

Railway applications — Communications, signalling and processing systems — Software for railway control and protection systems

ICS 35.240.60; 45.020

National foreword

This British Standard is the UK implementation of EN 50128:2001, incorporating corrigendum May 2010.

The UK participation in its preparation was entrusted by Technical Committee GEL/9, Railway electrotechnical applications, to Subcommittee GEL/9/1, Signalling and communications.

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

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

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

This British Standard, having been prepared under the direction of the Electrotechnical Sector Committee, was published under the authority of the Standards Committee and comes into effect on 15 May 2001

© BSI 2010

ISBN 978 0 580 71752 9

Amendments/corrigenda issued since publication

Date	Comments
31 August 2010	Implementation of CENELEC corrigendum May 2010. Addition of Annex ZZ

EUROPEAN STANDARD

EN 50128

NORME EUROPÉENNE

EUROPÄISCHE NORM

March 2001

ICS 29.280; 45.060.10

Incorporating corrigendum May 2010

English version

**Railway applications —
Communications, signalling and processing systems —
Software for railway control and protection systems**

Applications ferroviaires —
Systèmes de signalisation, de
télécommunication et de traitement —
Logiciels pour systèmes de commande
et de protection ferroviaire

Bahnanwendungen —
Telekommunikationstechnik, Signal-
technik und Datenverarbeitungssysteme —
Software für Eisenbahnsteuerungs- und
Überwachungssysteme

This European Standard was approved by CENELEC on 2000-11-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

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

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

CENELEC members are the national electrotechnical committees of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CENELEC

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

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

© 2001 CENELEC - All rights of exploitation in any form and by any means reserved worldwide for CENELEC members.

Foreword

This European Standard was prepared by SC 9XA, Communication, signalling and processing systems, of Technical Committee CENELEC TC 9X, Electrical and electronic applications for railways.

The text of the draft was submitted to the formal vote and was approved by CENELEC as EN 50128 on 2000-11-01.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2001-11-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2003-11-01

This European Standard should be read in conjunction with EN 50126, Railway applications — The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) and EN 50129, Railway applications — Safety related electronic systems for signalling.

Annexes designated “normative” are part of the body of the standard.

Annexes designated “informative” are given for information only.

In this standard, annex A is normative and annex B is informative.

Contents

	Pages
Introduction	5
1 Scope	7
2 Normative references	7
3 Definitions	8
4 Objectives and conformance	11
5 Software safety integrity levels	12
5.1 Objective	12
5.2 Requirements	12
6 Personnel and responsibilities	13
6.1 Objective	13
6.2 Requirements	13
7 Life cycle issues and documentation	15
7.1 Objectives	15
7.2 Requirements	15
8 Software Requirements Specification	18
8.1 Objectives	18
8.2 Input documents	18
8.3 Output documents	18
8.4 Requirements	18
9 Software architecture	20
9.1 Objectives	20
9.2 Input documents	20
9.3 Output documents	20
9.4 Requirements	20
10 Software design and implementation	21
10.1 Objectives	21
10.2 Input documents	22
10.3 Output documents	22
10.4 Requirements	22
11 Software verification and testing	25
11.1 Objective	25
11.2 Input documents	25
11.3 Output documents	25
11.4 Requirements	25
12 Software/hardware integration	28
12.1 Objectives	28
12.2 Input documents	28
12.3 Output documents	29
12.4 Requirements	29

13	Software validation	30
13.1	Objective	30
13.2	Input documents	30
13.3	Output documents	30
13.4	Requirements	30
14	Software assessment	32
14.1	Objective	32
14.2	Input documents	32
14.3	Output documents	32
14.4	Requirements	32
15	Software quality assurance	33
15.1	Objectives	33
15.2	Input documents	33
15.3	Output documents	33
15.4	Requirements	33
16	Software maintenance	35
16.1	Objective	35
16.2	Input documents	35
16.3	Output documents	36
16.4	Requirements	36
17	Systems configured by application data	37
17.1	Objectives	37
17.2	Input documents	37
17.3	Output documents	37
17.4	Requirements	38
17.4.1	Data preparation life cycle	38
17.4.2	Data preparation procedures and tools	38
17.4.3	Software development	39
Annex A	(normative) Criteria for the selection of techniques and measures	46
Annex B	(informative) Bibliography of techniques	61
Annex ZZ	(informative) Coverage of Essential Requirements of EC Directives	106
Figures		
Figure 1	— Integrity levels for safety-related systems	40
Figure 2	— Software safety route map	41
Figure 3	— Development life cycle 1	42
Figure 4	— Development life cycle 2	43
Figure 5	— Independence versus Software Integrity Level	44
Figure 6	— Relationship between generic system development and application development	45

Introduction

This standard is part of a group of related standards. The others are EN 50126, Railway applications — The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) and EN 50129, Railway applications — Safety related electronic systems for signalling. EN 50126 addresses system issues on the widest scale, while EN 50129 addresses the approval process for individual systems which may exist within the overall railway control and protection system. This standard concentrates on the methods which need to be used in order to provide software which meets the demands for safety integrity which are placed upon it by these wider considerations.

This standard owes much of its direction to earlier work done by Working Group 9 of IEC/TC 65. The work of WG 9 resulted in a generic standard for software for safety systems which is now part of IEC 61508. A particular aspect of the work by WG 9 is its inclusion of Software Integrity Level 0, which covers non-safety software, as well as Software Integrity Levels 1 to 4, which cover safety-related and safety-critical software. This standard also covers all five Software Integrity Levels.

Account has also been taken of the work of the Institution of Railway Signal Engineers (the IRSE), in particular its Technical Report Number 1, which addressed the same topic.

The key concept of this European Standard is that of levels of software safety integrity. The more dangerous the consequences of a software failure, the higher the software safety integrity level will be.

This European Standard has identified techniques and measures for 5 levels of software safety integrity where 0 is the minimum level and 4 the highest level. Four of these levels, 1 to 4, refer to safety-related software, whilst level 0 refers to non safety-related software. This level has been included as normative in order to allow a smooth transition between software developments for non safety-related systems and those for safety-related systems. The required techniques and measures for each software safety integrity level and for the non safety-related level are shown in the tables. In this version, the required techniques for level 1 are the same as for level 2, and the required techniques for level 3 are the same as for level 4. This European Standard does not give guidance on which level of software integrity is appropriate for a given risk. This decision will depend upon the many factors including the nature of the application, the extent to which other systems carry out safety functions and social and economic factors.

It is the function of EN 50126 and EN 50129 to specify the safety functions allocated to software.

This European Standard specifies those measures necessary to achieve these requirements. The process is illustrated in Figure 1.

EN 50126 and EN 50129 require that a systematic approach be taken to:

- i) identifying hazards, risks and risk criteria;
- ii) identifying the necessary risk reduction to meet the risk criteria;
- iii) defining an overall System Safety Requirements Specification for the safeguards necessary to achieve the required risk reduction;
- iv) selecting a suitable system architecture;
- v) planning, monitoring and controlling the technical and managerial activities necessary to translate the System Safety Requirements Specification into a safety-related system of a validated safety performance (or safety integrity).

As decomposition of the specification into a design comprising safety-related systems and components takes place, further allocation of safety integrity levels is performed. Ultimately this leads to the required software safety integrity levels.

The current state-of-the-art is such that neither the application of quality assurance methods (so-called fault avoiding measures) nor the application of software fault tolerant approaches can guarantee the absolute safety of the system. 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.

The principles applied in developing high integrity software include, but are not restricted to:

- top-down design methods;
- modularity;
- verification of each phase of the development life cycle;
- verified modules and module libraries;
- clear documentation;
- auditable documents; and
- validation testing.

These and related principles must be correctly applied. This standard specifies the level of assurance required to demonstrate this at each software safety integrity level.

After the System Safety Requirements Specification, which identifies all safety functions allocated to software and determines the system safety integrity level, has been obtained or produced, the functional steps in the application of this European Standard are shown in Figure 2 and are as follows:

- i) define the Software Requirements Specification and, in parallel, consider the software architecture. The software architecture is where the basic safety strategy is developed for the software and the software safety integrity level (clauses 5, 8 and 9);
- ii) design, develop and test the software according to the Software Quality Assurance Plan, software safety integrity level and the software life cycle (clause 10);
- iii) integrate the software on the target hardware (clause 12);
- iv) validate the software (clause 13);
- v) if software maintenance is required during operational life then re-activate this European Standard as appropriate (clause 16).

A number of activities run across the software development. These include verification (clause 11), assessment (clause 14) and quality assurance (clause 15).

Requirements are given for systems which are configured by application data (clause 17).

Requirements are also given for the competency of staff involved in software development (clause 6).

The standard does not mandate the use of a particular software development life cycle. However, a recommended life cycle and documentation set are given (clause 7 and Figures 3 and 4).

Tables have been formulated ranking various techniques/measures against the 5 software safety integrity levels. The tables are in annex A. Cross-referenced to the tables is a bibliography giving a brief description of each technique/measure with references to further sources of information. The bibliography is in annex B.

1 Scope

1.1 This European Standard specifies procedures and technical requirements for the development of programmable electronic systems for use in railway control and protection applications. It is aimed at use in any area where there are safety implications. These may range from the very critical, such as safety signalling to the non-critical, such as management information systems. These systems may be implemented using dedicated microprocessors, programmable logic controllers, multiprocessor distributed systems, larger scale central processor systems or other architectures.

1.2 This European Standard is applicable exclusively to software and the interaction between software and the system of which it is part.

1.3 Software safety integrity levels above 0 are for use in systems in which the consequences of failure could include loss of life. Economic or environmental considerations, however, may also justify the use of higher software safety integrity levels.

1.4 This European Standard applies to all software used in development and implementation of railway control and protection systems including:

- application programming;
- operating systems;
- support tools;
- firmware.

Application programming comprises high level programming, low level programming and special purpose programming (for example: Programmable Logic Controller ladder logic).

1.5 The use of standard, commercially available software and tools is also addressed in this European Standard.

1.6 This European Standard also addresses the requirements for systems configured by application data.

1.7 This European Standard is not intended to address commercial issues. These should be addressed as an essential part of any contractual agreement. All the clauses of this European Standard will need careful consideration in any commercial situation.

1.8 This European Standard is not intended to be retrospective. It therefore applies primarily to new developments and only applies in its entirety to existing systems if these are subjected to major modifications. For minor changes, only clause 16 applies.

2 Normative references

This European Standard incorporates by dated or undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this European Standard only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies (including amendments).

EN 50126, *Railway applications — The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*.

EN 50129*, *Railway applications — Safety related electronic systems for signalling*.

EN 50159-1, *Railway applications — Communication, signalling and processing systems — Part 1: Safety-related communication in closed transmission systems*.

EN 50159-2, *Railway applications — Communication, signalling and processing systems — Part 2: Safety-related communication in open transmission systems*.

EN ISO 9001, *Quality systems — Model for quality assurance in design/development, production, installation and servicing*.

EN ISO 9000-3, *Quality management and quality assurance standards — Part 3: Guidelines for the application of ISO 9001:1994 to the development, supply, installation and maintenance of computer software*.

3 Definitions

For the purposes of this European Standard, the following definitions apply. For terms not defined here, the following references should be consulted in order of priority:

EN ISO 8402, *Quality management and quality assurance — Vocabulary*.

IEC 60050-191, *International Electrotechnical Vocabulary of Chapter 191: Dependability and quality of service*.

IEEE 610.12, *IEEE standard glossary of software engineering terminology*.

ISO/IEC 2382, *Information Technology Vocabulary*.

ISO/IEC 9126, *Information Technology — Software Product Evaluation — Quality characteristics and guidelines for their use*.

3.1 assessment

process of analysis to determine whether the Design Authority and the Validator have achieved a product that meets the specified requirements and to form a judgement as to whether the product is fit for its intended purpose

3.2 assessor

person or agent appointed to carry out the assessment

3.3 availability

ability of a product to be in a state to perform a required function under given conditions at a given instant of time or over a given time interval, assuming the required external resources are provided

3.4 commercial off-the-shelf (COTS) software

software defined by market-driven need, commercially available and whose fitness for purpose has been demonstrated by a broad spectrum of commercial users

* at draft stage

3.5

design authority

body responsible for the formulation of a design solution to fulfil the specified requirements and for overseeing the subsequent development and setting to work of a system in its intended environment

3.6

designer

one or more persons assigned by the Design Authority to analyse and transform specified requirements into acceptable design solutions which have the required safety integrity

3.7

element

part of a product that has been determined to be a basic unit or building block. An element may be simple or complex

3.8

error

deviation from the intended design which could result in unintended system behaviour or failure

3.9

failure

deviation from the specified performance of a system. A failure is the consequence of a fault or error in a system

3.10

fault

abnormal condition that could lead to an error or a failure in a system. A fault can be random or systematic

3.11

fault avoidance

use of design techniques which aim to avoid the introduction of faults during the design and construction of the system

3.12

fault tolerance

built-in capability of a system to provide continued correct provision of service as specified, in the presence of a limited number of hardware or software faults

3.13

firmware

ordered set of instructions and associated data stored in a way that is functionally independent of main storage, usually in a ROM

3.14

generic software

generic software is software which can be used for a variety of installations purely by the provision of application-specific data

3.15

implementer

one or more persons assigned by the Design Authority to transform specified designs into their physical realization

3.16

product

collection of elements, interconnected to form a system, sub-system or item of equipment, in a manner which meets the specified requirements. In this European Standard, a product may be considered to consist entirely of elements of software or documentation

3.17

programmable logic controller (PLC)

solid-state control system which has a user programmable memory for storage of instructions to implement specific functions

3.18

reliability

ability of an item to perform a required function under given conditions for a given period of time

3.19

requirements traceability

objective of requirements traceability is to ensure that all requirements can be shown to have been properly met

3.20

risk

combination of the frequency, or probability, and the consequence of a specified hazardous event

3.21

safety

freedom from unacceptable levels of risk

3.22

safety authority

body responsible for certifying that the safety-related system is fit for service and complies with relevant statutory and regulatory safety requirements

3.23

safety-related software

software which carries responsibility for safety

3.24

software

intellectual creation comprising the programs, procedures, rules and any associated documentation pertaining to the operation of a system

NOTE Software is independent of the media used for transport.

3.25

software life cycle

activities occurring during a period of time that starts when software is conceived and ends when the software is no longer available for use. The software life cycle typically includes a requirements phase, development phase, test phase, integration phase, installation phase and a maintenance phase

3.26

software maintainability

capability of a system to be modified to correct faults, improve performance or other attributes, or adapt it to a different environment

3.27

software maintenance

action, or set of actions, carried out on software after its acceptance by the final user. The aim is to improve, increase and/or correct its functionality

3.28

software safety integrity level

classification number which determines the techniques and measures that have to be applied in order to reduce residual software faults to an appropriate level

3.29

system safety integrity level

number which indicates the required degree of confidence that a system will meet its specified safety features

3.30

traceability

degree to which a relationship can be established between two or more products of a development process, especially those having a predecessor/successor or master/subordinate relationship to one another

3.31

validation

activity of demonstration, by analysis and test, that the product meets, in all respects, its specified requirements

3.32

validator

person or agent appointed to carry out validation

3.33

verification

activity of determination, by analysis and test, that the output of each phase of the life cycle fulfils the requirements of the previous phase

3.34

verifier

person or agent appointed to carry out verification

4 Objectives and conformance

4.1 In each of the following clauses, the objectives and requirements of the clause are detailed.

4.2 To conform to this European Standard it shall be shown that each of the requirements have been satisfied to the software safety integrity level defined and therefore the clause objective has been met.

4.3 Where a requirement is qualified by the words "To the extent required by the software safety integrity level", this indicates that a range of techniques and measures can be used to satisfy that requirement.

4.4 Where 4.3 applies the tables detailed in this European Standard should be used to assist in the selection of techniques and measures appropriate to the software safety integrity level.

4.5 If a technique or measure is ranked as highly recommended (HR) in the tables then the rationale for not using that technique should be detailed and recorded either in the Software Quality Assurance Plan or in another document referenced by the Software Quality Assurance Plan. This is not necessary if an approved combination of techniques given in the corresponding table is used.

4.6 If a technique or measure is proposed to be used that is not contained in the tables then its effectiveness and suitability in meeting the particular requirement and overall objective of the clause shall be justified and recorded in either the Software Quality Assurance Plan or in another document referenced by the Software Quality Assurance Plan.

4.7 Compliance with the requirements of a particular clause and their respective techniques and measures detailed in the tables shall be assessed by the inspection of documents required by this standard, other objective evidence, auditing and the witnessing of tests.

4.8 This European Standard requires the use of a package of techniques and their correct application. These techniques are required from the tables and detailed in the bibliography.

5 Software safety integrity levels

5.1 Objective

To describe the assignment of software safety integrity levels to the software.

5.2 Requirements

5.2.1 There shall be produced, in accordance with EN 50126 and EN 50129:

- System Requirements Specification;
- System Safety Requirements Specification;
- System Architecture Description;
- System Safety Plan;

which include:

- safety functions;
- configuration or architecture of the system;
- hardware reliability requirements;
- safety integrity requirements.

The software safety integrity level shall be specified through following the general process for obtaining a safety integrity level identified in EN 50126.

5.2.2 The required software safety integrity level shall be decided on the basis of the level of risk associated with the use of the software in the system and the system safety integrity level.

5.2.3 Without further precautions, the software safety integrity level shall be, as a minimum, identical to the system safety integrity level. However, if mechanisms exist to prevent the failure of a software module from causing the system to go to an unsafe state, the software safety integrity level of the module may be reduced.

5.2.4 Risks which shall be taken into account are those associated with the following hazard consequences:

- loss of human life or lives;
- injuries to or illness of persons;
- environmental pollution; and
- loss of or damage to property.

5.2.5 Risk may be quantified but it is not possible to specify the software safety integrity in the same manner. Therefore for this European Standard the software safety integrity shall be specified as one of the following five levels:

Software safety integrity level	Description of software safety integrity
4	Very High
3	High
2	Medium
1	Low
0	Non safety-related

5.2.6 The software safety integrity level shall be specified in the Software Requirements Specification (clause 8). If different software components have different software safety integrity levels, these shall be specified in the Software Architecture Specification (clause 9).

6 Personnel and responsibilities

6.1 Objective

To ensure that all personnel who have responsibilities for the software are competent to discharge those responsibilities.

6.2 Requirements

6.2.1 As a minimum, the supplier and/or developer and the customer shall implement the relevant parts of EN ISO 9001, in accordance with the guidelines contained in EN ISO 9000-3.

6.2.2 Except at software safety integrity level 0, the safety process shall be implemented under the control of an appropriate safety organization which is compliant with the “Safety Organization” sub-clause in the “Evidence of Safety Management” clause of EN 50129.

6.2.3 All personnel involved in all the phases of the software life cycle, including management activities, shall have the appropriate training, experience and qualifications.

6.2.4 It is highly recommended that the training, experience and qualifications of all personnel involved in all the phases of the software life cycle, including management activities, be justified with respect to the particular application, except at software safety integrity level 0.

6.2.5 The justification contained in 6.2.4 shall be recorded in the Software Quality Assurance Plan, and shall include evidence of competency in the following areas, as appropriate:

- i) engineering appropriate to the application area;
- ii) software engineering;
- iii) computer-systems engineering;
- iv) safety engineering;
- v) legal and regulatory framework.

6.2.6 An independent assessor for the software shall be appointed. See also 6.2.10 and 14.4.1.

6.2.7 The assessor shall be given authority to perform the assessment of the software.

6.2.8 Throughout the software life cycle, the parties involved shall be independent, to the extent required by the software safety integrity level, in accordance with Figure 5, which shall be interpreted as follows.

At all software safety integrity levels, the Assessor shall be approved by the Safety Authority and independent from the supplier except in the circumstances defined in 6.2.10.

The Designer/Implementer, Verifier and Validator can all belong to the same company but the following rules for minimum independence shall be complied with:

At software safety integrity level 0:

There are no constraints; the Designer/Implementer, Verifier and Validator can all be the same person.

At software safety integrity level 1 and 2:

The Verifier and Validator can be the same person but they shall not be the Designer/Implementer. However, the Designer/Implementer, Verifier and Validator can all report through the Project Manager.

At software safety integrity level 3 and 4 there are two permissible arrangements:

- a) The Verifier and Validator can be the same person but they shall not also be the Designer/Implementer. In addition, the Verifier and Validator shall not all report through the Project Manager as the Designer/Implementer does and they shall have the authority to prevent the release of the product.
- b) The Designer/Implementer, Verifier and Validator must all be separate persons. The Designer/Implementer and Verifier can report through the Project Manager, whereas the Validator shall not and the Validator shall have the authority to prevent the release of the product.

6.2.9 The parties responsible for the various clauses are as follows:

Software Requirements Specification (clause 8)	Designer
Software Requirements Test Specification (clause 8)	Validator
Software architecture (clause 9)	Designer
Software design and development (clause 10)	Designer
Software verification and testing (clause 11)	Verifier
Software/hardware integration (clause 12)	Designer
Software validation (clause 13)	Validator
Software assessment (clause 14)	Assessor

6.2.10 At the discretion of the Safety Authority, the Assessor may be part of the supplier's organization or of the customer's organization but, in such cases, the Assessor shall:

- be authorized by the Safety Authority;
- be totally independent from the project team;
- report directly to the Safety Authority.

7 Life cycle issues and documentation

7.1 Objectives

- 7.1.1 To structure the development of the software into defined phases and activities.
- 7.1.2 To record all information pertinent to the software throughout the life cycle of the software.

7.2 Requirements

7.2.1 A life cycle model for the development of software shall be selected. It shall be detailed in the Software Quality Assurance Plan in accordance with clause 15 of this European Standard. For example, two life cycle models are shown in Figures 3 and 4.

7.2.2 Quality assurance procedures shall run in parallel with life cycle activities and use the same terminology.

7.2.3 All activities to be performed during a phase shall be defined prior to the phase commencing. Each phase of the software life cycle shall be divided into elementary tasks with a well-defined input, output and activity for each of them.

7.2.4 The Software Quality Assurance Plan shall describe which verification steps and reports are required.

7.2.5 All documents shall be structured to allow continued expansion in parallel with the design process.

7.2.6 Traceability of documents shall be provided for by each document having a unique reference number and a defined and documented relationship with other documents. Each term, acronym or abbreviation shall have the same meaning in every document. If, for historical reasons, this is not possible, the different meanings shall be listed and the references given.

In addition, each document except documents for COTS software (see 9.4.5) or previously developed software (see 9.4.6) shall be written according to the following rules:

- a) it shall contain or implement all applicable conditions and requirements of the predecessor document with which it has a hierarchical relationship;
- b) it shall not contradict the predecessor document;
- c) each term, acronym or abbreviation shall have the same meaning in every document;
- d) each item or concept shall be referred to by the same name or description in every document.

7.2.7 The contents of all documents shall be recorded in a form appropriate for manipulation, processing and storage.

7.2.8 To the extent required by the software safety integrity level, the documents listed in the Documents Cross-Reference Table (see below) shall be produced.

7.2.9 Depending upon the size, complexity and life cycle of the software being developed, the number of separate documents required to be produced will vary. Some documents may be combined (providing there is no loss of the required detail in the process). For large projects it may be necessary to sub-divide the documentation listed (in a hierarchical manner) into a number of more manageable child documents. Documents which have been produced by independent teams or entities shall not be combined into a single document.

7.2.10 The relationship between the various documents identified in clause 7 can also be defined by using a DCRT (a Document Cross-Reference Table). For each document listed in the “Documents” column, the phase and clause associated with its creation can be found by reading horizontally and vertically from the cell containing the symbol “■”. The phases in which it is used can be found by reading vertically from the cells marked with the symbol “◆”. The clause or other reference to the definition of the document can be found in the “Where defined” column. Where a clause is given, the following clauses should also be checked as they may contain further information. It should be noted that the reference for the Software Configuration Management Plan is shown in brackets because that clause simply references EN ISO 9001.

Documents Cross-Reference Table

clause	8	9	10	11	12	13	14	15	16	DOCUMENTS where defined	
title	SRS	SA	SDD	SVer	S/H I	SVal	Ass	Q	Ma		
PHASES (*) = in parallel with other phases											
SYSTEM INPUTS	◆			◆	◆					System Requirements Specification	EN 50129 annex B.2.3
	◆	◆		◆	◆	◆	◆			System Safety Requirements Specification	EN 50129 annex B.2.4
	◆				◆					System Architecture Description	EN 50129 annex B.2.1
										System Safety Plan	EN 50129 EN 50126
SW PLANNING (*)	◆	◆	◆	◆		◆	◆	■		Sw Quality Assurance Plan	15.4.3
						◆	◆	■		Sw Configuration Management Plan	(15.4.2)
				■		◆	◆			Sw Verification Plan	11.4.1
				■		◆	◆			Sw Integration Test Plan	11.4.5
					■	◆	◆			Sw/Hw Integration Test Plan	12.4.1
						■	◆			Sw Validation Plan	13.4.3
							◆	■		Sw Maintenance Plan	16.4.3
										Data Preparation Plan	17.4.2.1
										Data Test Plan	17.4.2.4
SW REQUIREMENTS	■	◆	◆	◆	◆	◆	◆			Sw Requirements Specification	8.4.1
										Application Requirements Specification	17.4.1.1
	■			◆	◆	◆	◆			Sw Requirements Test Specification	8.4.13
				■						Sw Requirements Verification Report	11.4.11
SW DESIGN		■	◆	◆	◆	◆	◆			Sw Architecture Specification	9.4.1
			■	◆	◆	◆	◆			Sw Design Specification	10.4.3
				■						Sw Arch. and Design Verification Report	11.4.12
SW MODULE DESIGN			■	◆	◆	◆	◆			Sw Module Design Specification	10.4.3
			■	◆	◆	◆	◆			Sw Module Test Specification	10.4.14
				■						Sw Module Verification Report	11.4.13
CODE			■	◆	◆	◆	◆			Sw Source Code	
				■		◆	◆			Sw Source Code Verification Report	11.4.14
MODULE TESTING			■	◆						Sw Module Test Report	10.4.14
SW INTEGRATION				■						Sw Integration Test Report	11.4.15
										Data Test Report	17.4.2.4
SW/HW INTEGRATION					■					Sw/Hw Integration Test Report	12.4.8
VALIDATION (*)						■				Sw Validation Report	13.4.10
ASSESSMENT (*)							■			Sw Assessment Report	14.4.9
MAINTENANCE									■	Sw Change Records	16.4.9
									■	Sw Maintenance Records	16.4.8

8 Software requirements specification

8.1 Objectives

8.1.1 To describe a document which defines a complete set of requirements for the software meeting all system requirements to the extent required by the software safety integrity level. It serves the purpose of a comprehensive document for each software engineer and makes it unnecessary for him to screen for requirements in any other document.

8.1.2 To describe the Software Requirements Test Specification.

8.2 Input documents

- 1) System Requirements Specification.
- 2) System Safety Requirements Specification.
- 3) System Architecture Description.
- 4) Software Quality Assurance Plan.

8.3 Output documents

- 1) Software Requirements Specification.
- 2) Software Requirements Test Specification.

8.4 Requirements

8.4.1 The Software Requirements Specification shall express the required properties of the software being developed, not the procedures to develop them. These properties, which are all (except safety) defined in ISO/IEC 9126, shall include:

- functionality (including capacity and response time performance);
- reliability and maintainability;
- safety (including safety functions and their associated software safety integrity levels);
- efficiency;
- usability;
- portability.

The software safety integrity level shall be derived as defined in clause 5 and recorded in the Software Requirements Specification.

8.4.2 To the extent required by the software safety integrity level the Software Requirements Specification shall be expressed and structured in such a way that it is:

- i) complete, clear precise, unequivocal, verifiable, testable, maintainable and feasible;
- ii) traceable back to all documents mentioned under 8.2.

8.4.3 The Software Requirements Specification shall include modes of expression and descriptions which are understandable to the responsible personnel involved in the whole life cycle of the system.

8.4.4 The Software Requirements Specification shall identify and document all interfaces with any other systems, either within or outside the equipment under control, including operators, wherever a direct connection exists or is planned.

8.4.5 All relevant modes of operation shall be detailed in the Software Requirements Specification.

8.4.6 All relevant modes of behaviour of the programmable electronics, in particular failure behaviour, shall be detailed in the Software Requirements Specification.

8.4.7 Any constraints between the hardware and the software shall be identified and documented in the Software Requirements Specification.

8.4.8 The Software Requirements Specification shall indicate the degree of software self-checking and the specified degree of hardware checking by the software. Software self-checking consists of the detection and reporting by the software of its own failures and errors.

8.4.9 The Software Requirements Specification shall include requirements for the periodic testing of functions to the extent required by the System Safety Requirements Specification.

8.4.10 The Software Requirements Specification shall include requirements to enable all safety functions to be testable during overall system operation to the extent required by the System Safety Requirements Specification.

8.4.11 When the software is required to perform functions especially those related to achieving the required system safety integrity level, then these shall be clearly identified in the Software Requirements Specification.

8.4.12 When the software is required to perform non-safety functions then these shall be clearly identified in the Software Requirements Specification.

8.4.13 A Software Requirements Test Specification shall be developed from the Software Requirements Specification. This test specification shall be used for verification of all the requirements as described in the Software Requirements Specification and also as a description of the tests to be performed on the completed software.

8.4.14 The Software Requirements Test Specification shall identify for each required function the test cases including:

- i) the required input signals with their sequences and their values;
- ii) the anticipated output signals with their sequences and their values;
- iii) the acceptance criteria, including performance and quality aspects.

8.4.15 Traceability to requirements shall be an important consideration in the validation of a safety-related system and means shall be provided to allow this to be demonstrated throughout all phases of the life cycle.

8.4.16 Any untraceable material shall be shown to have no bearing upon the safety or integrity of the system.

9 Software architecture

9.1 Objectives

- 9.1.1 To develop a software architecture that achieves the requirements of the Software Requirements Specification to the extent required by the software safety integrity level.
- 9.1.2 To review the requirements placed on the software by the system architecture.
- 9.1.3 To identify and evaluate the significance of hardware/software interactions for safety.
- 9.1.4 To choose a design method if one has not been previously defined.

9.2 Input documents

- 1) Software Requirements Specification.
- 2) System Safety Requirements Specification.
- 3) System Architecture Description.
- 4) Software Quality Assurance Plan.

9.3 Output documents

Software Architecture Specification.

9.4 Requirements

- 9.4.1 The proposed software architecture shall be established by the software supplier and/or developer and detailed in the Software Architecture Specification.
- 9.4.2 The Software Architecture Specification shall consider the feasibility of achieving the Software Requirements Specification at the required software safety integrity level.
- 9.4.3 The Software Architecture Specification shall identify, evaluate and detail the significance of all hardware/software interactions. As required by EN 50126 and EN 50129, the preliminary studies concerning the interactions between hardware and software shall have been recorded in the System Safety Requirements Specification.
- 9.4.4 The Software Architecture Specification shall identify all software components and for these components identify:
 - i) whether these components are new, existing or proprietary;
 - ii) whether these components have been previously validated and if so their validation conditions;
 - iii) the software safety integrity level of the component.
- 9.4.5 The use of COTS software shall be subject to the following restrictions:
 - i) for software safety integrity level 0, the use of the COTS software shall be accepted with no further precautions;

- ii) if COTS software is to be used at software safety integrity levels 1 or 2, it shall be included in the software validation process;
- iii) if COTS software is to be used at software safety integrity levels 3 or 4, the following precautions shall also be taken:
 - a) the COTS software shall be included in the validation testing;
 - b) an analysis of possible failures shall be carried out;
 - c) a strategy shall be defined to detect failures of the COTS software and to protect the system from these failures;
 - d) the protection strategy shall be the subject of validation testing;
 - e) error logs shall exist and shall be evaluated;
 - f) as far as practicable, only the simplest functions of the COTS software shall be used.

9.4.6 If previously developed software is to be used as part of the design then it shall be clearly identified and documented. The Software Architecture Specification shall justify the software's suitability in satisfying the Software Requirements Specification and the software safety integrity level. The effects of any changes to the software on the rest of the system must be carefully considered in order to decide whether they require a re-inspection and re-assessment. There shall be evidence that interface specifications to other modules which are not being re-verified, re-validated and re-assessed are being followed.

9.4.7 Whenever possible existing verified software modules developed according to this standard shall be used in the design.

9.4.8 The software architecture shall minimize the safety part of the application.

9.4.9 Where the software consists of components of different software safety integrity levels, then all of the software components shall be treated as belonging to the highest software safety integrity level unless there is evidence of independence between the higher software safety integrity level components and the lower software safety integrity level components. This evidence shall be recorded in the Software Architecture Specification.

9.4.10 The Software Architecture Specification shall identify the strategy for the software development to the extent required by the software safety integrity level. The Software Architecture Specification shall be expressed and structured in such a way that it is:

- i) complete, clear, precise, unequivocal, verifiable, testable, maintainable and feasible;
- ii) traceable back to the Software Requirements Specification.

9.4.11 The Software Architecture Specification shall justify the balance taken between the strategies of avoiding faults and handling faults.

9.4.12 The Software Architecture Specification shall justify that the techniques and measures chosen, form a set which satisfies the Software Requirements Specification at the required software safety integrity level.

10 Software design and implementation

10.1 Objectives

10.1.1 To design and implement software of a defined software safety integrity level from the Software Requirements Specification and the Software Architecture Specification.

10.1.2 To achieve software which is analysable, testable, verifiable and maintainable. Module testing is also included in this phase. As verification and test will be a critical element in the validation, particular consideration shall be given to verification and test needs throughout the design and development, in order to ensure the resultant system and its software will be readily testable from the outset.

10.1.3 To select a suitable set of tools, including languages and compilers, for the required software safety integrity level, over the whole life cycle of the software which assists verification, validation, assessment and maintenance.

10.1.4 To carry out software integration.

10.2 Input documents

- 1) Software Requirements Specification.
- 2) Software Architecture Specification.
- 3) Software Quality Assurance Plan.

10.3 Output documents

- 1) Software Design Specification.
- 2) Software Module Design Specification.
- 3) Software Module Test Specification.
- 4) Software Source code and supporting documentation.
- 5) Software Module Test Report.

10.4 Requirements

10.4.1 The Software Requirements Specification and the Software Architecture Specification shall be available, although not necessarily finalized, prior to the start of the design process.

10.4.2 The size and complexity of the software developed shall be kept to a minimum.

10.4.3 The Software Design Specification shall describe the software design based on a decomposition into modules with each module having a Software Module Design Specification and a Software Module Test Specification.

10.4.4 The Software Design Specification shall address:

- i) software components traced back to software architecture and their safety integrity level;
- ii) interfaces of software components with the environment;
- iii) interfaces between the software components;
- iv) data structure;
- v) partitioning of requirements on components;
- vi) main algorithms and sequencing;
- vii) diagrams.

10.4.5 The Software Module Design Specifications shall address (one Software Module Design Specification per module):

- i) identification of all lowest software components (called modules in the present standard) traced back to the upper level;
- ii) their detailed interfaces with environment and other modules with detailed inputs and outputs;
- iii) their safety integrity level;
- iv) detailed algorithms and data structures.

Each Software Module Design Specification should be self-consistent and allow coding of the corresponding module.

10.4.6 Each software module shall be readable, understandable and testable.

10.4.7 A suitable set of tools, including design methods, languages and compilers shall be selected for the required software safety integrity level over the whole life cycle of the software.

10.4.8 When applicable, automatic testing tools and integrated development tools shall be used. This shall take account of the needs of the Verifier and Validator.

10.4.9 To the extent required by the software safety integrity level, the programming language selected shall have a translator/compiler which has one of the following:

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

10.4.10 The language chosen shall meet the following requirements:

- i) the language chosen shall contain features that facilitate the identification of programming errors;
- ii) the language chosen shall support features that match the design method.

10.4.11 When 10.4.10 cannot be satisfied then a justification for any alternative language detailing its fitness for purpose shall be recorded in the Software Architecture Specification or Software Quality Assurance Plan.

10.4.12 Coding standards shall be developed and used for the development of all software. These shall be referenced in the Software Quality Assurance Plan (see 15.4.5).

10.4.13 The coding standards shall specify good programming practice, proscribe unsafe language features and describe procedures for source code documentation. As a minimum, each software module shall contain in the source code the information defined in the following (non-exhaustive) list:

- author;
- configuration history;
- short description.

The use of a standard form for this information is recommended. It should be the same for all the modules.

10.4.14 Software Module Testing: each module shall have a Software Module Test Specification which the module shall be tested against. These tests shall show that each module performs its intended function. The Software Module Test Specification shall define the required degree of test coverage.

A Software Module Test Report shall be produced and shall include the following features:

- i) a statement of the test results and whether each module has met the requirements of its Software Module Design Specification;
- ii) a statement of test coverage shall be provided for each module, showing that all source code instructions have been executed at least once;
- iii) it shall be in a form that is auditable;
- iv) test cases and their results shall be recorded in a machine readable form for subsequent analysis; tests should be repeatable and be performed by automatic means, if practicable.

Checking that the module has correctly satisfied its test specification is a verification activity, see clause 11.

10.4.15 In accordance with the required software safety integrity level the design method chosen shall possess features that facilitate:

- i) abstraction, modularity and other features which control complexity;
- ii) the clear and precise expression of:
 - functionality;
 - information flow between components;
 - sequencing and time related information;
 - concurrency;
 - data structure and properties;
- iii) human comprehension;
- iv) verification and validation.

10.4.16 The design method chosen shall possess features that facilitate software maintenance. Such features shall include modularity, information hiding and encapsulation.

10.4.17 The integration of software modules shall be the process of progressively combining individual and previously tested modules of software into a composite whole (or into a number of composite sub-systems) in order that the module interfaces and the assembled software may be adequately proven prior to system integration and test.

10.4.18 Within the context of this standard, and to a degree appropriate to the specified software safety integrity level, traceability shall particularly address:

- i) traceability of requirements to the design or other objects which fulfil them;
- ii) traceability of design objects to the implementation objects which instantiate them.

The output of the traceability process shall be the subject of formal configuration management.

11 Software verification and testing

11.1 Objective

To the extent required by the software safety integrity level, to test and evaluate the products of a given phase to ensure correctness and consistency with respect to the products and standards provided as input to that phase.

11.2 Input documents

- 1) System Requirements Specification.
- 2) System Safety Requirements Specification.
- 3) Software Requirements Specification.
- 4) Software Requirements Test Specification.
- 5) Software Architecture Specification.
- 6) Software Design Specification.
- 7) Software Module Design Specification.
- 8) Software Module Test Specification.
- 9) Software Source Code and supporting documentation.
- 10) Software Quality Assurance Plan.
- 11) Software Module Test Report.

11.3 Output documents

- 1) Software Verification Plan.
- 2) Software Requirements Verification Report.
- 3) Software Architecture and Design Verification Report.
- 4) Software Module Verification Report.
- 5) Software Source Code Verification Report.
- 6) Software Integration Test Plan.
- 7) Software Integration Test Report.

11.4 Requirements

11.4.1 A Software Verification Plan shall be created in order that verification activities may be properly directed and that particular design or other verification needs may be suitably provided for. During development (and depending upon the size of the system) the plan may be sub-divided into a number of child documents and be naturally added to, as the detailed needs of verification become clearer.

11.4.2 The Software Verification Plan shall document all the criteria, techniques and tools to be utilized in the verification process for that phase.

11.4.3 The Software Verification Plan shall describe the activities to be performed to ensure correctness and consistency with respect to the products and standards provided as input to that phase.

11.4.4 The Software Verification Plan shall address the following:

- i) the selection of verification strategies and techniques. To avoid undue complexity in the assessment of the verification and test activity, preference should be given to the selection of test cases and methods, etc., which are in themselves readily analysable;
- ii) the selection and utilization of the software test equipment;
- iii) the selection and documentation of verification activities;
- iv) the evaluation of verification results gained;
- v) the evaluation of the reliability requirements;
- vi) the roles and responsibilities of those involved in the test process;
- vii) the degree of test coverage required to be achieved.

11.4.5 The Software Integration Test Plan shall document the following:

- i) test cases and test data;
- ii) types of tests to be performed;
- iii) test environment, tools, configuration and programs;
- iv) test criteria on which the completion of the test will be judged.

11.4.6 In each development phase it shall be shown that the functional, reliability, performance and safety requirements are met.

11.4.7 Verification shall be carried out by an independent party to the extent required by the software safety integrity level as defined in Figure 5.

11.4.8 Testing which is not fully documented and is performed by the designer prior to verification shall not be regarded as part of the verification.

11.4.9 The results of each verification shall be retained in a form defined or referenced in the Software Verification Plan such that it is auditable.

11.4.10 After each verification activity a verification report shall be produced stating either that the software has passed the verification or the reasons for the failures. The verification reports shall address the following:

- i) items which do not conform to the Software Requirements Specification, Software Design Specification or Software Module Design Specifications;
- ii) items which do not conform to the Software Quality Assurance Plan;
- iii) modules, data, structures and algorithms poorly adapted to the problem;

- iv) detected errors or deficiencies;
- v) the identity and configuration of the items verified.

11.4.11 Software Requirements Verification: once the Software Requirements Specification has been established, verification shall address:

- i) the adequacy of the Software Requirements Specification in fulfilling the requirements set out in the System Requirements Specification, the System Safety Requirements Specification and the Software Quality Assurance Plan;
- ii) the adequacy of the Software Requirements Test Specification as a test of the Software Requirements Specification;
- iii) the internal consistency of the Software Requirements Specification.

The results shall be recorded in a Software Requirements Verification Report.

11.4.12 Software Architecture and Design Verification: after the Software Architecture Specification and the Software Design Specification have been established, verification shall address:

- i) the adequacy of the Software Architecture Specification and the Software Design Specification in fulfilling the Software Requirements Specification;
- ii) the adequacy of the Software Design Specification for the Software Requirements Specification with respect to consistency and completeness;
- iii) the adequacy of the Software Integration Test Plan as a set of test cases for the Software Architecture Specification and the Software Design Specification;
- iv) the internal consistency of the Software Architecture and Design Specifications.

The results shall be recorded in a Software Architecture and Design Verification Report.

11.4.13 Software Module Verification: after each Software Module Design Specification has been established, verification shall address:

- i) the adequacy of the Software Module Design Specification in fulfilling the Software Design Specification;
- ii) the adequacy of the Software Module Test Specification as a set of test cases for the Software Module Design Specification;
- iii) the decomposition of the Software Design Specification into software modules and the Software Module Design Specifications with reference to:
 - feasibility of the performance required;
 - testability for further verification;
 - readability by the development and verification team; and
 - maintainability to permit further evolution;
- iv) the adequacy of the Software Module Test Reports as a record of the tests carried out in accordance with the Software Module Test Specification.

The results shall be recorded in a Software Module Verification Report.

11.4.14 Software Source Code Verification: to the extent demanded by the software safety integrity level the Software Source Code shall be verified to ensure conformance to the Software Module Design Specification and the Software Quality Assurance Plan. This shall include a check to determine whether the coding standards have been applied correctly.

The results shall be recorded in a Software Source Code Verification Report.

11.4.15 A Software Integration Test Report shall be produced as follows:

- i) a Software Integration Test Report shall be produced stating the test results and whether the objectives and criteria of the Software Integration Test Plan have been met. If there is a failure, the reasons for the failure shall be recorded;
- ii) the Software Integration Test Report shall be in a form that is auditable;
- iii) test cases and their results shall be recorded, preferably in machine readable form for subsequent analysis;
- iv) tests should be repeatable and be performed by automatic means, if practicable;
- v) the identity and configuration of the items verified.

11.4.16 For software/hardware integration, see 12.4.8.

12 Software/hardware integration

12.1 Objectives

12.1.1 To demonstrate that the software and the hardware interact correctly to perform their intended functions.

12.1.2 To combine the software and hardware, ensuring their compatibility, to meet the System Safety Requirements Specification and the requirements of the intended software safety integrity level.

12.2 Input documents

- 1) System Requirements Specification.
- 2) System Safety Requirements Specification.
- 3) System Architecture Description.
- 4) Software Requirements Specification.
- 5) Software Requirements Test Specification.
- 6) Software Architecture Specification.
- 7) Software Design Specification.
- 8) Software Module Design Specification.
- 9) Software Module Test Specification.
- 10) Software Source code and supporting documentation.
- 11) Hardware documentation.

12.3 Output documents

- 1) Software/Hardware Integration Test Plan.
- 2) Software/Hardware Integration Test Report.

12.4 Requirements

12.4.1 For software safety integrity levels greater than zero, a Software/Hardware Integration Test Plan will be created early in the development life cycle, in order that integration activities may be properly directed and that particular design or other integration needs may be suitably provided for. During development (and depending upon the size of the system) the plan may be subdivided into a number of child documents and be naturally added to, as the hardware and software designs evolve and the detailed needs of integration become clearer.

12.4.2 The Software/Hardware Integration Test Plan shall document the following:

- i) test cases and test data;
- ii) types of tests to be performed;
- iii) test environment including tools, support software and configuration description; and
- iv) test criteria on which the completion of the test will be judged.

12.4.3 The Software/Hardware Integration Test Plan 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.

12.4.4 The Software/Hardware Integration Test Plan shall distinguish between the following activities:

- i) merging of software onto the target hardware;
- ii) system integration;

12.4.5 Tools and facilities identified in the Software/Hardware Integration Test Plan should be available at the earliest practicable time.

12.4.6 During Software/Hardware Integration any modification or change to the integrated system shall be subject to an impact study which shall identify all modules impacted and the necessary reverification activities.

12.4.7 Test cases and their results shall be recorded, preferably in machine readable form for subsequent analysis.

12.4.8 A Software/Hardware Integration Test Report shall be produced as follows:

- i) Software/Hardware Integration Test Report shall state the test results and whether the objectives and criteria of the Software/Hardware Integration Test Plan have been met. If there is a failure, it shall be recorded;
- ii) the Software/Hardware Integration Test Report shall be in a form that is auditable;
- iii) test cases and their results shall be recorded, preferably in a machine readable form for subsequent analysis;

- iv) the Software/Hardware Integration Test Report shall also contain evidence that the Verifier is satisfied with the adequacy of the Software/Hardware Integration Test Report as a record of the tests carried out in accordance with the Software/Hardware Integration Test Plan;
- v) the Software/Hardware Integration Test Report shall document the identity and configuration of all items involved.

13 Software validation

13.1 Objective

To analyse and test the integrated system to ensure compliance with the Software Requirements Specification with particular emphasis on the functional and safety aspects according to the software safety integrity level.

13.2 Input documents

- 1) Software Requirements Specification.
- 2) All Hardware and Software Documentation.
- 3) System Safety Requirements Specification.

13.3 Output documents

- 1) Software Validation Plan.
- 2) Software Validation Report.

13.4 Requirements

13.4.1 Analysing and testing shall be the main validation activity.

13.4.2 Simulation and modelling may be used to supplement the validation process.

13.4.3 A Software Validation Plan shall be established and detailed in suitable documentation.

13.4.4 The Software Validation Plan shall be developed, performed and results evaluated by an independent party to the extent required by the software safety integrity level.

13.4.5 If required by the software safety integrity level, the scope and contents of the Software Validation Plan shall be agreed with the assessor. This agreement shall also make a statement concerning the presence of the assessor during testing.

13.4.6 The Software Validation Plan shall include a summary justifying the validation strategy chosen. The justification shall include consideration, according to the required software safety integrity level, of:

- i) manual or automated techniques or both;
- ii) static or dynamic techniques or both;
- iii) analytical or statistical techniques or both.

13.4.7 The Software Validation Plan shall identify the steps necessary to demonstrate the adequacy of the:

- Software Requirements Specification;
- Software Architecture Specification;
- Software Design Specification;
- Software Module Design Specification;

in fulfilling the safety requirements set out in the System Safety Requirements Specification. The Validator shall check that the verification process is complete.

13.4.8 Measurement equipment used for validation shall be calibrated appropriately. Any tools, hardware or software, used for validation shall be shown to be suitable for the purpose.

13.4.9 The software shall be exercised by simulation of input signals present during normal operation, anticipated occurrences and undesired conditions requiring system action.

13.4.10 The results of the validation shall be documented in the Software Validation Report in an auditable form.

13.4.11 Once hardware/software integration is finished, a Software Validation Report shall be produced as follows:

- i) it shall state whether the objectives and criteria of the Software Validation Plan have been met. If there is a failure, it shall be recorded;
- ii) it shall state the tests results and whether the whole software on its target machine fulfils the requirements set out in the Software Requirements Specification;
- iii) an evaluation of the test coverage on the requirements of the Software Requirements Specification shall be provided;
- iv) the Software Validation Report shall be in a form that is auditable;
- v) test cases and their results shall be recorded in a machine readable form for subsequent analysis;
- vi) tests should be repeatable and be performed by automatic means, if practicable.

13.4.12 The Software Validation Report shall document the identity and configuration of all:

- i) the hardware and software used;
- ii) the equipment used;
- iii) the equipment's calibration;
- iv) the simulation models used;
- v) the discrepancies found;
- vi) the corrective actions performed.

13.4.13 Any discrepancies found, including detected errors, shall be clearly identified in a separate section of the Software Validation Report and included in any release note which accompanies the delivered software.