

BS EN 60880:2009

Incorporating corrigendum June 2015



BSI Standards Publication

Nuclear power plants — Instrumentation and control systems important to safety — Software aspects for computer- based systems performing category A functions

bsi.

...making excellence a habit.
AVA Documents™
Document Sharing Hub

National foreword

This British Standard is the UK implementation of EN 60880:2009. It is identical to IEC 60880:2006. It supersedes BS IEC 60880:2006 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee NCE/8, Reactor instrumentation.

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

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

© The British Standards Institution 2015.
Published by BSI Standards Limited 2015

ISBN 978 0 580 90531 5

ICS 27.120.20; 35.080

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

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2010.

Amendments/corrigenda issued since publication

Date	Text affected
30 June 2015	Page numbering corrected

**Nuclear power plants -
Instrumentation and control systems important to safety -
Software aspects for computer-based systems
performing category A functions
(IEC 60880:2006)**

Centrales nucléaires de puissance -
Instrumentation et contrôle-commande
importants pour la sûreté -
Aspects logiciels des systèmes
programmés réalisant des fonctions
de catégorie A
(CEI 60880:2006)

Kernkraftwerke -
Leittechnik für Systeme
mit sicherheitstechnischer Bedeutung -
Softwareaspekte für rechnerbasierte
Systeme zur Realisierung
von Funktionen der Kategorie A
(IEC 60880:2006)

This European Standard was approved by CENELEC on 2009-07-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

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

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

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

CENELEC

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

Central Secretariat: Avenue Marnix 17, B - 1000 Brussels

Foreword

The text of the International Standard IEC 60880:2006, prepared by SC 45A, Instrumentation and control of nuclear facilities, of IEC TC 45, Nuclear instrumentation, was submitted to the formal vote and was approved by CENELEC as EN 60880 on 2009-07-01 without any modification.

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) 2010-07-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2012-07-01

CLC/TC 45AX experts draw attention to the readers of this European standard to the fact that it should be read in conjunction with IAEA document INSAG-10, 1996, “Defence in Depth in Nuclear Safety” which applies.

Endorsement notice

The text of the International Standard IEC 60880:2006 was approved by CENELEC as a European Standard without any modification.

Annex ZA (normative)

Normative references to international publications with their corresponding European publications

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

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

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IEC 60671	- ¹⁾	Nuclear power plants - Instrumentation and control systems important to safety - Surveillance testing	-	-
IEC 61069-2	1993	Industrial-process measurement and control - Evaluation of system properties for the purpose of system assessment - Part 2: Assessment methodology	EN 61069-2	1994
IEC 61226	- ¹⁾	Nuclear power plants - Instrumentation and control systems important to safety - Classification of instrumentation and control functions	-	-
IEC 61508-4	- ¹⁾	Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 4: Definitions and abbreviations	EN 61508-4	2001 ²⁾
IEC 61513	- ¹⁾	Nuclear power plants - Instrumentation and control for systems important to safety - General requirements for systems	-	-
ISO/IEC 9126	Series	Software engineering - Product quality	-	-
IAEA guide NS-G-1.2	- ¹⁾	Safety assessment and verification for nuclear power plants	-	-
IAEA guide NS-G-1.3	- ¹⁾	Instrumentation and control systems important to safety in nuclear power plants	-	-

¹⁾ Undated reference.

²⁾ Valid edition at date of issue.

CONTENTS

INTRODUCTION.....	6
1 Scope and object.....	9
2 Normative references	9
3 Terms and definitions	10
4 Symbols and abbreviations.....	15
5 General requirements for software projects	15
5.1 General.....	15
5.2 Software types	17
5.3 Software development approach	18
5.4 Software project management	20
5.5 Software quality assurance plan	20
5.6 Configuration management.....	21
5.7 Software security.....	22
6 Software requirements.....	24
6.1 Specification of software requirements	24
6.2 Self-supervision	25
6.3 Periodic testing	25
6.4 Documentation	26
7 Design and implementation	26
7.1 Principles for design and implementation	27
7.2 Language and associated translators and tools	29
7.3 Detailed recommendations	30
7.4 Documentation	32
8 Software Verification	32
8.1 Software verification process.....	32
8.2 Software verification activities	33
9 Software aspects of system integration.....	37
9.1 Software aspects of system integration plan.....	38
9.2 System integration	39
9.3 Integrated system verification.....	39
9.4 Fault resolution procedures	40
9.5 Software aspects of integrated system verification report	40
10 Software aspects of system validation	41
10.1 Software aspects of the system validation plan.....	41
10.2 System validation	41
10.3 Software aspects of the system validation report.....	42
10.4 Fault resolution procedures	42
11 Software modification	42
11.1 Modification request procedure	43
11.2 Procedure for executing a software modification.....	44
11.3 Software modification after delivery.....	45

12	Software aspects of installation and operation	46
12.1	On-site installation of the software	46
12.2	On-site software security	46
12.3	Adaptation of the software to on-site conditions	47
12.4	Operator training	47
13	Defences against common cause failure due to software	48
13.1	General	48
13.2	Design of software against CCF	49
13.3	Sources and effects of CCF due to software	49
13.4	Implementation of diversity	50
13.5	Balance of drawbacks and benefits connected with the use of diversity	50
14	Software tools for the development of software	50
14.1	General	50
14.2	Selection of tools	51
14.3	Requirements for tools	52
15	Qualification of pre-developed software	57
15.1	General	57
15.2	General requirements	57
15.3	Evaluation and assessment process	58
15.4	Requirements for integration in the system and modification of PDS	66
	Annex A (normative) Software safety life cycle and details of software requirements	67
	Annex B (normative) Detailed requirements and recommendations for design and implementation	69
	Annex C (informative) Example of application oriented software engineering (software development with application-oriented language)	82
	Annex D (informative) Language, translator, linkage editor	86
	Annex E (informative) Software verification and testing	88
	Annex F (informative) Typical list of software documentation	96
	Annex G (informative) Considerations of CCF and diversity	97
	Annex H (informative) Tools for production and checking of specification, design and implementation	101
	Annex I (informative) Requirements concerning pre-developed software (PDS)	104
	Annex J (informative) Correspondence between IEC 61513 and this standard	106

INTRODUCTION

a) Technical background, main issues and organisation of the standard

Engineering of software based Instrumentation and Control (I&C) systems to be used for nuclear safety purposes is a challenge due to the safety requirements to be fulfilled. The safety software used in nuclear power plants (NPP) which are often required only in emergency cases, have to be fully validated and qualified before being used in operation. In order to achieve the high reliability required, special care has to be taken throughout the entire life cycle, from the basic requirements, the various design phases and V&V procedures for operation and maintenance. It is the main aim of this standard to address the related safety aspects and to provide requirements for achieving the high software quality necessary.

The first edition of this standard was issued in 1986 to interpret the basic safety principles applied so far in hardwired systems for the utilisation of digital systems — multiprocessor distributed systems as well as larger scale central processor systems — in the safety systems of nuclear power plants.

It has been used extensively within the nuclear industry to provide requirements and guidance for software of NPP safety I&C systems.

Although many of the requirements within the first edition continued to be relevant, there were significant factors which justified the development of this second edition:

- Since 1986, a number of new standards have been produced which address in detail the general requirements for systems (IEC 61513), hardware requirements (IEC 60987) and a standard to address software for I&C systems performing category B or C functions for NPP systems important to safety (IEC 62138). The Safety Guide 50-SG-D3 of the IAEA has been superseded by the guide NS-G-1.3. Additionally, IEC 60880-2 has been issued.
- Software engineering techniques have advanced significantly in the intervening years.

In this standard, utmost care has been taken to keep transparency with respect to the first edition. Where possible, the phrasing of requirements has been kept, otherwise it has been extended in a traceable way. In the same manner, IEC 60880-2 dealing with software aspects of defence against common cause failures, use of software tools and pre-developed software has been integrated, so that now this current standard covers entirely the software safety issues to be addressed.

It is intended that the standard be used by systems developers, systems purchasers/users (utilities), systems assessors and by licensors.

b) Situation of the current standard in the structure of the SC 45A standard series

IEC 60880 is directly referenced by IEC 61513 which deals with the system aspects of high integrity computer-based I&C used in safety systems of nuclear power plants together.

IEC 60880 is the second level SC 45A document tackling the issue of software aspects for I&C systems performing category A functions.

Software for categories B and C functions is dealt with in IEC 62138.

IEC 60880 and IEC 62138 together cover the domain of the software aspects of computer-based systems used in nuclear power plants to perform functions important to safety.

This second edition of IEC 60880 is to be read in conjunction with IEC 60987 and IEC 61226, the appropriate SC 45A standards on computer hardware and on classification.

For more details on the structure of the SC 45A standard series see item d) of this introduction.

c) Recommendation and limitation regarding the application of this standard

It is important to note that this standard establishes no additional functional requirements for safety systems.

Aspects for which special requirements and recommendations have been produced, are:

- 1) a general approach to software development to assure the production of the highly reliable software required including hardware and software interdependencies;
- 2) a general approach to software verification and to the software aspects of the computer-based system validation;
- 3) procedures for software modification and configuration control;
- 4) requirements for use of tools;
- 5) procedures for qualification of pre-developed software.

It is recognised that software technology is continuing to develop at a rapid pace and that it is not possible for a standard such as this to include references to all modern design technologies and techniques.

To ensure that the standard will continue to be relevant in future years the emphasis has been placed on issues of principle, rather than specific software technologies.

If new techniques are developed then it should be possible to assess the suitability of such techniques by applying the safety principles contained within this standard.

d) Description of the structure of the SC 45A standard series and relationships with other IEC documents and other bodies documents (IAEA, ISO)

The top level document of the SC 45A standard series is IEC 61513. This standard deals with requirements for NPP I&C systems important to safety and lays out the SC 45A standards series.

IEC 61513 refers directly to other SC 45A standards for general topics related to categorization of functions and classification of systems, qualification, separation of systems, defence against common cause failure, software aspects of computer-based systems, hardware aspects of computer-based systems, and control room design. The standards referenced directly at this second level should be considered together with IEC 61513 as a consistent document set.

At a third level, SC 45A standards not directly referenced by IEC 61513 are standards related to specific equipment, technical methods or specific activities. Usually these documents, which make reference to second level documents for general topics, can be used on their own.

A fourth level extending the SC 45A standard series corresponds to the technical reports which are not normative.

IEC 61513 has adopted a presentation format similar to the basic safety publication IEC 61508 with an overall safety life-cycle framework and a system life-cycle framework and provides an interpretation of the general requirements of IEC 61508 parts 1, 2 and 4, for the nuclear application sector. Compliance with this standard will facilitate consistency with the requirements of IEC 61508 as they have been interpreted for the nuclear industry. In this framework, IEC 60880 and IEC 62138 correspond to IEC 61508, part 3 for the nuclear application sector.

IEC 61513 refers to ISO standards as well as to IAEA 50-C-QA for topics related to quality assurance.

The SC 45A standards series consistently implement and detail the principles and basic safety aspects provided in the IAEA Code on the safety of nuclear power plants and in the IAEA safety series, in particular the Requirements NS-R-1, "Safety of Nuclear Power Plants: Design" and the Safety Guide NS-G-1.3, "Instrumentation and control systems important to safety in Nuclear Power Plants". The terminology and definitions used by SC 45A standards are consistent with those used by the IAEA.

NUCLEAR POWER PLANTS – INSTRUMENTATION AND CONTROL SYSTEMS IMPORTANT TO SAFETY – SOFTWARE ASPECTS FOR COMPUTER-BASED SYSTEMS PERFORMING CATEGORY A FUNCTIONS

1 Scope and object

This International Standard provides requirements for the software of computer-based I&C systems of nuclear power plants performing functions of safety category A as defined by IEC 61226.

According to the definition in IEC 61513, I&C systems of safety class 1 are basically intended to support category A functions, but may also support functions of lower categories. However the system requirements are always determined by the functions of the highest category implemented.

For software of I&C system performing only category B and C functions in NPP as defined by IEC 61226, requirements and guidance of IEC 62138 are applicable.

This standard provides requirements for the purpose of achieving highly reliable software. It addresses each stage of software generation and documentation, including requirements specification, design, implementation, verification, validation and operation.

The principles applied in developing these requirements include:

- best available practices;
- top-down design methods;
- modularity;
- verification of each phase;
- clear documentation;
- auditable documents;
- validation testing.

Additional guidance and information on how to comply with the requirements of the main part of this standard is given in Annexes A to I.

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 60671, *Periodic tests and monitoring of the protection system of nuclear reactors*

IEC 61069-2:1993, *Industrial-process measurement and control – Evaluation of system properties for the purpose of system assessment – Part 2: Assessment methodology*

IEC 61226, *Nuclear power plants – Instrumentation and control systems important for safety – Classification of instrumentation and control functions*

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

IEC 61513, *Nuclear power plants – Instrumentation and control for systems important to safety – General requirements for systems*

ISO/IEC 9126, *Software engineering – Product quality*

IAEA guide NS-G-1.2, *Safety Assessment and Verification for Nuclear power Plant*

IAEA guide NS-G-1.3, *Instrumentation and Control Systems Important to Safety in Nuclear Power Plants*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1

animation

process by which the behaviour defined by a specification is displayed with actual values derived from the stated behaviour expressions and from some input values

3.2

application function

function of an I&C system that performs a task related to the process being controlled rather than to the functioning of the system itself

[IEC 61513, 3.1]

3.3

application-oriented language

computer language specifically designed to address a certain type of application and to be used by persons who are specialists of this type of application

[IEC 62138, 3.3]

NOTE 1 Equipment families usually feature application-oriented languages so as to provide easy to use capability for adjusting the equipment to specific requirements.

NOTE 2 Application-oriented languages may be used to specify the functional requirements of an I&C system, and/or to specify or design application software. They may be based on texts, on graphics, or on both.

NOTE 3 Examples: function block diagram languages, language defined by IEC 61131-3.

3.4

application software

part of the software of an I&C system that implements the application functions

[IEC 61513, 3.2]

3.5

automated code generation

function of automated tools allowing transformation of the application-oriented language into a form suitable for compilation or execution

3.6

channel

arrangement of interconnected components within a system that initiates a single output. A channel loses its identity where single output signals are combined with signals from other channels, e.g., from a monitoring channel, or a safety actuation channel.

[IAEA NS-G-1.3, Glossary]

3.7

code compaction

purposeful reduction in memory size required for a program by the elimination of redundant or extraneous instructions

3.8

common cause failure (CCF)

failure of two or more structures, systems or components due to a single specific event or cause

[IAEA NS-G-1.3, Glossary]

3.9

computer

programmable functional unit that consists of one or more associated processing units and peripheral equipment, that is controlled by internally stored programs and that can perform substantial computation, including numerous arithmetic operations or logic operations, without human intervention during a run

[ISO 2382/1]

NOTE A computer may be a standalone unit or may consist of several interconnected units.

3.10

computer program

set of ordered instructions and data that specify operations in a form suitable for execution by a computer

3.11

computer-based system

I&C system whose functions are mostly dependent on, or completely performed by using microprocessors, programmed electronic equipment or computers

[IEC 61513, 3.10]

NOTE Equivalent to: digital systems, software-based system, programmed system.

3.12

data

presentation of information or instructions in a manner suitable for communication, interpretation, or processing by computers

[IEEE 610, modified]

NOTE Data which are required to define parameters and to instantiate application and service functions in the system are called "application data".

3.13

defence in depth

application of more than one protective measure for a given safety objective, such that the objective is achieved even if one of the protective measures fails

[IAEA Safety Glossary]

3.14

diversity

existence of two or more different ways or means of achieving a specified objective. Diversity is specifically provided as a defence against common cause failure. It may be achieved by providing systems that are physically different from each other or by functional diversity, where similar systems achieve the specified objective in different ways.

3.15

dynamic analysis

process of evaluating a system or component based on its behaviour during execution. In contrast to static analysis

[IEEE 610]

3.16

failure

deviation of the delivered service from the intended one

[IEC 61513, 3.21, modified]

3.17

fault

defect in a hardware, software or system component

[IEC 61513, 3.22]

3.18

fault tolerance

built-in capability of a system to provide continued correct execution in the presence of a limited number of hardware or software faults

3.19

functional diversity

application of the diversity at the functional level (for example, to have trip activation on both pressure and temperature limit)

3.20

general-purpose language

computer language designed to address all types of usage

[IEC 62138, 3.17]

NOTE 1 The operational system software of equipment families is usually implemented using general-purpose languages.

NOTE 2 Examples: Ada, C, Pascal.

3.21

human error

human action that produces an unintended result

3.22

initialise

to set counters, switches, addresses, or contents of storage devices to zero or other starting values at the beginning of, or at prescribed points in, the operation of a computer program

3.23

integration tests

tests performed during the hardware/software integration process prior to computer-based system validation to verify compatibility of the software and the computer hardware

3.24

library

collection of related software elements that are grouped together, but which are individually selected for inclusion in the final software product

3.25

N-version software

set of different programs, known as versions, developed to meet a common requirement and common acceptance test. Concurrent and independent execution of these versions takes place, generally in redundant hardware. Identical inputs in test systems or corresponding inputs in redundant systems are used. A predetermined strategy such as voting is used to decide between conflicting outputs in different versions.

3.26

operational system software

software running on the target processor during operation, such as: input/output drivers and services, interrupt management, scheduler, communication drivers, applications oriented libraries, on-line diagnostic, redundancy and graceful degradation management

3.27

postulated initiating event

PIE

events that lead to anticipated operational occurrences or accident conditions and their consequential failure effects

[IEC 61513, 3.41]

3.28

pre-developed software

PDS

software part that already exists, is available as a commercial or proprietary product, and is being considered for use

[IEC 62138, 3.24, modified]

3.29

redundancy

provision of alternative (identical or diverse) structures, systems or components, so that any one can perform the required function regardless of the state of operation or failure of any other

[IAEA NS-G-1.3, Glossary]

3.30

role-based access control

access control based on rules, defining the permitted access of users to objects (functions, data) on the basis of user groups with an identical role instead of individual users

3.31

safety function

specific purpose that must be accomplished for safety

[IAEA NS-R-1, Glossary]

3.32

safety system

system important to safety, provided to ensure the safety shutdown of the reactor or the residual heat removal from the core, or to limit the consequences of anticipated operational occurrences and design basis incidents

[IAEA NS-R-1, Glossary]

3.33

signal trajectory

time histories of all equipment conditions, internal states, input signals, and operator inputs which determine the outputs of a system

3.34

software

programs (i.e. sets of ordered instructions), data, rules and any associated documentation pertaining to the operation of a computer-based I&C system

[IEC 62138, 3.27]

3.35

software development

phase of the software lifecycle that leads to the creation of the software of an I&C system or of a software product. It covers all the activities from software requirements specification to validation and installation on site

[IEC 62138, 3.30]

3.36

software modification

change in an already agreed document (or documents) leading to an alteration of the executable code

NOTE Software modifications may occur either during initial software development (e.g. to remove faults found in later stages of development), or after the software is already in service.

3.37

software safety life cycle

necessary activities involved in the development and operation of the software of an I&C system important to safety occurring during a period of time that starts at a concept phase with the software requirements specification and finishes when the software is withdrawn from use

[IEC 62138, 3.31]

3.38

software version

instance of a software product derived by modification or correction of a preceding software product instance

[IEEE 610, modified]

3.39 specification

document that specifies, in a complete, precise, verifiable manner, the requirements, design behaviour or other characteristics of a system or component and, often, the procedures for determining whether these provisions have been satisfied

[IEEE 610]

NOTE There are different types of specifications, for example software requirements specification or design specification.

3.40 static analysis

process of evaluating a system or component based on its form, structure, content or documentation. In contrast to dynamic analysis

3.41 system software

part of the software of an I&C system designed for a specific computer or equipment family to facilitate the development, operation and modification of these items and associated programs

[IEC 62138, 3.33]

3.42 system validation

confirmation by examination and provision of other evidence that a system fulfils in its entirety the requirement specification as intended (functionality, response time, fault tolerance, robustness)

3.43 verification

confirmation by examination and by provision of objective evidence that the results of an activity meet the objectives and requirements defined for this activity

[IEC 62138, 3.35]

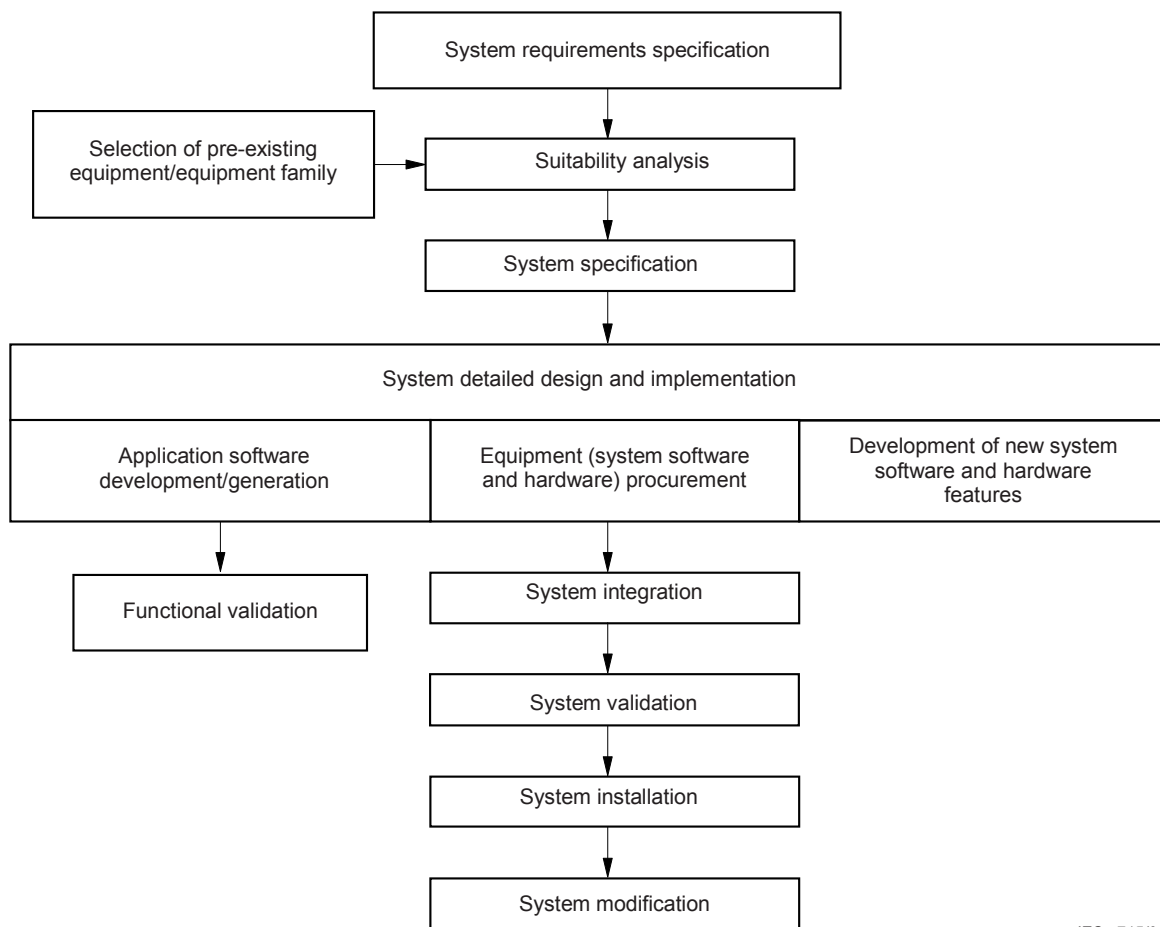
4 Symbols and abbreviations

CASE	Computer Aided Software Engineering
CCF	Common Cause Failure (see 3.8)
I&C	Instrumentation and Control
PDS	Pre-Developed Software (see 3.28)
PIE	Postulated Initiating Event (see 3.27)
QA	Quality Assurance
V&V	Verification and Validation

5 General requirements for software projects

5.1 General

The process of producing instrumentation and control systems for use in nuclear power plants is set down in IEC 61513 that introduces the concept of the system safety lifecycle as a vehicle by which the development process can be controlled and whose adoption should also result in the evidence necessary to justify the operation of safety systems. The system safety lifecycle described in IEC 61513 includes and places requirements on, but does not dictate, the project arrangements to be used for production of systems (see Figure 1).



IEC 715/06

Figure 1 – Activities of the system safety lifecycle (as defined by IEC 61513)

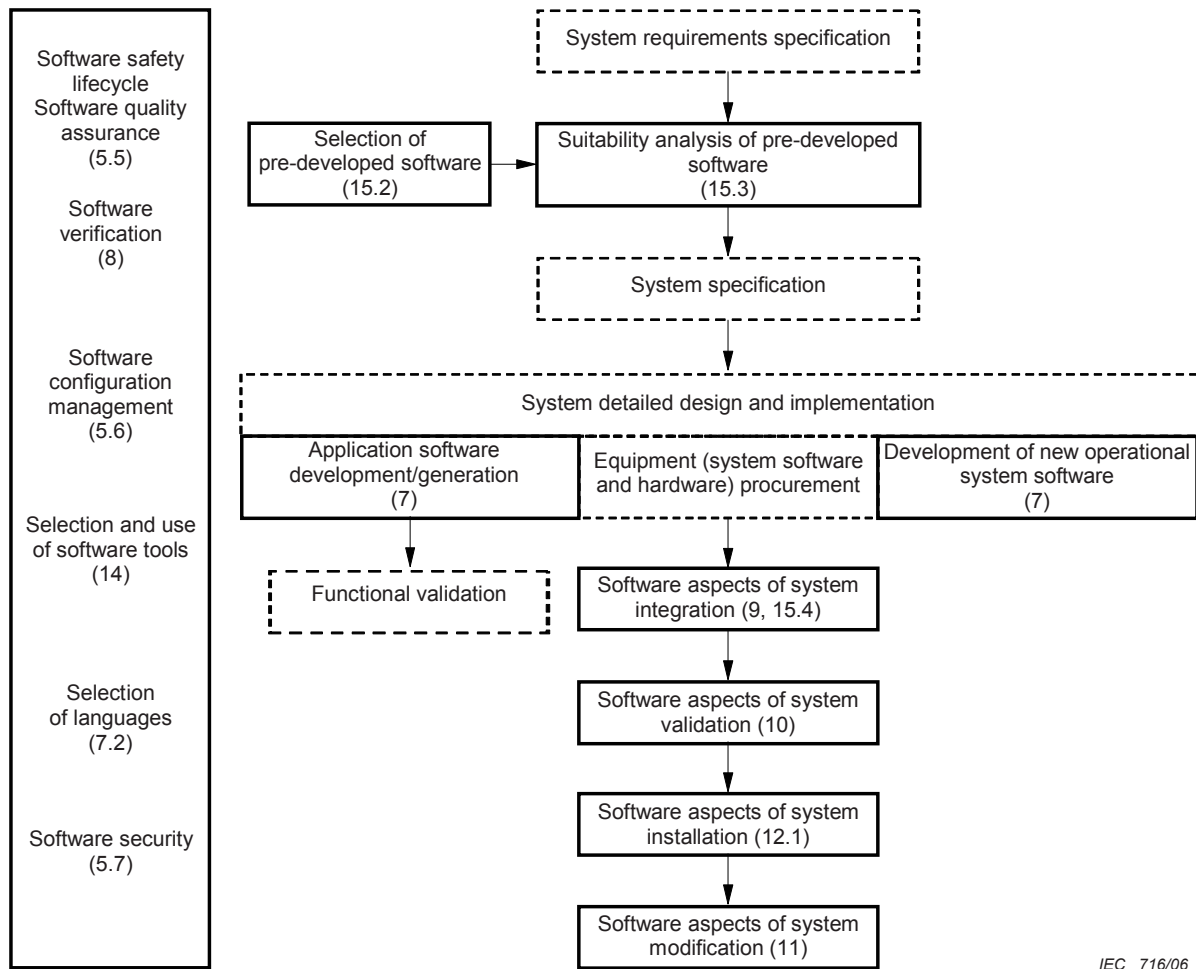
For computer-based (i.e. digital) systems the system safety lifecycle is further developed to introduce the concept of a software safety lifecycle (see Figure 2 for activities). This shows the hardware and software development being undertaken in parallel from a common specification but coming together at the integration and installation phases of the lifecycle.

The following processes support the phased development process of producing the software:

- software project management (5.4);
- software quality assurance and quality control (5.5);
- software configuration management (5.6);
- software security (5.7);
- software verification (Clause 8).

There are also activities involving selection of languages (7.2, Annex D), selection of software tools to support the development (Clause 14), prevention of CCF (Clause 13) and production of documentation (7.4, Annex F).

The resulting software related activities in the system safety lifecycle and the supporting processes are shown below in Figure 2 (boxes in bold lines with reference to related subclauses in brackets).



NOTE Boxes in thin dotted lines represent system activities not addressed in this standard.

Figure 2 – Software related activities in the system safety lifecycle

The approach to software development should be based on the traditional “V” model as this approach has been reflected and promulgated in other standards notably IAEA NS-G-1.3, but allowing necessary adjustments recognizing that some phases of the development can be done automatically by tools and that software development may be iterative.

Subclauses 5.2 and 5.3 introduce the different software types and the development approach considered in this standard.

5.2 Software types

The software components of a system are often defined as being either operational system software (communications, I/O management, standard functions, self-supervision, etc.) or application software (interlock logic, control loops, display formats, alarm logic, etc.).

Application software generally uses the facilities provided by the operational system software, thus reducing the need for duplication of code within modules and thus reducing the overall amount of software.

Application software is usually specific to one project.

Operational system software may be used in different projects.

Many system designs make extensive use of configuration data. Configuration data may be associated with operational system software or with application software. Configuration data associated with application software consists mainly of plant engineering data resulting from the design of the plant, and is often prepared by plant designers who are not required to have software skills.

Configuration data can be divided into:

- data items which are not intended to be modified on-line by plant operators, and which are submitted to the same requirements as apply to the rest of the software;
- parameters, i.e. data items which may be modified by operators during plant operation (for example, alarm limits, set points, data required to calibrate instrumentation) and which need specific requirements.

Many modern I&C equipment platforms are provided with extensive development tools which enable system engineers to design and produce executable codes.

As an example, a typical I&C system, developed using components of an equipment family, includes the following:

- pre-developed software components, such as the operational system software kernel and the application function libraries. Generally, these components have been developed using general-purpose languages;
- configuration data needed to adapt the operational system software kernel to the I/O environment and to the services required by the application;
- application software developed using an application-oriented language.

5.3 Software development approach

Software usually contributes strongly to the functions performed by the I&C system. It may also support additional functions introduced by system design (e.g. initialisation and surveillance of hardware, communication between, and synchronisation of, subsystems). Thus, the software safety lifecycle is in most cases strongly integrated with the system safety lifecycle. In particular, the software requirements specification is a part of, or is derived directly from, system specification and system design.

Though the verification of new software components is definitely a part of the software safety lifecycle, there is often no separate and well-identified boundary between software integration and system integration. Therefore, in this standard, software integration is considered to be a part of system integration. Software validation too is not a purely software activity: in this standard, it is considered a part of system integration and/or system validation.

The standard assumes that the software life cycle, originally intended for the development of software with general-purpose languages, is also applicable to application-oriented languages and configuration of pre-developed software.

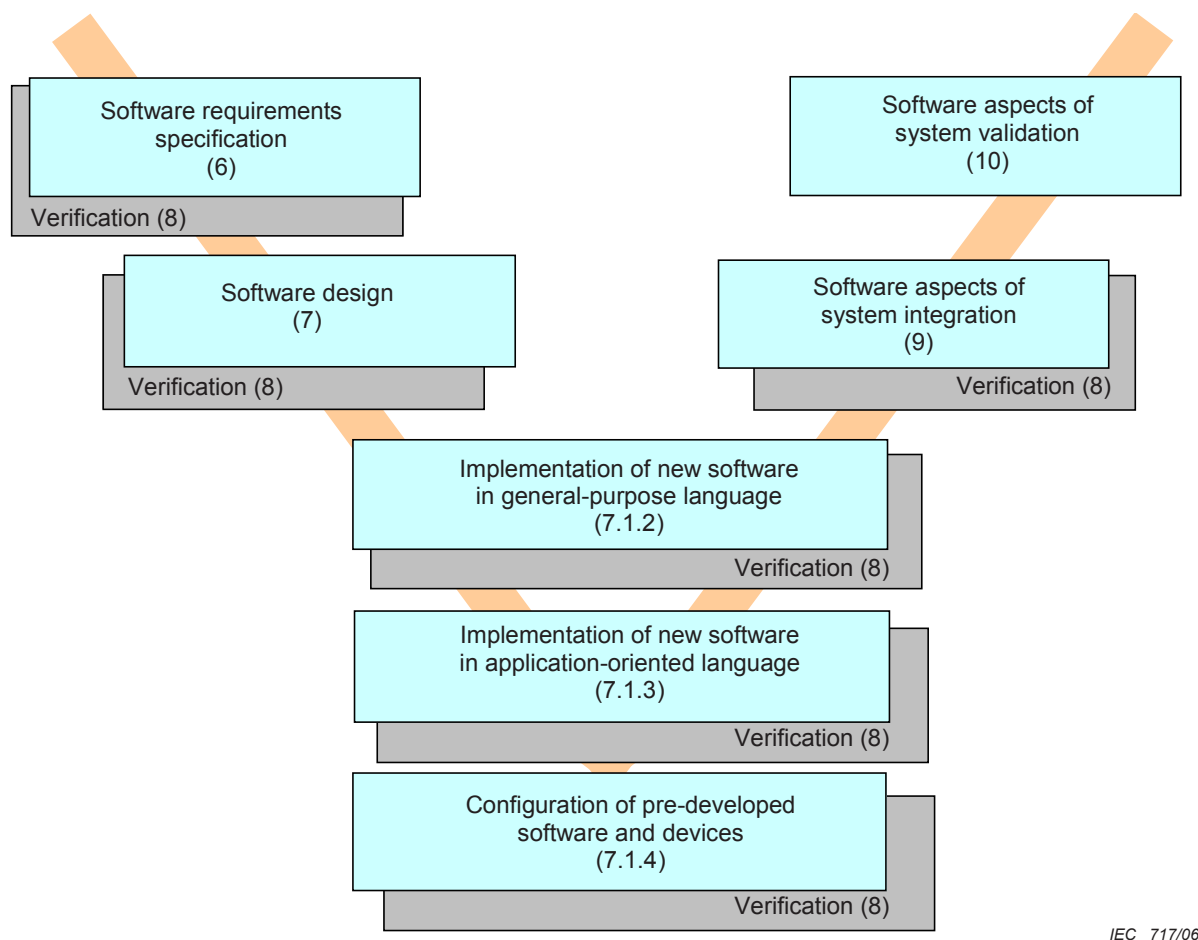
However, it recognises differences in the development process by introducing dedicated sub-processes for each software type at the implementation level:

- implementation using general-purpose languages;
- implementation using application-oriented languages with associated code generators;
- selection, use and configuration of pre-developed software products.

As boxes “Application software development/generation” and “Development of new operational system software” in Figure 2 represent a large and essential part of the software safety lifecycle, a “zoom” is provided in Figure 3, which illustrates in more detail the activities between software requirements specification and validation, with a clear representation of the three different implementation paths. In that figure, references to related subclauses of this standard are in brackets.

The standard also gives additional requirements for software in Annex B.

The principles, reflected in the requirements of this standard, are related to the quality of the final code, and are applicable regardless of whether the code is developed using general-purpose languages, application-oriented languages with automated code generation, or configuration.



IEC 717/06

Figure 3 – Development activities of the IEC 60880 software safety lifecycle

5.4 Software project management

5.4.1 Any software project shall be structured into a number of phases.

Each phase is to some extent self-contained but will depend on other phases and will, in its turn, be depended on by others. These phases are informally recognizable by the specific activities pertinent to them.

The phases and associated activities defined for a software project describe a process called in this standard the "software development". It is recognized that this process may be iterative provided the requirement from the last paragraph of the introduction to Clause 6 of the IEC 61513 is fulfilled.

The following general factors determine the activities and phases in implementing a software project.

5.4.2 The activities performed during the phases of the development process shall be identified, according to the software development approach chosen for the project (see 5.2 and 5.3).

5.4.3 Software development activities shall address the whole software safety lifecycle.

5.4.4 Each phase of the software development identified in 5.4.1 shall be divided into well-defined activities.

5.4.5 The phases of the software development shall be formalised and none of the identified phases shall be omitted.

5.4.6 When activities of the software development are automated using software tools, the activities automated shall be documented including the documentation of the inputs and outputs relevant to the phase.

5.4.7 The inputs and outputs of each phase shall be defined and documented.

5.4.8 Every output of each phase shall be systematically checked (item c) of B.1 and item g) of B.4).

5.4.9 Each phase shall include generation of the appropriate documents (Annex F).

5.4.10 Each phase shall be systematically terminated by a review including examination of relevant documents.

5.4.11 A list of documents required through the software safety life cycle shall be established during the software development. An example of a typical list is given in Annex F.

5.5 Software quality assurance plan

5.5.1 A quality assurance plan shall exist or be established at an early stage of the software safety life cycle.

Special quality assurance plans may be adopted for individual product phases or particular software components according to national or company standards provided the principles defined in this standard are addressed.

5.5.2 Any deviations from the requirements of this standard and its normative annexes shall be identified and justified.

5.5.3 If practices differing from those of normative annexes are used, they shall be documented and auditable according to the requirements of the main parts of this standard.

5.5.4 In particular, the impact of these practices on the I&C system and the software shall be considered.

5.5.5 All technical procedures required during each phase of the software safety life cycle shall be addressed by the quality assurance plan.

5.5.6 The quality assurance plan shall require that the implementation of the activities of the phases is assigned to competent persons equipped with adequate resources.

5.5.7 The quality assurance plan shall require that modifications in already approved documents are identified, reviewed and approved by authorized persons.

5.5.8 The quality assurance plan shall require that the methods, languages, tools, rules and standards used are identified and documented, and known to, and mastered by, the persons concerned.

5.5.9 The quality assurance plan shall require that when several methods, languages, tools, rules and/or standards are used, it is clear which ones were used for each activity.

5.5.10 The quality assurance plan shall require that project specific terms, expressions, abbreviations and conventions are explicitly defined.

5.5.11 The quality assurance plan shall require that any quality issues raised are tracked and resolved.

5.5.12 The quality assurance plan shall require that records resulting from its application are produced.

5.5.13 Every verification step or review shall result in a report on the analysis performed, the conclusions reached and the resolutions agreed. This report shall be included in the documentation.

5.5.14 Any deviation from the quality assurance plan shall be documented and justified.

5.6 Configuration management

Subclause 6.2.1.2 (system configuration management plan) of IEC 61513 provides requirements for configuration management at the I&C system level. This subclause provides additional requirements specific, or of particular importance, to software.

5.6.1 Configuration management for software shall be performed according to the provisions of a configuration management plan or of the quality assurance plan.

5.6.2 These provisions shall be consistent with those for system level configuration management.

5.6.3 Documented software configuration management procedures shall be established early in the lifecycle of a software project to address the following requirements.

5.6.4 Each produced version of each software entity shall be uniquely identified.

5.6.5 It shall be possible to identify the relevant versions of all software documentation associated with each software entity.

5.6.6 Software under development shall be segregated from software which has achieved release or verified status.

5.6.7 It shall be possible to identify the versions of all software entities which together constitute a complete version of the final product.

5.6.8 It shall be possible to verify the integrity of the software entities.

5.6.9 It shall be possible to identify the version of the software in the target system.

5.6.10 It shall be possible to retrospectively identify all software entities affected by the implementation of a modification.

5.6.11 Access to all software entities placed under configuration control shall be subject to adequate controls to ensure that software is not modified by unauthorised persons and that the security of the software is maintained.

5.6.12 It shall be possible to identify all translation tools and tool versions used to produce each executable entity (see 14.3.3).

5.7 Software security

The objective of security is to protect software and data so that unauthorised persons and systems cannot read or modify them and so that authorised persons and systems are not denied access to them.

Subclauses 5.4.2 (overall security plan) and 6.2.2 (system security plan) of IEC 61513 provide requirements for security at a level of the I&C architecture and of an individual I&C system.

Although the use of software does present certain potential security threats, the main countermeasures are usually on system level, for example physical protection measures, hardwired interlocking devices. The main security requirements put on the software may help minimizing the vulnerability and may enable and support the protective measures on the system level.

This subclause provides security requirements specific to or of particular importance to software.

5.7.1 Security analysis

5.7.1.1 An analysis of the potential security threats regarding the software shall be performed. It shall take into account the relevant phases of the system and software safety life cycles. It shall determine the requirements regarding the protection and the accessibility of data and software based upon the requirements of this subclause.

5.7.1.2 The software security analysis shall be taken into account in the software or system quality assurance plan or in the software or system security plan.

5.7.1.3 If the analysis shows that the countermeasures on the system level are not sufficient then the security analysis shall identify requirements for software design countermeasures.

5.7.2 Security design

5.7.2.1 The requirements for design countermeasures in the software identified in the security analysis shall be included in the software design requirements.

5.7.2.2 Any new software should be designed so as to minimize the vulnerability of the system.

5.7.2.3 Any pre-developed software should be configured and parameterised so as to minimize the vulnerability of the system, for example by minimizing the functions to the necessary extent or using existing security functions of the software.

5.7.2.4 The operator shall be prevented from altering stored programs.

5.7.2.5 If operator access is required to change data to operate the I&C functions then the human-machine-interface devices shall restrict access to the necessary extent.

5.7.2.6 Where required to counter possible security threats then effective protection measures shall be included in the design, configuration and/or parameter assignment of the software concerning:

- user selective access control to the software functions;
- data connections to systems with lower safety importance;
- traceability of software or parameter modifications.

5.7.2.7 The design documentation shall identify and describe the functions critical for security and the security features implemented into the software.

5.7.2.8 During verification of the software the effectiveness of the security functions shall be confirmed.

5.7.2.9 During the validation of the I&C system, effectiveness of the security functions implemented in the software shall be demonstrated through suitable tests.

5.7.3 User access

5.7.3.1 Where required the software shall support technical measures for an effective authentication procedure before user access is permitted.

5.7.3.2 Where user access is a feature critical for security that is implemented in the software then the authentication procedure implemented in the software should support technical means for a combination of knowledge (e.g. password), property (e.g. key, smart-card) and/or personal features (e. g. fingerprint) rather than rely solely on a password.

5.7.3.3 The software and the role-based access control shall be configured and parameterised so as to minimize the user access permitted to the functions and data to the necessary extent.

5.7.3.4 User access capabilities shall be protected to an appropriate extent against the opportunities and the consequences of potential security threats.

Cryptographic methods (data encryption) can be one of the possible ways to correctly implement this requirement.

5.7.3.5 No remote access that can influence the software functions or the data from outside the technical environment of the plant (e. g. from the administrative buildings or from outside of the plant) shall be implemented.

5.7.4 Security during development

5.7.4.1 The software safety development lifecycle should address potential security threats during development and maintenance activities.

5.7.4.2 There shall be provisions against hidden functions in the application software or system software (e.g. software code verification) because they could support potential unauthorised access.

5.7.4.3 If provisions could not be implemented for pre-developed software, the use of such software shall be justified considering the potential security threats, the safety importance of I&C functions concerned and the characteristics of the system and software.

5.7.4.4 Potential means for deliberate modification of software that may cause erroneous behaviour triggered by time or data conditions shall be identified and justified to be detectable during the verification activities.

6 Software requirements

6.1 Specification of software requirements

6.1.1 The software requirements shall be derived from requirements of the safety systems and are part of the computer-based system specification.

6.1.2 The software requirements shall describe what the software has to do and not how the software shall do it.

6.1.3 The software requirements shall specify:

- the application functions to be provided by the software;
- the different modes of behaviour of the software, and the corresponding conditions of transition;
- the interfaces and interactions of the software with its environment (e.g. with operators, with the rest of the I&C system, with the other systems, if any, with which it interacts or shares resources), including the roles, types, formats, ranges and constraints of inputs and outputs;
- the parameters of the software which can be modified manually during operation, if any, their roles, types, formats, ranges and constraints, and the checks to be performed by the software when they are modified;
- the required software performance, in particular response time requirements;
- what the software must not do or must avoid, when appropriate;
- the requirements of, or the assumptions made by, the software regarding its environment, when applicable;
- the requirements if any, of standard software packages.

6.1.4 Due to the significance of this phase of software development, the process of laying down software requirements shall be rigorous.

6.1.5 The software requirements specification shall be such that compliance of the I&C system to the requirements of IEC 61513 can be demonstrated.

Additional requirements for the software requirements specification are given in Annex A.

6.1.6 The constraints between hardware and software shall be described (A.2.1).

6.1.7 A reference to the hardware requirements specification shall be made within the software requirements specification for any hardware design impacts.

6.1.8 Special operating conditions such as plant commissioning and refuelling shall be described down to the software level for the functions that are impacted.

6.2 Self-supervision

6.2.1 The software of the computer-based system shall supervise the hardware during operation within specified time intervals and the software behaviour (A.2.2).

This is considered to be a primary factor in achieving high overall system reliability.

6.2.2 Those parts of the memory that contain code or invariable data shall be monitored to detect unintended changes.

6.2.3 The self-supervision should be able to detect to the extent practicable:

- random failure of hardware components;
- erroneous behaviour of software (e.g. deviations from specified software processing and operating conditions or data corruption);
- erroneous data transmission between different processing units.

6.2.4 If a failure is detected by the software during plant operation, the software shall take appropriate and timely response. Those shall be implemented according to the system reactions required by the specification and to IEC 61513 system design rules.

This may require giving due consideration to avoiding spurious actuation.

6.2.5 Self-supervision shall not adversely affect the intended system functions.

6.2.6 It should be possible to automatically collect all useful diagnostic information arising from software self-supervision.

6.3 Periodic testing

6.3.1 For computer-based safety systems, the general principles of IEC 60671 should be used for those components which are not covered adequately by self-supervision.

6.3.2 The software shall be designed so as to meet the requirements of periodic testing which takes place within specified maximum intervals (e.g. shut-down periods).

- 1) every safety function shall be coverable by periodic testing;
- 2) any failure of the safety functions shall be detected.

6.3.3 It should be possible to automatically collect all useful diagnostic information arising from software periodic testing.

6.3.4 The quality of the software of auxiliary devices for testing should correspond to the quality of equipment used for validation as mentioned in 10.2.

The software dedicated to auxiliary devices for testing of the class 1 systems need not be designed to comply with all software safety requirements of this standard.

6.4 Documentation

6.4.1 The main purpose of the software requirements specification document is to form the basis for software development. However, the licensing aspects should not be neglected as this document may be submitted to the regulator. Therefore, it may contain aspects of minor importance to software development which are, however, a background for licensing.

Such important aspects for licensing may be:

- risk considerations;
- recommendations for functions or engineered safety features;
- other items that provide the background for specific requirements;
- special regulatory requirements on software structure, code analysis, V&V, etc.

6.4.2 The software requirements specification shall be presented according to a standard whose formality should not preclude readability (A.2.3).

6.4.3 The software requirements specification shall be unequivocal, testable or verifiable, and achievable.

A formal language or an application-oriented language may be used to improve the coherence and completeness of aspects of the software requirements specification.

Automated tools may be used for this purpose.

6.4.4 The software requirements specification shall be provided to the relevant participants in the engineering process.

7 Design and implementation

This clause presents good practice for developing software with the appropriate safety features, which is as fault-free as possible and which is amenable to verification.

The software requirements specification shall be available before the design and implementation phases of program development begin.

The design and implementation phases implement the software requirements specification and provide the basis for the verification of the design and implementation of the software.

7.1 Principles for design and implementation

7.1.1 General

7.1.1.1 The software design shall include self-supervision (A.2.2).

7.1.1.2 On failure detection, appropriate action shall be taken in accordance with 6.2.

7.1.1.3 The program structure should be based on a decomposition into modules.

7.1.1.4 The program structure should be simple and easy to understand, both in its overall design and in its details.

7.1.1.5 Tricks, recursive structures and code compaction should be avoided.

7.1.1.6 The source program should be understandable by skilled engineers not involved in the software development process.

7.1.1.7 The source program should conform to documented rules designed to improve clarity, modifiability and testability.

7.1.1.8 Any non conformances against design rules should be justified.

7.1.1.9 Comprehensive and clearly written documentation shall be provided.

7.1.1.10 Communication links shall be designed in compliance with the requirements on data communication given in 5.3.1.3 of IEC 61513.

7.1.1.11 Communication links used inside the same redundancy train shall be deterministic.

7.1.1.12 The following recommendations are derived from these principles:

- 1) measures to implement the software safety requirements including self-supervision should be chosen at the beginning of the design (Clause B.3);
- 2) a top down approach to software design should be preferred to a bottom up approach (Clause B.1);
- 3) a conceptual model of the software architecture should be adopted at the beginning of each software project (Clause B.2);
- 4) the program should be written to allow easy verification (Clauses B.4 and B.5);
- 5) where standard software from a manufacturer or supplier is used, the requirements of Clause 15 apply in addition to item c) of Clause B.2);
- 6) application oriented languages should be used in preference to machine oriented ones (item e) of Clause B.5).

7.1.2 Implementation of new software in general-purpose languages

This subclause considers the situation where part or all of the category A safety functions are provided by developing new software components, using general-purpose languages.

General-purpose languages are usually either high level languages such as Ada, C, Pascal or assembly languages dedicated to the platform in use. They can be used to implement any kind of function, provided appropriate design and coding rules are followed.

7.1.2.1 The software design phase shall identify the software components to be developed with general-purpose languages.

7.1.2.2 For these components, the development process should define a detailed design phase and a coding phase.

7.1.2.3 The detailed design phase activities shall refine the design phase outputs to the point where coding with the chosen language can be performed in a systematic way (e.g. by defining the necessary algorithms, data structures, function interfaces, constraints, etc.).

7.1.2.4 The level of detail of the information provided by the detailed design phase depends on the general-purpose language used. If assembly language is used, the design shall provide detailed structured algorithms and data representation.

7.1.2.5 The coding phase shall translate the detailed design into source code, according to predefined programming rules based on the requirements of Annex B.

7.1.3 Implementation of new software in application-oriented languages

This subclause considers the situation where part or all of the category A safety functions are provided by developing new software components, using application-oriented languages.

Application-oriented languages support formalisms (such as logic diagrams or function block diagrams, etc.) that may be used to express all or part of the software requirements specification and/or part of the software design specification. These parts can be used as input to generate executable code by automated means.

7.1.3.1 The formalisms used should have the following properties: low complexity, clarity and standardisation of layout and presentation, modularity, presence of pertinent comments, avoidance of unsafe features. These properties generally facilitate understanding, verification, testing and later modification.

7.1.3.2 Application-oriented languages should have a format that is comprehensible to those engineers responsible for reviewing the software specification, for example the process engineers and the I&C engineers dealing with the systems for which the I&C functions are specified.

7.1.3.3 The application-oriented language should support a simple software structure, for example linear programs.

7.1.3.4 The application-oriented language should allow the developers to take into account the specification of the I&C system architecture design, e.g. enable the assignment of functions to system components and support any hardware fault tolerant design features.

7.1.4 Configuration of pre-developed software

This subclause considers the situation where category A safety functions are provided by software configured with application configuration data, i.e. data specific to the intended application.

Pre-developed software may be associated with an equipment family or may be a standalone product which is then integrated with a chosen hardware platform.

7.1.4.1 Where pre-developed software is to be used, the capabilities of the software shall be evaluated and assessed (see 15.3) to ensure that it is suitable for the intended role.

The requirements for the suitability analysis are addressed in 15.3.1.2. Should the use of the software be restricted, this may be achieved by using software to envelop the pre-developed software.

For the purpose of configuring software, a tool-based approach is preferable to reduce the scope for human error.

7.1.4.2 All constraints relevant to the data shall be documented, for example allowable data formats, ranges, calculation rules.

7.1.4.3 The configuration data shall be documented.

7.1.4.4 Adequate justification shall be provided for data values used with cross-reference to design input sources.

7.1.4.5 Traceability – it should be possible to determine when any modifications were made to configuration data and by whom.

7.1.4.6 Maintainability – the development process shall ensure that through a structured approach together with the use of comments in data and/or supporting documentation the data design may be understood and maintained throughout the intended life of the system it belongs to.

See 14.3.5 for further guidance concerning the use of application data tools.

7.2 Language and associated translators and tools

7.2.1 General requirements

Even though the use of specific languages cannot be required, the following may be considered as common basic rules for languages used for the design and implementation of software for class 1 systems.

7.2.1.1 The language in use shall follow strict (or well-defined) semantic and syntax rules.

7.2.1.2 The syntax of the language shall be completely and clearly defined and documented.

7.2.1.3 The use of the language should be restricted to a 'safe' subset where appropriate, for example be restricted to primitives that are suitable to specify the necessary functions.

7.2.1.4 Languages with a thoroughly tested translator should be used.

7.2.1.5 If no thoroughly tested translator is employed, additional verification shall provide evidence that the result of the translation is correct.

7.2.1.6 Tools for automated testing should be available.

The use of automated tools is recommended. The requirements of Clause 14 apply.

7.2.2 General purpose languages

General purpose languages for class 1 systems and their translators should not prevent the use of error-limiting constructs such as:

- translation-time variable type checking, subroutine parameters checking;
- run-time array bound checking.

Guidance for selection of language, translator, etc., is given in Annex D.

7.2.3 Application-oriented languages and associated automated code generation

7.2.3.1 Application-oriented languages should be transformed into a general purpose language by automated tools (e.g. by a code generator) prior to translation into an executable form.

7.2.3.2 The conformance of the generated code with the requirements for software design and software coding of this standard shall be assessed and non-conformances justified.

7.2.3.3 The program structure to be generated shall be defined generically, for example the position of declarations relative to code statements.

7.2.3.4 The generated code shall not be modified by direct manual action on the code.

7.2.3.5 The code shall be regenerated if the input specification has to be modified, for example with respect to findings from V&V activities.

Additional recommendations on automated code generation are given in item f) of Clause B.5.

7.3 Detailed recommendations

7.3.1 General

A set of recommendations is given in Annex B which specifies in detail the aspects identified in 7.1.

The headings of the individual recommendations of Annex B are applicable to the two major phases of software development, as shown in the following table.

Table 1 – Process and product aspects of design and implementation

Software development phase	Process aspects	Annex B	Product aspects	Annex B
Design	- Modifiability - Top down approach - Verification of intermediate design products - Modification control during the development	B1.a B1.b B1.c B1.d	- Control and access structures - Modules - Operational system software - Execution time - Interrupts - Arithmetic expressions - Plausibility checks - Safe output - Branches and loops - Subroutines - Nested structures - Data structures - Application-oriented languages	B2.a B2.b B2.c B2.d B2.e B2.f B3.a B3.b B4.a B4.b B4.c B4.e B5.e
Implementation	- Verification of intermediate design products - Modification control during the development - Unit and integration tests - Coding rules	B1.c B1.d B4.g B5.d	- Modules - Execution time - Interrupts - Arithmetic expressions - Plausibility checks - Safe output - Memory contents - Error checking - Branches and loops - Subroutines - Nested structures - Addressing and arrays - Data structures - Dynamic changes - Sequences and arrangements - Comments - Assembler - Automated code generation	B2.b B2.d B2.e B2.f B3.a B3.b B3.c B3.d B4.a B4.b B4.c B4.d B4.e B4.f B5.a B5.b B5.c B5.f

7.3.2 Use of the requirements and recommendations

7.3.2.1 The requirements and recommendations given in Annex B shall be met during software development or otherwise justified and documented.

7.3.2.2 The justification should be done at the beginning of the design process.

7.4 Documentation

7.4.1 During software development, the end of the design phase shall be marked by production of the software design specification.

This document serves as the basis for the formal design review and the subsequent program implementation.

7.4.2 Sufficient detail shall be included so that program implementation can proceed without further design clarification.

7.4.3 The document should be structured according to the levels of the software design process.

The software design specification may be expressed as one document or as an integrated set of documents.

7.4.4 In that case, each document shall have a defined relationship to the other documents and contain a well-bounded subject-matter.

7.4.5 Documentation formats should be selected according to the specific purpose, including:

- narrative description;
- arithmetic expressions;
- graphical representation.

7.4.6 Documents should contain appropriate diagrams and drawings.

As a general rule it is preferable to choose a graphical representation.

7.4.7 The documentation itself should comply with national standards if appropriate.

8 Software verification

8.1 Software verification process

The verification activities undertaken as part of the software development are usually the responsibility of the supplier and are undertaken by staff independent of those performing the software production; the most appropriate way is to engage a verification team.

Additional verification activities may be undertaken as part of a third party assessment of the software and of its development process in order to provide assurance that the software will meet its quality targets. There are many ways by which this independent verification role can be resourced and implemented, this often being a matter of national regulatory preference.

8.1.1 The verification team shall be composed of individuals who are not engaged in production and who have the necessary competencies and knowledge.

The following requirements define explicitly the level of independence required.

8.1.2 The management of the verification team shall be separate and independent from the management of the development team.

8.1.3 Communication between the verification team and the development team, whether for clarification or fault reporting, shall be conducted formally in writing at a level of detail which may be audited.

8.1.4 Interactions between the two parties should aim at maintaining the independence of judgment of the verification team.

8.1.5 The verification team shall be equipped with adequate resources and means. It shall be given the time necessary to perform the verification activities.

8.1.6 The verification team shall have clearly defined responsibilities and obligations.

8.1.7 The verification team shall have the necessary authority to report its conclusions.

8.1.8 The output of each software development phase (Figure 3) shall be verified.

8.1.9 The software verification activities shall confirm the adequacy of the software requirements specification in fulfilling the system requirements assigned to the software by the system specification.

8.1.10 The software verification activities shall confirm the adequacy of the software design specification in fulfilling the software requirements.

8.1.11 The software verification activities shall confirm the compliance of the code to the software design specification as derived by the design phase. When a CASE tool with features such as automated code generation is used, dedicated requirements are given in 8.2.3.2.

8.1.12 Each production activity should be started on a basis of verified input data/documents.

8.1.13 When undertaken as part of the software development, verification of the product of a phase should be performed before the start of the next phase and shall be performed before the completion (i.e. its verification) of the next phase.

Possible preparatory work for a subsequent phase may be done before the precedent phase has been verified and approved.

8.1.14 If input data/documents for an activity have been modified, that activity and subsequent activities shall be repeated as necessary to address potential impact.

8.1.15 All software verifications shall be completed before the system is placed into active service.

8.2 Software verification activities

The following verification activities are required.

8.2.1 Verification plan

8.2.1.1 The software verification plan shall be established prior to starting software verification activities.