important common measures are the introduction of guidelines and computer aid, i.e.

- guidelines, which
  - specify a grouping plan;
  - ask for checklists for the contents; and
  - determine the form of the document;
- administration of the documentation with the help of a computer-aided and organised project library.

Individual measures are:

- separation in the documentation
  - of the requirements,
  - of the system (user-documentation) and
  - of the development (including internal inspection);
- grouping of the development documentation according to the safety lifecycle;
- definition of standardised documentation modules, from which the documents can be compiled;
- clear identification of the constituent parts of the documentation;
- formalised revision update;
- selection of clear and intelligible means of description:
  - formal notation for determinations;
  - natural language for introductions, justifications and representations of intentions;

- graphical representations for examples;
- semantic definition of graphical elements; and
- directories of specialised words.

References:

IEC 61506:1997, *Industrial-process measurement and control – Documentation of application software*

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

B.1.3 Separation of E/E/PE system safety functions from non-safety functions

NOTE This technique/measure is referenced in Tables B.1 and B.6 of IEC 61508-2.

Aim: To prevent the non-safety-related part of the system from influencing the safety-related part in undesired ways.

Description: In the specification, it should be decided whether a separation of the safety-related systems and non-safety-related systems is possible. Clear specifications should be written for the interfacing of the two parts. A clear separation reduces the effort for testing the safety-related systems.

Reference:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

B.1.4 Diverse hardware

NOTE This technique/measure is referenced in Tables A.15, A.16 and A.18 of IEC 61508-2.

Aim: To detect systematic failures during operation of the EUC, using diverse components with different rates and types of failures.

Description: Different types of components are used for the diverse channels of a safety-related system. This reduces the probability of common cause failures (for example overvoltage, electromagnetic interference), and increases the probability of detecting such failures.

Existence of different means of performing a required function, for example different physical principles, offer other ways of solving the same problem. There are several types of diversity. Functional diversity employs the use of different approaches to achieve the same result.

Reference :

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

B.2 E/E/PE system design requirements specification

Global objective: To produce a specification which is, as far as possible, complete, free from mistakes, free from contradiction, and simple to verify.

B.2.1 Structured specification

NOTE This technique/measure is referenced in Tables B.1 and B.6 of IEC 61508-2.

Aim: To reduce complexity by creating a hierarchical structure of partial requirements. To avoid interface failures between the requirements.

Description: This technique structures the functional specification into partial requirements such that the simplest possible, visible relations exist between the latter. This analysis is successively refined until small clear partial requirements can be distinguished. The result of the final refinement is a hierarchical structure of partial requirements which provide a framework for the specification of the complete requirements. This method emphasises the interfaces of the partial requirements and is particularly effective for avoiding interface failures.

References:

ESA PSS 05-02, *Guide to the user requirements definition phase*, Issue 1, Revision 1, ESA Board for Software Standardisation and Control (BSSC), ESA, Paris, March 1995, ftp://ftp.estec.esa.nl/pub/wm/wme/bssc/PSS0502.pdf

Structured Analysis and System Specification. T. De Marco, Yourdon Press, Englewood Cliffs, 1979, ISBN-10: 0138543801, ISBN-13: 978-0138543808

B.2.2 Formal methods

NOTE 1 See C.2.4 for details of specific formal methods.

NOTE 2 This technique/measure is referenced in Tables B.1, B.2 and B.6 of IEC 61508-2.

Aim: Formal methods transfers the principles of mathematical reasoning to the specification and implementation of technical systems therefore increase the completeness, consistency or correctness of a specification or implementation.

Description: Formal methods provide a means of developing a description of a system during specification and/or implementation phase. These formal descriptions are mathematical models of the system function and/or structure.

Therefore unambiguous system description could be achieved (e.g. any state of an automaton is described by its initial state, inputs and the transition equations of the automaton) which increase understanding of the underlying system.

Choosing a suitable formal method is a difficult undertaking requiring full understanding of system, its development process and the range of mathematical models that could possibly be used (see following notes).

NOTE 3 The theorems of interest of the model (properties) represent guaranties about the system which provides far more confidence than simulation i.e. observing selected actions of the system.

NOTE 4 The disadvantages of formal methods can be:

– Fixed level of abstraction;
– limitations to capture all functionality that is relevant at the given stage;
– difficulty that implementation engineers have to understand the model;
– high efforts to develop, analyze and maintain model over the lifecycle of system;
– availability of efficient tools which support the building and analysis of model;
– availability of staff capable to develop and analyze model.

NOTE 5 The formal methods community's focus clearly was been the modeling of the target function of system often deemphasizing the fault robustness of a system. Therefore respective formal methods including system robustness has to be selected.

Reference:

Formal Specification: Techniques and Applications. N.Nissanke, Springer-Verlag Telos, 1999, ISBN-10: 1852330023

B.2.3 Semi-formal methods

NOTE 1 IEC 61508- 3 Table B.7 extends this Annex B list with other semi-formal software related techniques. art 3 lists:

– logic/function block diagrams: described in IEC 61131-3;
– sequence diagrams: described in IEC 61131-3;
– data flow diagrams: see C.2.2;
– finite state machines/state transition diagrams: see B.2.3.2;
– time Petri nets: see B.2.3.3;
– entity-relationship-attribute Data models: see B.2.4.4;
– message sequence charts: see C.2.14;
– decision/truth tables: see C.6.1.

Aim: To express parts of a specification unambiguously and consistently, so that some mistakes, omissions and wrong behaviour can be detected.

NOTE 2 This technique/measure is referenced in Tables B.1, B.2 and B.6 of IEC 61508-2 and in Tables A.1, A.2, A.4, B.7, C.1, C.2, C.4 and C.17 of IEC 61508-3.

B.2.3.1 General

Aim: To prove that the design meets its specification.

Description: Semi-formal methods provide a means of developing a description of a system at some stage in its development, i.e. specification, design or coding. The description can in some cases be analysed by machine or animated to display various aspects of the system behaviour. Animation can give extra confidence that the system meets the real requirement as well as the specified requirement.

Two semi-formal methods are described in the following subclauses.

B.2.3.2 Finite state machines / state transition diagrams

NOTE This technique/measure is referenced in Tables B.5, B.7, C.15 and C.17 of IEC 61508-3.

Aim: To model, verify, specify or implement the control structure of a system.

Description: Many systems can be described in terms of their states, their inputs, and their actions. Thus when in state S1, on receiving input I a system might carry out action A and move to state S2. By describing a system's actions for every input in every state we can describe a system completely. The resulting model of the system is called a finite state machine (or finite state automata). It is often drawn as a so-called state transition diagram showing how the system moves from one state to another, or as a matrix in which the dimensions are state and input, and the matrix cells contain the action and new state resulting from receiving the input when in the given state.

Where a system is complicated or has a natural structure, this can be reflected in a layered finite state machine. A Statechart is a type of state transition diagram in which nested states are allowed (the object state splits into two or more sub-states which can evolve in parallel, and possibly recombine into a single state at some point); this adds to the expressive power of the state transition notation but can add extra complexity which may be undesirable in a safety related system. Statecharts have a formal (mathematical) specification. State transition diagrams can apply to the whole system or to some object or element within it.

A specification or design expressed as a finite state machine can be checked for

- completeness (the system or object must have an action and new state for every input in every state);
- consistency (only one state transition is possible for each state/input pair); and
- reachability (whether or not it is possible to get from one state to another by any sequence of inputs); and
- absence of endless loops or dead-end states; etc.

These are important properties for critical systems. Tools to support these checks are easily developed and various models based on finite state automata (formal languages, Petri nets, Markov graphs, etc.) can be used. Algorithms also exist that allow the automatic generation of test cases for verifying a finite state machine implementation or for animating a finite state machine model. State transition diagrams and Statecharts are widely supported by tools which allow the diagrams to be drawn and checked, and which will generate code to implement the described state machine.

They can also be used for failure probability calculations, see B.6 and C.6.

References:

Introduction to Automata Theory, Languages, and Computation (3rd Edition). J. Hopcroft, R. Motwani, J. Ullman, Addison-Wesley Longman Publishing Co, 2006, ISBN: 0321462254

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès – Lavoisier, 2009, ISBN: 978-2-7462-1991-5

B.2.3.3 Time Petri nets

NOTE This technique/measure is referenced in Tables B.5, B.7, C.15 and C.17 of IEC 61508-3.

Aim: To model relevant aspects of the system behaviour and to assess and possibly improve safety and operational requirements through analysis and re-design.

Description: Petri nets are a particular case of finite state automata. They belong to a class of graph theoretic models which are suitable for representing information and control flow in systems that exhibit concurrency and have asynchronous behaviour.

A Petri net is a network of places and transitions. The places may be "marked" or "unmarked". A transition is "enabled" when all the input places to it are marked. When enabled, it is permitted (but not obliged) to "fire". If it fires, the input places to the transition become unmarked, and each output place from the transition is marked instead.

The potential hazards can be represented as particular states (markings) in the model. The Petri net model can be extended to allow for timing features of the system. Although "classical" Petri nets concentrate on control flow aspects, several extensions have been proposed to incorporate data flow into the model.

They also provide a very efficient support for performing Monte Carlo simulation in order to achieve failure probability calculations, see B.6.6.8.

References:

Timed Petri Nets: Theory and Application. Jiacun Wang, Springer, 1998, ISBN 0792382706

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès – Lavoisier, 2009, ISBN: 978-2-7462-1991-5

B.2.4 Computer-aided specification tools

NOTE This technique/measure is referenced in Tables B.1 and B.6 of IEC 61508-2 and in Tables A.1, A.2, C.1 and C.2 of IEC 61508-3.

B.2.4.1 General

Aim: To use formal specification techniques to facilitate automatic detection of ambiguity and completeness.

Description: The technique produces a specification in the form of a database that can be automatically inspected to assess consistency and completeness. The specification tool can animate various aspects of the specified system to the user. In general, the technique supports not only the creation of the specification but also of the design and other phases of the project lifecycle. Specification tools can be classified according to the following subclauses.

B.2.4.2 Tools oriented towards no specific method

Aim: To help the user write a good specification by providing prompts and links between related parts.

Description: The specification tool takes over some routine work from the user and supports the project management. It does not enforce any particular specification methodology. The relative independence with regard to method allows users a great deal of freedom but also gives them little of the specialised support necessary when creating specifications. This makes familiarisation with the system more difficult.

B.2.4.3 Model orientated procedure with hierarchical analysis

Aim: To avoid incompleteness, ambiguity and contradiction in the specification, e.g. supporting the user writing a good specification by ensuring consistency between descriptions of actions and data at various levels of abstraction.

Description: This method gives a functional representation of the desired system (structured analysis) at various levels of abstraction (degree of precision). There is a huge arsenal of such models: finite automata are a class of such models widely used to describe the evolution of discrete/digital systems. Differential equations are similar in spirit and aim at continuous/analogue systems. The analysis at various levels acts on both actions and data. Assessment of ambiguity and completeness is possible between hierarchical levels as well as between two functional units (modules) on the same level (e.g. any state of a system model is described by its initial state, inputs and the transition equations of the automaton).

NOTE Issues of model based descriptions may be the level of abstraction, limitations to capture all functionality that is relevant at the given stage, difficulty that practitioners have to understand the model (from reading the syntax to understanding), high efforts to develop, analyze and maintain a model over the lifecycle of a system, availability of efficient tools which support the building and analysis of model (development of such tools is certainly a high effort undertaking) and availability of staff capable to develop and analyze models.

Reference:

System requirements analysis. Jeffrey O. Grady, Academic Press, 2006, ISBN 012088514X, 9780120885145

B.2.4.4 Entity-relationship-attribute data models

Aim: To help the user write a good specification by focusing on entities within the system and relationships between them.

Description: The desired system is described as a collection of objects and their relationships. The tool enables one to determine which relationships can be interpreted by the

system. In general, the relationships permit a description of the hierarchical structure of the objects, the data flow, the relationships between the data, and which data are subject to certain manufacturing processes. The classical procedure has been extended for process control applications. Inspection capabilities and support for the user depend on the variety of relationships illustrated. On the other hand, a large number of representation possibilities makes the application of this technique complex.

Reference:

Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle. Karl Eugene Wiegers, Microsoft Press, 2003, ISBN 0735618798, 9780735618794

B.2.4.5 Incentive and answer

Aim: To help the user write a good specification by identifying stimulus-response relationships.

Description: The relationships between the objects of the system are specified in a notation of "incentives" and "answers". A simple and easily expanded language is used which contains language elements which represent objects, relationships, characteristics and structures.

B.2.5 Checklists

NOTE This technique/measure is referenced in Tables B.1, B.2 and B.6 of IEC 61508-2 and in Tables A.10, B.8, C.10 and C.18 of IEC 61508-3.

Aim: To draw attention to, and manage critical appraisal of, all important aspects of the system by safety lifecycle phase, ensuring comprehensive coverage without laying down exact requirements.

Description: A set of questions to be answered by the person performing the checklist. Many of the questions are of a general nature and the assessor must interpret them as seems most appropriate to the particular system being assessed. Checklists can be used for all phases of the overall, E/E/PE system safety and software safety lifecycles and are particularly useful as a tool to aid the functional safety assessment.

To accommodate wide variations in systems being validated, most checklists contain questions which are applicable to many types of system. As a result, there may well be questions in the checklist being used which are not relevant to the system being dealt with and which should be ignored. Equally there may be a need, for a particular system, to supplement the standard checklist with questions specifically directed at the system being dealt with.

In any case it should be clear that the use of checklists depends critically on the expertise and judgement of the engineer selecting and applying the checklist. As a result, the decisions taken by the engineer, with regard to the checklist(s) selected, and any additional or superfluous questions, should be fully documented and justified. The objective is to ensure that the application of the checklists can be reviewed and that the same results will be achieved unless different criteria are used.

The object in completing a checklist is to be as concise as possible. When extensive justification is necessary this should be done by reference to additional documentation. Pass, fail and inconclusive, or some similar restricted set of responses should be used to document the results for each question. This conciseness greatly simplifies the procedure of reaching an overall conclusion as to the results of the checklist assessment.

References:

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

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122, 9780471469124

Software Quality Assurance: From Theory to Implementation. Daniel Galin, Pearson Education, 2004, ISBN 0201709457, 9780201709452

IEC 61346 (all parts), *Industrial systems, installations and equipment and industrial products – Structuring principles and reference designation*

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

Risk Assessment and Risk Management for the Chemical Process Industry. H.R. Greenberg, J.J. Cramer, John Wiley and Sons, 1991, ISBN 0471288829, 9780471288824

B.2.6 Inspection of the specification

NOTE This technique/measure is referenced in Tables B.1 and B.6 of IEC 61508-2.

Aim: To avoid incompleteness and contradiction in the specification.

Description: Inspection is a general technique in which various qualities of a specification document are assessed by an independent team. The inspection team puts questions to the creator, who must answer them satisfactorily. The examination should (if possible) be carried out by a team that was not involved in the creation of the specification. The required degree of independence is determined by the safety integrity levels demanded of the system. The independent inspectors should be able to reconstruct the operational function of the system in an indisputable manner without referring to any further specifications. They must also check that all relevant safety and technical aspects in the operational and organisational measures are covered. This procedure has proved itself to be very effective in practice.

References:

IEC 61160:2005, *Design review*

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122, 9780471469124

Software Quality Assurance: From Theory to Implementation. D. Galin, Pearson Education, 2004, ISBN 0201709457, 9780201709452

B.3 E/E/PE system design and development

Global objective: To produce a stable design of the safety-related system in conformance with the specification.

B.3.1 Observance of guidelines and standards

NOTE This technique/measure is referenced in Table B.2 of IEC 61508-2.

Aim: To observe application sector standards (not specified in this standard).

Description: Guidelines should be complied with during the design of the safety-related system. These guidelines should firstly lead to safety-related systems which are practically

free from failures, and secondly facilitate the subsequent safety validation. They can be universally valid, specific to a project, or specific only to a single phase.

References:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

B.3.2 Structured design

NOTE This technique/measure is referenced in Tables B.2 and B.6 of IEC 61508-2.

Aim: To reduce complexity by creating a hierarchical structure of partial requirements. To avoid interface failures between the requirements. To simplify verification.

Description: When designing the hardware, specific criteria or methods should be used. For example, the following might be required:

– a hierarchically structured circuit design;

– use of manufactured and tested circuit parts.

Similarly, when designing the software, the use of structure charts enables an unambiguous structure of the software modules to be created. This structure shows how the modules relate to each other, the precise data which passes between modules, and the precise controls that exist between modules.

References:

IEC 61346 (all parts), *Industrial systems, installations and equipment and industrial products – Structuring principles and reference designation*

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

Software Design. D. Budgen, Pearson Education, 2003, ISBN 0201722194, 9780201722192

An Overview of JSD, J. R. Cameron, IEEE Trans SE-12 No. 2, February 1986

Structured Development for Real-Time Systems (3 Volumes). P. T. Yourdon, P. T. Yourdon Press, 1985

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward, S. J. Mellor, Yourdon Press, 1985

Applications and Extensions of SADT. D. T. Ross, Computer, 25-34, April 1985

Essential Systems Analysis. St. M. McMenamin, F. Palmer, Yourdon Inc, 1984

Structured Analysis (SA): A language for communicating ideas. D. T. Ross, IEEE Trans. Software Eng, Vol. SE-3 (1), 16-34

B.3.3 Use of well-tried components

NOTE This technique/measure is referenced in Tables B.2 and B.6 of IEC 61508-2.

Aim: To reduce the risk of numerous first time and undetected faults by the use of components with specific characteristics.

Description: The selection of well-tried components is carried out by the manufacturer, with regard to safety according to the reliability of the elements (for example the use of operationally tested physical units to meet high safety requirements, or the storing of safety-

related programs in safe memories only). The safety of memories can refer to unauthorised access as well as environmental influences (electromagnetic compatibility, radiation, etc) and the response of the elements in the event of a failure occurring.

References:

IEC 61163-1:2006, *Reliability stress screening – Part 1: Repairable assemblies manufactured in lots*

B.3.4 Modularisation

NOTE This technique/measure is referenced in Tables B.2 and B.6 of IEC 61508-2.

Aim: To reduce complexity and avoid failures, related to interfacing between subsystems.

Description: Every subsystem, at all levels of the design, is clearly defined and is of restricted size (only a few functions). The interfaces between subsystems are kept as simple as possible and the cross-section (i.e. shared data, exchange of information) is minimised. The complexity of individual subsystems is also restricted.

References:

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122, 9780471469124

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

Software Reliability – Principles and Practices. G. J. Myers, Wiley-Interscience, New York, 1976, ISBN-10: 0471627658, ISBN-13: 978-0471627654

B.3.5 Computer-aided design tools

NOTE This technique/measure is referenced in Tables B.2 and B.6 of IEC 61508-2 and in Tables A.4 and C.4 of IEC 61508-3.

Aim: To carry out the design procedure more systematically. To include appropriate automatic construction elements which are already available and tested.

Description: Computer-aided design tools (CAD) should be used during the design of both hardware and software when available and justified by the complexity of the system. The correctness of such tools should be demonstrated by specific testing, by an extensive history of satisfactory use, or by independent verification of their output for the particular safety-related system that is being designed.

Support tools should be selected for their degree of integration. 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 minimizing the possibility of introducing human error in the reworking of intermediate results.

References:

Overview of Technology Computer-Aided Design Tools and Applications in Technology Development, *Manufacturing and Design*. W. Fichtner, Journal of Computational and Theoretical Nanoscience, Volume 5, Number 6, June 2008, pp. 1089-1105(17)

The Electromagnetic Data Exchange: Much more than a Common Data Format. P.E. Frandsen et al. In *Proceeding of the 2nd European Conference on Antennas and Propagation*. The Institution of Engineering and Technology (IET), 2007, ISBN 978-0-86341-842-6

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

B.3.6 Simulation

NOTE This technique/measure is referenced in Tables B.2, B.5 and B.6 of IEC 61508-2.

Aim: To carry out a systematic and complete inspection of an electrical/electronic circuit, of both the functional performance and the correct dimensioning of the components.

Description: The function of the safety-related system circuit is simulated on a computer via a software behavioural model. Individual components of the circuit each have their own simulated behaviour, and the response of the circuit in which they are connected is examined by looking at the marginal data of each component.

B.3.7 Inspection (reviews and analysis)

NOTE This technique/measure is referenced in Tables B.2 and B.6 of IEC 61508-2.

Aim: To reveal discrepancies between the specification and implementation.

Description: Specified functions of the safety-related system are examined and evaluated to ensure that the safety-related system conforms to the requirements given in the specification. Any points of doubt concerning the implementation and use of the product are documented so they may be resolved. In contrast to a walk-through, the author is passive and the inspector is active during the inspection procedure.

References:

IEC 61160:2005, *Design Review*

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122, 9780471469124

ANSI/IEE 1028:1997, *IEEE Standard for software reviews*

Dependability of Critical Computer Systems 3. P. G. Bishop et al., Elsevier Applied Science, 1990, ISBN 1-85166-544-7

B.3.8 Walk-through

NOTE This technique/measure is referenced in Tables B.2 and B.6 of IEC 61508-2.

Aim: To reveal discrepancies between the specification and implementation.

Description: Specified functions of the safety-related system draft are examined and evaluated to ensure that the safety-related system complies with the requirements given in the specification. Doubts and potential weak points concerning the realisation and use of the product are documented so that they may be resolved. In contrast to an inspection, the author is active and the inspector is passive during the walk-through.

References:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

ANSI/IEEE 1028:1997, *IEEE Standard for software reviews*

Dependability of Critical Computer Systems 3. P. G. Bishop et al., Elsevier Applied Science, 1990, ISBN 1-85166-544-7

Methodisches Testen von Programmen. G. J. Myers, Oldenbourg Verlag, München, Wien, 1987

B.4 E/E/PE system operation and maintenance procedures

Global objective: To develop procedures which help to avoid failures during the operation and maintenance of the safety-related system.

B.4.1 Operation and maintenance instructions

NOTE This technique/measure is referenced in Table B.4 of IEC 61508-2.

Aim: To avoid mistakes during operation and maintenance of the safety-related system.

Description: User instructions contain essential information on how to use and how to maintain the safety-related system. In special cases, these instructions will also include examples on how to install the safety-related system in general. All instructions must be easily understood. Figures and schematics should be used to describe complex procedures and dependencies.

Reference: *Guidelines for Safe Automation of Chemical Processes*. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

B.4.2 User friendliness

NOTE This technique/measure is referenced in Table B.4 of IEC 61508-2.

Aim: To reduce complexity during operation of the safety-related system.

Description: The correct operation of the safety-related system may depend to some degree on human operation. By considering the relevant system design and the design of the workplace, the safety-related system developer must ensure that

- the need for human intervention is restricted to an absolute minimum;
- the necessary intervention is as simple as possible;
- the potential for harm from operator error is minimised;
- the intervention facilities and indication facilities are designed according to ergonomic requirements;
- the operator facilities are simple, well labelled and intuitive to use;
- the operator is not overstrained, even in extreme situations;
- training on intervention procedures and facilities is adapted to the level of knowledge and motivation of the trainee user.

B.4.3 Maintenance friendliness

NOTE This technique/measure is referenced in Table B.4 of IEC 61508-2.

Aim: To simplify maintenance procedures of the safety-related system and to design the necessary means for effective diagnosis and repair.

Description: Preventive maintenance and repair is often carried out under difficult circumstances and under pressure from deadlines. Therefore, the safety-related system developer should ensure that

- safety-related maintenance measures are necessary as seldom as possible or even, ideally, not necessary at all;
- sufficient, sensible and easy-to-handle diagnosis tools are included for those repairs that are unavoidable – tools should include all necessary interfaces;
- if separate diagnosis tools have to be developed or obtained, then these should be available on time.

B.4.4 Limited operation possibilities

NOTE This technique/measure is referenced in Tables B.4 and B.6 of IEC 61508-2.

Aim: To reduce the operation possibilities for the normal user.

Description: This approach reduces the operation possibilities by

- limiting the operation within special operating modes, for example by key switches;
- limiting the number of operating elements;
- limiting the number of generally possible operating modes.

Reference:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

B.4.5 Operation only by skilled operators

NOTE This technique/measure is referenced in Tables B.4 and B.6 of IEC 61508-2.

Aim: To avoid operating failures caused by misuse.

Description: The safety-related system operator is trained to a level which is appropriate to the complexity and safety integrity level of the safety-related system. Training includes studying the background of the production process and knowing the relationship between the safety-related system and the EUC.

Reference:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

B.4.6 Protection against operator mistakes

NOTE This technique/measure is referenced in Tables B.4 and B.6 of IEC 61508-2.

Aim: To protect the system against all classes of operator mistakes.

Description: Wrong inputs (value, time, etc) are detected via plausibility checks or monitoring of the EUC. To integrate these facilities into the design, it is necessary to state at a very early stage which inputs are possible and which are permissible.

B.4.7 (Not used)

B.4.8 Modification protection

NOTE This technique/measure is referenced in Tables A.17 and A.18 of IEC 61508-2.

Aim: To protect the safety-related system against hardware modifications by technical means.

Description: Modifications or manipulations are detected automatically, for example by plausibility checks for the sensor signals, detection by the technical process and by automatic start-up tests. If a modification is detected, then emergency action is taken.

B.4.9 Input acknowledgement

NOTE This technique/measure is referenced in Tables A.17 and A.18 of IEC 61508-2.

Aim: A mistake during operation is detected by the operator himself before activating the EUC.

Description: An input to the EUC via the safety-related system is echoed to the operator before being sent to the EUC so that the operator has the possibility to detect and correct a mistake. As well as abnormal, unprovoked personnel action, the system design should consider top/bottom speed limits and direction of human reaction. This would avoid, for example, the operator pressing keys faster than expected, causing the system to read a double keystroke as a single one, or a key to be pressed twice because the system (display) was too slow to react to the first instance. The same key stroke should not be valid more than once in succession for critical data entry; pressing the "enter" or "yes" key unlimited times must not lead to an unsafe action of the system.

Time-out procedures should be included with multiple choice questions (yes/no, etc.) to cater for when the operator may not make up his mind and leave the system waiting.

Ability to reboot a safety-related PES makes the system vulnerable unless both software/hardware are designed with such occasions in mind.

B.5 E/E/PE system integration

Global objective: To avoid failures during the integration phase and to reveal any failures that are made during this and previous phases.

B.5.1 Functional testing

NOTE This technique/measure is referenced in Tables B.3 and B.5 of IEC 61508-2 and in Tables A.5, A.6, A.7, C.5, C.6 and C.7 of IEC 61508-3.

Aim: To reveal failures during the specification and design phases. To avoid failures during implementation and the integration of software and hardware.

Description: During the functional tests, reviews are carried out to see whether the specified characteristics of the system have been achieved. The system is given input data which adequately characterises the normally expected operation. The outputs are observed and their response is compared with that given by the specification. Deviations from the specification and indications of an incomplete specification are documented.

Functional testing of electronic components designed for a multi-channel architecture usually involves the manufactured components being tested with pre-validated partner components. In addition to this, it is recommended that the manufactured components be tested in combination with other partner components of the same batch, in order to reveal common mode faults which would otherwise have remained masked.

Also, the working capacity of the system has to be sufficient, see guidance in C.5.20.

References:

Software Testing and Quality Assurance. K. Naik, P. Tripathy, Wiley Interscience, 2008, Print ISBN: 9780471789116 Online ISBN: 9780470382844

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122, 9780471469124

Practical Software Testing: A Process-oriented Approach. I. Burnstein, Springer, 2003, ISBN 0387951318, 9780387951317

Dependability of Critical Computer Systems 3. P. G. Bishop et al., Elsevier Applied Science, 1990, ISBN 1-85166-544-7

B.5.2 Black-box testing

NOTE This technique/measure is referenced in Tables B.3, B.5 and B.6 of IEC 61508-2 and in Tables A.5, A.6, A.7, C.5, C.6 and C.7 of IEC 61508-3.

Aim: To check the dynamic behaviour under real functional conditions. To reveal failures to meet functional specification, and to assess utility and robustness.

Description: The functions of a system or program are executed in a specified environment with specified test data which is derived systematically from the specification according to established criteria. This exposes the behaviour of the system and permits a comparison with the specification. No knowledge of the internal structure of the system is used to guide the testing. The aim is to determine whether the functional unit carries out correctly all the functions required by the specification. The technique of forming equivalence classes is an example of the criteria for blackbox test data. The input data space is subdivided into specific input value ranges (equivalence classes) with the aid of the specification. Test cases are then formed from the

- data from permissible ranges;
- data from inadmissible ranges;
- data from the range limits;
- extreme values;
- and combinations of the above classes.

Other criteria can be effective in order to select test cases in the various test activities (module test, integration test and system test). For example, the criterion "extreme operational conditions" is relied upon for the system test within the framework of a validation.

References:

Software Testing and Quality Assurance. K. Naik, P. Tripathy, Wiley Interscience, 2008, Print ISBN: 9780471789116 Online ISBN: 9780470382844

Essentials of Software Engineering. Frank F. Tsui, Orlando Karam. Jones & Bartlett, 2006. ISBN 076373537X, 9780763735371

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122, 9780471469124

Systematic Software Testing. Rick D. Craig, Stefan P. Jaskiel. Artech House, 2002. ISBN 1580535089, 9781580535083

B.5.3 Statistical testing

NOTE This technique/measure is referenced in Tables B.3, B.5 and B.6 of IEC 61508-2.

Aim: To check the dynamic behaviour of the safety-related system and to assess its utility and robustness.

Description: This approach tests a system or program with input data selected according to the expected statistical distribution of the real operating inputs – the operational profile.

References:

A discussion of statistical testing on a safety-related application. S Kuball, J H R May, Proc. IMechE Vol. 221 Part O: J. Risk and Reliability, Institution of Mechanical Engineers, 2007

Practical Reliability Engineering. P. O'Connor, D. Newton, R. Bromley, John Wiley and Sons, 2002, ISBN 0470844639, 9780470844632

Dependability of Critical Computer Systems 3. P. G. Bishop et al., Elsevier Applied Science, 1990, ISBN 1-85166-544-7

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0

B.5.4 Field experience

NOTE 1 This technique/measure is referenced in Tables B.3, B.5 and B.6 of IEC 61508-2.

NOTE 2 See also C.2.10 for a similar measure and Annex D for a statistical approach, both in the context of software.

Aim: To use field experience from different applications as one of the measures to avoid faults either during E/E/PE system integration and/or during E/E/PE system safety validation.

Description: Use of components or subsystems, which have been shown by experience to have no, or only unimportant, faults when used, essentially unchanged, over a sufficient period of time in numerous different applications. Particularly for complex components with a multitude of possible functions (for example operating system, integrated circuits), the developer shall pay attention to which functions have actually been tested by the field experience. For example, consider self-test routines for fault detection: if no break-down of the hardware occurs within the operating period, the routines cannot be said to have been tested, since they have never performed their fault detection function.

For field experience to apply, the following requirements must have been fulfilled:

- unchanged specification;
- 10 systems in different applications;
- 10^5 operating hours and at least one year of service history.

NOTE 3 Sector standards may specify different numbers.

The field experience is demonstrated through documentation of the vendor and/or operating company. This documentation must contain at least

- the exact designation of the system and its component, including version control for hardware;
- the users and time of application;
- the operating hours;
- the procedures for the selection of the systems and applications procured to the proof;

– the procedures for fault detection and fault registration as well as fault removal.

References:

IEC 60300-3-2:2004, *Dependability management – Part 3-2: Application guide – Collection of dependability data from the field*

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

B.6 E/E/PE system safety validation

Global objective: To prove that the E/E/PE safety-related system conforms to the E/E/PE system safety requirements specification and E/E/PE system design requirements specification.

B.6.1 Functional testing under environmental conditions

NOTE This technique/measure is referenced in Table B.5 of IEC 61508-2.

Aim: To assess whether the safety-related system is protected against typical environmental influences.

Description: The system is put under various environmental conditions (for example according to the standards in the IEC 60068 series or the IEC 61000 series), during which the safety functions are assessed for their reliability (and compatibility with the standards mentioned).

References:

IEC 60068-1:1988, *Environmental testing – Part 1: General and guidance*
Amendment 1(1992)

IEC 61000-4-1:2006, *Electromagnetic compatibility (EMC) – Part 4-1: Testing and measurement techniques – Overview of IEC 61000-4 series*

Dependability of Critical Computer Systems 3. P. G. Bishop et al., Elsevier Applied Science, 1990, ISBN 1-85166-544-7

B.6.2 Interference surge immunity testing

NOTE This technique/measure is referenced in Tables B.5 and B.6 of IEC 61508-2.

Aim: To check the capacity of the safety-related system to handle peak surges.

Description: The system is loaded with a typical application program, and all the peripheral lines (all digital, analogue and serial interfaces as well as the bus connections and power supply, etc.) are subjected to standard noise signals. In order to obtain a quantitative statement, it is sensible to approach the surge limit carefully. The chosen class of noise is not complied with if the function fails.

References:

IEC 61000-4-5:2005, *Electromagnetic compatibility (EMC) – Part 4-5: Testing and measurement techniques – Surge immunity test*

C37.90.1-2002, *IEEE Standard for Surge Withstand Capability (SWC) Tests for Relays and Relay Systems Associated with Electric Power Apparatus*

B.6.3 (Not used)

B.6.4 Static analysis

NOTE This technique/measure is referenced in Tables B.5 and B.6 of IEC 61508-2 and in Tables A.9, B.8, C.9 and C.18 of IEC 61508-3.

Aim: To avoid systematic faults that can lead to breakdowns in the system under test, either early or after many years of operation.

Description: This systematic and possibly computer-aided approach inspects specific static characteristics of the prototype system to ensure completeness, consistency, lack of ambiguity regarding the requirement in question (for example construction guidelines, system specifications, and an appliance data sheet). A static analysis is reproducible. It is applied to a prototype which has reached a well-defined stage of completion. Some examples of static analysis, for hardware and software, are

- consistency analysis of the data flow (such as testing if a data object is interpreted everywhere as the same value);
- control flow analysis (such as path determination, determination of non-accessible code);
- interface analysis (such as investigation of variable transfer between various software modules);
- dataflow analysis to detect suspicious sequences of creating, referencing and deleting variables;
- testing adherence to specific guidelines (for example creepage distances and clearances, assembly distance, physical unit arrangement, mechanically sensitive physical units, exclusive use of the physical units which were introduced).

References:

Static Analysis and Software Assurance. D. Wagner, Lecture Notes in Computer Science, Volume 2126/2001, Springer, 2001, ISBN 978-3-540-42314-0

An Industrial Perspective on Static Analysis. B A Wichmann, A A Canning, D L Clutterbuck, L A Winsborrow, N J Ward and D W R Marsh. Software Engineering Journal., 69 – 75, March 1995

Dependability of Critical Computer Systems 3. P. G. Bishop et al,. Elsevier Applied Science, 1990, ISBN 1-85166-544-7

B.6.5 Dynamic analysis and testing

NOTE This technique/measure is referenced in Tables B.5 and B.6 of IEC 61508-2 and in Tables A.5, A.9, B.2, C.5, C.9 and C.12 of IEC 61508-3.

Aim: To detect specification failures by inspecting the dynamic behaviour of a prototype at an advanced state of completion.

Description: The dynamic analysis of a safety-related system is carried out by subjecting a near-operational prototype of the safety-related system to input data which is typical of the intended operating environment. The analysis is satisfactory if the observed behaviour of the safety-related system conforms to the required behaviour. Any failure of the safety-related system must be corrected and the new operational version must then be reanalysed.

References:

The Concept of Dynamic Analysis. T. Ball, ESEC/FSE ’99, Lecture Notes in Computer Science, Springer, 1999, ISBN 978-3-540-66538-0

Dependability of Critical Computer Systems 3. P. G. Bishop et al,. Elsevier Applied Science, 1990, ISBN 1-85166-544-7

B.6.6 Failure analysis

NOTE This technique/measure is referenced in Tables B.5 and B.6 of IEC 61508-2.

B.6.6.1 Failure modes and effects analysis (FMEA)

Aim: To analyse a system design, by examining systematically all possible sources of failure of a system's components and determining the effects of these failures on the behaviour and safety of the system.

Description: The analysis usually takes place through a meeting of engineers. Each component of a system is analysed in turn to give a set of failure modes for the component, their causes and effects (locally and at overall system level), detection procedures and recommendations. If the recommendations are acted upon, they are documented as remedial action taken.

References:

IEC 60812:2006, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*

Risk Assessment and Risk Management for the Chemical Process Industry. H.R. Greenberg, J.J. Cramer, John Wiley and Sons, 1991, ISBN 0471288829, 9780471288824

Reliability Technology. A. E. Green, A. J. Bourne, Wiley-Interscience, 1972, ISBN 0471324809

B.6.6.2 Cause consequence diagrams

NOTE This technique/measure is referenced in Tables B.3, B.4, C.13 and C.14 of IEC 61508-3.

Aim: To analyse and model, in a compact diagrammatic form, the sequence of events that can develop in a system as a consequence of combinations of basic events.

Description: The technique can be regarded as a combination of fault tree and event tree analysis. It starts from a critical (initiating) event and the consequence graph is traced forwards by using YES/NO gates describing success and failure of some operations. This allows building event sequences leading either to an accident or to a mastered situation. Then cause graphs (i.e. fault trees) are built for each failure. Then starting from an accidental situation and going in the backward direction gives a global fault tree with this accidental situation as top event. In the forward direction the possible consequences arising from an event are determined. The graph can contain vertex symbols which describe the conditions for propagation along different branches from the vertex. Time delays can also be included. These conditions can also be described with fault trees. The lines of propagation can be combined with logical symbols, to make the diagram more compact. A set of standard symbols is defined for use in cause consequence diagrams. The diagrams can be used for generating fault trees and to compute the probability of occurrence of certain critical consequences. It can also be used to produce event trees.

References:

IEC 62502, *Analysis techniques for dependability – Event tree analysis (ETA)*[1]

The Cause Consequence Diagram Method as a Basis for Quantitative Accident Analysis. B. S. Nielsen, Danish Atomic Energy Commission, Riso-M-1374, 1971

---

1 Under consideration.

B.6.6.3 Event tree analysis (ETA)

NOTE This technique/measure is referenced in Tables B.4 and C.14 of IEC 61508-3.

Aim: To model, in a diagrammatic form, the sequence of events that can develop in a system after an initiating event, and thereby indicate how serious consequences can occur. An event tree is difficult to build from scratch and using consequence diagram is helpful.

Description: On the top of the diagram is written the sequence conditions that are relevant in the progression of events that follow the initiating event. Starting under the initiating event, which is the target of the analysis, a line is drawn to the first condition in the sequence. There the diagram branches off into "yes" and "no" branches, describing how future events depend on the condition. For each of these branches, one continues to the next condition in a similar way. Not all conditions are, however, relevant for all branches. One continues to the end of the sequence, and each branch of the tree constructed in this way represents a possible consequence. Provided the conditions in the sequences are independent, the event tree can be used to compute the probability of the various consequences, based on the probability and number of conditions in the sequence. As conditions are rarely fully independent, such calculation shall be considered cautiously and performed by skilled analysts.

References:

IEC 62502, *Analysis techniques for dependability – Event tree analysis (ETA)*[2]

Risk Assessment and Risk Management for the Chemical Process Industry. H.R. Greenberg, J.J. Cramer, John Wiley and Sons, 1991, ISBN 0471288829, 9780471288824

B.6.6.4 Failure modes, effects and criticality analysis (FMECA)

NOTE Failure analysis is referenced in Tables A.10, B.4, C.10 and C.14 of IEC 61508-3.

Aim: To rank the criticality of components which could result in injury, damage or system degradation through single-point failures, in order to determine which components might need special attention and necessary control measures during design or operation.

Description: This method is comparable to FMEA, but there are one or more columns for indicating the criticality, which can be ranked in many ways. The most laborious method is described by the Society for Automotive Engineers (SAE) in ARP 926. In this procedure, the criticality number for any component is indicated by the number of failures of a specific type expected during each million operations occurring in a critical mode. The criticality number is a function of nine parameters, most of which have to be measured. A very simple method for criticality determination is to multiply the probability of component failure by the damage that could be generated; this method is similar to simple risk factor assessment.

References:

IEC 60812:2006, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*

Software criticality analysis of COTS/SOUP. P.Bishop, T.Clement, S.Guerra. In Reliability Engineering & System Safety, Volume 81, Issue 3, September 2003, Elsevier Ltd., 2003

Software FMEA techniques. P.L.Goddard. In Proc Annual 2000 Reliability and Maintainability Symposium, IEEE, 2000, ISBN: 0-7803-5848-1

---

2 Under consideration.

B.6.6.5 Fault tree analysis (FTA)

NOTE This technique/measure is referenced in Tables B.4 and C.14 of IEC 61508-3.

Aim: To aid in the analysis of events, or combinations of events, that will lead to a hazard or serious consequence and to perform the probability calculation of the top event.

Description: Starting at an event which would be the immediate cause of a hazard or serious consequence (the "top event"), analysis is carried out in order to identify the causes of this event. This is done in several steps through the use of logical operators (and, or, etc). Intermediate causes are analysed in the same way, and so on, back to basic events where analysis stops.

The method is graphical, and a set of standardised symbols are used to draw the fault tree. At the end of the analysis, the fault tree represents the logical function linking the basic events (generally components failures) to the top event (the overall system failure).The technique is mainly intended for the analysis of hardware systems, but there have also been attempts to apply this approach to software failure analysis. This technique can be used qualitatively for failure analysis (identification failure scenarios: minimal cut sets or prime implicants), semi-quantitatively (by ranking scenarios according to their probabilities) and quantitatively for probabilistic calculations of the top event (see C.6).

References:

IEC 61025:2006, *Fault tree analysis (FTA)*

From safety analysis to software requirements. K.M. Hansen, A.P. Ravn, A.P, V Stavridou. IEEE Trans Software Engineering, Volume 24, Issue 7, Jul 1998

B.6.6.6 Markov models

NOTE See B.1 of IEC 61508-6 for the use of this technique against reliability block diagrams, in the context of analysing hardware safety integrity.

Aim: To model the behaviour of the system by a state transition graph and to evaluate probabilistic system parameters (e.g, un-reliability, un-availability, *MTTF*, *MUT*, *MDT*, etc.) of a system.

Description: It is a finite state automaton (see B.2.3.2) represented by a directed graph. The nodes (circles) represent the states and the edges (arrows) between nodes represent the transitions (failure, repairs, etc.) occurring between the states. Edges are weighted with the corresponding failure rates or repair rates. The fundamental property of homogeneous Markov processes is that the future depends only of the present: a change of state, *N*, to a subsequent state, *N*+1, is independent of the previous state, *N*-1. This implies that all the probabilistic laws of the models are exponential.

The failure events, states and rates can be detailed in such a way that a precise description of the system is obtained, for example detected or undetected failures, manifestation of a larger failure, etc. Proof test intervals may also be modelled properly by using the so-called multi-phase Markov processes where the probabilities of the states at the end of one phase (e.g. just before a proof test) can be used to calculate the initial conditions for the next phase (e.g. the probabilities of the various states after a proof test has been performed).

The Markov technique is suitable for modelling multiple systems in which the level of redundancy varies with time due to component failure and repair. Other classical methods, for example, FMEA and FTA, cannot readily be adapted to modelling the effects of failures throughout the lifecycle of the system since no simple combinatorial formulae exist for calculating the corresponding probabilities.

In the simplest cases, the formulae which describe the probabilities of the system are readily available in the literature or can be calculated manually and some methods of simplification (i.e. reducing the number of states) also exist to handle more complex cases.

Nevertheless, mathematically speaking, a homogeneous Markov graph is only a simple and common set of linear differential equations with constant coefficients. This has been analysed for a long time and powerful algorithms have been developed and are available to handle them. Therefore when the size of the model increases it is very efficient to use the above algorithms which are implemented in various computer software packages.

It has to be noted that the size of the graph increases exponentially with the number of components: this is the so-called combinatorial explosion. Therefore this technique is usable without approximations only for small systems.

When non-exponential laws have to be handled -semi-Markov models- then Monte Carlo simulation (see B.6.6.8) should be used.

References:

IEC 61165:2006, *Application of Markov techniques*

The Theory of Stochastic Processes. R. E. Cox and H. D. Miller, Methuen and Co. Ltd., London, UK, 1963

Finite MARKOV Chains. J. G. Kemeny and J. L. Snell. D. Van Nostrand Company Inc, Princeton, 1959

The Theory and Practice of Reliable System Design. D. P. Siewiorek and R. S. Swarz, Digital Press, 1982

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès – Lavoisier, 2009, ISBN: 978-2-7462-1991-5

B.6.6.7 Reliability block diagrams (RBD)

NOTE 1 This technique/measure is used in Annex B of IEC 61508-6.

NOTE 2 See also C.6.4 “Reliability block diagrams”.

Aim: To model, in a diagrammatic form, the set of events that must take place and conditions which must be fulfilled for a successful operation of a system or a task. It is more a method of representation than a method of analysis.

Description: The target of the analysis is represented as a success path consisting of blocks, lines and logical junctions. A success path starts from one side of the diagram and continues via the blocks and junctions to the other side of the diagram. A block represents a condition or an event, and the path can pass it if the condition is true or the event has taken place. If the path comes to a junction, it continues if the logic of the junction is fulfilled. If it reaches a vertex, it may continue along all outgoing lines. If there exists at least one success path through the diagram, the target of the analysis is operating correctly.

A RBD is a structural representation of the modelled system. It is a kind of electrical circuit: when the current find a path from the input to the output, that means that the modelled system is working properly, when the circuit is cut that means that the modelled system is failed. This lead to the concept of minimal cut sets which represent the combinations of failures (i.e. places where the RBD is "cut") leading to the failure of the modelled system.

Mathematically a RBD is similar to a fault tree. It represents the logical function linking the states of the individual components (failed or working) to the state of the whole system (failed or working). Therefore the calculations are similar as those described for fault trees.

References:

IEC 61078:2006, *Analysis techniques for dependability – Reliability block diagram and boolean methods*

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès – Lavoisier, 2009, ISBN: 978-2-7462-1991-5

B.6.6.8 Monte-Carlo simulation

NOTE This technique/measure is referenced in Table B.4 of IEC 61508-3 and used in IEC 61508-6 Annex B.

Aim: To simulate real world phenomena by generating random numbers when analytical methods are not practicable.

Description: Monte-Carlo simulations are used to solve two classes of problems:

– probabilistic, where random numbers are used to generate stochastic phenomena; and
– deterministic, which are mathematically translated into an equivalent probabilistic problem (e.g. integral calculations).

The principle of Monte-Carlo simulation is to use random numbers to animate a behavioural functional and dysfunctional model of the system under study. Such behavioural models are provided by state transition models (Markov graph, Petri nets, formal languages, etc.). The Monte-Carlo simulation is run to produce a large statistical sample from which statistical results are obtained.

When using Monte-Carlo simulations care must be taken to ensure that the biases, tolerances or noise have reasonable values. This shall be managed through the confidence interval which is easily obtained from the simulations. Contrary to analytical methods, Monte Carlo simulation is self approximating. Negligible events just not appear without need to identify them to simplify the model.

A general principle of Monte-Carlo simulations is to restate and reformulate the problem so that the results obtained are as accurate as possible rather than tackling the problem as initially stated.

In the context of this standard, Monte Carlo simulation may be used for SIL calculations and to take into consideration the reliability data uncertainties. With present time computers, SIL4 calculations are easily achieved.

References:

Monte Carlo Methods. J. M. Hammersley, D. C. Handscomb, Chapman & Hall, 1979

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès – Lavoisier, 2009, ISBN: 978-2-7462-1991-5

B.6.6.9 Fault tree models

NOTE 1 See IEC 61508-6 for the use of this technique in the context of analysing hardware safety integrity.

NOTE 2 Fault trees have already been described above as safety validation means (see B.6.6.5). They are also widely used for failure analysis and probabilistic calculations.

Aim: To build by a systematic topdown graphical (effect-cause) approach, the logical function linking the basic events (failure modes) to the top event (unwanted event).

Description: This is both a method of analysis helping the analyst to develop the model step by step and a mathematical model for probabilistic calculations. It allows performing:

– qualitative analysis by identifying and sorting failure scenarios (minimal cut sets or prime implicants),

– semi quantitative analysis by ranking scenarios according to their probabilities of occurrence,

– quantitative analysis by calculating the probability of the top event.

Like RBD (Reliability Block Diagrams), a fault tree represents the logical (boolean) function linking the states of the individual components (failed or working) to the state of the whole system (failed or working). Therefore when the components are independent, probabilistic calculations may be performed just by applying the basic properties of probabilities applied on logical function. This is not so easy because this is a static model which basically works only with genuine (i.e. constant) probabilities). Time dependent probabilities shall be handled cautiously. For example the PFD_{avg} of safety systems comprising periodically proof tested components cannot be calculated straightforwardly and this is even more difficult for *PFH* of safety systems working in continuous mode. Therefore only reliability engineers with a sound understanding of the underlying mathematics should perform un-availability/*PFD* and un-reliability/*PFH* calculations with this method.

Calculations may be done by hand for very simple fault trees but a lot of algorithms have been developed and implemented to handle complex logical equations over the last 50 past years. The state of the art at the present time is using BDD (Binary Decision Diagrams) which is a technique of compact encoding of the logical equation into a computer memory. It is, at the present time, the only method able to perform the probabilistic calculations without approximations on industrial size systems. It is also sufficiently fast to allow handling uncertainties by Monte Carlo simulation.

Reference:

IEC 61025:2006, *Fault tree analysis (FTA)*

B.6.6.10 Generalised Stochastic Petri net models (GSPN)

NOTE 1 See IEC 61508-6 for the use of this technique in the context of analysing hardware safety integrity.

NOTE 2 Petri nets have already been mentioned as semi-formal method (see B.2.3.3). They can also efficiently used in the context of hardware safety integrity.

Aim: To graphically build a functionnal and dysfunctional model behaving as close as possible as the actual modelled system in order to provide an efficient support for Monte carlo simulation.

Description: This is an asynchronous finite state automata as described in B.2.3.3, except that the good property tracked when performing semi-formal validation are no longer existing when modelling the dysfunctionnal behaviour of a safety system. The so-called places (pictured by circles) represent the potental states and the so-called transitions (pictured by rectangles) represents the events likely to occur. In addition of the marking of the places (see B.2.3.3) messages or predicates may be used to validate (enable) the transitions and the delay elapsing from the validation of a transition to its firing may be deterministic or stochastic. This is why those Petri Nets are called "generalised stochastic" Petri nets.

Petri nets constitute flexible behavioural models which prove to be very efficient as Monte Carlo simulation support (see B.6.6.8). Except the accuracy of the Monte Carlo simulation itself which, anyway, is always known, all the limitations of other methods (dependancies,

combinatory explosion, non-exponentail laws, etc.) are overcome. With present time computer this is no longer a problem even for SIL4 evaluations.

References:

IEC 62551, *Analysis techniques for dependability – Petri net modelling (CD1)*[3]

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès – Lavoisier, 2009, ISBN: 978-2-7462-1991-5

B.6.7 Worst-case analysis

NOTE This technique/measure is referenced in Tables B.5 and B.6 of IEC 61508-2.

Aim: To avoid systematic failures which arise from unfavourable combinations of the environmental conditions and the component tolerances.

Description: The operational capacity of the system and the component dimensioning is examined or calculated on a theoretical basis. The environmental conditions are changed to their highest permissible marginal values. The most essential responses of the system are inspected and compared with the specification.

B.6.8 Expanded functional testing

NOTE This technique/measure is referenced in Tables B.5 and B.6 of IEC 61508-2.

Aim: To reveal failures during the specification and design and development phases. To check the behaviour of the safety-related system in the event of rare or unspecified inputs.

Description: Expanded functional testing reviews the functional behaviour of the safety-related system in response to input conditions which are expected to occur only rarely (for example major failure), or which are outside the specification of the safety-related system (for example incorrect operation). For rare conditions, the observed behaviour of the safety-related system is compared with the specification. Where the response of the safety-related system is not specified, one should check that the plant safety is preserved by the observed response.

References:

Software Testing and Quality Assurance. K. Naik, P. Tripathy, Wiley Interscience, 2008, Print ISBN: 9780471789116 Online ISBN: 9780470382844

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122, 9780471469124

Dependability of Critical Computer Systems 3. P. G. Bishop et al,. Elsevier Applied Science, 1990, ISBN 1-85166-544-7

B.6.9 Worst-case testing

NOTE This technique/measure is referenced in Tables B.5 and B.6 of IEC 61508-2.

Aim: To test the cases specified during worst-case analysis.

Description: The operational capacity of the system and the component dimensioning is tested under worst-case conditions. The environmental conditions are changed to their highest permissible marginal values. The most essential responses of the system are inspected and compared with the specification.

---

[3] Under consideration.

B.6.10 Fault insertion testing

NOTE This technique/measure is referenced in Tables B.5 and B.6 of IEC 61508-2.

Aim: To introduce or simulate faults in the system hardware and document the response.

Description: This is a qualitative method of assessing dependability. Preferably, detailed functional block, circuit and wiring diagrams are used in order to describe the location and type of fault and how it is introduced; for example: power can be cut from various modules; power, bus or address lines can be open/short-circuited; components or their ports can be opened or shorted; relays can fail to close or open, or do it at the wrong time, etc. Resulting system failures are classified, as in Tables 1 and 2 of IEC 60812, for example. In principle, single steady-state faults are introduced. However, in case a fault is not revealed by the built-in diagnostic tests or otherwise does not become evident, it can be left in the system and the effect of a second fault considered. The number of faults can easily increase to hundreds.

The work is done by a multidisciplinary team and the vendor of the system should be present and consulted. The mean operating time between failure for faults that have grave consequences should be calculated or estimated. If the calculated time is low, modifications should be made.

References:

IEC 60812:2006, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*

IEC 61069-5:1994, *Industrial-process measurement and control – Evaluation of system properties for the purpose of system assessment – Part 5: Assessment of system dependability*

Annex C
(informative)

Overview of techniques and measures for achieving software safety integrity (see IEC 61508-3)

C.1 General

The overview of techniques contained in this annex should not be regarded as either complete or exhaustive.

C.2 Requirements and detailed design

NOTE Relevant techniques and measures are found in B.2.

C.2.1 Structured diagrammatic methods

NOTE This technique/measure is referenced in Tables A.2 and A.4 of IEC 61508-3.

C.2.1.1 General

Aim: The main aim of structured methods is to promote the quality of software development by focusing attention on the early parts of the lifecycle. The methods aim to achieve this through both precise and intuitive procedures and notations (assisted by computers), to determine and document requirements and implementation features in a logical order and a structured manner.

Description: A range of structured methods exists. Some are designed for traditional data-processing and transaction processing functions, while others are more oriented to process control and real-time applications (which tend to be more safety critical). UML (see C.3.12) contains many examples of structured notations.

Structured methods are essentially "thought tools" for systematically perceiving and partitioning a problem or system. Their main features are the following:

– a logical order of thought, breaking a large problem into manageable stages;

– analysis and documentation of the total system, including the environment as well as the required system;

– decomposition of data and function in the required system;

– checklists, i.e. lists of the sort of things that need analysis;

– low intellectual overhead – simple, intuitive, pragmatic;

– often with a strong emphasis on developing a diagrammatic model of the intended system, and CASE tool support for the overall method.

The supporting notations for analysing and documenting problems and system entities (for example processes and data flows) tend to be precise, but notations for expressing the processing functions performed by these entities tend to be more informal. However, some methods do make partial use of (mathematically) formal notations (for example, regular expressions or finite state machines). Increased precision not only reduces the scope for misunderstanding, it provides scope for automatic processing.

Another benefit of structured notations is their visibility, enabling a specification or design to be checked intuitively by a user, against his powerful but unstated knowledge.

This overview describes several structured methods in more detail.

Reference:

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

Software Design. D. Budgen, Pearson Education, 2003, ISBN 0201722194, 9780201722192

C.2.1.2 Controlled Requirements Expression (CORE)

Aim: To ensure that all the requirements are determined and expressed.

Description: This approach is intended to bridge the gap between the customer/end user and the analyst. It is not mathematically rigorous but aids communication – CORE is designed for requirements expression rather than specification. The approach is structured, and the expression goes through various levels of refinement. The CORE method encourages a wider view of the problem, bringing in a knowledge of the environment in which the system will be used and the differing viewpoints of the various types of user. CORE includes guidelines and tactics for recognising departures from the "grand design". Departures can be corrected or explicitly identified and documented. Thus specifications may not be complete, but unresolved problems and high-risk areas are detected and have to be considered in the subsequent design.

Reference:

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

Requirements Engineering. E. Hull, K. Jackson, J. Dick. Springer, 2005, ISBN 1852338792, 9781852338794

C.2.1.3 Jackson System Development (JSD)

Aim: A development method covering the development of software systems from requirements through to code, with special emphasis on real-time systems.

Description: JSD is a staged development procedure in which the developer models the real world behaviour upon which the system functions are to be based, determines the required functions and inserts them into the model, and transforms the resulting specification into one that is realisable in the target environment. It therefore covers the traditional phases of specification and design and development but takes a somewhat different view from the traditional methods in not being top-down.

Moreover, it places great emphasis on the early stage of discovering the entities in the real world that are the concern of the system being built and on modelling them and what can happen to them. Once this analysis of the "real world" has been done and a model created, the system’s required functions are analysed to determine how they can fit into this real-world model. The resulting system model is augmented with structured descriptions of all the processes in the model and the whole is then transformed into programs that will operate in the target software and hardware environment.

References:

Systems Analysis and Design. D. Yeates, A. Wakefield. Pearson Education, 2003, ISBN 0273655361, 9780273655367

An Overview of JSD. J. R. Cameron. IEEE Transactions on Software Engineering, SE-12, No. 2, February 1986

C.2.1.4 Real-time Yourdon

Aim: The specification and design of real-time systems.

Description: The development scheme underlying this technique assumes a three-stage evolution of a system being developed. The first stage involves the building of an "essential model", one that describes the behaviour required by the system. The second involves the building of an implementation model which describes the structures and mechanisms that, when implemented, embody the required behaviour. The third stage involves the actual building of the system in hardware and software. The three stages correspond roughly to the traditional specification and design and development phases but lay greater emphasis on the fact that at each stage the developer is engaged in a modelling activity.

The essential model is in two parts:

– the environmental model, containing a description of the boundary between the system and its environment and a description of the external events to which the system must respond; and

– the behavioural model, which contains schemes describing the transformation carried out by the system in response to events and a description of the data the system must hold in order to respond.

The implementation model also divides into submodels, covering the allocation of individual processes to processors and the decomposition of the processes into software modules.

To capture these models, the technique combines a number of other well-known techniques: data-flow diagrams, transformation graphs, structured English, state transition diagrams and Petri nets. Additionally, the method contains techniques for simulating a proposed system design, either on paper or mechanically from the models that are drawn up.

References:

Real-time Systems Development. R. Williams. Butterworth-Heinemann, 2006, ISBN 0750664711, 9780750664714

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward and S. J. Mellor. Yourdon Press, 1985

C.2.2 Data flow diagrams

NOTE This technique/measure is referenced in Tables B.5 and B.7 of IEC 61508-3.

Aim: To describe the data flow through a program in a diagrammatic form.

Description: Data flow diagrams document how data input is transformed to output, with each stage in the diagram representing a distinct transformation.

Data flow diagrams have three aspects:

– annotated arrows – represent data flow in and out of the transformation centres, with the annotations documenting what the data is;

– annotated bubbles – represent transformation centres, with the annotation documenting the transformation;

– operators (and, xor) – these operators are used to link the annotated arrows.

Each bubble in a data flow diagram can be considered as a stand-alone black box which, as soon as its inputs are available, transforms them to its outputs. One of the principal advantages is that they show transformations without making any assumptions about how these transformations are implemented. A pure data flow diagram does not include control

information or sequencing information, but this is catered for by real-time extensions to the notation, as in real-time Yourdon (see C.2.1.4).

The preparation of data flow diagrams is best approached by considering system inputs and working towards system outputs. Each bubble must represent a distinct transformation – its output should, in some way, be different from its input. There are no rules for determining the overall structure of the diagram and constructing a data flow diagram is one of the creative aspects of system design. Like all design, it is an iterative procedure with early attempts refined in stages to produce the final diagram.

References:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

ISO 5807:1985, *Information processing – Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts*

ISO/IEC 8631:1989, *Information technology – Program constructs and conventions for their representation*

C.2.3 Structure diagrams

NOTE This technique/measure is referenced in Table B.5 of IEC 61508-3.

Aim: To show the structure of a program diagrammatically.

Description: Structure diagrams are a notation which complements data flow diagrams. They describe the programming system and a hierarchy of parts and display this graphically, as a tree. They document how elements of a data flow diagram can be implemented as a hierarchy of program units.

A structure chart shows relationships between program modules without including any information about the order of activation of these units. They are drawn using the following four symbols:

- a rectangle annotated with the name of the module;
- a line connecting these rectangles creating structure;
- a circled arrow (circle empty), annotated with the name of data passed to and from elements in the structure chart (normally, the circled arrow is drawn parallel to the line connecting the rectangles in the chart);
- a circled arrow (circle filled), annotated with the name of the control signal passing from one module to another in the structure chart, again the arrow is drawn parallel to the line connecting the two modules.

From any non-trivial data flow diagram, it is possible to derive a number of different structure charts.

Data flow diagrams depict the relationship between information and functions in the system. Structure charts depict the way elements of the system are implemented. Both techniques present valid, though different, views of the system.

References:

Software Design & Development. G. Lancaster. Pascal Press, 2001, ISBN 1741251753, 9781741251753

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

C.2.4 Formal methods

NOTE This technique/measure is referenced in Tables A.1, A.2, A.4 and B.5 of IEC 61508-3.

C.2.4.1 General

Aim: The development of software in a way that is based on mathematics. This includes formal design and formal coding techniques.

Description: Formal methods provide a means of developing a description of a system at some stage in its specification, design or implementation. The resulting description is in a strict notation that can be subjected to mathematical analysis to detect various classes of inconsistency or incorrectness. Moreover, the description can in some cases be analysed by machine with a rigour similar to the syntax checking of a source program by a compiler, or animated to display various aspects of the behaviour of the system described. Animation can give extra confidence that the system meets the real requirement as well as the formally specified requirement, because it improves human recognition of the specified behaviour.

A formal method will generally offer a notation (generally some form of discrete mathematics being used), a technique for deriving a description in that notation, and various forms of analysis for checking a description for different correctness properties.

Several formal methods are described in the following subclauses of this overview : CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z. Note that other techniques, such as finite state machines and Petri nets (see Annex B), may be considered as formal methods, depending on how strictly the techniques, as used, conform to a rigorous mathematical basis.

References:

Formal Specification: Techniques and Applications. N.Nissanke, Springer-Verlag Telos, 1999, ISBN 1852330023

The Practice of Formal Methods in Safety-Critical Systems. S. Liu, V. Stavridou, B. Dutertre, J. Systems Software 28, 77-87, Elsevier, 1995

Formal Methods: Use and Relevance for the Development of Safety-Critical Systems. L. M. Barroca, J. A. McDermid, The Computer Journal 35 (6), 579-599, 1992

How to Produce Correct Software – An Introduction to Formal Specification and Program Development by Transformations. E. A. Boiten et al, The Computer Journal 35 (6), 547-554, 1992

C.2.4.2 Calculus of Communicating Systems (CCS)

Aim: CCS is a means of describing and reasoning about the behaviour of systems of concurrent, communicating processes.

Description: CCS is a mathematical calculus concerned with the behaviour of systems. The system design is modelled as a network of independent processes operating sequentially or in parallel. Processes can communicate via ports (similar to CSP's channels), the communication only taking place when both processes are ready. Non-determinism can be modelled. Starting from a high-level abstract description of the entire system (known as a trace), it is possible to carry out a step-wise refinement of the system into a composition of communicating processes whose total behaviour is that required of the whole system. Equally, it is possible to work in a bottom-up fashion, combining processes and deducing the properties of the resulting system using inference rules related to the composition rules.

Reference:

Communication and Concurrency. R. Milner. Pearson Education, 1989, ISBN 9780131150072

C.2.4.3 Communicating Sequential Processes (CSP)

Aim: CSP is a technique for the specification of concurrent software systems, i.e. systems of communicating processes operating concurrently.

Description: CSP provides a language for the specification of systems of processes and proof for verifying that the implementation of processes satisfies their specifications (described as a trace, i.e. a permissible sequence of events).

A system is modelled as a network of independent processes, composed sequentially or in parallel. Each process is described in terms of all of its possible behaviours. Processes can communicate (synchronise or exchange data) via channels, the communication only taking place when both processes are ready. The relative timing of events can be modelled.

The theory behind CSP was directly incorporated into the architecture of the INMOS transputer, and the OCCAM language allows a CSP-specified system to be directly implemented on a network of transputers.

Reference:

Communicating Sequential Processes: The First 25 Years. A. Abdallah, C. Jones, J. Sanders (Eds.). Springer, 2004, ISBN 3540258132, 9783540258131

C.2.4.4 Higher Order Logic (HOL)

Aim: This is a formal language intended as a basis for hardware specification and verification.

Description: HOL refers to a particular logic notation and its machine support system, both of which were developed at the University of Cambridge computer laboratory. The logic notation is mostly taken from Church's simple theory of types and the machine support system is based upon the logic of computable functions (LCF) system.

Reference:

Higher-Order Computational Logic. J. Lloyd. In Computational Logic: Logic Programming and Beyond, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2002, ISBN 978-3-540-43959-2

C.2.4.5 LOTOS

Aim: LOTOS is a means for describing and reasoning about the behaviour of systems of concurrent, communicating processes.

Description: LOTOS (language for temporal ordering specification) is based on CCS with additional features from the related algebras CSP and CIRCAL (circuit calculus). It overcomes the weakness of CCS in the handling of data structures and value expressions by combining it with aspects of the abstract data type language ACT ONE. The process description aspect of LOTOS could, however, be used with other formalisms for the description of abstract data types.

References:

Model Checking for Software Architectures. R. Mateescu. In Software Architecture, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2004, ISBN 978-3-540-22000-8

ISO 8807:1989, *Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*

C.2.4.6 OBJ

Aim: To provide a precise system specification with user feed-back and system validation prior to implementation.

Description: OBJ is an algebraic specification language. Users specify requirements in terms of algebraic equations. The behavioural, or constructive, aspects of the system are specified in terms of operations acting on abstract data types (ADT). An ADT is like an ADA package where the operator behaviour is visible whilst the implementation details are "hidden".

An OBJ specification, and subsequent step-wise implementation, is amenable to the same formal proof techniques as other formal approaches. Moreover, since the constructive aspects of the OBJ specification are machine-executable, it is straightforward to achieve system validation from the specification itself. Execution is essentially the evaluation of a function by equation substitution (rewriting) which continues until specific output value is obtained. This executability allows end-users of the envisaged system to gain a "view" of the eventual system at the system specification stage without the need to be familiar with the underlying formal specification techniques.

As with all other ADT techniques, OBJ is only applicable to sequential systems, or to sequential aspects of concurrent systems. OBJ has been used for the specification of both small- and large-scale industrial applications.

References:

Software Engineering with OBJ: Algebraic Specification in Action. J. Goguen, G. Malcolm. Springer, 2000, ISBN 0792377575, 9780792377573

C.2.4.7 Temporal logic

Aim: Direct expression of safety and operational requirements and formal demonstration that these properties are preserved in the subsequent development steps.

Description: Standard first-order predicate logic contains no concept of time. Temporal logic extends first-order logic by adding modal operators (for example "henceforth" and "eventually"). These operators can be used to qualify assertions about the system. For example, safety properties might be required to hold "henceforth", whilst other desired system states might be required to be attained "eventually" from some other initiating state. Temporal formulas are interpreted on sequences of states (behaviours). What constitutes a "state" depends on the chosen level of description. It can refer to the whole system, a system element or the computer program.

Quantified time intervals and constraints are not handled explicitly in temporal logic. Absolute timing has to be handled by creating additional time states as part of the state description.

Reference:

Mathematical Logic for Computer Science. M. Ben-Ari. Springer, 2001, ISBN 1852333197, 9781852333195

C.2.4.8 VDM, VDM++ – Vienna Development Method

Aim: The systematic specification and implementation of sequential (VDM) and concurrent real-time (VDM++) programs.

Description: VDM is a mathematically based specification technique and a technique for refining implementations in a way that allows proof of their correctness with respect to the specification.

The specification technique is model-based in that the system state is modelled in terms of set-theoretic structures on which are described invariants (predicates), and operations on that state are modelled by specifying their pre- and post-conditions in terms of the system state. Operations can be proved to preserve the system invariants.

The implementation of the specification is done by the reification of the system state in terms of data structures in the target language and by refinement of the operations in terms of the program in the target language. Reification and refinement steps give rise to proof obligations that establish their correctness. Whether or not these obligations are carried out is determined by the designer.

VDM is principally used in the specification stage but can be used in the design and implementation stages leading to source code. It can only be applied to sequentially structured programs or the sequential processes in concurrent systems.

The object-oriented and concurrent real-time extension of VDM, VDM++, is a formal specification language based on the ISO language VDM-SL and on the object-oriented language Smalltalk.

VDM++ provides a wide range of constructs such that a user can formally specify concurrent real-time systems in an object-oriented fashion. In VDM++ a complete formal specification consists of a collection of class specifications and optionally a workspace.

Real-time provisions of VDM++ are:

- temporal expressions are provided to denote both the current moment and the method invocation moment within a method body;
- a timed post expression can be added to a method to specify the upper (or lower) bounds of the execution time for correct implementations;
- time continuous variables have been introduced. With assumption and effect clauses one can specify relations (for example differential equations) between these functions of time. This feature has proven to be very useful in the specification of requirements of systems which operate in a time continuous environment. Refinement steps lead to discrete software solutions for these kinds of systems.

References:

ISO/IEC 13817-1:1996, *Information technology – Programming languages, their environments and system software interfaces – Vienna Development Method – Specification Language – Part 1: Base language*

Systematic Software Development using VDM. C. B. Jones. Prentice-Hall. 2nd Edition, 1990

Conformity Clause for VDM-SL, G. I. Parkin and B. A. Wichmann, Lecture Notes in Computer Science 670, FME'93 Industrial-Strength Formal Methods, First International Symposium of Formal Methods in Europe. Editors: J. C. P. Woodcock and P. G. Larsen. Springer Verlag, 501-520

C.2.4.9 Z

Aim: Z is a specification language notation for sequential systems and a design technique that allows the developer to proceed from a Z specification to executable algorithms in a way that allows proof of their correctness with respect to the specification.

Z is principally used in the specification stage but a method has been devised to go from specification into a design and an implementation. It is best suited to the development of data-oriented, sequential systems.

Description: Like VDM, the specification technique is model-based in that the system state is modelled in terms of set-theoretic structures on which are described invariants (predicates), and operations on that state are modelled by specifying their pre- and post-conditions in terms of the system state. Operations can be proved to preserve the system invariants thereby demonstrating their consistency. The formal part of a specification is divided into schemas which allow the structuring of specifications through refinement.

Typically, a Z specification is a mixture of formal Z and informal explanatory text in natural language. (Formal text on its own can be too terse for easy reading and often its purpose needs to be explained, while the informal natural language can easily become vague and imprecise.)

Unlike VDM, Z is a notation rather than a complete method. However, an associated method (called B) has been developed which can be used in conjunction with Z. The B method is based on the principle of step-wise refinement.

References:

Formal Specification using Z, 2nd Edition. D. Lightfoot. Palgrave Macmillan, 2000, ISBN 9780333763278

The B-Method. S. Schneider. Palgrave Macmillan, 2001, ISBN 9780333792841

C.2.5 Defensive programming

NOTE This technique/measure is referenced in Table A.4 of IEC 61508-3.

Aim: To produce programs which detect anomalous control flow, data flow or data values during their execution and react to these in a predetermined and acceptable manner.

Description: Many techniques can be used during programming to check for control or data anomalies. These can be applied systematically throughout the programming of a system to decrease the likelihood of erroneous data processing.

There are two overlapping areas of defensive techniques. Intrinsic error-safe software is designed to accommodate its own design shortcomings. These shortcomings may be due to mistakes in design or coding, or to erroneous requirements. The following lists some of the defensive techniques:

– variables should be range checked;

– where possible, values should be checked for plausibility;

– parameters to procedures should be type, dimension and range checked at procedure entry.

These first three recommendations help to ensure that the numbers manipulated by the program are reasonable, both in terms of the program function and physical significance of the variables.

Read-only and read-write parameters should be separated and their access checked. Functions should treat all parameters as read-only. Literal constants should not be write-accessible. This helps detect accidental overwriting or mistaken use of variables.

Fault tolerant software is designed to "expect" failures in its own environment or use outside nominal or expected conditions, and behave in a predefined manner. Techniques include the following.

– Input variables and intermediate variables with physical significance should be checked for plausibility.
– The effect of output variables should be checked, preferably by direct observation of associated system state changes.
– The software should check its configuration, including both the existence and accessibility of expected hardware and also that the software itself is complete – this is particularly important for maintaining integrity after maintenance procedures.

Some of the defensive programming techniques, such as control flow sequence checking, also cope with external failures.

References:

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

Dependability of Critical Computer Systems: Guidelines Produced by the European Workshop on Industrial Computer Systems, Technical Committee 7 (EWICS TC7, Systems Reliability, Safety, and Security). Elsevier Applied Science, 1989, ISBN 1851663819, 9781851663811

C.2.6 Design and coding standards

NOTE This technique/measure is referenced in Table A.4 of IEC 61508-3.

C.2.6.1 General

Aim: To facilitate verifiability, to encourage a team-centred, objective approach and to enforce a standard design method.

Description: The rules to be adhered to are agreed at the outset of the project between the participants. These rules comprise the design and development methods to be followed (for example JSP, Petri nets, etc.) and the related coding standards (see C.2.6.2).

These rules are made to allow for ease of development, verification, assessment and maintenance. Therefore they should take into account available tools, in particular analysers and reverse engineering tools.

References:

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

Verein Deutscher Ingenieure. Software-Zuverlassigkeit – Grundlagen, Konstruktive Mass-nahmen, Nachweisverfahren. VDI-Verlag, 1993, ISBN 3-18-401185-2

C.2.6.2 Coding standards

NOTE This technique/measure is referenced in Table B.1 of IEC 61508-3.

Aim: To reduce the likelihood of errors in the safety-related code and to facilitate its verification.

Description: The following principles indicate how safety-related coding rules (for any programming language) can assist in complying with the IEC 61508-3 normative requirements and in achieving the informative "desirable properties" (see Annex F). Account should be taken of available support tools.

IEC 61508-3 Requirements & Recommendations	*Coding Standards Suggestions*
Modular approach (Table A.2-7, Table A.4-4)	Software module size limit (Table B.9–1) and software complexity control (Table B.9–2). Examples: • Specification of "local" size and complexity metrics and limits (for modules) • Specification of "global" complexity metrics and limits (for overall modules organisation) • Parameter number limit / fixed number of subprogram parameters (Table B.9–4) Information hiding/encapsulation (Table B.9–3): e.g., incentives for using particular language features. Fully defined interface (Table B.9–6). Examples: • Explicit specification of function signatures • Failure assertion programming (Table A.2-3a) and data verification (7.9.2.7), with explicit specification of pre-conditions and post-conditions for functions, of assertions, of data types invariants
Code understandability • Promote code understandability (7.4.4.13) • Readable, understandable and testable (7.4.6)	Naming conventions promoting meaningful, unambiguous names. Example: avoidance of names that could be confounded (e.g., IO and I0). Symbolic names for numeric values. Procedures and guidelines for source code documentation (7.4.4.13). For example: • Explain why's and meanings (and not only what), • Caveats • Side effects Where practicable, the following information shall be contained in the source code (7.4.4.13): • Legal entity (for example: company, author(s), etc.) • Description • Inputs and outputs • Configuration management history (See also Modular Approach)

IEC 61508-3 Requirements & Recommendations	*Coding Standards Suggestions*
Verifiability and testability • Facilitate verification and testing (7.4.4.13) • Facilitate the detection of design or programming mistakes (7.4.4.10) • Formal verification (Table A.5 - 9) • Formal proof (Table A.9 - 1)	• Wrappers for “critical” library functions, to check pre- and post-conditions • Incentives for using language features that can express restrictions on the use of particular data elements or functions (e.g., const) • For tool supported verification: rules for complying with the limitations of the selected tools (provided this does not impair more essential goals) • Limited use of recursion (Table B.1 – 6) and other forms of circular dependencies (See also Modular Approach)
Static verification of conformance to the specified design (7.9.2.12)	Coding guidelines for the implementation of specific design concepts or constraints. For example: • Coding guidelines for cyclic behaviour, with guaranteed maximum cycle time (Table A.2-13a) • Coding guidelines for time-triggered architecture (Table A.2-13b) • Coding guidelines for event-driven architecture, with guaranteed maximum response time (Table A.2-13c) • Loops with a statically determined maximum number of iterations (except for the infinite loop of the cyclic design) • Coding guidelines for static resource allocation (Table A.2-14) and avoidance of dynamic objects (Table B.1–2) • Coding guidelines for static synchronisation of access to shared resources (Table A.2-15) • Coding guidelines to comply with limited use of interrupts (Table B.1–4) • Coding guidelines to avoid dynamic variables (Table B.1–3a) • Online checking of the installation of dynamic variables (Table B.1–3b) • Coding guidelines to ensure compatibility with other programming languages used (7.4.4.10) Guidelines to facilitate traceability with design

IEC 61508-3 Requirements & Recommendations	*Coding Standards Suggestions*
Language subset (Table A.3 - 3) • Proscribe unsafe language features (7.4.4.13) • Use only defined language features (7.4.4.10) • Structured programming (Table A.4 - 6) • Strongly typed programming language (Table A.3 - 2) • No automatic type conversion (Table B.1 - 8)	Exclusion of language features leading to unstructured designs. E.g., • Limited use of pointers (Table B.1–5) • Limited use of recursion (Table B.1–6) • Limited use of C-like unions • Limited use of Ada or C++-like exceptions • No unstructured control flow in programs in higher level languages (Table B.1–7) • One entry/one exit point in subroutines and functions (Table B.9-5) • No automatic type conversion • Limited use of side effects not apparent from functions signatures (e.g., of static variables). No side effects in evaluation of conditions and all forms of assertions. Limited or documented-only use of compiler-specific features. Limited use of potentially misleading language constructs. Rules to be applied when these language features are used nonetheless.
Good programming practice (7.4.4.13)	When applicable: • Coding guidelines to ensure that, when necessary, floating point expressions are evaluated in the right order (e.g., “a-b+c” is not always equal to “a+c-b”) • In floating point comparisons: use only inequalities (less than, less or equal to, greater than, greater or equal to) instead of strict equality • Guidelines regarding conditional compilation and “pre-processing” • Systematic checking of return conditions (success / failure) Documentation, and, when possible, automation of the production of executable code (makefiles). Avoidance of side effects not apparent from functions signatures. When such side effects exist, guidelines to document them. Bracketing when operators precedence is not absolutely obvious. Catching of supposedly impossible situations (e.g., a “default” case in C “switches”). Use of “wrappers” for critical modules, in particular to check pre- and post-conditions and return conditions. Coding guidelines to comply with known compiler errors and limits set by compiler assessment.

C.2.6.3 No dynamic variables or dynamic objects

NOTE This technique/measure is referenced in Tables A.2 and B.1 of IEC 61508-3.

Aim: To exclude

- unwanted or undetected overlay of memory;
- bottlenecks of resources during (safety-related) run-time.

Description: In the case of this measure, dynamic variables and dynamic objects are those variables and objects that have their memory allocated and absolute addresses determined at run-time. The value of allocated memory and its addresses depend on the state of the system at the moment of allocation, which means that it cannot be checked by the compiler or any other off-line tool.

Because the number of dynamic variables and objects, and the existing free memory space for allocating new dynamic variables or objects, depends on the state of the system at the moment of allocation, it is possible for faults to occur when allocating or using the variables or objects. For example, when the amount of free memory at the location allocated by the system is insufficient, the memory contents of another variable can be inadvertently overwritten. If dynamic variables or objects are not used, these faults are avoided.

Restrictions on the use of dynamic objects are needed where the dynamic behaviour cannot be accurately predicted by some static analysis (i.e. in advance of the program execution), and therefore predictable program execution cannot be guaranteed.

C.2.6.4 On-line checking during creation of dynamic variables or dynamic objects

NOTE 1 This technique/measure is referenced in Table B.1 of IEC 61508-3.

Aim: To check that the memory to be allocated to dynamic variables and objects is free before allocation takes place, ensuring that the allocation of dynamic variables and objects during run-time does not impact existing variables, data or code.

Description: In the case of this measure, dynamic variables are those variables that have their memory allocated and absolute addresses determined at run-time (variables in this sense are also the attributes of object instances).

By means of hardware or software, the memory is checked to ensure it is free before a dynamic variable or object is allocated to it (for example, to avoid stack overflow). If allocation is not allowed (for example if the memory at the determined address is not sufficient), appropriate action must be taken. After a dynamic variable or object has been used (for example, after exiting a subroutine) the whole memory which was allocated to it must be freed.

NOTE 2 An alternative is to demonstrate statically that memory will be adequate in all cases.

C.2.6.5 Limited use of interrupts

NOTE This technique/measure is referenced in Table B.1 of IEC 61508-3.

Aim: To keep software verifiable and testable.

Description: The use of interrupts should be restricted. Interrupts may be used if they simplify the system. Software handling of interrupts should be inhibited during critical parts (for example time critical, critical to data changes) of the executed functions. If interrupts are used, then parts not interruptible should have a specified maximum computation time, so that the maximum time for which an interrupt is inhibited can be calculated. Interrupt usage and masking should be thoroughly documented.

C.2.6.6 Limited use of pointers

NOTE This technique/measure is referenced in Table B.1 of IEC 61508-3.

Aim: To avoid the problems caused by accessing data without first checking range and type of the pointer. To support modular testing and verification of software. To limit the consequence of failures.

Description: In the application software, pointer arithmetic may be used at source code level only if pointer data type and value range (to ensure that the pointer reference is within the correct address space) are checked before access. Inter-task communication of the application software should not be done by direct reference between the tasks. Data exchange should be done via the operating system.

C.2.6.7 Limited use of recursion

NOTE This technique/measure is referenced in Table B.1 of IEC 61508-3.

Aim: To avoid unverifiable and untestable use of subroutine calls.

Description: If recursion is used, there must be a clear criterion which makes predictable the depth of recursion.

C.2.7 Structured programming

NOTE This technique/measure is referenced in Table A.4 of IEC 61508-3.

Aim: To design and implement the program in a way that it is practical to analyse without it being executed. The program may contain only an absolute minimum of statistically untestable behaviour.

Description: The following principles should be applied to minimise structural complexity:

- divide the program into appropriately small software modules, ensuring they are decoupled as far as possible and all interactions are explicit;
- compose the software module control flow using structured constructs, that is sequences, iterations and selection;
- keep the number of possible paths through a software module small, and the relation between the input and output parameters as simple as possible;
- avoid complicated branching and, in particular, avoid unconditional jumps (goto) in higher level languages;
- where possible, relate loop constraints and branching to input parameters;
- avoid using complex calculations as the basis of branching and loop decisions.

Features of the programming language which encourage the above approach should be used in preference to other features which are (allegedly) more efficient, except where efficiency takes absolute priority (for example some safety critical systems).

References:

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985, 9780521780988

A Discipline of Programming. E. W. Dijkstra. Englewood Cliffs NJ, Prentice-Hall, 1976

C.2.8 Information hiding/encapsulation

NOTE This technique/measure is referenced in Table B.9 of IEC 61508-3.

Aim: To prevent unintended access to data or procedures and thereby support a good program structure.

Description: Data that is globally accessible to all software elements can be accidentally or incorrectly modified by any of these elements. Any changes to these data structures may require detailed examination of the code and extensive modifications.

Information hiding is a general approach for minimising these difficulties. The key data structures are "hidden" and can only be manipulated through a defined set of access procedures. This allows the internal structures to be modified or further procedures to be added without affecting the functional behaviour of the remaining software. For example, a name directory might have access procedures "insert", "delete" and "find". The access procedures and internal data structures could be re-written (for example to use a different look-up method or to store the names on a hard disk) without affecting the logical behaviour of the remaining software using these procedures.

In this connection, the concept of abstract data types should be used. If direct support is not provided, then it may be necessary to check that the abstraction has not been inadvertently broken.

References:

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985, 9780521780988

On the Design and Development of Program Families. D. L. Parnas. IEEE Trans SE-2, March 1976

C.2.9 Modular approach

NOTE This technique/measure is referenced in Tables A.4 and B.9 of IEC 61508-3.

Aim: Decomposition of a software system into small comprehensible parts in order to limit the complexity of the system.

Description: A modular approach or modularisation contains several rules for the design, coding and maintenance phases of a software project. These rules vary according to the design method employed during design. Most methods contain the following rules:

- a software module (or equivalently, subprogram) should have a single well-defined task or function to fulfil;
- connections between software modules should be limited and strictly defined, coherence in one software module shall be strong;
- collections of subprograms should be built providing several levels of software modules;
- subprogram sizes should be restricted to some specified value, typically two to four screen sizes;
- subprograms should have a single entry and a single exit only;
- software modules should communicate with other software modules via their interfaces – where global or common variables are used they should be well structured, access should be controlled and their use should be justified in each instance;
- all software module interfaces should be fully documented;
- any software module’s interface should contain only those parameters necessary for its function. However, this recommendation is complicated by the possibility that a programming language may permit default parameters, or that an object-oriented approach is used.

References:

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985, 9780521780988

C.2.10 Use of trusted/verified software elements

NOTE This technique/measure is referenced in Tables A.2, C.2, A.4 and C.4 of IEC 61508-3.

Aim: To avoid the need for software designs and elements to be extensively revalidated or redesigned for each new application. To take advantage of designs which have not been formally or rigorously verified, but for which considerable operational history is available. To take advantage of a pre-existing software element which has been verified for a different application and for which a body of verification evidence exists.

Description: This measure verifies that the software elements are sufficiently free from systematic design faults and/or operational failures.

It is generally impractical to build a complex system from the most rudimentary parts. It is generally necessary to make use of major subassemblies (“elements”, see IEC 61508-4, 3.4.5 and 3.2.8) that have been previously developed to provide some useful function and which can be reused to implement some part of the new system.

Well-designed and structured PESs are made up of a number of software elements which are clearly distinct and which interact with each other in clearly specified ways. Building up a library of such generally applicable software elements which can be reused in several applications allows much of the resource necessary for validating the designs to be shared by more than one application.

However, for safety related applications it is essential to have sufficient confidence that the new system incorporating these pre-existing elements has the required safety integrity, and that safety is not compromised by some incorrect behaviour of the pre-existing element.

Two viewpoints are possible in order to gain confidence that the behaviour of a pre-existing element can be accurately known:

- to analyse a comprehensive operational history of the element to demonstrate that the element has been “proven-in-use”;

- to assess a body of verification evidence that has been gathered on the behaviour of the element to determine if the element meets the requirements of this standard.

C.2.10.1 Proven-in-use

Only in rare cases will “proven-in-use” (see IEC 61508-4, 3.8.18) be a sufficient argument that a trusted software element achieves the necessary safety integrity. For complex elements with many possible functions (e.g. an operating system), it is essential to establish which functions of the element are actually sufficiently proven-in-use. For example, where a self-test routine is provided to detect faults, if no failure occurs within the operating period, one cannot consider the self-test routine for fault detection as being proven-in-use.

A software element can be considered to be proven-in-use if it fulfils the following criteria:

- unchanged specification;
- systems in different applications;
- at least one year of service history;
- operating time according to the safety integrity level or suitable number of demands; demonstration of a non-safety-related failure rate of less than
  - 10^{-2} per demand (year) with a confidence of 95 % requires 300 operational runs (years),
  - 10^{-5} per demand (year) with a confidence of 99,9 % requires 690 000 operational runs (years);

NOTE 1 See Annex D for some mathematical aspects supporting the above numerical estimates. See also B.5.4 for a similar measure and statistical approach.

- all of the operating experience must relate to a known demand profile of the functions of the software element, to ensure that increased operating experience genuinely leads to an increased knowledge of the behaviour of the software element relative to that demand profile;
- no safety-related failures.

NOTE 2 A failure which may not be safety critical in one context can be safety critical in another, and vice versa.

To enable verification that software element fulfils the criteria, the following must be documented:

- exact identification of each system and its elements, including version numbers (for both software and hardware);
- identification of users, and time of application;
- operating time;
- procedure for the selection of the user-applied systems and application cases;
- procedures for detecting and registering failures, and for removing faults.

C.2.10.2 Assess a body of verification evidence

A pre-existing software element (see IEC 61508-4, 3.2.8) is one that already exists and has not been developed specifically for the current project or SRS. 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.

In order to assess the safety integrity of the new system incorporating the pre-existing software, a body of verification evidence is needed to determine the behaviour of the pre-existing element. This may be derived (1) from the element supplier's own documentation and records of the development process of the element, or (2) it may be created or supplemented by additional qualification activities undertaken by the developer of the new safety related system, or by third parties. This is the "Safety Manual for compliant items" that defines the capabilities and limitations of the potentially reusable software element.

In any case, a Safety Manual for compliant items must exist (or must be created) that is adequate to make possible an assessment of the integrity of a specific Safety Function that depends wholly or partly on the reused element. If not, a conservative conclusion must be drawn that the element has not been justified for safety-related reuse. (This is not to say that the element cannot be justified in any case, but simply that insufficient evidence was found in this particular case.)

This standard has specific requirements for the contents of the Safety Manual for compliant items, see IEC 61508-2 Annex D and IEC 61508-3 Annex D, and IEC 61508-3 7.4.2.12 and 7.4.2.13.

As a very brief indication of content, the Safety Manual for compliant items will address the following:

- that the element's design is known and documented;
- 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;
- that unused and unneeded functions of the element will not prevent the new system from meeting its safety requirements;
- that all credible failure mechanisms of the element in the new system have been identified and that appropriate mitigation has been implemented.

A functional safety assessment of the new system must establish that the reused element is applied strictly within the limits of capability that have been justified by the evidence and assumptions in the element's Compliant Item Safety Manual.

References:

Component-Based Software Development: Case Studies. Kung-Kiu Lau. World Scientific, 2004, ISBN 9812388281, 9789812388285

Software Reuse and Reverse Engineering in Practice. P. A. V. Hall (ed.), Chapman & Hall, 1992, ISBN 0-412-39980-6

Software criticality analysis of COTS/SOUP. P.Bishop, T.Clement, S.Guerra. In Reliability Engineering & System Safety, Volume 81, Issue 3, September 2003, Elsevier Ltd., 2003

C.2.11 Traceability

Aim: To maintain consistency between lifecycle stages.

Description: In order to ensure that the software that results from lifecycle activities meets the requirements for correct operation of the safety-related system, it is essential to ensure consistency between the lifecycle stages. A key concept here is that of "traceability" between activities. This is essentially an impact analysis to check (1) that decisions made at an earlier stage are adequately implemented in later stages (forward traceability), and (2) that decisions made at a later stage are actually required and mandated by earlier decisions.

Forward traceability is broadly concerned with checking that a requirement is adequately addressed in later lifecycle stages. Forward traceability is valuable at several points in the safety lifecycle:

- from the system safety requirements to the software safety requirements;
- from the Software Safety Requirements Specification, to the software architecture;
- from the Software Safety Requirements Specification, to the software design;
- from the Software Design Specification, to the module and integration test specifications;
- from the system and software design requirements for hardware/software integration, to the hardware/software integration test specifications;
- from the Software Safety Requirements Specification, to the software safety validation plan;
- from the Software Safety Requirements Specification, to the software modification plan (including reverification and revalidation);
- from the Software Design Specification, to the software verification (including data verification) plan;

- from the requirements of IEC 61508-3 Clause 8, to the plan for software functional safety assessment.

Backward traceability is broadly concerned with checking that every implementation (interpreted in a broad context, and not confined to code implementation) decision is clearly justified by some requirement. If this justification is absent, then the implementation contains something unnecessary that will add to the complexity but not necessarily address any genuine requirement of the safety-related system. Backward traceability is valuable at several points in the safety lifecycle:

- from the safety requirements, to the perceived safety needs;
- from the software architecture, to the Software Safety Requirements Specification;
- from the software detailed design to the software architecture;
- from the software code to the software detailed design;
- from the software safety validation plan, to the Software Safety Requirements Specification;
- from the software modification plan, to the Software Safety Requirements Specification;
- from the software verification (including data verification) plan, to the Software Design Specification.

Reference:

Requirements Engineering. E. Hull, K. Jackson, J. Dick. Springer, 2005, ISBN 1852338792, 9781852338794

C.2.12 Stateless software design (or limited state design)

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To limit the complexity of software behaviour.

Description: Consider a software program that processes a sequence of transactions: it receives a sequence of inputs and produces an output in response to each input. The program may also memorise some or all of its history in a “state”, and may take this state into account when calculating how to respond to future inputs.

Where the program’s output in response to a specific input is completely determined by that input, then the program is said to be memoryless or stateless. Each input/output transaction is complete in the sense that no transaction is influenced by any earlier transaction, and a specific input always results in the same associated output.

In contrast, where the program takes account of both its specific input and also its memorised state in calculating its output, then the program is capable of more complex behaviour because it can deliver different outputs in response to the same input on different occasions. The response to a specific input may depend on the context (i.e. the previous inputs and outputs) in which the input is received. A further consideration that is relevant to some applications (typically communications) is that the program’s behaviour can be particularly sensitive to changes in the stored state, whether inadvertently or maliciously introduced.

Stateless (or limited state) design is a general approach that aims to minimise the potential complexity of software behaviour by avoiding or minimising the use of state information in the software design.

References:

Introduction to Automata Theory, Languages, and Computation (3rd Edition). J. Hopcroft, R. Motwani, J. Ullman, Addison-Wesley Longman Publishing Co, 2006, ISBN:0321462254

Stateless connections. T. Aura, P Nikander. In Proc International Conference on Information and Communications Security (ICICS'97), ed Yongfei Han. Springer, 1997, ISBN 354063696X, 9783540636960

C.2.13 Offline numerical analysis

NOTE This technique/measure is referenced in Table A.9 of IEC 61508-3.

Aim: To ensure the accuracy of numerical calculations.

Description: Numerical inaccuracy may arise in the calculation of a mathematical function as a consequence of using finite representations of ideal functions and numbers. Truncation error is introduced when a function is approximated by a finite number of terms of an infinite series such as a Fourier series. Rounding error is introduced by the finitely accurate representation of real numbers in a physical computer. When anything but the simplest calculation is performed in floating point, the validity of the calculation must be checked to ensure that the accuracy required by the application is actually achieved.

Reference:

Guide to Scientific Computing. P.R. Turner. CRC Press, 2001, ISBN 0849312426, 9780849312427

C.2.14 Message sequence charts

NOTE This technique/measure is referenced in Tables B.7 and C.17 of IEC 61508-3.

Aim: To assist the capture of system requirements in the early design stages of software development including requirements and software architectural design. In UML, the name "System Sequence Diagram" is used for this notation.

Description: The Message Sequence Chart is a diagrammatic mechanism for describing the behaviour of a system in terms of the communication that takes place between the system actors (and actor may be to a human being, a computer system, or a software element or object, depending on the design phase). For each actor, a vertical "lifeline" is drawn on the diagram and arrows between the lifelines are used to represent messages. Actions upon receipt of messages can be optionally shown on the diagrams as boxes. A collection of scenarios (describing both desirable and undesirable behaviour) is built up as a specification of the required system behaviour. These scenarios have several uses. They can be animated to demonstrate the system behaviour to end-users. They can be transformed into an executable implementation of the system. They can form the basis of test data.

UML contains extensions to the orginal concept of the Message Squence Chart in the form of selection and iteration constructs which allow scenarios to branch and loop, providing a more compact notation. Sub-diagrams can also be defined which can be referenced from a number of higher level sequence diagrams. Timer and external events can also be represented.

References:

"*Message Sequence charts*", D. Harel, P. Thiagarajan. In UML for Real: Design of Embedded Real-Time Systems. ed. L. Lavagno. Springer, 2003, ISBN 1402075014, 9781402075018

ISO/IEC 19501:2005, *Information technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2*

C.3 Architecture design

C.3.1 Fault detection and diagnosis

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To detect faults in a system, which might lead to a failure, thus providing the basis for counter-measures in order to minimise the consequences of failures.

Description: Fault detection is the activity of checking a system for erroneous states (caused by a fault within the (sub)system to be checked). The primary goal of fault detection is to inhibit the effect of wrong results. A system which acts in combination with parallel elements, relinquishing control when it detects its own results are incorrect, is called self-checking.

Fault detection is based on the principles of redundancy (mainly to detect hardware faults – see IEC 61508-2 Annex A) and diversity (software faults). Some sort of voting is needed to decide on the correctness of results. Special methods applicable are: assertion programming, N-version programming and the diverse monitor technique; and for hardware: introducing additional sensors, control loops, error checking codes, etc.

Fault detection may be achieved by checks in the value domain or in the time domain on different levels, especially physical (temperature, voltage etc), logical (error detecting codes), functional (assertions) or external (plausibility checks). The results of these checks may be stored and associated with the data affected to allow failure tracking.

Complex systems are composed of subsystems. The efficiency of fault detection, diagnosis and fault compensation depends on the complexity of the interactions among the subsystems, which influences the propagation of faults.

Fault diagnosis should be applied at the smallest subsystem level, since smaller subsystems allow a more detailed diagnosis of faults (detection of erroneous states).

Integrated enterprise-wide information systems can routinely communicate the status of safety systems, including diagnostic testing information, to other supervisory systems. If an anomaly is detected, it can be highlighted and used to trigger corrective action before a hazardous situation develops. Lastly, if an incident does occur, documentation of such anomalies can aid the subsequent investigation.

Reference: *Dependability of Critical Computer Systems 1*. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0

C.3.2 Error detecting and correcting codes

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To detect and correct errors in sensitive information.

Description: For an information of n bits, a coded block of k bits is generated which enables r errors to be detected and corrected. Two example types are Hamming codes and polynomial codes.

It should be noted that in safety-related systems it will normally be necessary to discard faulty data rather than try to correct it, since only a predetermined fraction of errors may be corrected properly.

Reference:

Fundamentals of Error-correcting Codes, W. Huffman, V. Pless. Cambridge University Press, 2003, ISBN 0521782805, 9780521782807

C.3.3 Failure assertion programming

NOTE This technique/measure is referenced in Table A.17 of IEC 61508-2, and Tables A.2 and C.2 of IEC 61508-3.

Aim: To detect residual software design faults during execution of a program, in order to prevent safety critical failures of the system and to continue operation for high reliability.

Description: The assertion programming method follows the idea of checking a pre-condition (before a sequence of statements is executed, the initial conditions are checked for validity) and a post-condition (results are checked after the execution of a sequence of statements). If either the pre-condition or the post-condition is not fulfilled, the processing reports the error.

For example,

```
assert < pre-condition>;
    action 1;
       :
       :
    action x;
assert < post-condition>;
```

References:

Exploiting Traces in Program Analysis. A. Groce, R. Joshi. Lecture Notes in Computer Science vol 3920, Springer Berlin / Heidelberg, 2006, ISBN 978-3-540-33056-1

Software Development – A Rigorous Approach. C. B. Jones, Prentice-Hall, 1980

C.3.4 Diverse monitor

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To protect against residual specification and implementation faults in software which adversely affect safety.

Description: Two monitoring approaches can be distinguished: (1) the monitor and the monitored function in the same computer, with some guarantee of independence between them; and (2) the monitor and the monitored function in separate computers.

A diverse monitor is an external monitor, implemented on an independent computer to a different specification. This diverse monitor is solely concerned with ensuring that the main computer performs safe, not necessarily correct, actions. The diverse monitor continuously monitors the main computer. The diverse monitor prevents the system from entering an unsafe state. In addition, if it detects that the main computer is entering a potentially hazardous state, the system has to be brought back to a safe state either by the diverse monitor or the main computer.

Hardware and software of the diverse monitor should be classified and qualified according to the appropriate SIL.

Reference:

Requirements based Monitors for Real-Time Systems, D. Peters, D. Parnas. IEEE Transactions on Software Engineering, vol. 28, no. 2, 2002

C.3.5 Software diversity (diverse programming)

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: Detect and mask residual software design and implementation faults during execution of a program, in order to prevent safety critical failures of the system, and to continue operation for high reliability.

Description: In diverse programming a given program specification is designed and implemented *N* times in different ways. The same input values are given to the *N* versions, and the results produced by the *N* versions are compared. If the result is considered to be valid, the result is transmitted to the computer outputs.

An essential requirement is that the *N* versions be independent of each other in some sense, so that they do not all fail simultaneously due to the same cause. In practice it may be very difficult to achieve and to demonstrate the version independence that is the foundation of the *N*-version approach.

The *N* versions can run in parallel on separate computers, alternatively all versions can be run on the same computer and the results subjected to an internal vote. Different voting strategies can be used on the *N* versions, depending on the application requirements, as follows.

– If the system has a safe state, then it is feasible to demand complete agreement (all *N* agree) otherwise an output value is used that will cause the system to reach the safe state. For simple trip systems the vote can be biased in the safe direction. In this case the safe action would be to trip if either version demanded a trip. This approach typically uses only two versions (*N*=2).

– For systems with no safe state, majority voting strategies can be employed. For cases where there is no collective agreement, probabilistic approaches can be used in order to maximise the chance of selecting the correct value, for example, taking the middle value, temporary freezing of outputs until agreement returns, etc.

This technique does not eliminate residual software design faults, nor does it avoid errors in the interpretation of the specification, but it provides a measure to detect and mask before they can affect safety.

References:

Modelling software design diversity – a review, B. Littlewood, P. Popov, L. Strigini. ACM Computing Surveys, vol 33, no 2, 2001

The N-Version Approach to Fault-Tolerant Software, A.Avizienis, IEEE Transactions on Software Engineering, vol. SE-11, no. 12 pp.1491-1501, 1985

An experimental evaluation of the assumption of independence in multi-version programming, J.C. Knight, N.G. Leveson. IEEE Transactions on Software Engineering, vol. SE-12, no 1, 1986

In Search of Effective Diversity: a Six Language Study of Fault-Tolerant Flight Control Software. A. Avizienis, M. R. Lyu and W. Schutz. 18th Symposium on Fault-Tolerant Computing, Tokyo, Japan, 27-30 June 1988, IEEE Computer Society Press, 1988, ISBN 0-8186-0867-6

C.3.6 Backward recovery

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To provide correct functional operation in the presence of one or more faults.

Description: If a fault has been detected, the system is reset to an earlier internal state, the consistency of which has been proven before. This method implies saving of the internal state frequently at so-called well-defined checkpoints. This may be done globally (for the complete database) or incrementally (changes only between checkpoints). Then the system has to compensate for the changes which have taken place in the meantime by using journalling (audit trail of actions), compensation (all effects of these changes are nullified) or external (manual) interaction.

References:

Looking into Compensable Transactions. Jing Li, Huibiao Zhu, Geguang Pu, Jifeng He. In Software Engineering Workshop, 2007. SEW 2007. IEEE, 2007, ISBN 978-0-7695-2862-5

Software Fault Tolerance (Trends in Software, No. 3), M. R. Lyu (ed.), John Wiley & Sons, April 1995, ISBN 0471950688

C.3.7 Re-try fault recovery mechanisms

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To attempt functional recovery from a detected fault condition by re-try mechanisms.

Description: In the event of a detected fault or error condition, attempts are made to recover the situation by re-executing the same code. Recovery by re-try can be as complete as a reboot and a re-start procedure or a small re-scheduling and re-starting task, after a software time-out or a task monitoring action. Re-try techniques are commonly used in communication fault or error recovery, and re-try conditions could be flagged from a communication protocol error (checksum, etc.) or from a communication acknowledgement response time-out.

Reference:

Reliable Computer Systems: Design and Evaluation, D.P. Siewiorek, R.S. Schwartz. A.K. Peters Ltd., 1998, ISBN 156881092X, 9781568810928

C.3.8 Graceful degradation

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To maintain the more critical system functions available, despite failures, by dropping the less critical functions.

Description: This technique gives priorities to the various functions to be carried out by the system. The design ensures that if there is insufficient resources to carry out all the system functions, the higher priority functions are carried out in preference to the lower ones. For example, error and event logging functions may be lower priority than system control functions, in which case system control would continue if the hardware associated with error logging were to fail. Further, should the system control hardware fail, but not the error logging hardware, then the error logging hardware would take over the control function.

This is predominantly applied to hardware but is applicable to the total system including software. It must be taken into account from the topmost design phase.

References:

Towards the Integration of Fault, Resource, and Power Management, T. Siridakis. In Computer Safety, Reliability, and Security: 23rd International Conference, SAFECOMP 2004. Eds. Maritta Heisel et. al. Springer, 2004, ISBN 3540231765, 9783540231769

Achieving Critical System Survivability Through Software Architectures, J.C. Knight, E.A. Strunk. Springer Berlin / Heidelberg, 2004, 978-ISBN 3-540-23168-4

The Evolution of Fault-Tolerant Computing. Vol. 1 of Dependable Computing and Fault-Tolerant Systems, Edited by A. Avizienis, H. Kopetz and J. C. Laprie, Springer Verlag, 1987, ISBN 3-211-81941-X

Fault Tolerance, Principle and Practices. T. Anderson and P. A. Lee, Vol. 3 of Dependable Computing and Fault-Tolerant Systems, Springer Verlag, 1987, ISBN 3-211-82077-9

C.3.9 Artificial intelligence fault correction

NOTE 1 This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To be able to react to possible hazards in a very flexible way by introducing a combination of methods and process models and some kind of on-line safety and reliability analysis.

Description: Fault forecasting (calculating trends), fault correction, maintenance and supervisory actions may be supported by artificial intelligence (AI) based systems in a very efficient way in diverse channels of a system, since the rules might be derived directly from the specifications and checked against these. Certain common faults which are introduced into specifications, by implicitly already having some design and implementation rules in mind, may be avoided effectively by this approach, especially when applying a combination of models and methods in a functional or descriptive manner.

The methods are selected in such a way that faults may be corrected and the effects of failures be minimised, in order to meet the desired safety integrity.

NOTE 2 See C.3.2 for warning about correcting faulty data, and item 5, Table A.2 of IEC 61508-3 for negative recommendations concerning this technique.

Reference:

Fault Diagnosis: Models, Artificial Intelligence, Applications. J. Korbicz, J. Koscielny, Z. Kowalczuk, W. Cholewa. Springer, 2004, ISBN 3540407677, 9783540407676

C.3.10 Dynamic reconfiguration

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To maintain system functionality despite an internal fault.

Description: The logical architecture of the system has to be such that it can be mapped onto a subset of the available resources of the system. The architecture needs to be capable of detecting a failure in a physical resource and then remapping the logical architecture back onto the restricted resources left functioning. Although the concept is more traditionally restricted to recovery from failed hardware units, it is also applicable to failed software units if there is sufficient "run-time redundancy" to allow a software re-try or if there is sufficient redundant data to make the individual and isolated failure be of little importance.

This technique must be considered at the first system design stage.

Reference:

Dynamic Reconfiguration of Software Architectures Through Aspects. C. Costa et al. Lecture Notes in Computer Science, Volume 4758/2007, Springer Berlin / Heidelberg, 2007, ISBN 978-3-540-75131-1

C.3.11 Safety and Performance in real time: Time-Triggered Architecture

NOTE 1 This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: Composability and transparent implementation of fault-tolerance into safety-critical real-time systems with predictable behaviour.

Description: In a Time-Triggered Architecture (TTA) system, all system activities are initiated and based on the progression of a globally synchronised time-base. Each application is assigned a fixed time slot on the time-triggered bus, which contains the messages exchanged between the jobs of each application which can therefore be exchanged only according to a defined schedule. In event-driven systems, system activities are triggered by arbitrary events at unpredictable points in time. The key advantages of a TTA are (see reference Scheidler, Heiner et. al.):

– composability, which greatly reduces the effort required for testing and certifying the system;
– transparent implementation of fault-tolerance, which makes the architecture highly recommendable for safety-critical applications;
– provision of a globally synchronised time-base, which facilitates the design of distributed real-time systems.

Communication between nodes is done using the *Time-Triggered Protocol TTP/C* (see reference Kopetz, Hexel et. al.) according to a static schedule, deciding when to transmit a message and whether a received message is relevant for the particular electronic module or not. Access to the bus is controlled by a cyclic *time-division multiple access (TDMA)* schema derived from the global notion of time.

The TTP/C protocol guarantees (see reference Rushby) four basic services (core services) in a network of TTA nodes (see reference Kopetz, Bauer):

– Deterministic and timely message transport: Transport of messages from the output port of the sending element to the input ports of the receiving elements within an a priori known time bound. A fault-tolerant transport service is offered by a time-triggered communication service that is available via the temporal firewall interface which eliminates control error propagation by design and minimises coupling between elements. The timely transport of messages with minimal latency and jitter is crucial for the achievement of control stability in real-time applications.
– Fault-tolerant Clock Synchronization: The communication controller generates a fault-tolerant synchronised global time base (with a precision within a few clock tics) that is provided to the host subsystem.
– Consistent Diagnosis of Failing Nodes (Membership Service): The communication controller informs every SRU (“smallest replaceable unit”) about the state of every other SRU in a cluster with a latency of less than one TDMA round.
– Strong Fault Isolation: A maliciously faulty host subsystem (including its software) can produce erroneous data outputs, but can never interfere in any other way with the correct operation of the rest of a TTP/C cluster. Fail silence in the temporal domain is guaranteed by the time-triggered behaviour of the communication controller.

NOTE 2 Other time-triggered protocols are FlexRay and TT-Ethernet (time-triggered Ethernet).

References:

Time-Triggered Architecture (TTA). C. Scheidler, G. Heiner, R. Sasse, E. Fuchs, H. Kopetz, C. Temple. In Advances in Information Technologies: The Business Challenge, ed. J-Y. Roger. IOS Press, 1998, ISBN 9051993854, 9789051993851

A Synchronisation Strategy for a TTP/C Controller. H. Kopetz, R. Hexel, A. Krueger, D. Millinger, A. Schedl. SAE paper 960120, Application of Multiplexing Technology SP 1137, Detroit, SAE Press, Warrendale, 1996

The Time-Triggered Architecture. H. Kopetz, G. Bauer. Proceedings of the IEEE Special Issue on Modeling and Design of Embedded Software, October 2002

An Overview of Formal Verification for the Time-Triggered Architecture. J. Rushby: Invited paper, Oldenburg, Germany, September 9-12, 2002. Proceedings FTRTFT 2002, Springer LNCS 2469, 2002, ISBN 978-3-540-44165-6

C.3.12 UML

NOTE This technique/measure is referenced in Table B.7 of IEC 61508-3.

Aim: To provide a comprehensive set of notations for modelling the desired behaviour of complex systems

Description: UML is, as the name implies, a collection of requirements and design notations which is intended to provide comprehensive support for software development. Some parts of UML are based upon notations first introduced in other methods (such as system sequence diagrams and state transition diagrams) and other notations are unique to UML. UML is strongly biased towards object-oriented concepts although some of the notations can be used without any necessity to proceed to an object-oriented programming. UML is supported by a number of commercially available CASE tools, many of which are capable of automatically generating code from the UML models.

The UML notations which are most generally applicable to the specification and design of safety related systems are the following:

- Class diagrams
- Use cases
- Activity diagrams
- State transition diagrams (Statecharts)
- System sequence diagrams

Other UML notations are relevant to the expression of software architectural design (software structure) but are not listed specifically here.

State transition diagrams are described in B.2.3.2 and system sequence diagrams in C.2.14. The other notations are described in the following three subsections.

C.3.12.1 Class diagrams

Class diagrams define the classes of objects with which the software has to deal. They are based upon earlier entity-relationship-attribute diagrams but are adapted for object oriented design. Each class (of which there will be one or more instances known as objects at run time) is represented as a rectangle and the various relationships between the classes are shown as lines or arrows. The operations or methods offered by each class, and the data attributes of each class, can be added to the diagram. The relationships which can be represented consist of both reference relationships with their cardinality (an instance of class

A may refer to one or many instances of class B) and specialisation relationships (Class X is a refinement of class Y) with possibly additional methods and attributes. Multiple inheritance can be depicted.

C.3.12.2 Use cases

Use cases provide a textual description of the desired behaviour of the system in reponse to a particular scenario, usually from the point of view of external actors including human users of the system and external systems. Alternative sub-scenarios within a given use case can be used to represent optional behaviour, especially in error response cases. A collection of use cases is developed to provide a sufficiently complete specification of the system requirements. Use clauses can be the starting point for the development more rigorous models such as system sequence diagrams and activity diagrams.

Use case diagrams provide a pictorial representation of the system and the actors who are involved in the use cases, but are not rigorous and only the text of the use case is important for specification.

C.3.12.3 Activity diagrams

An activity diagram shows the intended sequence of actions carried out by a software element (often an object in an object-oriented design) including sequential and iterative behaviour (some aspects look remarkably like a flowchart). Activity diagrams however allow the actions of a number of elements to be described in parallel, with the interactions between the elements shown by arrows on the diagram. Synchronisation points where an activity must wait for one or more inputs from other activities before it can proceed are shown by a symbol similar to a Petri net node.

Reference:

ISO/IEC 19501:2005, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

C.4 Development tools and programming languages

C.4.1 Strongly typed programming languages

NOTE This technique/measure is referenced in Table A.3 of IEC 61508-3.

Aim: Reduce the probability of faults by using a language which permits a high level of checking by the compiler.

Description: When a strongly typed programming language is compiled, many checks are made on how variable types are used, for example in procedure calls and external data access. Compilation will fail and an error message be produced for any usage that does not conform to predefined rules.

Such languages usually allow user-defined data types to be defined from the basic language data types (such as integer, real). These types can then be used in exactly the same way as the basic type. Strict checks are imposed to ensure the correct type is used. These checks are imposed over the whole program, even if this is built from separately compiled units. The checks also ensure that the number and the type of procedure arguments match even when referenced from separately compiled software modules.

Strongly typed languages usually support other aspects of good software engineering practice such as easily analysable control structures (for example if.. then.. else, do.. while, etc.) which lead to well-structured programs.

Typical examples of strongly typed languages are Pascal, Ada and Modula 2.

Reference:

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985, 9780521780988

C.4.2 Language subsets

NOTE This technique/measure is referenced in Table A.3 of IEC 61508-3.

Aim: To reduce the probability of introducing programming faults and increase the probability of detecting any remaining faults.

Description: The language is examined to determine programming constructs which are either error-prone or difficult to analyse, for example, using static analysis methods. A language subset is then defined which excludes these constructs.

References:

Practical Experiences of Safety- and Security-Critical Technologies, P. Amey, A.J. Hilton. Ada User Journal, June, 2004

Safer C: Developing Software for High-integrity and Safety-critical Systems. L. Hatton, McGraw-Hill, 1994, ISBN 0077076400, 9780077076405

Requirements for programming languages in safety and security software standard. B. A. Wichmann. Computer Standards and Interfaces. Vol. 14, pp 433-441, 1992

C.4.3 Certified tools and certified translators

NOTE This technique/measure is referenced in Table A.3 of IEC 61508-3.

Aim: Tools are necessary to help developers in the different phases of software development. Wherever possible, tools should be certified so that some level of confidence can be assumed regarding the correctness of the outputs.

Description: The certification of a tool will generally be carried out by an independent, often national, body, against independently set criteria, typically national or international standards. Ideally, the tools used in all development phases (specification, design, coding, testing and validation) and those used in configuration management, should be subject to certification.

To date, only compilers (translators) are regularly subject to certification procedures; these are laid down by national certification bodies and they exercise compilers (translators) against international standards such as those for Ada and Pascal.

It is important to note that certified tools and certified translators are usually certified only against their respective language or process standards. They are usually not certified in any way with respect to safety.

References:

The certification of software tools with respect to software standards, P. Bunyakiati et al. In IEEE International Conference on Information Reuse and Integration, IRI 2007, IEEE, 2007, ISBN 1-4244-1500-4

Certified Testing of C Compilers for Embedded Systems. O. Morgan. In: 3rd Institution of Engineering and Technology Conference on Automotive Electronics. IEEE, 2007, ISBN 978-0-86341-815-0

The Ada Conformity Assessment Test Suite (ACATS), version 2.5, Ada Conformity Assessment Authority, http://www. ic.org/compilerstesting.html , Apr. 2002

C.4.4 Tools and translators: increased confidence from use

NOTE This technique/measure is referenced in Table A.3 of IEC 61508-3.

Aim: To avoid any difficulties due to translator failures which can arise during development, verification and maintenance of a software package.

Description: A translator is used, where there has been no evidence of improper performance over many prior projects. Translators without operating experience or with any serious known faults should be avoided unless there is some other assurance of correct performance (for example, see C.4.4.1).

If the translator has shown small deficiencies, the related language constructs are noted down and carefully avoided during a safety related project.

Another version to this way of working is to restrict the usage of the language to only its commonly used features.

This recommendation is based on the experience from many projects. It has been shown that immature translators are a serious handicap to any software development. They make a safety-related software development generally infeasible.

It is also known, presently, that no method exists to prove the correctness for all tool or translator parts.

C.4.4.1 Comparison of source program and executable code

Aim: To check that the tools used to produce a PROM image have not introduced any errors into the PROM image.

Description: The PROM image is reverse-engineered to obtain the constituent "object" modules. These "object" modules are reverse-engineered into assembly language files. Using suitable techniques the reverse generated assembly language files are compared with the actual source files originally used to produce the PROM.

The major advantage of the technique is that the tools (compilers, linkers etc.) used to produce the PROM image do not have to be validated for all programs. The technique verifies that source file used for the particular safety-related system are correctly transformed.

References:

Demonstrating Equivalence of Source Code and PROM Contents. D. J. Pavey and L. A. Winsborrow. The Computer Journal Vol. 36, No. 7, 1993

Formal demonstration of equivalence of source code and PROM contents: an industrial example. D. J. Pavey and L. A. Winsborrow. Mathematics of Dependable Systems, Ed. C. Mitchell and V. Stavridou, Clarendon Press, 1995, ISBN 0-198534-91-4

Assuring Correctness in a Safety Critical Software Application. L. A. Winsborrow and D. J. Pavey. High Integrity Systems, Vol. 1, No. 5, pp 453-459, 1996

C.4.5 Suitable programming languages

NOTE This technique/measure is referenced in Table A.3 of IEC 61508-3.

Aim: To support the requirements of this International Standard as much as possible, in particular defensive programming, strong typing, structured programming and possibly assertions. The programming language chosen should lead to an easily verifiable code with a minimum of effort and facilitate program development, verification and maintenance.

Description: The language should be fully and unambiguously defined. The language should be user- or problem-orientated rather than processor/platform machine-orientated. Widely used languages or their subsets are preferred to special purpose languages.

In addition to the already referenced features the language should provide for

- block structure;
- translation time checking; and
- run-time type and array bound checking.

The language should encourage

- the use of small and manageable software modules;
- restriction of access to data in specific software modules;
- definition of variable subranges; and
- any other type of error-limiting constructs.

If safe operation of the system is dependent upon real-time constraints, then the language should also provide for exception/interrupt handling.

It is desirable that the language is supported by a suitable translator, appropriate libraries of pre-existing software modules, a debugger and tools for both version control and development.

Currently, at the time of developing this standard, it is not clear whether object-oriented languages are to be preferred to other conventional ones.

Features which make verification difficult and therefore should be avoided are

- unconditional jumps excluding subroutine calls;
- recursion;
- pointers, heaps or any type of dynamic variables or objects;
- interrupt handling at source code level;
- multiple entries or exits of loops, blocks or subprograms;
- implicit variable initialisation or declaration;
- variant records and equivalence; and
- procedural parameters.

Low-level languages, in particular assembly languages, present problems due to their processor/platform machine-orientated nature.

A desirable language property is that its design and use should result in programs whose execution is predictable. Given a suitably defined programming language, there is a subset which ensures that program execution is predictable. This subset cannot (in general) be statically determined, although many static constraints may assist in ensuring predictable execution. This would typically require a demonstration that array indices are within bounds, and that numeric overflow cannot arise, etc.

Table C.1 gives recommendations for specific programming languages.

References:

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985, 9780521780988

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

IEC 61131-3:2003, *Programmable controllers – Part 3: Programming languages*

ISO/IEC 1539-1:2004, *Information technology – Programming languages – Fortran – Part 1:Base language*

ISO/IEC 7185:1990, *Information technology – Programming languages – Pascal*

ISO/IEC 8652:1995, *Information technology – Programming languages – Ada*

ISO/IEC 9899:1999, *Programming languages – C*

ISO/IEC 10206:1991, *Information technology – Programming languages – Extended Pascal*

ISO/IEC 10514-1:1996, *Information technology – Programming languages – Part 1: Modula-2, Base Language*

ISO/IEC 10514-3:1998, *Information technology – Programming languages – Part 3: Object Oriented Modula-2*

ISO/IEC 14882:2003, *Programming languages – C++*

ISO/IEC/TR 15942:2000, *Information technology — Programming languages — Guide for the use of the Ada programming language in high integrity systems*

Table C.1 – Recommendations for specific programming languages

	Programming language	SIL1	SIL2	SIL3	SIL4
1	ADA	HR	HR	R	R
2	ADA with subset	HR	HR	HR	HR
3	Java	NR	NR	NR	NR
4	Java with subset (including either no garbage collection or garbage collection which will not cause the application code to stop for a significant period of time). See Annex G for guidance on use of object oriented facilities.	R	R	NR	NR
5	PASCAL (see Note 1)	HR	HR	R	R
6	PASCAL with subset	HR	HR	HR	HR
7	FORTRAN 77	R	R	R	R
8	FORTRAN 77 with subset	HR	HR	HR	HR
9	C	R	–	NR	NR
10	C with subset and coding standard, and use of static analysis tools	HR	HR	HR	HR
11	C++ (see Annex G for guidance on use of object oriented facilities)	R	–	NR	NR
12	C++ with subset and coding standard, and use of static analysis tools (see Annex G for guidance on use of object oriented facilities)	HR	HR	HR	HR
13	Assembler	R	R	–	–
14	Assembler with subset and coding standard	R	R	R	R
15	Ladder diagrams	R	R	R	R
16	Ladder diagram with defined subset of language	HR	HR	HR	HR

	Programming language	SIL1	SIL2	SIL3	SIL4
17	Functional block diagram	R	R	R	R
18	Function block diagram with defined subset of language	HR	HR	HR	HR
19	Structured text	R	R	R	R
20	Structured text with defined subset of language	HR	HR	HR	HR
21	Sequential function chart	R	R	R	R
22	Sequential function chart with defined subset of language	HR	HR	HR	HR
23	Instruction list	R	–	NR	NR
24	Instruction list with defined subset of language	HR	R	R	R

NOTE 1 The recommendationsHR, R and – NR are explained in Annex A of IEC 61508-3.

NOTE 2 System software includes the operating system, drivers, embedded functions and software modules provided as part of the system. The software is typically provided by the safety system vendor. The language subset should be carefully selected to avoid complex structures which may result in implementation faults. Checks should be performed to check for proper use of the language subset.

NOTE 3 The application software is the software developed for a specific safety application. In many cases this software is developed by the end user or by an application oriented contractor. Where a number of programming languages have the same recommendation, the developer should select one which is commonly used by personnel in the industry or facility. The language subset should be carefully selected to avoid complex structures which may result in implementation faults. Checks should be performed to check for proper use of the language subset.

NOTE 4 If a specific language is not listed in the table, it must not be assumed that it is excluded. It should conform to this International Standard.

NOTE 5 There are a number of extensions to the Pascal language including Free Pascal. References to Pascal include these extensions.

NOTE 6 Java is designed to have a run-time garbage collector. A subset of Java can be defined which does not require garbage collection. Some Java implementations provide progressive garbage collection which recovers free memory as the program executes and prevent execution stopping for a period when available memory is exhausted. Hard real time applications should not use any form of garbage collection.

NOTE 7 If the Java implementation requires a run-time interpreter of Java intermediate code, then the interpreter must be treated as part of the safety related software and treated in accordance with the requirements of IEC 61508-3.

NOTE 8 For entries 15-24, see IEC 61131-3.

C.4.6 Automatic software generation

NOTE This technique/measure is referenced in Table A.2 of IEC 61508-3.

Aim: To automate the more error-prone tasks of software implementation.

Description: The system design is described by a model (an executable specification) at a higher level of abstraction than the traditional executable code. The model is transformed automatically by a code generator into executable form. The aim is to improve software quality by eliminating the error-prone manual tasks of coding. A further potential benefit is that more complex designs can be undertaken at the higher abstract level.

References:

Embedded Software Generation from System Level Design Languages, H Yu, R. Domer, D. Gajski. In “ASP-DAC 2004: Proceedings of the ASP-Dac 2004 Asia and South Pacific Design Automation Conference, 2004”, IEEE Circuits and Systems Society. IEEE, 2004, ISBN 0780381750, 9780780381759

Transforming Process Algebra Models into UML State Machines: Bridging a Semantic Gap?. M.F. van Amstel et. al. In Theory and Practice of Model Transformations: First International Conference, ICMT”. ed. A. Vallecillo. Springer, 2008, ISBN 3540699260, 9783540699262

C.4.7 Test management and automation tools

NOTE This technique/measure is referenced in Table A.5 of IEC 61508-3.

Aim: To encourage a systematic and thorough approach to software and system testing.

Description: The use of appropriate support tools mechanises the more labour-intensive and error-prone tasks in system development and brings the capability for a systematic approach to test management. The availability of support encourages a more thorough approach to both normal and regression testing.

Reference:

Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing. R.Black, John Wiley and Sons, 2002, ISBN 0471223980, 9780471223986

C.5 Verification and modification

C.5.1 Probabilistic testing

NOTE This technique/measure is referenced in Tables A.5, C.15, A.7 and C.17 of IEC 61508-3.

Aim: To gain a quantitative figure about the reliability properties of the investigated software.

Description: The method produces a statistical estimate of software reliability. This quantitative figure may take into account the related levels of confidence and significance and can give

– a failure probability per demand;
– a failure probability during a certain period of time; and
– a probability of error containment.

From these figures, other parameters may be derived such as:

– probability of failure free execution;
– probability of survival;
– availability;
– MTBF or failure rate; and
– probability of safe execution.

Probabilistic considerations are either based on a probabilistic test or on operating experience. Usually, the number of test cases or observed operating cases is very large. Typically, the testing of the demand mode of operation involves considerably less elapsed time than the continuous mode of operation.

Automated testing tools are normally employed to provide test data and supervise test outputs. Large tests are run on large host computers with the appropriate process simulation periphery. Test data is selected both according to systematic and random hardware viewpoints. The overall test control, for example, guarantees a test data profile, while random selection can govern individual test cases in detail.

Individual test harnesses, test executions and test supervisions are determined by the detailed test aims as described above. Other important conditions are given by the mathematical prerequisites that must be fulfilled if the test evaluation is to meet its intended test aim.

Probabilistic figures about the behaviour of any test object may also be derived from operating experience. Provided the same conditions are met, the same mathematics can be applied as for the evaluation of test results.

In practice, it is very difficult to demonstrate ultra-high levels of reliability using these techniques.

References:

A discussion of statistical testing on a safety-related application. S Kuball, J H R May, Proc IMechE Vol. 221 Part O: J. Risk and Reliability, Institution of Mechanical Engineers, 2007

Estimating the Probability of Failure when Testing Reveals No Failures, W.K. Miller, L.J. Morell, et al.. IEEE Transactions on Software Engineering, VOl. 18, NO.1, pp33-43, January 1992

Reliability estimation from appropriate testing of plant protection software, J. May, G. Hughes, A.D. Lunn. IEE Software Engineering Journal, v10 n6 pp 206-218, Nov 1995 (ISSN: 0268-6961)

Validation of ultra high dependability for software based systems, B. Littlewood and L. Strigini. Comm. ACM 36 (11), 69-80, 1993

C.5.2 Data recording and analysis

NOTE This technique/measure is referenced in Tables A.5 and A.8 of IEC 61508-3.

Aim: To document all data, decisions and rationale in the software project to allow for easier verification, validation, assessment and maintenance.

Description: Detailed documentation is maintained during a project, which could include

– testing performed on each software module;

– decisions and their rationale;

– problems and their solutions.

During and at the conclusion of the project this documentation can be analysed to establish a wide variety of information. In particular, data recording is very important for the maintenance of computer systems as the rationale for certain decisions made during the development project is not always known by the maintenance engineers.

Reference:

Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989, ISBN ISBN 1851663819, 9781851663811

C.5.3 Interface testing

NOTE This technique/measure is referenced in Table A.5 of IEC 61508-3.

Aim: To detect errors in the interfaces of subprograms.

Description: Several levels of detail or completeness of testing are feasible. The most important levels are tests for

– all interface variables at their extreme values;

- all interface variables individually at their extreme values with other interface variables at normal values;
- all values of the domain of each interface variable with other interface variables at normal values;
- all values of all variables in combination (this will only be feasible for small interfaces);
- the specified test conditions relevant to each call of each subroutine.

These tests are particularly important if the interfaces do not contain assertions that detect incorrect parameter values. They are also important after new configurations of pre-existing subprograms have been generated.

C.5.4 Boundary value analysis

NOTE This technique/measure is referenced in Tables B.2, B.3 and B.8 of IEC 61508-3.

Aim: To detect software errors occurring at parameter limits or boundaries.

Description: The input domain of the program is divided into a number of input classes according to the equivalence relation (see C.5.7). The tests should cover the boundaries and extremes of the classes. The tests check that the boundaries in the input domain of the specification coincide with those in the program. The use of the value zero, in a direct as well as in an indirect translation, is often error-prone and demands special attention:

- zero divisor;
- blank ASCII characters;
- empty stack or list element;
- full matrix;
- zero table entry.

Normally the boundaries for input have a direct correspondence to the boundaries for the output range. Test cases should be written to force the output to its limited values. Consider also if it is possible to specify a test case which causes the output to exceed the specification boundary values.

If the output is a sequence of data, for example a printed table, special attention should be paid to the first and the last elements and to lists containing none, one and two elements.

References:

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

C.5.5 Error guessing

NOTE This technique/measure is referenced in Tables B.2 and B.8 of IEC 61508-3.

Aim: To remove common programming mistakes.

Description: Testing experience and intuition combined with knowledge and curiosity about the system under test may add some uncategorised test cases to the designed test case set.

Special values or combinations of values may be error-prone. Some interesting test cases may be derived from inspection checklists. It may also be considered whether the system is robust enough. For example: can the buttons be pushed on the front-panel too fast or too often? What happens if two buttons are pushed simultaneously?

Reference:

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

C.5.6 Error seeding

NOTE This technique/measure is referenced in Table B.2 of IEC 61508-3.

Aim: To ascertain whether a set of test cases is adequate.

Description: Some known types of mistake are inserted (seeded) into the program, and the program is executed with the test cases under test conditions. If only some of the seeded errors are found, the test case set is not adequate. The ratio of found seeded errors to the total number of seeded errors is an estimate of the ratio of found real errors to total number errors. This gives a possibility of estimating the number of remaining errors and thereby the remaining test effort.

$$\frac{\text{Found seeded errors}}{\text{Total number of seeded errors}} = \frac{\text{Found real errors}}{\text{Total number of real errors}}$$

The detection of all the seeded errors may indicate either that the test case set is adequate, or that the seeded errors were too easy to find. The limitations of the method are that in order to obtain any usable results, the types of mistake as well as the seeding positions must reflect the statistical distribution of real errors.

References:

Software Fault Injection: Inoculating Programs Against Errors. J. Voas, G. McGraw. Wiley Computer Pub., 1998, ISBN 0471183814, 9780471183815

Faults, Injection Methods, and Fault Attacks. Chong Hee Kim, Jean-Jacques Quisquater, IEEE Design and Test of Computers, vol. 24, no. 6, pp. 544-545, Nov., 2007

Fault seeding for software reliability model validation. A. Pasquini, E. De Agostino. Control Engineering Practice, Volume 3, Issue 7, July 1995. Elsevier Science Ltd

C.5.7 Equivalence classes and input partition testing

NOTE This technique/measure is referenced in Tables B.2 and B.3 of IEC 61508-3.

Aim: To test the software adequately using a minimum of test data. The test data is obtained by selecting the partitions of the input domain required to exercise the software.

Description: This testing strategy is based on the equivalence relation of the inputs, which determines a partition of the input domain.

Test cases are selected with the aim of covering all the partitions previously specified. At least one test case is taken from each equivalence class.

There are two basic possibilities for input partitioning which are

– equivalence classes derived from the specification – the interpretation of the specification may be either input orientated, for example the values selected are treated in the same way, or output orientated, for example the set of values lead to the same functional result;

– equivalence classes derived from the internal structure of the program – the equivalence class results are determined from static analysis of the program, for example the set of values leading to the same path being executed.

References:

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

Static Analysis and Software Assurance. D. Wagner, Lecture Notes in Computer Science, Volume 2126/2001, Springer, 2001, ISBN 978-3-540-42314-0

C.5.8 Structure-based testing

NOTE This technique/measure is referenced in Table B.2 of IEC 61508-3.

Aim: To apply tests which exercise certain subsets of the program structure.

Description: Based on analysis of the program, a set of input data is chosen so that a large (and often prespecified target) percentage of the program code is exercised. Measures of code coverage will vary as follows, depending upon the level of rigour required. In all cases, 100 % of the selected coverage metric should be the aim; if it is not possible to achieve 100 % coverage, the reasons why 100 % cannot be achieved should be documented in the test report (for example, defensive code which can only be entered if a hardware problem arises). The first four techniques in the following list are mentioned specifically in the recommendations in Table B.3 of IEC 61508-3 and are widely supported by testing tools; the remaining techniques could also be considered.

– **Entry point (call graph) coverage**: ensure that every subprogram (subroutine or function) has been called at least once (this is the least rigorous structural coverage measurement).

  NOTE In object-oriented languages, there can be several subprograms of the same name which apply to different variants of a polymorphic type (overriding subprograms) which can be invoked by dynamic dispatching. In these cases every such overriding subprogram should be tested.

– **Statements**: ensure that all statements in the code have been executed at least once.

– **Branches**: both sides of every branch should be checked. This may be impractical for some types of defensive code.

– **Compound conditions**: every condition in a compound conditional branch (i.e. linked by AND/OR) is exercised. See MCDC (modified condition decision coverage, ref. DO178B).

– **LCSAJ**: a linear code sequence and jump is any linear sequence of code statements, including conditional statements, terminated by a jump. Many potential subpaths will be infeasible due to constraints on the input data imposed by the execution of earlier code.

– **Data flow**: the execution paths are selected on the basis of data usage; for example, a path where the same variable is both written and read.

– **Basis path**: one of a minimal set of finite paths from start to finish, such that all arcs are included. (Overlapping combinations of paths in this basis set can form any path through that part of the program.) Tests of all basis path has been shown to be efficient for locating errors.

References:

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

RTCA, Inc. document DO-178B and EUROCAE document ED-12B, *Software Considerations in Airborne Systems and Equipment Certification*, dated December 1, 1992

C.5.9 Control flow analysis

NOTE This technique/measure is referenced in Table B.8 of IEC 61508-3.

Aim: To detect poor and potentially incorrect program structures.

Description: Control flow analysis is a static testing technique for finding suspect areas of code that do not follow good programming practice. The program is analysed producing a directed graph which can be further analysed for

– inaccessible code, for instance unconditional jumps which leaves blocks of code unreachable;

– knotted code. Well-structured code has a control graph which is reducible by successive graph reductions to a single node. In contrast, poorly structured code can only be reduced to a knot composed of several nodes.

References:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

C.5.10 Data flow analysis

NOTE This technique/measure is referenced in Table B.8 of IEC 61508-3.

Aim: To detect poor and potentially incorrect program structures.

Description: Data flow analysis is a static testing technique that combines the information obtained from the control flow analysis with information about which variables are read or written in different portions of code. The analysis can check for

– variables that may be read before they are assigned a value – this can be avoided by always assigning a value when declaring a new variable;

– variables that are written more than once without being read – this could indicate omitted code;

– variables that are written but never read – this could indicate redundant code.

A data flow anomaly will not always directly correspond to a program fault, but if anomalies are avoided the code is less likely to contain faults.

References:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

C.5.11 Symbolic execution

NOTE This technique/measure is referenced in Table B.8 of IEC 61508-3.

Aim: To show the agreement between the source code and the specification.

Description: The program variables are evaluated after substituting the left-hand side by the right- hand side in all assignments. Conditional branches and loops are translated into Boolean expressions. The final result is a symbolic expression for each program variable. This expression is a formula for the value that the program would calculate if given real data. This can be checked against the expected expression.

A related use of symbolic execution is as a systematic way of generating test data for the paths through the program logic. The symbolic execution facility may be incorporated into an integrated toolset to provide a facility for software element test and code analysis.

References:

Using symbolic execution for verifying safety-critical systems. A. Coen-Porisini, G. Denaro, C. Ghezzi, M. Pezzé. Proceedings of the 8th European software engineering conference, and 9th ACM SIGSOFT international symposium on Foundations of software engineering. ACM, 2001, ISBN:1-58113-390-1

Using symbolic execution to guide test generation. G. Lee, J. Morris, K. Parker, G. Bundell, P. Lam. In Software Testing, Verification and Reliability, vol 15, no 1, 2005. John Wiley & Sons, Ltd

C.5.12 Formal proof (verification)

NOTE This technique/measure is referenced in Tables A.5 and A.9 of IEC 61508-3.

Aim: To prove the correctness of a program with respect to some abstract model of the program, using theoretical and mathematical models and rules.

Description: Testing is a common way to examine the correctness of a program. However, exhaustive testing is generally unachievable given the complexity of programs of practical value, and therefore only a fraction of the possible program behaviour can be examined in this way. In contrast, formal verification applies mathematical operations to a mathematical representation of a program in order to establish that the program behaves as defined for all possible inputs.

Formal verification of a system requires an abstract model of the program and of its required behaviour (i.e. a specification) in a language with a precise mathematical meaning. The specification may be complete, or it may be restricted to specific program properties:

- functional correctness properties, i.e. the program should exhibit a particular functionality.
- safety (i.e. some bad behaviour will never occur) and liveness (i.e. some good behaviour will occur eventually) properties.
- timing properties, i.e. some behaviour will occur at a particular time.

The outcome of formal verification is a rigorous argument that the abstract model of the program is correct with respect to the specification for all possible inputs i.e. the model satisfies the specified properties.

However, the correctness of the model does not directly prove the correctness of the actual program, and a further necessary step is to show that the model is an accurate abstraction of the actual program for the properties of interest. Some program properties of practical interest cannot be formalised mathematically (e.g. most timing and scheduling, or subjective properties such as a “clear and simple” user interface, or indeed any property or design

objective that cannot readily be expressed in a formal language). Formal verification therefore does not completely replace simulation and testing, but instead complements these techniques by providing further evidence to support the program's correct operation for all inputs. While formal verification can ensure the correctness of an abstract model of a program, testing ensures that the actual program behaves as expected.

The use of formal verification at the design phase may significantly reduce development time by discovering significant errors and oversights early in the design phase, and thus reducing the time required iterating between design and testing.

Several formal methods in practical use are described in C.2.4: for instance, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z.

C.5.12.1 Model checking

Model checking is a method for the formal verification of reactive and concurrent systems. Given a finite state structure which describes the behaviors of the system, a property written as a temporal logic formula is checked if it holds or not against the structure. Efficient algorithms (e.g. SPIN, SMV, and UPPAAL) are employed to traverse the whole states of the structure automatically and exhaustively. When the property does not hold, a counterexample is generated. It shows how the property is violated in the structure, and contains very useful information to investigate the system. Model checking can detect "deep bugs" that could escape from the traditional inspection and testing.

Note that model checking is helpful in analysing subtle complexity. This may be useful in some low-SIL applications but caution is needed if subtle complexity exists in high-SIL applications.

References:

Is Proof More Cost-Effective Than Testing?. S. King, R. Chapman, J. Hammond, A. Pryor. IEEE Transactions on Software Engineering, vol. 26 no. 8, August 2000

Model Checking. E. M. Clarke, O. Grumberg, and D. A. Peled. MIT Press, 1999, ISBN 0262032708, 9780262032704

Systems and Software Verification: Model-Checking Techniques and Tools. B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, and P. Mckenzie, Springer, 2001, ISBN 3-540-41523-8

Logic in Computer Science: Modelling and Reasoning about Systems. M.Huth and M. Ryan. Cambridge University Press, 2000, ISBN 0521652006, 0521656028

The Spin Model Checker: Primer and Reference Manual. G. J. Holzmann. Addison-Wesley, 2003, ISBN 0321228626, 9780321228628

C.5.12.2 (void)

C.5.13 Complexity metrics

NOTE This technique/measure is referenced in Tables B.9 and C.19 of IEC 61508-3.

Aim: To predict the attributes of programs from properties of the software itself or from its development or test history.

Description: These models evaluate some structural properties of the software and relate this to a desired attribute such as reliability or complexity. Software tools are required to evaluate most of the measures. Some of the metrics which can be applied are given below:

- graph theoretic complexity – this measure can be applied early in the lifecycle to assess trade-offs, and is based on the complexity of the program control graph, represented by its cyclomatic number;
- number of ways to activate a certain software module (accessibility) – the more a software module can be accessed, the more likely it is to be debugged;
- Halstead type metrics science – this measure computes the program length by counting the number of operators and operands; it provides a measure of complexity and size that forms a baseline for comparison when estimating future development resources;
- number of entries and exits per software module – minimising the number of entry/exit points is a key feature of structured design and programming techniques.

Reference:

Metrics and Models in Software Quality Engineering. S.H. Kan. Addison-Wesley, 2003, ISBN 0201729156, 9780201729153

C.5.14 Formal inspections

NOTE This technique/measure is referenced in Table B.8 of IEC 61508-3.

Aim: To reveal defects in a software element.

Description: Formal inspection is a structured process to inspect software material that is carried out by peers of the person producing the material to find defects and to enable the producer to improve the material. The producer should take no part in the inspection process, other than to brief the inspectors during the familiarization stage. Formal inspections may be carried out on specific software elements produced at any phase of the software development life-cycle.

Prior to the inspection taking place the inspectors should become familiar with the materials to be inspected. The inspectors' roles in the inspection process should be clearly defined. An inspection agenda should be prepared. Entry and exit criteria should be defined based on the properties required for the software element. Entry criteria are the criteria or requirements which must be met prior to the inspection taking place. Exit criteria are the criteria or requirements which must be met to complete a specific process.

During the inspection the findings of the inspection should be formally recorded by the moderator, whose role is to facilitate the inspection. A consensus on the findings should be reached by all inspectors. Defects should be classified as either a) requiring rectification prior to acceptance or b) requiring rectification by a given time / milestone. Defects identified should be referred to the producer for subsequent rectification after completion of the inspection. Dependent on the number and scope of identified defects, the moderator may determine it to be necessary for a further inspection of the software material.

References:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

Fagan, M. *Design and Code Inspections to Reduce Errors in Program Development*. IBM Systems Journal 15, 3 (1976): 182-211

C.5.15 Walk-through (software)

NOTE This technique/measure is referenced in Table B.8 of IEC 61508-3.

Aim: To reveal discrepancies between the specification and implementation.

Description: Walk-through is an informal technique, carried out by the producer of a software element in the presence of his peers with the objective of finding defects in the software element. They may be carried out on specific software elements produced at any phase of the software development life-cycle.

Specified functions of the safety-related system are examined and evaluated to ensure that the safety-related system conforms to the requirements given in the specification. Any points of doubt concerning the implementation and use of the product are documented so they may be resolved. In contrast with a formal inspection, the author is active during the walkthrough procedure.

References:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

C.5.16 Design review

NOTE This technique/measure is referenced in Table B.8 of IEC 61508-3.

Aim: To reveal defects in the design of the software.

Description: A design review is a formal, documented, comprehensive and systematic examination of the software design to evaluate the design requirements and the capability of the design to meet these requirements and identify problems and propose solutions.

Design Reviews provide the means to assess the status of the design against the input requirements, and the means to identify opportunities for further improvement. As the development life-cycle activities progress, and major detailed design milestones are met, Design Reviews should be held to review all interface aspects, ensure that the design can be verified to ensure that the design meets its requirements, and ensure that the most appropriate design is consistent with the safety requirements. Such a review is primarily intended to verify the work of the designers and should be treated as a confirmation and refining activity.

A rigorous inspection technique such as “sneak circuit analysis” may be used to detect incorrect software behaviour such as an unexpected path or logic flow, unintended outputs, incorrect timing, undesired actions.

References:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

IEC 61160:2005, *Design review*

Space Product Assurance, Sneak analysis - Part 2: Clue list. ECSS-Q-40-04A Part 2. ESA Publications Division, Noordwijk, 1997, ISSN 1028-396X, http://www.everyspec.com/ESA/ECSS-Q-40-04A_Part-2_14981/

C.5.17 Prototyping/animation

NOTE This technique/measure is referenced in Tables B.3 and B.5 of IEC 61508-3.

Aim: To check the feasibility of implementing the system against the given constraints. To communicate the specifier's interpretation of the system to the customer, in order to locate misunderstandings.

Description: A subset of system functions, constraints, and performance requirements are selected. A prototype is built using high-level tools. At this stage, constraints such as the target computer, implementation language, program size, maintainability, reliability and availability need not be considered. The prototype is evaluated against the customer's criteria and the system requirements may be modified in the light of this evaluation.

Reference:

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

C.5.18 Process simulation

NOTE This technique/measure is referenced in Tables A.7, C.7, B.3 and C.13 of IEC 61508-3.

Aim: To test the function of a software system, together with its interface to the outside world, without allowing it to modify the real world in any way.

Description: The creation of a system, for testing purposes only, which mimics the behaviour of the equipment under control (EUC).

The simulation may be software only or a combination of software and hardware. It must

- provide inputs, equivalent to the inputs which will exist when the EUC is actually installed;
- respond to outputs from the software being tested in a way which faithfully represents the controlled plant;
- have provision for operator inputs to provide any perturbations with which the system under test is required to cope.

When software is being tested the simulation may be a simulation of the target hardware with its inputs and outputs.

References:

EmStar: An Environment for Developing Wireless Embedded Systems Software. J Elson et al. http://cens.ucla.edu/TechReports/9_emstar.pdf

A hardware-software co-simulator for embedded system design and debugging. A. Ghosh et al. In Proceedings of the IFIP International Conference on Computer Hardware Description Languages and Their Applications, IFIP International Conference on Very Large Scale Integration, 1995. IEEE, 1995, ISBN 4930813670, 9784930813671

C.5.19 Performance requirements

NOTE This technique/measure is referenced in Table B.6 of IEC 61508-3.

Aim: To establish demonstrable performance requirements of a software system.

Description: An analysis is performed of both the system and the software requirements specifications to specify all general and specific, explicit and implicit performance requirements.

Each performance requirement is examined in turn to determine

- the success criteria to be obtained;
- whether a measure against the success criteria can be obtained;
- the potential accuracy of such measurements;
- the project stages at which the measurements can be estimated; and
- the project stages at which the measurements can be made.

The practicability of each performance requirement is then analysed in order to obtain a list of performance requirements, success criteria and potential measurements. The main objectives are:

- each performance requirement is associated with at least one measurement;
- where possible, accurate and efficient measurements are selected which can be used as early in the development as possible;
- essential and optional performance requirements and success criteria are specified; and
- where possible, advantage should be taken of the possibility of using a single measurement for more than one performance requirement.

Reference:

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

C.5.20 Performance modelling

NOTE This technique/measure is referenced in Tables B.2 and B.5 of IEC 61508-3.

Aim: To ensure that the working capacity of the system is sufficient to meet the specified requirements.

Description: The requirements specification includes throughput and response requirements for specific functions, perhaps combined with constraints on the use of total system resources. The proposed system design is compared against the stated requirements by

- producing a model of the system processes, and their interactions;
- determining the use of resources by each process, for example, processor time, communications bandwidth, storage devices, etc;
- determining the distribution of demands placed upon the system under average and worst-case conditions;
- computing the mean and worst-case throughput and response times for the individual system functions.

For simple systems an analytic solution may be sufficient, while for more complex systems some form of simulation may be more appropriate to obtain accurate results.

Before detailed modelling, a simpler "resource budget" check can be used which sums the resources requirements of all the processes. If the requirements exceed designed system capacity, the design is infeasible. Even if the design passes this check, performance modelling may show that excessive delays and response times occur due to resource starvation. To avoid this situation, engineers often design systems to use some fraction (for example 50 %) of the total resources so that the probability of resource starvation is reduced.

Reference:

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

C.5.21 Avalanche/stress testing

NOTE This technique/measure is referenced in Table B.6 of IEC 61508-3.

Aim: To burden the test object with an exceptionally high workload in order to show that the test object would stand normal workloads easily.

Description: There are a variety of test conditions which can be applied for avalanche/stress testing. Some of these test conditions are:

– if working in a polling mode then the test object gets much more input changes per time unit as under normal conditions;

– if working on demands then the number of demands per time unit to the test object is increased beyond normal conditions;

– if the size of a database plays an important role then it is increased beyond normal conditions;

– influential devices are tuned to their maximum speed or lowest speed respectively;

– for the extreme cases, all influential factors, as far as is possible, are put to the boundary conditions at the same time.

Under these test conditions, the time behaviour of the test object can be evaluated. The influence of load changes can be observed. The correct dimension of internal buffers or dynamic variables, stacks, etc. can be checked.

Reference:

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202, 9780201596205

C.5.22 Response timing and memory constraints

NOTE This technique/measure is referenced in Table B.6 of IEC 61508-3.

Aim: To ensure that the system will meet its temporal and memory requirements.

Description: The requirements specification for the system and the software includes memory and response requirements for specific functions, perhaps combined with constraints on the use of total system resources.

An analysis is performed to determine the distribution demands under average and worst-case conditions. This analysis requires estimates of the resource usage and elapsed time of each system function. These estimates can be obtained in several ways, for example comparison with an existing system or the prototyping and benchmarking of time critical systems.

C.5.23 Impact analysis

NOTE This technique/measure is referenced in Table A.8 of IEC 61508-3.

Aim: To determine the effect that a change or an enhancement to a software system will have to other software modules in that software system as well as to other systems.

Description: Prior to a modification or enhancement being performed on the software, an analysis should be undertaken to determine the impact of the modification or enhancement on the software, and to also determine which software systems and software modules are affected.

After the analysis has been completed a decision is required concerning the reverification of the software system. This depends on the number of software modules affected, the criticality of the affected software modules and the nature of the change. Possible decisions are:

– only the changed software module is reverified;

– all affected software modules are reverified; or

– the complete system is reverified.

Reference:

Requirements Engineering. E. Hull, K. Jackson, J. Dick. Springer, 2005, ISBN 1852338792, 9781852338794

C.5.24 Software configuration management

NOTE This technique/measure is referenced in Table A.8 of IEC 61508-3.

Aim: Software configuration management aims to ensure the consistency of groups of development deliverables as those deliverables change. Configuration management in general applies to both hardware and software development.

Description: Software configuration management is a technique used throughout development (see IEC 61508-3, 6.2.3). In essence, it requires documenting the production of every version of every significant deliverable and of every relationship between different versions of the different deliverables. The resulting documentation allows the developer to determine the effect on other deliverables of a change to one deliverable (especially one of its elements). In particular, systems or subsystems can be reliably re-built from consistent sets of element versions.

References:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

Software Configuration Management: Coordination for Team Productivity. W.A. Babich. Addison-Wesley, 1986, ISBN 0201101610, 9780201101614

CMMI: guidelines for process integration and product improvement, Mary Beth Chrissis, Mike Konrad, Sandy Shrum, Addison-Wesley, 2003, ISBN 0321154967, 9780321154965

C.5.25 Regression validation

NOTE This technique/measure is referenced in Table A.8 of IEC 61508-3.

Aim: To ensure that valid conclusions are drawn from regression testing.

Description: Complete regression testing of a large or complex system will usually require much effort and resource. Where possible, it is desirable to restrict the regression testing to cover only the system aspects of direct interest at that point in the system development. In this partial regression testing it is essential to have a clear understanding of the scope of the partial testing and to draw only valid conclusions regarding the tested state of the system.

Reference:

Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing. R.Black, John Wiley and Sons, 2002, ISBN 0471223980, 9780471223986

C.5.26 Animation of specification and design

NOTE This technique/measure is referenced in Table A.9 of IEC 61508-3.

Aim: To guide the software verification by means of a systematic examination of the specification

Description: A representation of the software that is more abstract than the executable code (i.e. a specification or a high level design) is examined to determine the behaviour of the eventual executable software. The examination is automated in some way (depending on the possibilities afforded by the nature and level of abstraction of the higher level representation) so as to simulate the behaviour and outputs of the executable software. One application of this approach is to generate tests (or “oracles”) that can be later applied to the executable software, thus automating to some degree the testing process. Another application is to animate a user interface so that non-technical end-users can gain some appreciation of the detailed meaning of the specification to which the software developers will work. This provides a valuable method of communication between the two groups.

References:

Supporting the Software Testing Process through Specification Animation. T.Miller, P.Strooper. In Proceedings of the First International Conference on Software Engineering and Formal Methods (SEFM'03), ed. P.Lindsay. IEEE Computer Society, IEEE Computer Society, 2003, ISBN 0769519490, 9780769519494

B model animation for external verification. H.Waeselynck, S.Behnia, In Proceedings. of the Second International Conference on Formal Engineering Methods, 1998. IEEE Computer Society, 1998, ISBN 0-8186-9198-0

C.5.27 Model based testing (test case generation)

NOTE This technique/measure is referenced in Table A.5 of IEC 61508-3.

Aim: To facilitate efficient automatic test case generation from system models and to generate highly repeatable test suites.

Description: Model-based Testing (MBT) is a black-box approach in which common testing tasks such as test case generation (TCG) and test results evaluation are based on a model of the system (application) under test (SUT). Typically, but not only, the systems data and user behaviour are modelled using Finite state machines, Markov processes, decision tables or the like (El-Far, 2001, generalized). Additionally, model-based testing can be combined with source code level test coverage measurement, and functional models can be based on existing source code.

Model-based testing is the automatic generation of efficient test cases/procedures using models of system requirements and specified functionality (SoftwareTech, 2009).

Since testing is very expensive, there is a huge demand for automatic test case generation tools. Therefore, model-based testing is currently a very active field of research, resulting in a

large number of available Test Case Generation (TCG) tools. These tools typically extract a test suite from the behavioural part of the model, guaranteeing to meet certain coverage requirements.

The model is an abstract, partial representation of the desired behaviour of the system under test (SUT). From this model, test models are derived, building an abstract test suite. Test cases are derived from this abstract test suite and executed against the system, and tests can be run against the system model as well. MBT with TCG is based on and strongly related to use of formal methods, so recommendations are similar with respect to safety integrity levels (SIL): HR (highly recommended) for higher SILs, and not required for lower SILs.

The specific activities in general are:

– build the model (from system requirements)
– generate expected inputs
– generate expected outputs
– run tests
– compare actual outputs with expected outputs
– decide on further action (modify model, generate more tests, estimate reliability/quality of the software)

Tests can be derived with different methods and techniques for expressing models of user/system behaviour, e.g.

– by using decision tables
– by using finite state machines
– by using grammars
– by using Markov Chain models
– by using state charts
– by theorem proving
– by constraint logic programming
– by model checking
– by symbolic execution
– by using an event-flow model
– reactive system tests: parallel hierarchical finite automaton
– etc.

Model-based testing is specifically targeting recently the safety critical domain. It allows for early exposure of ambiguities in specification and design, provides the capability to automatically generate many non-repetitive efficient tests, to evaluate regression test suites and to assess software reliability and quality, and eases updating of test suites.

A thorough overview is provided by ElFar (2001) and SoftwareTech 2009 (see references), other details and domain specific issues are discussed in the other references.

References:

T. Bauer, F. Böhr, D. Landmann, T. Beletski, R. Eschbach, Robert and J.H. Poore, *From Requirements to Statistical Testing of Embedded Systems* Software Engineering for Automotive Systems - SEAS 2007, ICSE Workshops, Minneapolis, USA

Eckard Bringmann, Andreas Krämer; *Model-based Testing of Automotive Systems* In: ICST, pp.485-493, 2008 International Conference on Software Testing, Verification, and Validation, 2008

Broy M., *Challenges in automotive software engineering*, International conference on Software engineering (ICSE '06),Shanghai, China, 2006

I. K. El-Far and J. A. Whittaker, *Model-Based Software Testing*. Encyclopedia of Software Engineering (edited by J. J. Marciniak). Wiley, 2001

Heimdahl, M.P.E.: *Model-based testing: challenges ahead*, Computer Software and Applications Conference (COMPSAC 2005), 25-28 July 2005, Edinburgh, Scotland, UK, 2005

Jonathan Jacky, Margus Veanes, Colin Campbell, and Wolfram Schulte, *Model-Based Software Testing and Analysis with C#*, ISBN 978-0-521-68761-4, Cambridge University Press 2008

A. Paradkar, *Case Studies on Fault Detection Effectiveness of Model-based Test Generation Techniques*, in ACM SIGSOFT SW Engineering Notes, Proc. of the first int. workshop on Advances in model-based testing A-MOST '05, Vol. 30 Issue 4. ACM Press 2005

S. J. Prowell, *Using Markov Chain Usage Models to Test Complex Systems*, HICSS '05: 38th Annual Hawaii, International Conference on System Sciences, 2005

Mark Utting and Bruno Legeard, *Practical Model-Based Testing: A Tools Approach*, ISBN 978-0-12-372501-1, Morgan-Kaufmann 2007

Hong Zhu et al. (2008). AST '08: *Proceedings of the 3rd International Workshop on Automation of Software Test*. ACM Press. ISBN 978-1-60558-030-2

Model-Based Testing of Reactive Systems Advanced Lecture Series, LNCS 3472, Springer-Verlag, 2005, ISBN 978-3-540-26278-7

Model-based Testing, SoftwareTech July 2009, Vol. 12, No. 2, Software Testing: A Life Cycle Perspective, http://www.goldpractices.com/practices/mbt/

C.6 Functional safety assessment

NOTE Relevant techniques and measures may also be found in B.6.

C.6.1 Decision tables (truth tables)

NOTE This technique/measure is referenced in Tables A.10 and B.7 of IEC 61508-3.

Aim: To provide a clear and coherent specification and analysis of complex logical combinations and relationships.

Description: This method uses two dimensional tables to concisely describe logical relationships between Boolean program variables.

The conciseness and tabular nature of the method makes it appropriate as a means of analysing complex logical combinations expressed in code.

The method is potentially executable if used as a specification.

C.6.2 Software Hazard and Operability Study (CHAZOP, FMEA)

Aim: To determine safety hazards in a proposed or existing system, their possible causes and consequences, and recommend action to minimise the chance of their occurrence.

Description: A team of engineers, with expertise covering the whole system under consideration, participate in a structured examination of a design, through a series of scheduled meetings. They consider both the functional aspects of the design and how the system would operate in practice (including human activity and maintenance). A leader encourages team members to be creative in exposing potential hazards, and drives the procedure by presenting each part of the system in connection with several guide words: "none", "more of", "less of", "part of", "more than" (or "as well as") and "other than". Every applied condition or failure mode is considered for its feasibility, how it could arise, the possible consequences (is there a hazard?), how it could be avoided and if the avoidance technique is worth the expense.

Hazard studies may take place at many stages of project development, but are most effective when performed early enough to influence major design and operability decisions.

The HAZOP technique evolved in the process industry and requires modification for software application. Different derivative methods (or Computer HAZOPs – "CHAZOPs") have been proposed which in general introduce new guide words and/or suggest schemes for systematically covering the system and software architecture.

References:

OF-FMEA: an approach to safety analysis of object-oriented software intensive systems, T. Cichocki, J. Gorski. In Artificial Intelligence and Security in Computing Systems: 9th International Conference, ACS '2002. Ed. J. Soldek. Springer, 2003, ISBN 1402073968, 9781402073960

Software FMEA techniques. P.L.Goddard. In Proc Annual 2000 Reliability and Maintainability Symposium, IEEE, 2000, ISBN: 0-7803-5848-1

Software criticality analysis of COTS/SOUP. P.Bishop, T.Clement, S.Guerra. In Reliability Engineering & System Safety, Volume 81, Issue 3, September 2003, Elsevier Ltd., 2003

C.6.3 Common cause failure analysis

NOTE 1 This technique/measure is referenced in Table A.10 of IEC 61508-3.

NOTE 2 See also Annex D of IEC 61508-6.

Aim: To determine potential failures in multiple systems or multiple subsystems which would undermine the benefits of redundancy, because of the appearance of the same failures in the multiple parts at the same time.

Description: Systems intended to take care of the safety of a plant often use redundancy in hardware and majority voting. This is to avoid random hardware failures in components or subsystems which would tend to prevent the correct processing of data.

However, some failures can be common to more than one component or subsystem. For example, if a system is installed in one single room, shortcomings in the air-conditioning, might reduce the benefits of redundancy. The same is true for other external effects on the system such as fire, flooding, electromagnetic interference, plane crashes, and earthquakes. The system may also be affected by incidents related to operation and maintenance. It is essential, therefore, that adequate and well- documented procedures are provided for operation and maintenance, and operating and maintenance personnel are extensively trained.

Internal effects are also major contributors to common cause failures. They can stem from design faults in common or identical components and their interfaces, as well as ageing of components. Common cause failure analysis has to search the system for such potential common failures. Methods of common cause failure analysis are: general quality control;

design reviews; verification and testing by an independent team; and analysis of real incidents with feedback of experience from similar systems. The scope of the analysis, however, goes beyond hardware. Even if software diversity is used in different channels of a redundant system, there might be some commonality in the software approaches which could give rise to common cause failure, for example, faults in the common specification.

When common cause failures do not occur exactly at the same time, precautions can be taken by means of comparison methods between the multiple channels which should lead to detection of a failure before this failure is common to all channels. Common cause failure analysis should take this technique into account.

References:

Reliability analysis of hierarchical computer-based systems subject to common-cause failures. L.Xing, L.Meshkat, S.Donohue. Reliability Engineering & System Safety Volume 92, Issue 3, March 2007

C.6.4 Reliability block diagrams

NOTE 1 This technique/measure is referenced in Table A.10 of IEC 61508-3 and is used in Annex B of IEC 61508-6.

NOTE 2 See also B.6.6.7 “Reliability block diagrams”.

Aim: To model, in a diagrammatic form, the set of events that must take place and conditions which must be fulfilled for a successful operation of a system or a task.

Description: The target of the analysis is represented as a success path consisting of blocks, lines and logical junctions. A success path starts from one side of the diagram and continues via the blocks and junctions to the other side of the diagram. A block represents a condition or an event, and the path can pass it if the condition is true or the event has taken place. If the path comes to a junction, it continues if the logic of the junction is fulfilled. If it reaches a vertex, it may continue along all outgoing lines. If there exists at least one success path through the diagram, the target of the analysis is operating correctly.

References:

IEC 61025:2006, *Fault tree analysis (FTA)*

From safety analysis to software requirements. K.M. Hansen, A.P. Ravn, A.P, V Stavridou. IEEE Trans Software Engineering, Volume 24, Issue 7, Jul 1998

IEC 61078:2006, *Analysis techniques for dependability – Reliability block diagram and boolean methods*

Annex D
(informative)

A probabilistic approach to determining software safety integrity for pre-developed software

D.1 General

This annex provides initial guidelines on the use of a probabilistic approach to determining software safety integrity for pre-developed software based on operational experience. This approach is considered particularly appropriate as part of the qualification of operating systems, library modules, compilers and other system software. The annex provides an indication of what is possible, but the techniques should be used only by those who are competent in statistical analysis.

NOTE This annex uses the term confidence level, which is described in IEEE 352. An equivalent term, significance level, is used in IEC 61164.

The techniques could also be used to demonstrate an increase in the safety integrity level of software over time. For example, software built to the requirements of IEC 61508-3 to SIL1 may, after a suitable period of successful operation in a large number of applications, be shown to achieve SIL2.

Table D.1 below shows the number of failure-free demands experienced or hours of failure-free operation needed to qualify for a particular safety integrity level. This table is a summary of the results given in D.2.1 and D.2.3.

Operating experience can be treated mathematically as outlined in D.2 below to supplement or replace statistical testing, and operating experience from several sites may be combined (i.e. by adding the number of treated demands or hours of operation), but only if

– the software version to be used in the E/E/PE safety-related system is identical to the version for which operating experience is being claimed;
– the operational profile of the input space is similar;
– there is an effective system for reporting and documenting failures; and
– the relevant prerequisites (see D.2 below) are satisfied.

Table D.1 – Necessary history for confidence to safety integrity levels

SIL	Low demand mode of operation	Number of treated demands		High demand or continuous mode of operation	Hours of operation in total	
	(Probability of failure to perform its design function on demand)	$1-\alpha = 0,99$	$1-\alpha = 0,95$	(Probability of a dangerous failure per hour)	$1-\alpha = 0,99$	$1-\alpha = 0,95$
4	$\geq 10^{-5}$ to $< 10^{-4}$	$4,6 \times 10^5$	3×10^5	$\geq 10^{-9}$ to $< 10^{-8}$	$4,6 \times 10^9$	3×10^9
3	$\geq 10^{-4}$ to $< 10^{-3}$	$4,6 \times 10^4$	3×10^4	$\geq 10^{-8}$ to $< 10^{-7}$	$4,6 \times 10^8$	3×10^8
2	$\geq 10^{-3}$ to $< 10^{-2}$	$4,6 \times 10^3$	3×10^3	$\geq 10^{-7}$ to $< 10^{-6}$	$4,6 \times 10^7$	3×10^7
1	$\geq 10^{-2}$ to $< 10^{-1}$	$4,6 \times 10^2$	3×10^2	$\geq 10^{-6}$ to $< 10^{-5}$	$4,6 \times 10^6$	3×10^6

NOTE 1 $1-\alpha$ represents the confidence level.

NOTE 2 See D.2.1 and D.2.3 for prerequisites and details of how this table is derived.

D.2 Statistical testing formulae and examples of their use

D.2.1 Simple statistical test for low demand mode of operation

D.2.1.1 Prerequisites

a) Test data distribution equal to distribution for demands during on-line operation.

b) Test runs are statistically independent from each other, with respect to the cause of a failure.

c) An adequate mechanism exists to detect any failures which may occur.

d) Number of test cases $n > 100$.

e) No failure occurs during the n test cases.

D.2.1.2 Results

Failure probability p (per demand), at the confidence level $1-\alpha$, is given by

$$p \leq 1 - \sqrt[n]{\alpha} \qquad \text{or} \qquad n \geq -\frac{\ln \alpha}{p}$$

D.2.1.3 Example

Table D.2 – Probabilities of failure for low demand mode of operation

$1-\alpha$	P
0,95	$3/n$
0,99	$4{,}6/n$

For a probability of failure on demand of SIL 3 at 95 % confidence the application of the formula gives 30 000 test cases under the conditions of the prerequisites. Table D.1 summarises the results for each safety integrity level.

D.2.2 Testing of an input space (domain) for a low demand mode of operation

D.2.2.1 Prerequisites

The only prerequisite is that the test data is selected to give a random uniform distribution over the input space (domain).

D.2.2.2 Results

The objective is to find the number of tests, n, that are necessary based on the threshold of accuracy, δ, of the inputs for the low demand function (such as a safety shut-down) that is being tested.

Table D.3 – Mean distances of two test points

Dimension of the domain	Mean distance of two test points in direction of an arbitrary axis
1	$\delta = 1/n$
2	$\delta = \sqrt[2]{1/n}$
3	$\delta = \sqrt[3]{1/n}$
k	$\delta = \sqrt[k]{1/n}$
NOTE *k* can be any positive integer. The values 1, 2 and 3 are just examples.	

D.2.2.3 Example

Consider a safety shut-down that is dependent on just two variables, A and B. If it has been verified that the thresholds that partition the input pair of variables A and B are treated correctly to an accuracy of 1 % of A or B's measuring range, the number of uniformly distributed test cases required in the space of A and B is

$$n = 1/\delta^2 = 10^4$$

D.2.3 Simple statistical test for high demand or continuous mode of operation

D.2.3.1 Prerequisites

a) Test data distribution equal to distribution during on-line operation.

b) The relative reduction for the probability of no failure is proportional to the length of the considered time interval and constant otherwise.

c) An adequate mechanism exists to detect any failures which may occur.

d) The test extends over a test time *t*.

e) No failure occurs during *t*.

D.2.3.2 Results

The relationship between the probability of failure λ, the confidence level 1-α and the testing time *t* is

$$\lambda = -\frac{\ln \alpha}{t}$$

The probability of failure is indirectly proportional to the mean operating time between failures:

$$\lambda = \frac{1}{\text{MTBF}}$$

NOTE This standard does not distinguish between the probability of failure per hour and the rate of failures in 1 h. Strictly, the probability of failure, *F*, is related to the failure rate, *f*, by the equation $F = 1\text{-}e^{-ft}$, but the scope of this standard is for failure rates of less than 10^{-5}, and for values this small $F \approx ft$.

D.2.3.3 Example

Table D.4 – Probabilities of failure for high demand or continuous mode of operation

$1-\alpha$	λ
0,95	$3/t$
0,99	$4{,}6/t$

To verify that the mean time between failures is at least 10^8 h with a confidence level of 95 %, a test time of 3×10^8 h is required and the prerequisites must be satisfied. Table D.1 summarises the number of tests required for each safety integrity level.

D.2.4 Complete test

The program is considered as an urn containing a known number *N* of balls. Each ball represents a program property of interest. Balls are drawn at random and replaced after inspection. A complete test is achieved if all the balls are drawn.

D.2.4.1 Prerequisites

a) Test data distribution is such that each of the *N* program properties is tested with equal probability.

b) Test runs are independent from each other.

c) Each occurring failure is detected.

d) Number of test cases $n >> N$.

e) No failure occurs during the *n* test cases.

f) Each test run tests one program property (a program property is what can be tested during one run).

D.2.4.2 Results

The probability *p* to test all program properties is given by

$$p = \sum_{j=0}^{N-1} (-1)^j \binom{N}{j} \left(\frac{N-j}{N}\right)^n \quad \text{or} \quad p = 1 + \sum_{j=1}^{N} (-1)^j \, C_{j,N} \left(\frac{N-j}{N}\right)^n$$

where

$$C_{j,N} = \frac{N(N-1)\ldots(N-j+1)}{j!}$$

For evaluation of this formula usually only the first terms matter since realistic cases are characterised by $n >> N$. The last factor makes all terms for large *j* very small. This is also visible in Table D.5.

D.2.4.3 Example

Consider a program that has been used at several installations for several years. In total, at least $7{,}5 \times 10^6$ runs have been executed. It is estimated that each 100th run fulfils the above prerequisites. So $7{,}5 \times 10^4$ runs made can be taken for statistical evaluation. It is estimated that 4 000 test runs would perform an exhaustive test. The estimates are conservative. According to Table D.5, the probability of not having tested everything equals $2{,}87 \times 10^{-5}$.

For N = 4 000, the values of the first terms depending on *n* are:

Table D.5 – Probability of testing all program properties

n	*P*
5×10^4	$1 - 1{,}49 \times 10^{-2} + 1{,}10 \times 10^{-4} - \ldots$
$7{,}5 \times 10^4$	$1 - 2{,}87 \times 10^{-5} + 4 \times 10^{-10} - \ldots$
1×10^5	$1 - 5{,}54 \times 10^{-8} + 1{,}52 \times 10^{-15} - \ldots$
2×10^5	$1 - 7{,}67 \times 10^{-19} + 2{,}9 \times 10^{-37} - \ldots$

In practice, such estimates should be made so that they are conservative.

D.3 References

Further information on the above techniques can be found in the following documents:

IEC 61164:2004, *Reliability growth – Statistical test and estimation methods*

Verification and Validation of Real-Time Software, Chapter 5. W. J. Quirk (ed.). Springer Verlag, 1985, ISBN 3-540-15102-8

Combining Probabilistic and Deterministic Verification Efforts. W. D. Ehrenberger, SAFECOMP 92, Pergamon Press, ISBN 0-08-041893-7

Ingenieurstatistik. Heinhold/Gaede, Oldenburg, 1972, ISBN 3-486-31743-1

IEEE 352:1987, *IEEE Guide for general principles of reliability analysis of nuclear power generating station safety systems*

Annex E
(informative)

Overview of techniques and measures for design of ASICs

NOTE The overview of techniques and measures contained in this annex and referenced by IEC 61508-2. This annex should not be regarded as either complete or exhaustive.

E.1 Design description in (V)HDL

Aim: Functional description at high level in hardware description language, for example VHDL or Verilog.

Description: Functional description at high abstraction level in hardware description language, for example VHDL or Verilog. The applied hardware description language should allow functional and/or application oriented description and should be abstracted from later implementation details. Dataflows, branches, arithmetical and/or logical operations should be implemented by assignment and operators of the hardware description language, without manual conversion in logical gates of the applied library.

NOTE For simplification “functional description at high abstraction level in hardware description language” will be denoted in the rest of the document as (V)HDL.

Reference:

IEEE VHDL, *Verilog + Standard VHDL Design guide*

E.2 Schematic entry

Aim: Functional description of the circuitry by drawing a circuit plan using gates and/or macros of the vendor library.

Description: Description of the circuit functionality by schematic entry of the circuit plan. The function to be realised should be implemented by instancing (import) the elementary logical circuit elements such as AND, OR, NOT along with macros consisting of complex arithmetical and logical functions, which are then interconnected. Complex circuits should be partitioned considering the functional viewpoints and can be distributed on different drawings, which are hierarchically interconnected. The interconnection signals should be uniquely defined and have explicit signal names over the entire hierarchy. The use of global signals (Labels) should be avoided as far as applicable.

E.3 Structured description

NOTE See also C.2.7 "Structured Programming" and E.12 "Modularization".

Aim: The description of the circuit's functionality should be structured in such a fashion that it is easily readable, i.e. circuit function can be intuitively understood on basis of description without simulation efforts.

Description: Description of the circuit functionality with (V)HDL or by schematic entry. An easily recognisable and modular structure is recommended. Each module should be implemented likewise in the same fashion and should be described in such a way that it is easily readable with clear defined sub functions. A strict distinction between implemented function and interconnection is recommended, i.e. the module, which is implemented by instancing other sub modules, contains explicitly interconnections of the instanced modules and should not contain any circuit logic.

E.4 Proven-in-use tools

Aim: Application of proven-in-use tools to avoid systematic failure by sufficient long-approved practice of the tools in various projects.

Description: Most of the used tools for designing ASICs and FPGAs comprise of sophisticated software, which cannot be considered to operate without any faults with respect to its correct functionality and it is also quite likely that faults might occur due to faulty operation. Therefore only tools with the attribute "proven-in-use" should be preferred for designing ASICs and FPGAs. This implies:

– Application of tools which have been used (in a comparable software version) over a long period of time or high number of users in various projects with equivalent complexity.
– Adequate experience of each ASIC/FPGA designer with the operation of the tool over a long period of time.
– Use of commonly used tools (adequate number of users) so that information regarding known failures with work arounds (version control with "Bug-List") is available. This information should be readily integrated in design flow and helps to avoid systematic failures.
– The consistency check of the internal tool database and the plausibility check avoid faulty output data. Standard tools check the consistency of the internal database, for example the consistency of database between synthesis- and place-and-route-tool, in order to operate with unique data.

NOTE The consistency check is an inherent attribute of the tool under use and the designer has limited influence on it. Therefore, if the possibility of manual consistency check is provided, the designer should use it adequately.

E.5 (V)HDL simulation

NOTE See also E.6 "Functional test on module level".

Aim: Functional verification of circuit described in (V)HDL by means of simulation.

Description: Verification of the function by simulating the entire circuit or each sub module. The (V)HDL simulator detects a sequence of outputs caused by the internal change of the circuit states as the result of applied input stimuli. The verification of the detected output sequence can be carried out either by pretraced sequence of output signals ("Wave form") or by a special environment known as test bench, which is installed during the design process. The chosen simulator should have an attribute "proven-in-use" in order to provide correct results and to mask faulty timing behaviour of the signals (Spikes, tri-state tracing), which might be caused by the simulator itself or faulty modelling.

E.6 Functional test on module level

NOTE See also E.5 "(V)HDL Simulation" and E.13 "Coverage of the verification scenarios".

Aim: Functional verification "Bottom-up".

Description: Verification of the implemented function - for example by simulation - at module level. The module under test will be instanced in a typical virtual test environment known as ”test bench” and stimulated by the test pattern contained in the code. A sufficient high coverage of specified function including all special cases if they exist is at least required. Automatic verification of output sequence by the code of "test bench" should be preferred against manual inspection of output signals.

E.7 Functional test on top level

NOTE See also E.8 "Functional test embedded in system environment”.

Aim: Verification of the ASIC (entire circuit).

Description: The objective of the test is the verification of the entire circuit (ASIC).

E.8 Functional test embedded in system environment

NOTE See also E.7 "Functional test on top level".

Aim: Verification of the specified function embedded in system environment.

Description: This test will verify the entire functionality of the circuit (ASIC) in its system environment, for example with all other components that are located on the circuit boards or elsewhere. A modelling of all relevant components on the circuit board and simulation of ASIC together with the created model to verify the correct functionality inclusive of timing behaviour is recommended. A complete functional test includes also testing of modules that are activated only during presence of failure.

E.9 Restricted use of asynchronous constructs

Aim: Avoidance of typical timing problems during synthesis, avoidance of ambiguity during simulation and synthesis caused by insufficient modelling, design for testability.

Description: Asynchronous constructs such as SET and RESET signals derived over combinatorial logic are susceptible during synthesis and produce circuits with spikes or inverse timing sequence and therefore should be avoided. Also insufficient modelling may not be interpreted properly by the synthesis tool, which causes ambiguous results during simulation. Additionally asynchronous constructs are poorly testable or not at all testable, so that the test coverage of production and on-line test is effectively reduced. The implementation of completely synchronous design with limited number of clock signals is therefore recommended. In systems with multiphase clocks, all the clocks should be derived from one central clock. Clock input of sequential logic should be always supplied exclusively by the clock signal, which does not contain any combinatorial logic. Asynchronous SET and RESET inputs of sequential cells should be always supplied by synchronous signals that do not contain any combinatorial logic. Master SET and RESET should be synchronised using two Flip-flops.

E.10 Synchronisation of primary inputs and control of metastabilities

Aim: Avoidance of ambiguous circuit behaviour as a result of set-up and hold timing violation.

Description: Input signals from external peripherals are generally asynchronous and can change their state arbitrarily. A direct processing of such signals by the synchronous sequential circuit elements of ASIC/FPGA, for example flip-flops leads to set-up and hold time violation resulting in unpredictable timing and functional behaviour of the ASIC/FPGA. Ultimately the metastability of the memory element might occur. Each asynchronous input signal should be therefore synchronised with respect to the synchronous ASIC circuit to avoid the functional ambiguity. Following measures are recommended:

– Input signals should be synchronised with two consecutive memory elements (Flip-flops) or some equivalent circuit in order to achieve a predictable functional behaviour.

– Each asynchronous input signal should be fundamentally synchronised in the above defined manner, i.e. each asynchronous signal is connected with exact one such synchronising circuit. If necessary the output of the synchronising circuit can be used for multiple access.

– The synchronising circuit should be used for stability test of parallel bus signals and to control the data consistency near sampling point

E.11 Design for testability

NOTE 1 See also E.31 "Implementation of test structures".

Aim: Avoidance of not testable or poorly testable structures in order to achieve high test coverage for production test or on-line test.

Description: Design for testability is governed by the avoidance of

– asynchronous constructs
– latches and on-chip tri-state signals
– wired-and / wired-or logic and redundant logic.

The combinatorial depth of the sub circuits plays an important role during the testing. The test pattern required for a complete test increases exponentially with the combinatorial depth of the circuit. Therefore, circuits with high combinatorial depth are only poorly testable with adequate means.

A design for testability orientated approach ensures that the desired test coverage is achieved. As the actual test coverage can be determined at a very late stage in the design process, insufficient consideration of “design for testability” issues might dramatically reduce the achievable test coverage, leading to additional effort.

NOTE 2 The test coverage is usually determined by the percentage of stuck-at faults detected.

E.12 Modularisation

NOTE See also C.2.8 "Information hiding/encapsulation", C.2.9 "Modular approach" and E.3 "Structured description".

Aim: Modular description of the circuit functions.

Description: Distinct partitioning of the total functionality in different modules with limited functions. So the transparency of the modules with the precisely defined interface is established. Every subsystem, at all levels of the design, is clearly defined and is of restricted size (only a few functions). The interfaces between subsystems are kept as simple as possible and the cross-section (i.e. shared data, exchange of information) is minimised. The complexity of individual subsystems is also restricted.

E.13 Coverage of the verification scenarios (test benches)

Aim: Quantitative and qualitative assessment of the applied verification scenarios during the functional test.

Description: The quality of the verification scenarios that is defined during the functional test, i.e. the applied test pattern (stimuli) to verify specified function including all special cases, if they exist, should be qualitatively and/or quantitatively documented. During a quantitative approach the achieved test coverage and the depth of the applied functional tests should be documented. The resulting coverage should meet the levels established for each of the coverage metrics. Any exception will be documented. In the case of a qualitative approach, the number of verified code lines, instructions or paths (“Code coverage”) of the circuit code to be verified should be estimated.

NOTE Exclusive ”Code coverage”-analysis has only a limited relevance, because of high parallelism of the hardware description, and will be justified by exhaustive checks. The code coverage” generally serves to demonstrate the not covered functional code.

E.14 Observation of coding guidelines

Aim: Strict observation of the coding style results in a syntactic and semantic correct circuit code.

Description: Syntactic coding rules help to create an easily readable code and allow a better documentation including version control. Typically, the rules for organising and commenting the instruction blocks or modules can be mentioned here.

Semantic coding rules help avoiding typical implementation problems by avoidance of constructs that lead to faulty synthesis with ambiguous implementation of the circuit function. Typical rules are for example the avoidance of asynchronous constructs or constructs that produce unpredictable timing sequence. In general the use of latches or coupling of data with clock signals lead to such ambiguities.

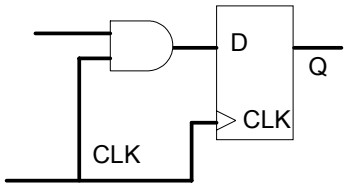
Design guidelines are recommended to avoid systematic design failures during ASIC development process. A coding style in certain aspect limits the design efficiency, offers however in turn the advantage of failure avoidance during ASIC development process. These are in particular:

– avoidance of typical coding infirmity or failure;
– restrictive usage of problematic constructs that produce ambiguous synthesis results;
– design for testability;
– transparent and easy to use code.

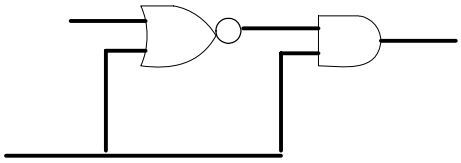
Example of a Coding Style

1. The code should contain as many comments as necessary to understand the function and implementation details. The used conventions have to be defined before the beginning of the design. The compliance of defined conventions should be checked during the design phase.
   1.1. Standard headers include history, cross references to specification, responsibility and design accompanying data such as version number, change requests etc.
   1.2. Easily readable templates: equivalent processes should be described with the same procedure, i.e. usage of predefined templates for recurrent processes (if-then-else, for etc.).
   1.3. Precise and readable naming convention e.g. capital/small letter, pre- and postfix, precise differentiation between port name, internal signals, constants, variables, low active level (xxx_n) etc.
   1.4. Module size restriction and number of ports per module should be limited to increase the readability of code.
   1.5. Structural and defensive code development, e.g. state information should be encapsulated in FSM (Information hiding) to provide the easy alteration of code.
   1.6. Plausibility checks such as range checking etc. should be implemented.
   1.7. Avoidance of following constructs/instructions
      – use of ascending range (x to y) for bus signals;
      – “Disable” instruction in Verilog (corresponds the instruction goto);
      – multidimensional arrays (> 2), records;
      – combination of signed and unsigned data type.
2. Complete synchronous design (clocks derived from central clocks are allowed)
   2.1. Module outputs should be synchronised, it also supports the testability and static timing analysis.
   2.2. Gated clocks should be handled with special precaution.

3. Avoidance of the coupling of data with clock increases the testability, reproducibility between pre- and post-layout data and compliance with RTL (register transfer level) behaviour.
4. Redundant logic is not testable and should be avoided:



Coupling of data and clock signals



Redundant logic

5. Feedback loops in the combinatorial logic should be avoided because they produce unstable design and will not be testable.
6. Full-scan design is recommended.
7. Avoidance of latches increases the testability and reduces the timing constraints during synthesis.
8. Master reset and all asynchronous inputs should be synchronised with two consecutive memory elements (Flip-flops) or an equivalent circuit(s) (metastability).
9. It is recommended to avoid asynchronous set/reset except for master reset.
10. The signals at the module port level should be of the type std_logic or std_logic_vector.

E.15 Application of code checker

Aim: Automatic verification of coding rules ("Coding style") by code checker tool.

Description: The application of code checker helps to a great extent automatically to observe the coding style and generates on-line documentation. However automatic code checker can generally verify the syntax and semantic of the code. The application of such tools should be therefore accompanied by the extension of general coding rules ("tool specific") with the project specific coding rules that the designer has to implement and evaluate separately.

E.16 Defensive programming

See C.2.5.

E.17 Documentation of simulation results

Aim: Documentation of all data needed for a successful simulation in order to verify the specified circuit function.

Description: All the data needed for the functional simulation at module-, chip- or system level should be well documented and archived with the following aims:

– To repeat the simulation at any later phase in turn key fashion.
– To demonstrate the correctness and completeness of all functions specified.

The following database should be archived for this purpose:

– Simulation set-up including complete software of the applied tools, for example simulator, synthesiser with corresponding version and the necessary simulation library.
– Log file of the simulation with full details regarding the time of simulation, applied tools with version and complete report of the work around if it was necessary.
– All relevant simulation results inclusive signal flow, especially in case of manual inspection and documentation of acquired results.

E.18 Code-Inspection

NOTE 1 See also C.5.14 "Formal inspections".

Aim: Review of circuit description.

Description: Review of circuit description should be carried out by

– Checking the coding style.
– Verification of the described functionality against the specification.
– Checking for defensive coding, error and exception handling.

NOTE 2 If the (V)HDL Simulation is not carried out, the completeness of the code inspection and the achieved results should have the equivalent quality that would be achieved by a (V)HDL-simulation.

E.19 Walk-through

NOTE 1 See also C.5.15 "Walk-through".

Aim: Review of the circuit description by walk-through.

Description: A code walk-through consists of a walk-through team selecting a small set of test cases, representative sets of inputs and corresponding expected outputs for the program. The test data is then manually traced through the logic of the program.

NOTE 2 As stand alone measure it should be applied only to the circuits with very low complexity. In the case of the failing (V)HDL-Simulation the completeness of the walk-through and the quality of the achieved results should have the equivalent quality that will be achieved by a (V)HDL-simulation.

Reference:

IEC 61160:2005, *Design review*

E.20 Application of validated soft cores

Aim: Avoidance of failure during the operation of soft cores by application of validated soft cores.

Description: If the vendor validates the soft core, following requirements should be fulfilled:

– The validation of the soft core should be carried out for the operation of the safety related system, having at least an equivalent or higher safety integrity level than the system under plan.
– All the assumptions and confinements, which are necessary for the validation of the soft core, should be complied.
– All the necessary documents for the validation of the soft core should be easily available, see also E.17 "Documentation of simulation results".
– Each vendor specification should be strictly observed and the evidence of the compliance should be documented.

E.21 Validation of the soft core

NOTE See also E.6 "Functional test on module level".

Aim: Avoidance of failure during the operation of the soft core by validation of the soft core during design life cycle.

Description: If the soft core is not explicitly developed for the operation in a safety related system, the generated code should be validated under the same premises that apply for the validation of any source code. This means that all possible test cases should be defined and implemented. The functional verification should be then derived by simulation.

E.22 Simulation of the gate netlist to check timing constraints

Aim: Independent verification of the achieved timing constraint during synthesis.

Description: Simulation of the gate netlist produced by the synthesis including the back-annotation of line delays and gate delays. The stimuli should be derived to stimulate the circuit in such a fashion that it will cover a high percentage of the timing constraints and include all the worst case timing paths. In general, the stimuli needed to perform E.6 "Functional test on module level" or E.7 "Functional test" provide a suitable criteria for the selection of the stimuli, provided sufficient test coverage can be claimed during the functional test. The circuit should be tested under best- and worst-case condition at the maximum specified clock rate.

The timing verification can be carried out by the automatic check of the set-up and hold time of the memory elements (flip-flops) of the target library as well as by the functional verification of the circuit. The functional verification should be primarily performed by observing the outputs of the chip. This can be automated by comparing the output signals of the circuit with an adequate reference model or (V)HDL source code of the circuit. This test is known as a “regression test” and should be preferred against a manual check of the output signals.

NOTE By applying this measure, the timing behaviour of only those paths can be verified which are actually stimulated during the simulation and, therefore, the bespoke measure cannot provide a complete timing analysis of the circuit in general.

E.23 Static analysis of the propagation delay (STA)

Aim: Independent verification of the timing constraints realised during the synthesis.

Description: Static Timing Analysis (STA) analyses all the paths of a netlist (circuit) generated by the synthesis tool considering the back-annotation, i.e. estimated line delays by the synthesis tool, as well as gate delays without performing the actual simulation. Therefore it allows in general a complete analysis of the timing constraint of the entire circuit. The circuit to be tested should be analysed under best- and worst-case condition operating at maximum specified clock rate and accounting for applicable clock jitter and duty cycle skew. The number of non-relevant timing paths can be limited to a certain minimum by adopting a suitable design technique. It is recommended to investigate, analyse and define the used technique that allows easily readable results before beginning with the design.

NOTE It can be assumed that STA covers explicit all the existing timing paths if

a) The timing constraints are properly specified.

b) The circuit under test contains only such timing paths that can be analysed by STA tools, i.e. generally the case with full synchronous circuits.

E.24 Verification of the gate netlist against reference model by simulation

Aim: Functional equivalence check of the synthesised gate netlist.

Description: Simulation of the gate netlist generated by synthesis tool. The applied stimuli for the verification of the circuit by simulation correspond exactly to the stimuli applied during the E.6 "Functional test on module level" and the E.7 "Functional test on top level" for the verification of the function at module level and top level respectively. The functional verification should be primarily performed by observing the outputs of the chip. This can be automated by comparing the output signals of the circuit with an adequate reference model or (V)HDL source code of the circuit. This test is known as a “regression test” and should be preferred to a manual check of the output signals.

NOTE By applying this measure the functional behaviour of only those paths are verified which are actually stimulated during the simulation. The test coverage can, therefore, only be as good as during the original functional test at module- or top-level, respectively. It is possible to complement this measure with a formal equivalence test. In all cases a functional verification of the (V)HDL source code should be carried out with the final netlist generated by the synthesis tool.

E.25 Comparison of the gate netlist with the reference model (formal equivalence test)

Aim: Functional equivalence check which is independent of simulation.

Description: Comparison of the circuit functionality described by the (V)HDL source code with the circuit functionality of the gate netlist generated by synthesis. The tools based on the formal equivalence principle are capable of verifying the functional equivalence of a different representation form of the circuit for example (V)HDL description or netlist description. By applying this measure a functional simulation is not necessary and an independent functional check is feasible. The successful application of this measure can only be guaranteed, if the applied tool is capable of proving complete equivalence and all the discrepancies reported are evaluated either by manual inspection or automatically.

NOTE It is advantageous to combine this measure with E.24 "Verification of the gate netlist against reference model by simulation". In all cases, a functional verification of (V)HDL source code should be carried out with the final netlist generated by the synthesis tool.

E.26 Check of vendor requirements and constraints

Aim: Avoidance of failure during production by checking the vendor requirements.

Description: A careful checking of the vendor requirements and constraints for example minimum and maximum fan-in and fan-out, maximum wire length (line delay), maximum slew rate of the signals, clock skew and so on by the synthesis tool enhances the reliability of the product. Besides the importance of the requirements for the production process, their violation has a great impact on the validity of the applied models that are used for the simulation. So that any violation of the vendor requirements and constraints leads to faulty simulation results producing undesired functionality.

E.27 Documentation of synthesis constraint, results and tools

Aim: Documentation of all defined constraints that are necessary for an optimal synthesis to generate the final gate netlist.

Description: The documentation of all the synthesis constraints and results is indispensable because of the following reasons:

– to reproduce the synthesis at any later phase.
– to generate an independent synthesis results for verification.

Essential documents are:

– Synthesis set-up including the applied tools and synthesis software with the actual version, the applied synthesis library and the defined constraints and scripts.
– Synthesis log file with the time remark, applied tool with version and complete documentation of the synthesis.
– The generated netlist with estimated time delays (standard delay format (SDF) File).

E.28 Application of proven-in-use synthesis tool

Aim: Tool based conversion of (V)HDL description of a circuit in gate netlist.

Description: Tool based mapping of the (V)HDL source code of circuit functionality by connection of the suitable gates and circuit primitives of the target ASIC library. The selected implementation out of a variety of possible implementations that fulfil the desired functionality depends on the most optimal result that is derived by the synthesis constraints such as timing (clock rate) and chip area.

E.29 Application of proven-in-use target library

NOTE See also E.4 "Proven-in-use tools".

Aim: Avoidance of systematic failures caused by a faulty target library.

Description: The synthesis and simulation target library for the development of an ASIC are derived from a common database and, therefore, are not independent. A systematic failure such as:

– ambiguity between real and modelled behaviour of the circuit elements,
– insufficient modelling for example of set-up and hold time,

is one of the typical examples.

Therefore only “proven-in-use” technologies and target libraries should be used for the design of ASICs that perform safety functions. This means:

– The application of target libraries that have been used over a significant long time in projects with comparable complexity and clock rating.
– Availability of the technology and corresponding target library over a sufficient long period, so that enough modelling accuracy of the library can be expected.

E.30 Script based procedures

Aim: Reproducibility of results and automation of the synthesis cycles.

Description: Automatic and script based control of the synthesis cycles including the definition of the applied constraints. Besides a precise documentation of a complete synthesis constraint, it helps to reproduce the netlist after the alteration of the (V)HDL source code under identical conditions.

E.31 Implementation of test structures

Aim: Design of testable ASICs that guarantees the final production test.

Description: Design for testability allows easily testable circuits by implementation of different test structures, for example:

– Scan-path: In a scan technique, either all (full scan design) or part of flip-flops (partial scan design) is connected in a single chain or multiple chains building a chain of shift registers. The scan-path allows an automatic generation of test pattern of the entire logic of a circuit. The tool generating test pattern is called ATPG “Automatic Test Pattern Generator”. The implementation of scan-path improves the testability of a circuit tremendously and allows more than 98 % of test coverage with reasonable effort. It is therefore recommended to implement, if possible, a full scan-path.

– NAND-Tree: In a NAND-tree, all the primary inputs of a circuit are connected in cascading fashion to build a chain. By applying a suitable test pattern (“walking bit”) it is possible to test the switching behaviour (timing and triggering level) of the inputs. NAND-tree offers a straightforward means for the characterisation of primary inputs. Its implementation is recommended, if the switching behaviour of the circuit cannot be tested otherwise.

– Build-in self test (BIST): Self test of the circuit and in particular the self test of the embedded memory can be carried out very efficiently by implementing on-chip test pattern generator. BIST allows an automatic verification of the circuit structure by applying a pseudo-random test pattern and evaluating the signature of the implemented circuit structure. BIST is recommended as additive measure particularly for memory test. The scan-path test can be replaced by a BIST.

– Quiescent current test (IDDQ-test): A static CMOS-circuit consumes a current mainly during switching event. An absolutely defect free circuit consumes therefore negligibly small amount of current (< 1μA, leakage current) as long as the test pattern is held stationary. IDDQ-test is very effective and provides more than 50 % test coverage just after the application of a couple of test patterns. IDDQ-test can be applied on functional test patterns as well as on synthesised test patterns generated by ATPG. This test method has proven to be most helpful in practice and is capable of detecting failure that other tests rarely or even cannot detect. This measure should therefore be applied additive to the regular production tests.

– Boundary-scan: Test architecture implemented for the verification of the interconnection of the components on a printed circuit board according to JTAG standard. Same philosophy can also be applied to verify the interconnection of modules on chip level. Boundary-scan is primarily recommended to improve the testability of the printed circuit board.

E.32 Estimation of test coverage by simulation

Aim: Determination of the achieved test coverage by the implemented test architecture during production test.

Description: Test coverage achieved by the scan-path test, BIST, functional test pattern or any other measures can be determined by fault simulation. During the fault simulation a test pattern is applied to a circuit in which the faults are inserted. A faulty response of the circuit to the applied stimuli corresponds to the faults inserted and thus contributes to the test coverage. The fault simulation allows the detection of stuck-at-faults ”stuck-at-1” and ”stuck-at-0” and the achieved test coverage represents the quality of the test pattern applied. The fault simulation in general can be used very effectively to detect faults associated with logic that is not a part of the scan-path, for example in case of partial scan-paths.

E.33 Estimation of the test coverage by application of ATPG tool

Aim: Determination of the test coverage that can be expected by synthesised test pattern (Scan-path, BIST) during the production test.

Description: Currently, there is variety of procedures, which generate pseudo-random or algorithmic test patterns for a circuit implemented with scan-path. The synthesis tool such as ATPG creates during the synthesis a catalogue of undetected faults. The test coverage can thus be estimated and defines the lower limit of the achieved test coverage with the applied test pattern. It is important to notice that the test coverage is limited to the circuit logic, which

is covered by the scan-path. The modules such as memory, BIST or part of circuits that are not integrated in scan-path are not considered in the estimation of test coverage.

E.34 Justification of proven-in-use for applied hard cores

Aim: Avoidance of systematic failure during the application of hard cores

Description: A hard core is generally regarded as a black box representing the desired functionality and is composed of layout data basis in target technology that provides the desired circuit component. The possible functional failure can be treated in analogy to discrete components like, standard microprocessors, memories etc. The operation of such hard cores without the verification of correct functionality is possible, if for the applied target technology the used core can be considered as proven-in-use component. The rest of the circuit should then be verified intensively.

E.35 Application of validated hard cores

NOTE See also E.6 "Functional test on module level".

Aim: Avoidance of systematic failure during the application of hard cores.

Description: The core validation should be carried out by vendors, because of the complex nature of the core and assumed constraints, during the design phase on the basis of the (V)HDL source code. The validation can be justified only for the configuration and the target technology of the applied component.

E.36 On-line testing of hard cores

NOTE See also E.13 "Coverage of the verification scenarios (test benches)".

Aim: Avoidance of systematic failure during the application of hard cores.

Description: Verification of correct function and implementation of used hard cores by application of on-line tests. In applying this measure an efficient test concept is necessary and the evaluation of the applied concept should be documented.

E.37 Design rule check (DRC)

Aim: Verification of vendor design rules.

Description: Verification of the generated layout with respect to vendor design rules, for example minimum wire lengths, maximum wire lengths and several rules regarding placement of layout structures. A complete and correct run of DRC should be documented in detail.

E.38 Verification of layout versus schematic (LVS)

Aim: Independent verification of the layout.

Description: LVS extracts the circuit functionality from the layout data basis and compares the extracted circuit elements including interconnections with the input netlist. This assures the equivalence of circuit layout with the netlist specifying the circuit functionality. A complete and correct run of LVS should be documented in detail.

E.39 Additional slack (>20 %) for process technologies in use for less than 3 years

Aim: Assurance of the robustness of the implemented circuit functionality even under strong process and parameter fluctuation.

Description: The actual circuit behaviour is defined by number of overlapping physical effects particularly for small structures (for example below 0,5 µm). As a matter of fact, due to the lack of detail knowledge and necessary simplifications an exact model of circuit elements cannot be derived. With decreasing geometrical structures line delays play more and more dominant role. Signal delays along the wire and cross-coupling capacities between the wires grow over proportional. Signal delays are no longer negligible compared to gate delays. Estimated line delays depict increasing risk with decreasing geometrical structures.

Therefore it is recommended to plan an adequate amount of slack (> 20 %) with respect to minimal and maximal timing constraints for circuits designed using processes in use for less than 3 years, in order to guarantee correct operation of the circuit functionality in presence of strongly fluctuating parameters during the production or due to lack of precise modelling.

E.40 Burn-in Test

Aim: Assurance of the robustness of the manufactured chip. Weed out early failures. Bare die chip products do not have to prove their robustness by burn-in but, e.g., by wafer-level stress methods.

Description: The burn-in test should be carried out at the highest tolerable operating temperature (generally 125 °C). The test duration is depending on the aimed SIL-Level or on specific burn-in recommendations for example of the ASIC manufacturer. Burn in can be used to:

– weed out early failures (beginning of bathtube curve with decreasing failure rate);
– prove that early failures are already weeded out during manufacturing and testing (i.e. that devices out of the production line are already in the region of constant failure rate of the bathtube curve).

E.41 Application of proven-in-use device-series

Aim: Assurance of the reliability of the manufactured chips.

Description: The manufacturer of a safety design should have sufficient application experience with the used programmable device technology and the concerning developing tools.

E.42 Proven-in-use production process

Aim: Assurance of the reliability of the manufactured chips.

Description: A proven-in-use production process is characterised by a sufficient series production experience.

E.43 Quality control of the production process

Quality measures and control mechanisms during the device production process ensure a continuous process control. For example optical or electrical control of test structures,

temperature humidity bias-tests or temperature cycle test (see IEC 60068-2-1, IEC 60068-2-2 etc.).

E.44 Manufacturing quality pass of the device

The device quality will be proved by carrying out selected part-stress tests, for example temperature humidity bias-tests or change of temperature tests (see IEC 60068-2-1, IEC 60068-2-2 etc.). The device manufacturer will give proof of it.

E.45 Functional quality pass of the device

All devices will be functionally tested. The device manufacturer will give proof of it.

E.46 Quality standards

The ASIC manufacturer should provide for a sufficient quality management, for example documented within a Quality & Reliability Handbook: ISO 9000 certification or SSQA-, Standard Supplier Quality Assessment

Annex F
(informative)

Definitions of properties of software lifecycle phases

Table F.1 – Software Safety Requirements Specification

(see IEC 61508-3 7.2 and IEC 61508-3 Table C.1)

	Property	Definition
1.1	Completeness with respect to the safety needs to be addressed by software	The Software Safety Requirements Specification addresses all the safety needs and constraints resulting from earlier phases of the safety lifecycle and allocated to the Software. Safety needs and constraints are usually stated in the inputs to the Software Safety Requirements Specification activity. This may include the specification of what the Software must not do or must avoid.
1.2	Correctness with respect to the safety needs to be addressed by software	The Software Safety Requirements Specification providing an appropriate answer to the safety needs and constraints assigned to the Software. The objective is to assure that what is specified will really guarantee safety in all the necessary conditions.
1.3	Freedom from intrinsic specification faults, including freedom from ambiguity	Internal completeness and consistency of the Software Safety Requirements Specification: providing all necessary information for all the functions and situations that can be derived from its statements; expressing no contradicting or inconsistent statements. Contrary to completeness and consistency with respect to safety needs, internal completeness and consistency can be assessed based on the Software Safety Requirements Specification only
1.4	Understandability of safety requirements	The Software Safety Requirements Specification is fully understandable without excessive effort by all those who need to read it, even if they have not been involved earlier in the project, provided that they have the required knowledge. One important objective is to facilitate verification and, possibly, modifications.
1.5	Freedom from adverse interference of non-safety functions with the safety needs to be addressed by software	The Software Safety Requirements Specification avoids requirements that are not necessary to safety of the EUC. The objective is to avoid unnecessary complexity in the design and implementation of the software, so as to reduce the risk of faults and of functions not important to safety interfering with, or jeopardizing, those that are important to safety
1.6	Capability of providing a basis for verification and validation	The Software Safety Requirements Specification gives rise to tests and examinations that generate objective evidence that the software satisfies the Software Safety Requirements Specification.

Table F.2 – Software design and development: software architecture design

(see IEC 61508-3 7.4.3 and IEC 61508-3 Table C.2)

	Property	Definition
2.1	Completeness with respect to Software Safety Requirements Specification	The Software Architecture Design addresses all the safety needs and constraints raised by the Software Safety Requirements Specification.
2.2	Correctness with respect to Software Safety Requirements Specification	The Software Architecture Design provides an appropriate answer to the specified software safety requirements.
2.3	Freedom from intrinsic design faults	The Software Architecture Design and the Design Documentation are free from faults that can be identified independently of any specified software Safety Requirement. Examples: deadlocks, access to unauthorised resources, resource leaks, intrinsic incompleteness (i.e., failure to address all the situations that derive from the design itself).
2.4	Simplicity and understandability. Predictability of behaviour.	The Software Architecture Design allowing a correct and accurate prediction, in all specified situations, of the functioning of the Software. In particular, these situations include erroneous and failure situations. Predictability implies in particular that the functioning does not depend on items that cannot be controlled by designers or users.
2.5	Verifiable and testable design	The Software Architecture Design and the Design Documentation allow and facilitate the production of credible evidence that all the specified software safety requirements are correctly taken into account by the Design and that the Design is free from intrinsic faults. Verifiability may imply derived properties like simplicity, modularity, clarity, testability, provability, etc., depending on the verification techniques used.
2.6	Fault tolerance	The Software Architecture Design gives assurance that the software will have a safe behaviour in the presence of errors (internal errors, errors of operators or of external systems). Defensive design may be active or passive. Active defensive designs may include, features like detection, reporting and containment of errors, graceful degradation and cleaning up of any undesirable side effects prior to the resumption of normal operation. Passive defensive designs include features that guarantee the imperviousness to particular types of errors or particular conditions (avalanches of inputs, particular dates and times) without the software taking any specific action.
2.7	Defence against Common Cause Failure from external events	The Software Architecture Design facilitates the identification of common cause failure modes and effective precautions against failure.

Table F.3 – Software design and development: support tools and programming language

(see IEC 61508-3 7.4.4 and IEC 61508-3 Table C.3)

	Property	Definition
3.1	Support the production of software with the required software properties	Means to provide detection of errors or the elimination of error prone constructs
3.2	Clarity of the operation and functionality of the tool	The provision of comprehensive coverage and feedback on all aspects of operation of the tool
3.3	Correctness and repeatability of output	The consistency and accuracy of the tool output for any given input

Table F.4 – Software design and development: detailed design

(see IEC 61508-3 7.4.5 and IEC 61508-3 7.4.6 and IEC 61508-3 Table C.4)

	Property	Definition
4.1	Completeness with respect to Software Safety Requirements Specification	Methods of detailed software design and production are adopted that ensure that the resulting software addresses all the safety needs and constraints assigned to the Software.
4.2	Correctness with respect to Software Safety Requirements Specification	There exists specific evidence to claim that the safety requirements assigned to the Software have been met by the developed software.
4.3	Freedom from intrinsic design faults	The developed software is free from intrinsic faults. Examples: deadlocks, access to unauthorised resources, resource leaks.
4.4	Simplicity and understandability Predictability of behaviour	The behaviour of the developed software is predictable by objective and convincing testing and analysis.
4.5	Verifiable and testable design	The developed software is verifiable and testable.
4.6	Fault tolerance / Fault detection	Techniques and designs give assurance that the developed software will behave safely in the presence of errors.
4.7	Freedom from common cause failure	Techniques and designs identify common cause failure modes and provide effective precautions against software failure.

Table F.5 – Software design and development: software module testing and integration

(see IEC 61508-3 7.4.7 and IEC 61508-3 7.4.8 and IEC 61508-3 Table C.5)

	Property	Definition
5.1	Completeness of testing and integration with respect to the design specifications	The software testing examines the software behaviour sufficiently thoroughly to ensure that all the requirements of the Software Design Specification have been addressed.
5.2	Correctness of testing and integration with respect to the design specifications (succesful completion)	The module testing task is completed, and there exists specific evidence to claim that the safety requirements have been met.
5.3	Repeatability	Consistent results are produced on repeating the individual assessments carried out as part of the module testing and integration.
5.4	Precisely defined testing configuration	The module testing and integration has been applied to the right version of the elements and the software, with the results claimed, and allows the results to be linked to the specific configuration of the “as-built” software.

Table F.6 – Programmable electronics integration (hardware and software)

(see IEC 61508-3 7.5 and IEC 61508-3 Table C.6)

	Property	Definition
6.1	Completeness of integration with respect to the design specifications	The integration provides the appropriate depth and coverage of the system elements to demonstrate that it can perform the intended functions and does not perform unintended functions under all foreseeable operating conditions and under system failure. This covers the principles used for the verification, the targeted levels of design and aspects of the integration (for example verification of completeness of interaction between modules)
6.2	Correctness of integration with respect to the design specifications (successful completion)	The integration is based on correct assumptions. E.g., correctness of expected results, of the conditions of use considered, representativeness of test environments. The integration task is completed, and there exists specific evidence to claim that the safety requirements have been met.
6.3	Repeatability	Consistent results are produced on repeating the individual assessments carried out as part of the integration.
6.4	Precisely defined integration configuration	The integration gives appropriate assurance that it has been effectively applied as documented, to the right version of the elements and the Software, with the results claimed, and allows the results to be linked to the specific configuration of the “as-built” Software.

Table F.7 – Software aspects of system safety validation

(see IEC 61508-3 7.7 and IEC 61508-3 Table C.7)

	Property	Definition
7.1	Completeness of validation with respect to the Software Design Specification	The software validation addresses all the requirements of the Software Design Specification.
7.2	Correctness of validation with respect to the Software Design Specification (successful completion)	The software validation task is completed, and there exists specific evidence to claim that the safety requirements have been met.
7.3	Repeatability	Consistent results are produced on repeating the individual assessments carried out as part of the software validation.
7.4	Precisely defined validation configuration	The clear and concise definition of the system the requirements the environment

Table F.8 – Software modification

(see IEC 61508-3 7.8 and IEC 61508-3 Table C.8)

	Property	Definition
8.1	Completeness of modification with respect to its requirements	The modification has been properly approved by authorised personnel, with an appropriate understanding of its functional, safety, technical and operational consequences.
8.2	Correctness of modification with respect to its requirements	The modification achieves its specified objectives.
8.3	Freedom from introduction of intrinsic design faults	The modification does not introduce new systematic faults. Examples: division by zero, out of bound indexes or pointers, use of non initialised variables.
8.4	Avoidance of unwanted behaviour	The modification does not introduce any behaviour that, according to constraints stated in the Software Safety Requirements Specification, must be avoided.
8.5	Verifiable and testable design	The software design is such that the effect of the modification is capable of being examined thoroughly.
8.6	Regression testing and verification coverage	The software design is such that effective and thorough regression testing is possible to demonstrate that the software after modification continues to satisfy the Software Safety Requirements Specification.

Table F.9 – Software verification

(see IEC 61508-3 7.9 and IEC 61508-3 Table C.9)

	Property	Definition
9.1	Completeness of verification with respect to the previous phase	The verification is capable of establishing that the software satisfies all relevant requirements of the Software Safety Requirements Specification.
9.2	Correctness of verification with respect to the previous phase (successful completion)	The verification task is completed, and there exists specific evidence to claim that the safety requirements have been met.
9.3	Repeatability	Consistent results are produced on repeating the individual assessments carried out as part of the verification.
9.4	Precisely defined verification configuration;	The verification has been applied to the right version of the elements and the Software, with the results claimed, and allows the results to be linked to the specific configuration of the “as-built” Software.

Table F.10 – Functional safety assessment

(see IEC 61508-3 Clause 8 and IEC 61508-3 Table C.10)

	Property	Definition
10.1	Completeness of functional safety assessment with respect to this standard	The software functional safety assessment produces a clear statement on the extent of compliance found, the judgements made, remedial actions and timescales recommended, the conclusions reached and the recommendations arising for acceptance, qualified acceptance, or rejection and for any time constraints placed on these recommendations.
10.2	Correctness of software functional safety assessment with respect to the Design Specifications (succesful completion)	The software functional safety assessment task is completed, and there exists specific evidence to claim that the safety requirements have been met.
10.3	Traceable closure of all identified issues	There is a clear statement on the extent to which the issues arising during software functional safety assessment have been addressed.
10.4	The ability to modify the software functional safety assessment after change without the need for extensive re-work of the assessment	The software functional safety assessment is capable of being reworked to allow parts of the software functional safety assessment to be re-assessed after software change and for revised conclusions to be achieved, without the need for extensive rework of the complete software functional safety assessment.
10.5	Repeatability	The functional safety assessment is carried out against a consistent, planned and open process on identified individuals and document, which allows scrutiny of the basis for the assessments and the judgement achieved to all those affected by its judgements including system providers, users, maintainers and regulators. The functional safety assessment allows independent competent personnel to repeat the individual assessments carried out as part of the assessment.
10.6	Timeliness	The functional safety assessment is carried out at an appropriate frequency linked to the software safety lifecycle phases and at least prior to determined hazards being present, and it provides timely reporting of deficiencies. The outcome of tests, inspections, analyses etc. are actually available when they are required as input to an assessment decision.
10.7	Precisely defined configuration	The software functional safety assessment allows the results to be linked to the specific configuration of the system which is to be substantiated by the functional safety assessment results.

Annex G
(informative)

Guidance for the development of safety-related object oriented software

All of the recommendations of this standard for software design apply to object oriented software. Because the object-oriented approach presents information differently from procedural or functional approaches, the following list contains those recommendations that need specific consideration:

– understanding class hierarchies, and identification of the software function(s) that will be executed upon the invocation of a given method (including when using an existing class library);
– structure-based testing (IEC 61508-3, Table B.2 and IEC 61508-7, C.5.8)

Tables G.1 and G.2 provide informative guidance for the use of object oriented software to supplement the more general normative guidance provided in IEC 61508-3 Tables A.2 and A.4.

Table G.1 – Object Oriented Software Architecture

	Recommendation	Details	SIL1	SIL2	SIL3	SIL4
G1.1	Traceability of the concept of the application domain to the classes of the architecture.	Note 1	R	HR	HR	HR
G1.2	Use of suitable frames, commonly used combinations of classes and design patterns. NOTE When using existing frames and design patterns, the requirements of pre-developed software apply to these frames and patterns.	Note 2	R	HR	HR	HR

NOTE 1 Traceability from application domain to class architecture is less important.

NOTE 2 EXAMPLE 1: For a part of the intended safety-related project a frame might exist from a non safety-related project that has successfully solved a similar task and that is well known to the project participants. Then use of that frame is recommended.

EXAMPLE 2: It may happen that different algorithms are needed for solving closely connected sub tasks of the safety-related project. The strategy pattern can be chosen for accessing the algorithms.

EXAMPLE 3: Part of the safety-related project may consist of issuing proper warnings to inner and outer stake holders. The observer pattern can be chosen for organizing these warnings. The requirement does not apply for libraries.

NOTE 3 It is usually an abstract basic class that provides access to the derived concrete classes.

Table G.2 – Object Oriented Detailed Design

	Recommendation	SIL1	SIL2	SIL3	SIL4
G2.1	Classes should have only one objective	R	R	HR	HR
G2.2	Inheritance used only if the derived class is a refinement of its basic class	HR	HR	HR	HR
G2.3	Depth of inheritance limited by coding standards	R	R	HR	HR
G2.4	Overriding of operations (methods) under strict control	R	HR	HR	HR
G2.5	Multiple inheritance used only for interface classes	HR	HR	HR	HR
G2.6	Inheritance from unknown classes			NR	NR
G2.7	Verification that the reused object oriented libraries meet the recommendations of this table	HR	HR	HR	HR

NOTE 1 In other words: One class is characterised by having one responsibility, i.e. taking care of closely connected data and the operations on these data.

NOTE 2 Care is required to avoid circular dependencies between objects.

The following terms used above are informally defined here.

Table G.3 – Some Oriented Detailed terms

Term	Informal definition
Basic class	Class that has derived classes. A Basic Class is sometimes called upper class or parent class.
Derived Class	Class (assembly of attributes and operations) that inherits attributes and/or operations from another class (basic class). A derived class is sometimes called subclass or child class.
Frame	Structure of a program, in many cases pre-developed in order to be filled out for the specific application
Overriding	Replacing an operation (method, subroutine) by another operation (method, subroutine) of the same signature and inheritance hierarchy during run-time; property of object-oriented languages or programs; implements polymorphism
Signature of an operation	Name of an operation (subroutine, method), together with its parameters (arguments) and their types, occasionally also their return types. Two signatures are equal if they have the same names, number and types of parameters; in some languages also the return types have to be equal.

Bibliography

[1] IEC 60068-1:1988, *Environmental testing – Part 1: General and guidance*

[2] IEC 60529:1989, *Degrees of protection provided by enclosures (IP Code)*

[3] IEC 60812:2006, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*

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

[5] IEC 61000-4-1:2006, *Electromagnetic compatibility (EMC) – Part 4-1: Testing and measurement techniques – Overview of IEC 61000-4 series*

[6] IEC 61000-4-5:2005, *Electromagnetic compatibility (EMC) – Part 4-5: Testing and measurement techniques – Surge immunity test*

[7] IEC/TR 61000-5-2:1997, *Electromagnetic compatibility (EMC) – Part 5: Installation and mitigation guidelines – Section 2: Earthing and cabling*

[8] IEC 61025:2006, *Fault tree analysis (FTA)*

[9] IEC 61069-5:1994, *Industrial-process measurement and control – Evaluation of system properties for the purpose of system assessment – Part 5: Assessment of system dependability*

[10] IEC 61078:2006, *Analysis techniques for dependability – Reliability block diagram and boolean methods*

[11] IEC 61131-3:2003, *Programmable controllers – Part 3: Programming languages*

[12] IEC 61160:2005, *Design review*

[13] IEC 61163-1:2006, *Reliability stress screening – Part 1: Repairable assemblies manufactured in lots*

[14] IEC 61164:2004, *Reliability growth – Statistical test and estimation methods*

[15] IEC 61165:2006, *Application of Markov techniques*

[16] IEC 61326-3-1:2008, *Electrical equipment for measurement, control and laboratory use – EMC requirements – Part 3-1: Immunity requirements for safety-related systems and for equipment intended to perform safety-related functions (functional safety) – General industrial applications*

[17] IEC 61326-3-2:2008, *Electrical equipment for measurement, control and laboratory use – EMC requirements – Part 3-2: Immunity requirements for safety-related systems and for equipment intended to perform safety-related functions (functional safety) – Industrial applications with specified electromagnetic environment*

[18] IEC 81346-1:2009, *Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations – Part 1: Basic rules*

[19] IEC 61506:1997, *Industrial-process measurement and control – Documentation of application software*

[20] IEC/TR 61508-0:2005, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 0: Functional safety and IEC 61508*

[21] IEC 61511 (all parts), *Functional safety – Safety instrumented systems for the process industry sector*

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

[23] IEC 62308:2006, *Equipment reliability – Reliability assessment methods*

[24] ISO/IEC 1539-1:2004, *Information technology – Programming languages – Fortran – Part 1: Base language*

[25] ISO 5807:1985, *Information processing – Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts*

[26] ISO/IEC 7185:1990, *Information technology – Programming languages – Pascal*

[27] ISO/IEC 8631:1989, *Information technology – Program constructs and conventions for their representation*

[28] ISO/IEC 8652:1995, *Information technology – Programming languages – Ada*

[29] ISO 8807:1989, *Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*

[30] ISO/IEC 9899:1999, *Programming languages – C*

[31] ISO/IEC 10206:1991, *Information technology – Programming languages – Extended Pascal*

[32] ISO/IEC 10514-1:1996, *Information technology – Programming languages – Part 1: Modula-2, Base Language*

[33] ISO/IEC 10514-3:1998, *Information technology – Programming languages – Part 3: Object Oriented Modula-2*

[34] ISO/IEC 13817-1:1996, *Information technology – Programming languages, their environments and system software interfaces – Vienna Development Method – Specification Language – Part 1: Base language*

[35] ISO/IEC 14882:2003, *Programming languages – C++*

[36] ISO/IEC/TR 15942:2000, *Information technology — Programming languages — Guide for the use of the Ada programming language in high integrity systems*

[37] IEC 61800-5-2, *Electromagnetic compatibility (EMC) – Part 5: Installation and mitigation guidelines – Section 2: Earthing and cabling*

[38] IEC 60601 (all parts), *Medical electrical equipment*

[39] IEC 60068-2-1, *Environmental testing – Part 2-1: Tests – Test A: Cold*

[40] IEC 60068-2-2, *Environmental testing – Part 2-2: Tests – Test B: Dry heat*

[41] ISO 9000*, Quality management systems – Fundamentals and vocabulary*

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

[43] 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*

[44] IEC 61508-3:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements*

[45] IEC 61508-6: 2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3*

Index