NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
60880

Deuxième édition
Second edition
2006-05

Centrales nucléaires de puissance – Instrumentation et contrôle-commande importants pour la sûreté – Aspects logiciels des systèmes programmés réalisant des fonctions de catégorie A

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

Numéro de référence
Reference number
CEI/IEC 60880:2006

Numérotation des publications

Depuis le 1er janvier 1997, les publications de la CEI sont numérotées à partir de 60000. Ainsi, la CEI 34-1 devient la CEI 60034-1.

Editions consolidées

Les versions consolidées de certaines publications de la CEI incorporant les amendements sont disponibles. Par exemple, les numéros d'édition 1.0, 1.1 et 1.2 indiquent respectivement la publication de base, la publication de base incorporant l'amendement 1, et la publication de base incorporant les amendements 1 et 2.

Informations supplémentaires sur les publications de la CEI

Le contenu technique des publications de la CEI est constamment revu par la CEI afin qu'il reflète l'état actuel de la technique. Des renseignements relatifs à cette publication, y compris sa validité, sont disponibles dans le Catalogue des publications de la CEI (voir ci-dessous) en plus des nouvelles éditions, amendements et corrigenda. Des informations sur les sujets à l'étude et l'avancement des travaux entrepris par le comité d'études qui a élaboré cette publication, ainsi que la liste des publications parues, sont également disponibles par l'intermédiaire de:

- **Site web de la CEI (www.iec.ch)**
- **Catalogue des publications de la CEI**

  Le catalogue en ligne sur le site web de la CEI (www.iec.ch/searchpub) vous permet de faire des recherches en utilisant de nombreux critères, comprenant des recherches textuelles, par comité d'études ou date de publication. Des informations en ligne sont également disponibles sur les nouvelles publications, les publications remplacées ou retirées, ainsi que sur les corrigenda.
- **IEC Just Published**

  Ce résumé des dernières publications parues (www.iec.ch/online_news/justpub) est aussi disponible par courrier électronique. Veuillez prendre contact avec le Service client (voir ci-dessous) pour plus d'informations.
- **Service clients**

  Si vous avez des questions au sujet de cette publication ou avez besoin de renseignements supplémentaires, prenez contact avec le Service clients:

  Email: custserv@iec.ch
  Tél: +41 22 919 02 11
  Fax: +41 22 919 03 00

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**
- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (www.iec.ch/searchpub) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.
- **IEC Just Published**

  This summary of recently issued publications (www.iec.ch/online_news/justpub) is also available by email. Please contact the Customer Service Centre (see below) for further information.
- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

  Email: custserv@iec.ch
  Tel: +41 22 919 02 11
  Fax: +41 22 919 03 00

.

NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
60880

Deuxième édition
Second edition
2006-05

Centrales nucléaires de puissance– Instrumentation et contrôle-commande importants pour la sûreté – Aspects logiciels des systèmes programmés réalisant des fonctions de catégorie A

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

International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

CODE PRIX
PRICE CODE XE

Pour prix, voir catalogue en vigueur
For price, see current catalogue

SOMMAIRE

CONTENTS

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

CENTRALES NUCLÉAIRES DE PUISSANCE – INSTRUMENTATION ET CONTRÔLE-COMMANDE IMPORTANTS POUR LA SÛRETÉ – ASPECTS LOGICIELS DES SYSTÈMES PROGRAMMÉS RÉALISANT DES FONCTIONS DE CATÉGORIE A

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI n'a prévu aucune procédure de marquage valant indication d'approbation et n'engage pas sa responsabilité pour les équipements déclarés conformes à une de ses Publications.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de propriété intellectuelle ou de droits analogues. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de propriété et de ne pas avoir signalé leur existence.

La Norme internationale CEI 60880 a été établie par le sous-comité 45A: Instrumentation et contrôle-commande des installations nucléaires, du comité d'études 45 de la CEI: Instrumentation nucléaire.

Cette deuxième édition annule et remplace la première édition publiée en 1986 ainsi que la CEI 60880-2 publiée en 2000. Elle constitue une révision technique.

L'objectif de la révision de la norme est de:

- Prendre en compte le fait que les techniques de génie logiciel ont progressé de façon significative ces dernières années.
- Mettre en cohérence la norme avec les nouvelles révisions des documents de l'AIEA NS-R-1 et NS-G-1.3, ceci comprenant autant que possible une adaptation des définitions.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

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

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 60880 has been prepared by subcommittee 45A: Instrumentation and control of nuclear facilities, of IEC technical committee 45: Nuclear instrumentation.

This second edition cancels and replaces the first edition published in 1986 and IEC 60880-2 published in 2000. It constitutes a technical revision.

The revision of the standard is intended to accomplish the following:

- To take into account the fact that software engineering techniques advanced significantly in the intervening years.
- To align the standard with the new revisions of IAEA documents NS-R-1 and NS-G-1.3. This includes as far as possible adaptation of the definitions.

- Remplacer, autant que possible, les exigences associées aux normes publiées depuis la parution de la première édition de la CEI 60880, plus particulièrement la CEI 61513, la CEI 61226 édition 2, la CEI 62138 et la CEI 60987.
- Intégrer complètement au niveau des chapitres 13, 14, 15 et des Annexes G, H, I, la CEI 60880-2 publiée en 2000.
- Faire la revue des exigences existantes et mettre à jour les définitions et la terminologie.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
45A/613/FDIS	45A/621/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de maintenance indiquée sur le site web de la CEI sous «http://webstore.iec.ch» dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

- To replace, as far as possible, requirements associated with standards published since the first edition of IEC 60880, especially IEC 61513, IEC 61226 edition 2, IEC 62138 and IEC 60987.
- To fully integrate IEC 60880-2 published in 2000 as chapters 13, 14, 15 and annexes G, H, I.
- To review the existing requirements and to update the terminology and definitions.

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

FDIS	Report on voting
45A/613/FDIS	45A/621/RVD

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

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

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

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

INTRODUCTION

a) Contexte technique, questions importantes et structure du document

Le développement des logiciels des systèmes de contrôle-commande numériques employés à des fins de sûreté nucléaire est un défi du fait des exigences de sûreté à satisfaire. Les logiciels de sûreté employés dans les centrales nucléaires, qui, souvent, ne sont sollicités qu'en cas d'urgence doivent être totalement validés et qualifiés avant leur mise en exploitation. Afin d'atteindre le haut niveau de fiabilité requis, une attention particulière doit être apportée durant tout le cycle de vie, depuis la spécification des exigences de base jusqu'à l'exploitation et la maintenance, en passant par les différentes étapes de conception et de V&V. L'objectif principal de cette norme est de traiter des différents aspects de sûreté correspondants et d'énoncer les exigences permettant d'atteindre le haut niveau de qualité logicielle nécessaire.

La première édition de cette norme, publiée en 1986, avait été développée pour interpréter les principes de sûreté de base jusqu'alors applicables aux systèmes câblés, pour les systèmes numériques – systèmes multiprocesseurs répartis, grands systèmes mono-processeurs – employés dans les systèmes de sûreté des centrales nucléaires.

Elle fut largement utilisée par l'industrie nucléai re pour fournir des exigences et des recom-mandations applicables aux logiciels des systèmes de contrôle commande des centrales nucléaires.

Bien que la plupart des exigences de cette première édition soient toujours pertinentes, des éléments significatifs ont justifié le développement de cette seconde édition:

– Depuis 1986 un certain nombre de nouvelles normes ont été produites; celles-ci traitent en détail des exigences générales portant sur les systèmes (CEI 61513) et des exigences relatives au matériel (CEI 60987). Une norme traite du logiciel des systèmes de contrôle-commande réalisant des fonctions de catégories B ou C pour les systèmes importants pour la sûreté des centrales nucléaire (CEI 62138). Le Guide de sûreté de l'AIEA 50-SG-D3 a été remplacé par le guide NS-G-1.3. Enfin, la CEI 60880-2 a été diffusée.
– Les techniques de génie logiciel ont progressé de façon significative au cours des dernières années.

Lors du développement de cette norme, le plus grand soin a été apporté au maintien de la transparence par rapport à la première édition. Lorsque cela a été possible, la formulation des exigences a été conservée, sinon elle a été modifiée tout en maintenant la traçabilité. De la même façon, la CEI 60880-2 traitant des aspects logiciels relatifs à la défense contre les défaillances de cause commune, à l'utilisation des d'outils logiciels et de logiciels prédéveloppés, a été intégrée de telle façon que la présente norme couvre aujourd'hui la totalité des sujets considérés.

L'objectif de cette norme est d'être utilisée par les développeurs de systèmes, les acheteurs et les utilisateurs de systèmes (exploitants), les évaluateurs de systèmes et autorités réglementaires.

b) Position de la présente norme dans la collection de normes du SC 45A

La CEI 61513, qui traite des systèmes de contrôle-commande numériques de haute intégrité employés dans les systèmes de sûreté des centrales nucléaires de puissance fait référence à la CEI 60880.

La CEI 60880 est le document de deuxième niveau qui traite les aspects logiciels des systèmes de contrôle-commande réalisant des fonctions de catégorie A.

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.

Le logiciel relatif aux fonctions de catégories B et C est traité dans la CEI 62138.

Prises ensemble, la CEI 60880 et la CEI 62138 couvrent les aspects logiciels relatifs aux systèmes numériques employés dans les centrales nucléaires de puissance pour réaliser les fonctions importantes pour la sûreté.

Cette seconde édition de la CEI 60880 doit être lue conjointement avec la CEI 60987 et la CEI 61226, qui sont les normes du SC 45A traitant des aspects matériels et du classement des systèmes.

Pour plus de détails sur la collection de normes du SC 45A, voir le point d) de cette introduction.

c) Recommandations et limites relatives à l'application de cette norme

Il est important de noter que cette norme n'établit pas d'exigences fonctionnelles supplémentaires pour les systèmes de sûreté.

Les aspects pour lesquels des exigences et des recommandations particulières ont été produites sont:

1) une approche générale du développement logiciel pour garantir une production de logiciel hautement fiable prenant en compte les interdépendances entre le matériel et le logiciel;
2) une approche générale pour la vérification du logiciel et pour les aspects logiciels de la validation du système programmé;
3) des procédures pour le contrôle des modifications et des configurations du logiciel;
4) des exigences applicables à l'utilisation des outils;
5) des procédures pour la qualification des logiciels prédéveloppés.

Il est reconnu que les techniques logicielles se développent de façon continue à un rythme soutenu et qu'il n'est pas possible, pour une norme, de faire référence à toutes les techniques et technologies nouvelles de conception.

Pour garantir la pertinence de la norme pour les années futures, l'accent a été mis sur les principes, plutôt que sur les techniques logicielles.

Si de nouvelles techniques sont développées, l'application des principes devrait permettre d'en apprécier l'utilité.

d) Description de la structure de la collection des normes du SC 45A et relations avec d'autres documents de la CEI et d'autres organisations (AIEA, ISO)

Le document de niveau supérieur de la collection de normes produites par le SC 45A est la CEI 61513. Cette norme traite des exigences relatives aux systèmes et équipements d'instrumentation et de contrôle-commande (systèmes d'I&C) utilisés pour accomplir les fonctions importantes pour la sûreté des centrales nucléaires, et structure la collection de normes du SC 45A.

La CEI 61513 fait directement référence aux autres normes du SC 45A traitant de sujets génériques, tels que la catégorisation des fonctions et le classement des systèmes, la qualification, la séparation des systèmes, les défaillances de cause commune, les aspects logiciels et les aspects matériels relatifs aux systèmes programmés, et la conception des salles de commande. Il convient de considérer que ces normes, de second niveau, forment, avec la CEI 61513, un ensemble documentaire cohérent.

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.

Au troisième niveau, les normes du SC 45A, qui ne sont généralement pas référencées directement par la CEI 61513, sont relatives à des matériels particuliers, à des méthodes ou à des activités spécifiques. Généralement ces documents, qui font référence aux documents de deuxième niveau pour les activités génériques, peuvent être utilisés de façon isolée.

Un quatrième niveau qui est une extension de la collection de normes du SC 45A correspond aux rapports techniques qui ne sont pas des documents normatifs.

La CEI 61513 a adopté une présentation similaire à celle de la CEI 61508, avec un cycle de vie et de sûreté global, un cycle de vie et de sûreté des systèmes, et une interprétation des exigences générales des parties 1, 2 et 4 de la CEI 61508 pour le secteur nucléaire. La conformité à la CEI 61513 facilite la compatibilité avec les exigences de la CEI 61508 telles qu'elles ont été interprétées dans l'industrie nucléaire. Dans ce cadre, la CEI 60880 et la CEI 62138 correspondent, pour l'application au secteur nucléaire, à la CEI 61508-3.

La CEI 61513 fait référence aux normes ISO ainsi qu'au document AIEA 50-C-QA pour ce qui concerne l'assurance qualité.

Les normes produites par le SC 45A sont élaborées de façon à être en accord avec les principes de sûreté fondamentaux du Code AIEA sur la sûreté des centrales nucléaires, ainsi qu'avec les guides de sûreté de l'AIEA, en particulier le guide NS-R-1 «Safety of Nuclear Power Plants: Design – Requirements» et le guide NS-G-1.3 «Instrumentation and Control Systems Important to Safety in Nuclear Power Plants». La terminologie et les définitions utilisées dans les normes produites par le SC 45A sont conformes à celles utilisées par l'AIEA.

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.

CENTRALES NUCLÉAIRES DE PUISSANCE – INSTRUMENTATION ET CONTRÔLE-COMMANDE IMPORTANTS POUR LA SÛRETÉ – ASPECTS LOGICIELS DES SYSTÈMES PROGRAMMÉS RÉALISANT DES FONCTIONS DE CATÉGORIE A

1 Domaine d'application et objet

Cette Norme internationale énonce des exigences pour les logiciels des systèmes d'instrumentation et de contrôle-commande (I&C) programmés des centrales nucléaires de puissance, réalisant des fonctions de catégorie A telle que définie par la CEI 61226.

Selon la définition donnée par la CEI 61513, les systèmes d'I&C de sûreté de classe 1 sont, à la base, destinés aux fonctions de catégorie A, mais peuvent également réaliser des fonctions de catégories inférieures. Cependant, les exigences des systèmes sont toujours déterminées par les fonctions de la plus haute catégorie concernée.

Pour les logiciels des systèmes d'I&C réalisant uniquement des fonctions de catégories B et C, telles que définies par la CEI 61226, les exigences et recommandations de la CEI 62138 sont applicables.

La présente norme énonce des exigences pour la production de logiciels de haute fiabilité. Elle prend en compte chaque étape de développement et de documentation du logiciel, c'est-à-dire la spécification des exigences, la conception, le développement, la vérification, la validation et l'exploitation.

Les principes appliqués pour développer ces exigences comprennent:

– l'utilisation des meilleures pratiques existantes;
– l'utilisation de méthodes de conception descendante;
– la modularité;
– la vérification de chaque phase;
– la clarté de la documentation;
– la vérifiabilité des documents;
– la réalisation de tests de validation.

Des recommandations et informations supplémentaires sur la façon de se conformer aux exigences de la partie principale de cette norme sont données dans les Annexes A à I.

2 Références normatives

Les documents référencés ci-après sont indispensables à l'application de cette norme. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document référencé (y compris les amendements) s'applique.

CEI 60671, *Essais périodiques et surveillance du système de protection des réacteurs nucléaires*

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*

CEI 61069-2:1993, *Mesure et commande dans les processus industriels – Appréciation des propriétés d'un système en vue de son évaluation – Partie 2: Méthodologie à appliquer pour l'évaluation*

CEI 61226, *Centrales nucléaires de puissance – Systèmes d'instrumentation et de contrôle-commande importants pour la sûreté – Classement des fonctions d'instrumentation et de contrôle-commande*

CEI 61508-4, *Sécurité fonctionnelle des systèmes électriques/électroniques/programmables relatifs à la sécurité – Partie 4: Définitions et abréviations*

CEI 61513, *Centrales nucléaires – Instrumentation et contrôle-commande des systèmes importants pour la sûreté – Exigences générales pour les systèmes*

ISO/CEI 9126, *Génie logiciel – Qualité du produit*

Guide AIEA NS-G-1.2, *Evaluation de sûreté et vérification pour les centrales nucléaires*

Guide AIEA NS-G-1.3, *Instrumentation et systèmes de commande importants pour la sûreté dans les centrales nucléaires*

3 Termes et définitions

Pour les besoins du présent document, les termes et définitions suivants s'appliquent.

3.1
animation
processus par lequel le comportement défini par une spécification est visualisé avec ses valeurs effectives dérivées des équations de comportement et des valeurs d'entrée

3.2
fonction d'application
fonction d'un système d'I&C qui accomplit une tâche relative au processus sous contrôle plutôt qu'au fonctionnement du système lui-même

[CEI 61513, 3.1]

3.3
langage orienté application
langage informatique spécifiquement conçu pour un certain type d'application et pour être utilisé par les spécialistes de ce type d'application

[CEI 62138, 3.3]

NOTE 1 Les familles d'équipements offrent en général des langages orientés application de façon à faciliter l'adaptation des équipements à des exigences particulières.

NOTE 2 Les langages orientés application peuvent être utilisés pour la spécification d'exigences fonctionnelles que doit satisfaire un système d'I&C, et/ou pour spécifier ou concevoir le logiciel d'application. Ils peuvent être basés sur du texte, des diagrammes ou une combinaison des deux.

NOTE 3 Exemples: les langages à blocs fonctionnels, les langages définis par la CEI 61131-3.

3.4
logiciel d'application
partie du logiciel d'un système d'I&C qui exécute des fonctions d'application

[CEI 61513, 3.2]

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
génération automatique de code
fonction d'outils automatiques permettant une transformation du langage orienté application en un langage de programmation généraliste apte à une compilation en format exécutable

3.6
canal
ensemble de composants interconnectés dans un système générant un signal de sortie unique. Un canal perd son identité lorsque plusieurs signaux de ce type sont combinés avec des signaux provenant d'autres canaux, tels que les canaux de surveillance ou les canaux des actionneurs de sûreté

[AIEA NS-G-1.3, Glossaire]

3.7
compactage de code
réduction voulue de l'espace mémoire nécessaire à un programme, obtenue par élimination des instructions redondantes ou sans rapport avec ledit programme

3.8
défaillance de cause commune (CCF)
défaillance de deux ou plusieurs structures, systèmes ou composants due à un seul évènement ou cause spécifique

[AIEA NS-G-1.3, Glossaire]

3.9
ordinateur
unité fonctionnelle programmable se composant d'une ou plusieurs unités de traitement associées et de périphériques, qui est commandée par des programmes rangés en mémoire interne et qui peut effectuer des traitements importants, comportant de nombreuses opérations arithmétiques ou logiques, sans intervention humaine pendant l'exécution

[ISO 2382/1]

NOTE Un ordinateur peut être composé d'une unité isolée ou de plusieurs unités interconnectées.

3.10
programme (d'ordinateur)
ensemble ordonné d'instructions et de données qui spécifie des opérations sous une forme adaptée à l'exécution par un ordinateur

3.11
système informatique
système d'I&C dont les fonctions dépendent en grande partie ou sont totalement effectuées à l'aide de microprocesseurs, d'un matériel électronique programmé ou d'ordinateurs

[CEI 61513, 3.10]

NOTE Équivalent à: système numérique, système programmé.

3.12
données
représentation d'informations ou d'instructions adaptée à la communication, l'interprétation, ou le traitement au moyen d'un ordinateur

[IEEE 610, modifiée]

NOTE Les données qui sont nécessaires pour définir des paramètres et initier des fonctions d'application et de service dans le système, sont appelées «données d'application».

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
défense en profondeur
mise en œuvre de plusieurs mesures de protections en vue d'un même objectif de sûreté, de façon à atteindre l'objectif même en cas d'échec d'une des mesures

[AIEA Safety Glossary]

3.14
diversité
existence de deux ou plusieurs manières différentes d'atteindre un objectif donné. La diversité est en particulier utilisée comme moyen de défense contre une défaillance de cause commune. Elle peut être réalisée par la mise en œuvre de systèmes physiquement différents les uns des autres, ou par une diversité fonctionnelle dans laquelle des systèmes similaires réalisent l'objectif spécifié de manière différente

3.15
analyse dynamique
processus consistant à évaluer un système ou un composant sur la base de son comportement pendant l'exécution. S'oppose à l'analyse statique

[IEEE 610]

3.16
défaillance
déviation du service délivré par rapport au service attendu

[CEI 61513, 3.21, modifiée]

3.17
défaut
imperfection dans un composant matériel, logiciel ou système

[CEI 61513, 3.22]

3.18
tolérance aux fautes
capacité intrinsèque d'un système de fonctionner correctement de manière continue en présence d'un nombre limité de défauts matériels ou logiciels

3.19
diversité fonctionnelle
application de la diversité au niveau fonctionnel (par exemple le déclenchement d'une action de protection sur seuil de pression ou sur seuil de température)

3.20
langage généraliste
langage informatique conçu pour s'adresser à tout type de besoin

[CEI 62138, 3.17]

NOTE 1 Le logiciel système d'une famille d'équipements est en général réalisé à l'aide de langages généralistes.

NOTE 2 Exemples: Ada, C, Pascal.

3.21
erreur humaine
action humaine conduisant à un résultat indésirable

3.22
initialiser
forcer les compteurs, clés, adresses ou contenus des dispositifs de stockage à zéro ou à des valeurs prédéterminées au début ou en certains points prescrits du fonctionnement d'un programme

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
tests d'intégration
tests effectués lors du processus d'intégration du matériel et du logiciel avant la phase de validation du système programmé, pour vérifier la compatibilité du logiciel et du matériel du système programmé

3.24
bibliothèque
ensemble d'éléments logiciels connexes contenus dans un fichier unique mais sélectionnés individuellement pour inclusion dans le produit logiciel final

3.25
logiciel N-versions
ensemble de programmes différents, appelés versions, développés pour répondre à des exigences identiques et aux mêmes tests d'acceptation. L'exécution simultanée et indépendante de ces versions a lieu en général sur des matériels redondants. On utilise des entrées identiques dans les systèmes de test ou des entrées correspondantes dans des systèmes redondants. Une stratégie prédéterminée telle que le vote est utilisée pour choisir en cas de sorties contradictoires générées par les différentes versions

3.26
logiciel système opérationnel
logiciel fonctionnant sur le processeur cible en exploitation, comme des pilotes et services d'entrée/sortie, la gestion des interruptions, la gestion de l'ordonnancement, des pilotes de communication, des bibliothèques orientées application, le diagnostic en ligne, la redondance et la gestion des modes de fonctionnement dégradé

3.27
événement initiateur hypothétique
PIE
événement engendrant des incidents opérationnels prévus ou des situations accidentelles, ainsi que les effets de défaillances produites

[CEI 61513, 3.41]

3.28
logiciel prédéveloppé (PDS)
logiciel qui existe déjà, qui peut ou non être un produit commercial, et dont l'utilisation est envisagée

[CEI 62138, 3.24, modifiée]

3.29
redondance
présence de structures, de systèmes ou de composants supplémentaires (identiques ou différents), tels que chacun d'entre eux puisse remplir la fonction requise quel que soit l'état de fonctionnement ou de défaillance des autres

[AIEA NS-G-1.3, Glossaire]

3.30
contrôle d'accès «basé fonction»
contrôle d'accès basé sur des règles définissant les droits d'accès des utilisateurs aux objets (fonctions, données) sur la base de groupes d'utilisateurs assurant une fonction identique et non pas sur la base d'utilisateurs individuels

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
fonction de sûreté
objectif particulier à satisfaire pour la sûreté

[AIEA NS-R-1, Glossaire]

3.32
système de sûreté
système important pour la sûreté assurant l'arrêt sûr du réacteur, ou l'évacuation de la chaleur résiduelle du cœur, ou la limitation des conséquences des incidents opérationnels prévus et des incidents de dimensionnement

[AIEA NS-R-1, Glossaire]

3.33
trajectoire de signal
historique de toutes les conditions des équipements, des états internes, des signaux d'entrée, et des entrées opérateur qui déterminent les valeurs de sortie d'un système

3.34
logiciel
programmes (ensembles ordonnés d'instructions), données, règles et toute documentation associée relatifs au fonctionnement d'un système informatique

[CEI 62138, 3.27]

3.35
développement logiciel
phase du cycle de vie du logiciel conduisant à la création du logiciel d'un système d'I&C ou d'un produit logiciel. Cette phase couvre toutes les activités, depuis la spécification d'exigences jusqu'à la validation et l'installation sur site

[CEI 62138, 3.30]

3.36
modification du logiciel
changement dans un document ou des documents déjà approuvés conduisant à un changement dans le code exécutable

NOTE Une modification du logiciel peut être effectuée durant le développement initial (par exemple pour éliminer des défauts mis en évidence dans les phases finales du développement) ou après la mise en service du logiciel.

3.37
cycle de vie et de sûreté du logiciel
activités nécessaires au développement et à l'exploitation du logiciel d'un système d'I&C important pour la sûreté. Elles couvrent la période allant de la spécification des exigences sur le logiciel au retrait de service du logiciel

[CEI 62138, 3.31]

3.38
version de logiciel
nouvelle édition d'un produit logiciel consécutive à une modification ou à une correction d'un produit logiciel précédent

[IEEE 610, modifiée]

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
spécification
document qui spécifie de manière complète, précise et vérifiable les exigences, le comportement de la conception ou autres caractéristiques d'un système ou composant et, souvent, les procédures permettant de déterminer si ces dispositions ont été satisfaites

[IEEE 610]

NOTE Il existe différents types de spécifications, par exemple les spécifications d'exigence logicielle ou les spécifications de conception.

3.40
analyse statique
processus d'évaluation d'un système ou d'un composant basé sur sa forme, sa structure, son contenu ou sa documentation. S'oppose à l'analyse dynamique

3.41
logiciel système
partie du logiciel d'un système d'I&C, d'un équipement ou d'une famille d'équipements conçue pour faciliter le développement, l'exploitation et la modification de ces systèmes et des programmes associés

[CEI 62138, 3.33]

3.42
validation du système
confirmation par examen et apport d'autres éléments justificatifs qu'un système satisfait à la totalité des exigences spécifiées (fonctionnalités, temps de réponse, tolérance aux fautes, robustesse)

3.43
vérification
confirmation par examen et apport d'éléments objectifs que les résultats d'une activité sont conformes aux objectifs et exigences établis pour cette activité

[CEI 62138, 3.35]

4 Symboles et abréviations

CASE	Atelier de Génie Logiciel
CCF	Défaillance de Cause Commune (voir 3.8)
I&C	Instrumentation et Contrôle-Commande
PDS	Logiciel Prédéveloppé (voir 3.28)
PIE	Événement Initiateur Hypothétique (voir 3.27)
QA	Assurance Qualité
V&V	Vérification et Validation

5 Exigences générales pour les projets logiciel

5.1 Généralités

Le processus de production des systèmes d'instrumentation et de contrôle-commande destinés aux centrales nucléaires est décrit dans la CEI 61513 qui introduit la notion de cycle de vie et de sûreté du système comme moyen d'assurer la maîtrise du processus de développement et comme cadre permettant d'apporter les justifications nécessaires à la mise en service de ces systèmes. Le cycle de vie et de sûreté du système décrit par la CEI 61513 inclut et énonce des exigences sur les dispositions nécessaires à la production des systèmes, mais n'impose pas de disposition particulière (voir Figure 1).

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).

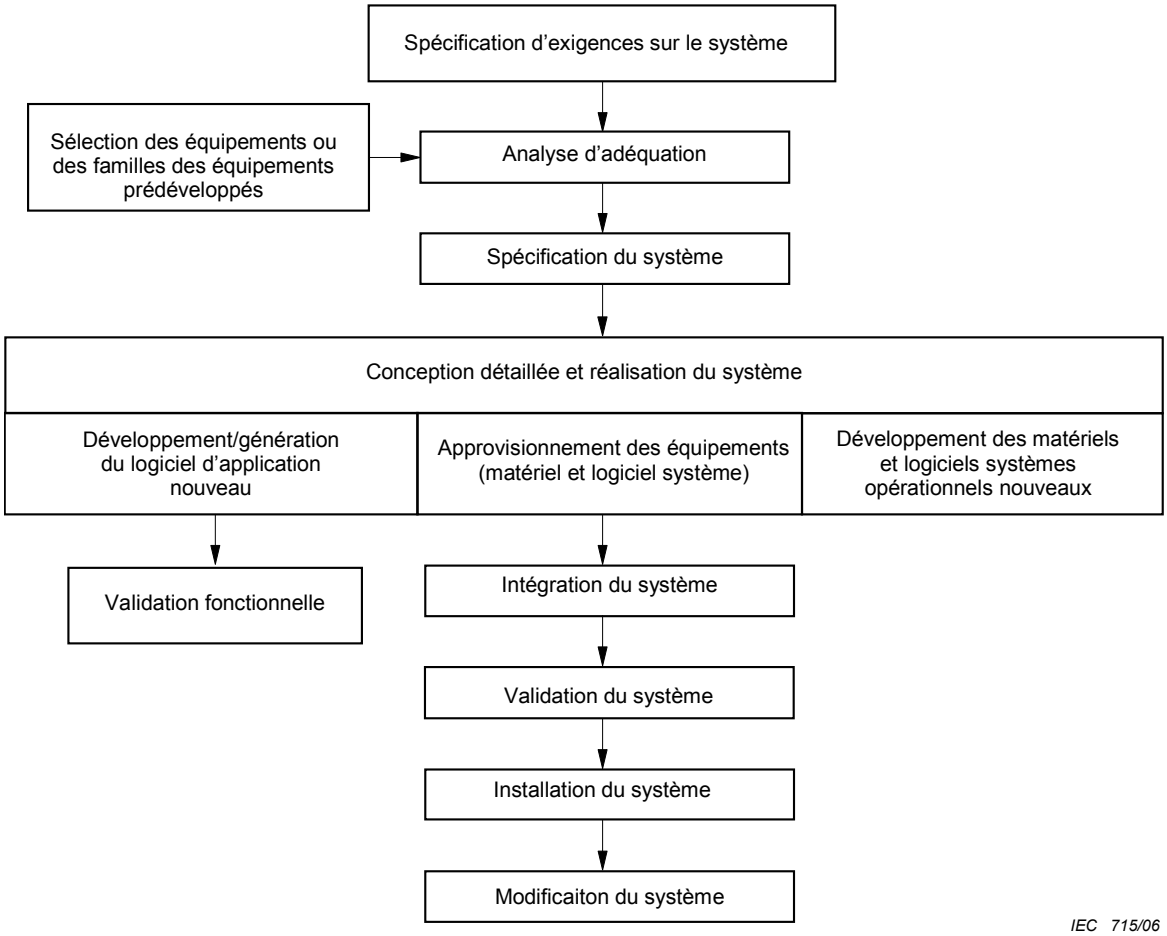


Figure 1 – Activités du cycle de vie et de sûreté du système (telles que définies par la CEI 61513)

Pour les systèmes programmés (c'est-à-dire numériques), le cycle de vie et de sûreté du système est précisé pour introduire le concept de cycle de vie et de sûreté du logiciel (voir Figure 2 pour les activités). Il montre des développements matériel et logiciel pouvant être entrepris en parallèle à partir d'une spécification commune et se rejoignant aux phases d'intégration et d'installation.

Les processus suivants complètent le processus de développement en phases du logiciel:

– la gestion du projet logiciel (5.4);
– l'assurance qualité et le contrôle de la qualité du logiciel (5.5);
– la gestion de configuration du logiciel (5.6);
– la sécurité du logiciel (5.7);
– la vérification du logiciel (Article 8).

Il y a également des activités correspondant à la sélection de langages (7.2, Annexe D), la sélection des outils logiciels contribuant au développement (Article 14), la prévention des CCF (Article 13) et la production de la documentation (7.4, Annexe F).

Les activités relatives au logiciel dans le cycle de vie et de sûreté du système et les processus complémentaires sont décrits ci-dessous dans la Figure 2 (cases encadrées en gras avec référence aux paragraphes donnée entre parenthèses).

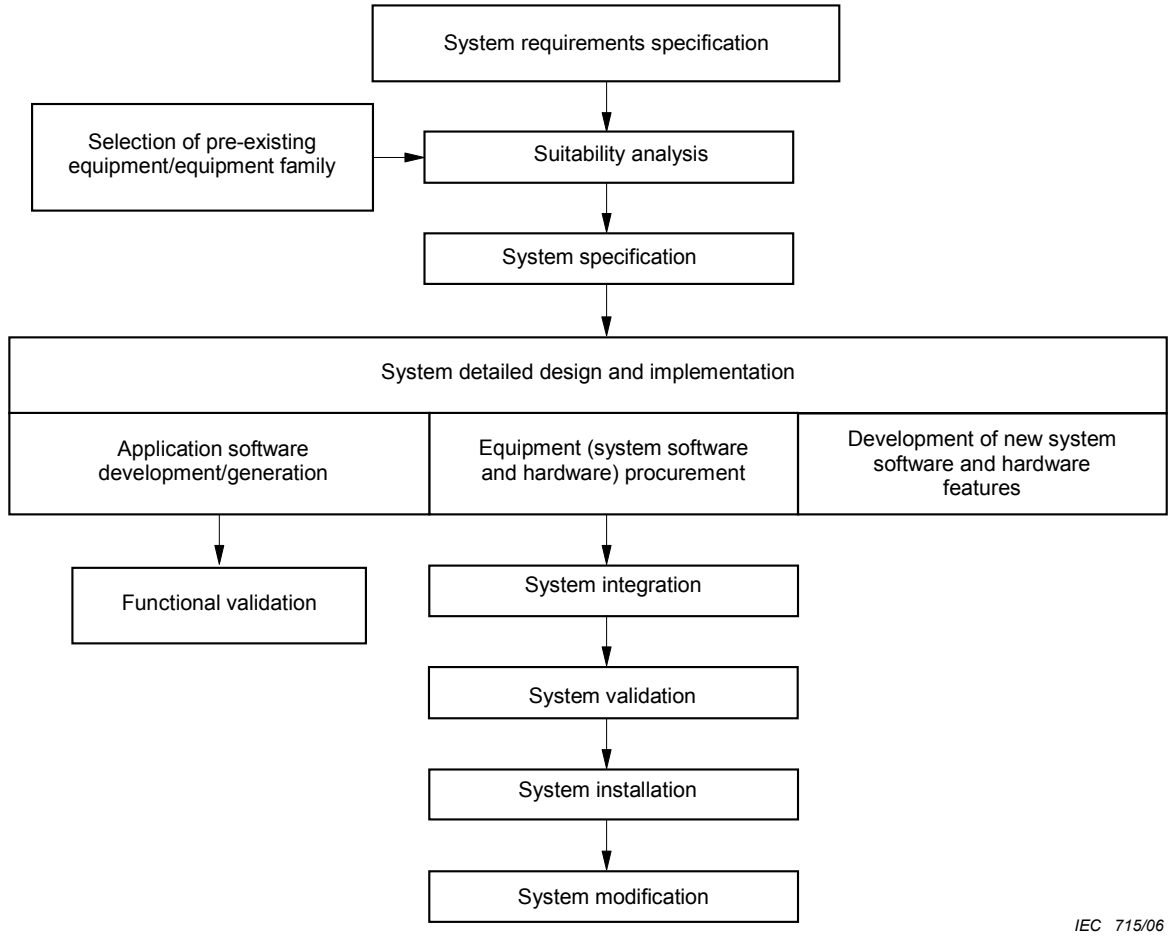


Figure 1 – Activities of the system safety lifecycle (as defined by IEC 61513)

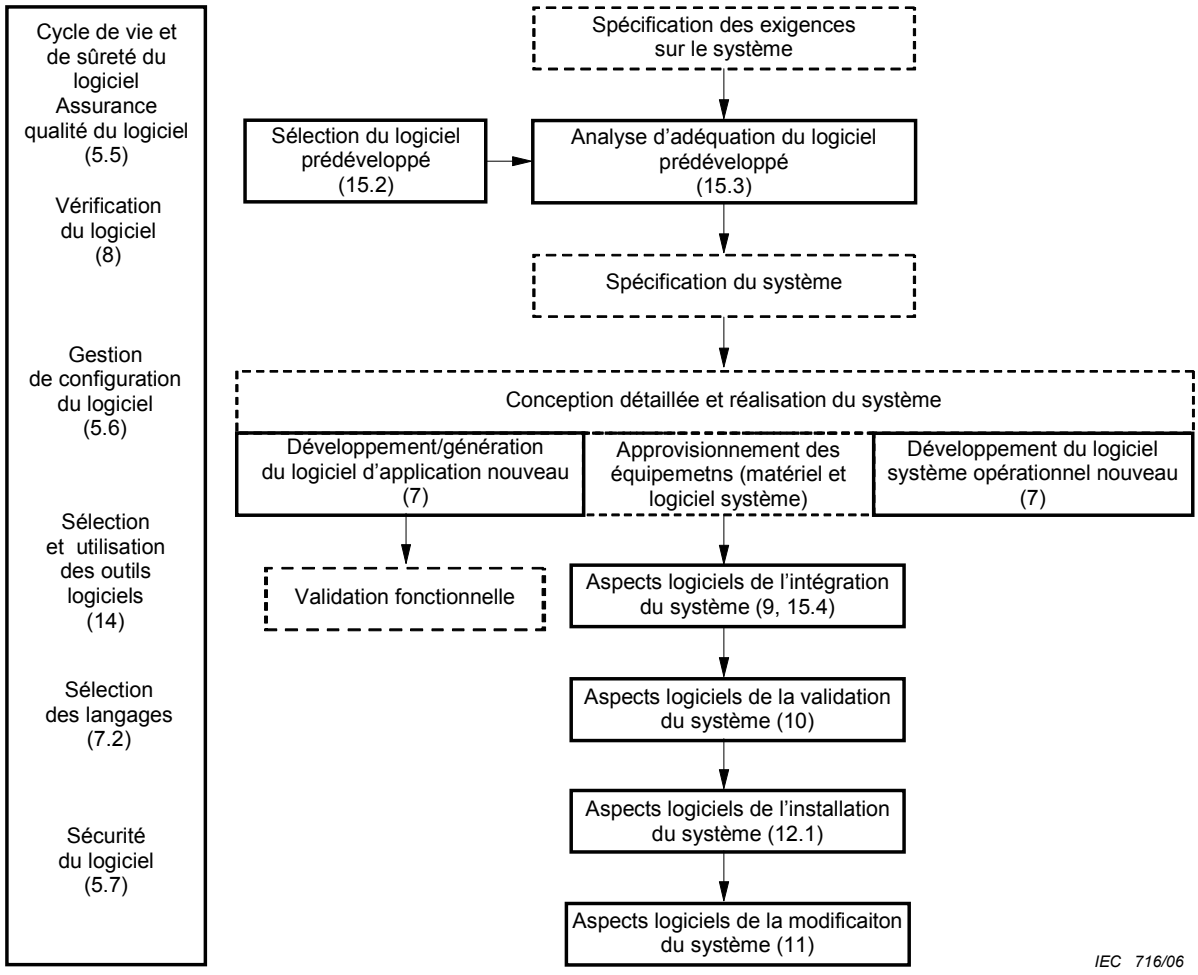
For computer-based (i.e. digital) systems the system safety lifecycle is further developed to introduce the concept of a software safety lifecycle (see Figure 2 for activities). This shows the hardware and software development being undertaken in parallel from a common specification but coming together 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 Les cases en pointillé représentent les activités du cycle de vie et de sûreté du système non traitées dans cette norme.

Figure 2 – Activités liées au logiciel, dans le cycle de vie et de sûreté du système

Il convient que le principe de développement du logiciel repose sur le modèle en «V» traditionnel, adopté par d'autres normes, notamment par le guide de sûreté AIEA NS-G-1.3, avec cependant les ajustements nécessaires lorsque certaines phases du développement sont faites automatiquement à l'aide d'outils ou lorsque le développement du logiciel suit un processus itératif.

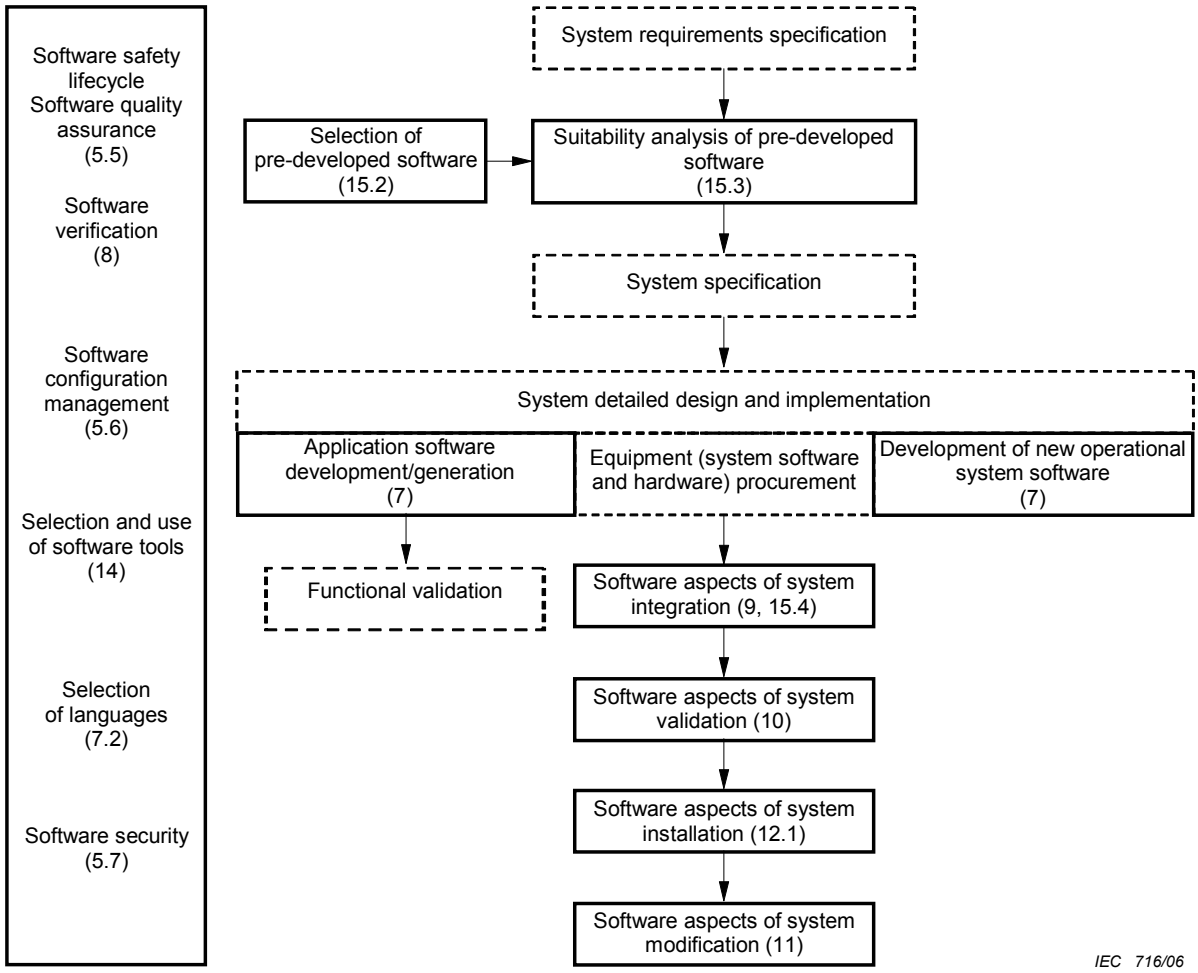
Les paragraphes 5.2 et 5.3 introduisent les différents types de logiciel et le principe de développement considérés dans cette norme.

5.2 Types de logiciel

Un composant logiciel d'un système programmé est souvent caractérisé comme étant soit du logiciel système opérationnel (gestion des communications, gestion des E/S, fonctions standards, fonctions d'auto-surveillance, etc.), soit du logiciel d'application (logique de verrouillage, boucles de commande, formats d'affichage, logique d'alarme, etc.).

Le logiciel d'application utilise généralement les fonctions fournies par le logiciel système opérationnel, réduisant ainsi le besoin de duplication de code et par conséquent la taille globale du logiciel.

Le logiciel d'application est généralement spécifique à un projet.



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.

Le logiciel système opérationnel peut être réutilisé dans différents projets.

De nombreuses conceptions de système font un usage important des données de configuration. Les données de configuration peuvent être associées au logiciel système opérationnel ou au logiciel d'application. Les données de configuration associées au logiciel d'application consistent principalement en données d'ingénierie résultant de la conception de l'installation, et sont souvent préparées par des concepteurs de centrale n'ayant pas nécessairement de compétences en logiciel.

Les données de configuration peuvent être divisées en:

– données qui ne sont pas destinées à être modifiées en ligne par des exploitants, et qui sont soumises aux mêmes exigences que le reste du logiciel,
– paramètres, c'est-à-dire des données pouvant être modifiées par des exploitants pendant le fonctionnement de l'installation (par exemple, des seuils d'alarme, des points de réglage, des données d'étalonnage de l'instrumentation), et qui nécessitent des exigences spécifiques.

De nombreuses familles d'équipements d'I&C modernes sont fournies avec des outils de développement complets qui permettent aux développeurs de systèmes de concevoir et produire des codes exécutables.

Par exemple, un système d'I&C typique développé en utilisant des composants d'une famille d'équipements comprend:

– des composants logiciels prédéveloppés, tels que le logiciel système opérationnel et les bibliothèques de fonctions d'application. Ces composants sont généralement développés avec des langages généralistes,
– les données de configuration nécessaires à l'adaptation du logiciel système opérationnel aux entrées-sorties et aux services exigés par l'application,
– le logiciel d'application, développé en langage orienté application.

5.3 Principe de développement du logiciel

Le logiciel contribue en général fortement aux fonctions réalisées par le système d'I&C. Il peut aussi contribuer à des fonctions supplémentaires introduites par la conception du système (par exemple l'initialisation et la surveillance du matériel, la communication entre et la synchronisation des sous-systèmes). C'est pourquoi le cycle de vie et de sûreté du logiciel est, dans la plupart des cas, fortement intégré au cycle de vie et de sûreté du système. En particulier, la spécification des exigences du logiciel fait partie, ou découle directement des spécifications du système et de sa conception.

Alors que la vérification des composants logiciels nouveaux fait clairement partie du cycle de vie et de sûreté du logiciel, il n'y a souvent pas de frontière nette entre l'intégration du logiciel et l'intégration du système. Par conséquent, dans cette norme, l'intégration du logiciel est considérée comme faisant partie de l'intégration du système. La validation du logiciel n'est pas non plus une activité purement logicielle: dans cette norme, elle est considérée comme faisant partie de l'intégration du système et/ou de la validation du système.

La norme considère que le cycle de vie du logiciel, conçu à l'origine pour des développements en langages généralistes, est également applicable aux développements en langages orientés application et à la configuration de logiciels prédéveloppés.

Cependant, des différences sont admises dans le processus de développement, par l'introduction, au niveau de la réalisation, de sous-processus dédiés pour chaque type de logiciel:

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:

– réalisation avec des langages généralistes;
– réalisation avec des langages orientés application, et des générateurs de code associés;
– sélection, utilisation et configuration de produits logiciels prédéveloppés.

Comme les encadrés «Développement/Génération du logiciel d'application nouveau» et «Développement du logiciel système opérationnel nouveau» de la Figure 2 représentent une part importante et essentielle du cycle de vie et de sûreté du logiciel, un «zoom» est fourni en Figure 3, illustrant avec plus de détails les activités entre la spécification des exigences du logiciel et la validation du logiciel, avec une représentation claire des trois différentes voies de réalisation. Dans cette figure, les références aux articles et paragraphes de cette norme sont entre parenthèses.

L'Annexe B de cette norme donne également des exigences supplémentaires pour le logiciel.

Les principes sous-tendus par les exigences de cette norme sont en relation avec la qualité du code final et sont applicables indépendamment du fait que le code soit développé avec des langages généralistes, avec des langages orientés application et une génération de code automatisée, ou par configuration de produits logiciels.

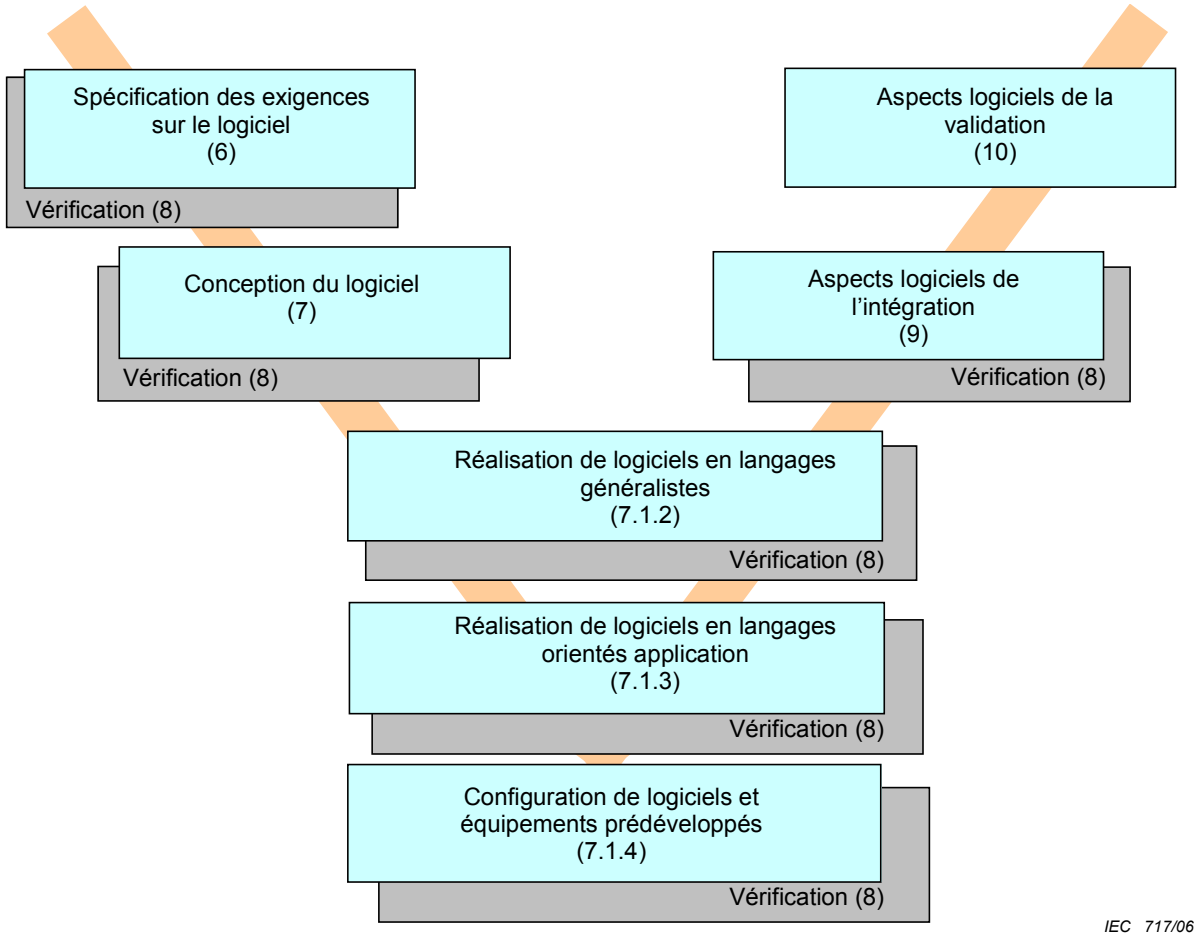


Figure 3 – Activités de développement dans le cycle de vie et de sûreté du logiciel pour la CEI 60880

– 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.

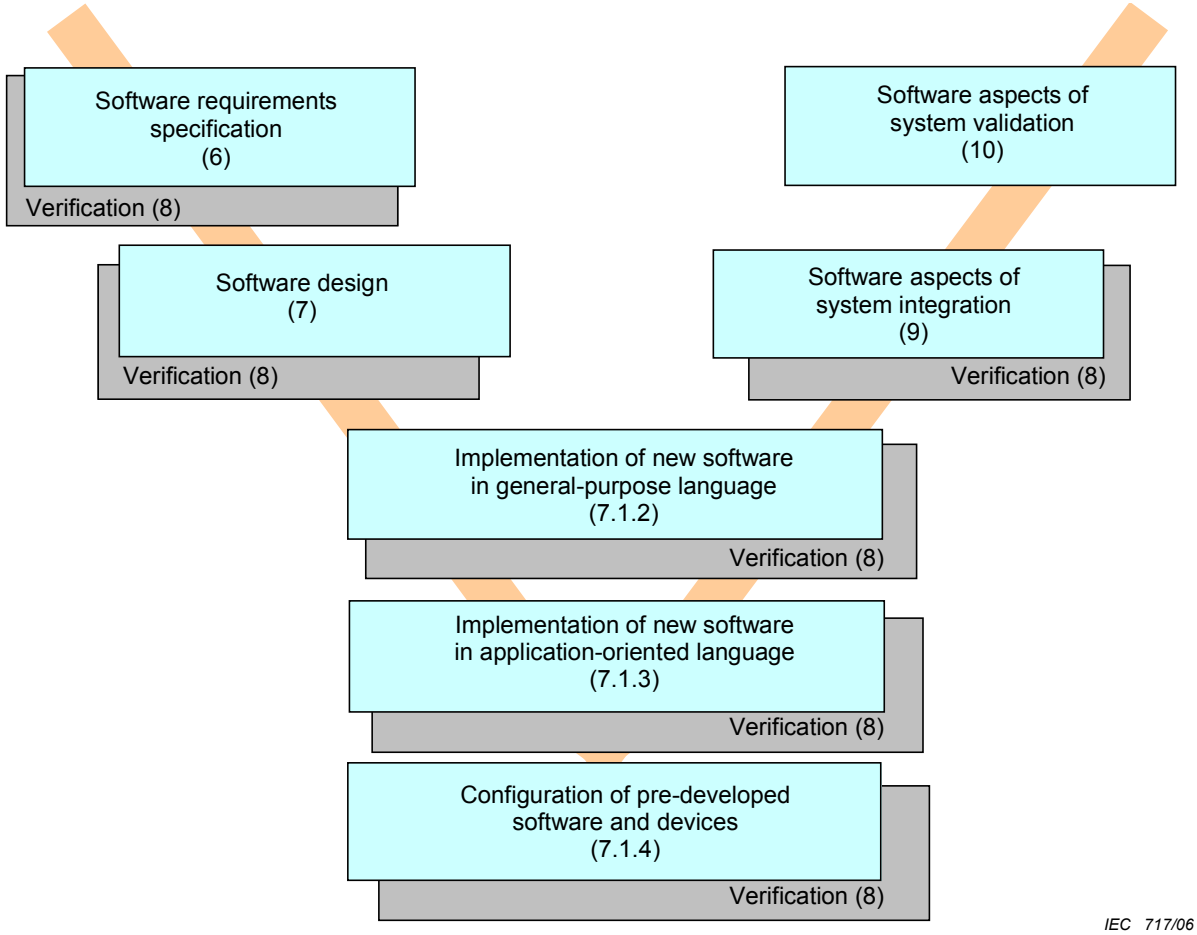


Figure 3 – Development activities of the IEC 60880 software safety lifecycle

5.4 Gestion du projet logiciel

5.4.1 Tout projet logiciel doit être découpé en phases.

Chaque phase constitue une entité en soi, mais elle est dépendante de certaines phases, tandis que d'autres phases en dépendent. Ces phases sont caractérisées de manière informelle par les activités spécifiques qui s'y rapportent.

Les phases et activités associées définies pour un projet logiciel décrivent un processus appelé «développement logiciel» dans cette norme. Il est reconnu que ce processus peut être itératif sous réserve que l'exigence du dernier paragraphe d'introduction de l'Article 6 de la CEI 61513 soit satisfaite.

Les facteurs généraux suivants déterminent les activités et phases d'un projet logiciel.

5.4.2 Les activités réalisées lors du développement logiciel doivent être identifiées en accord avec l'approche choisie pour le projet (voir 5.2 et 5.3).

5.4.3 Les activités de développement du logiciel doivent concerner l'ensemble du cycle de vie et de sûreté du logiciel.

5.4.4 Chaque phase du développement logiciel identifiée en 5.4.1 doit être divisée en activités bien définies.

5.4.5 Les phases du développement logiciel doivent être formalisées, et aucune des phases identifiées ne doit être omise.

5.4.6 Quand des activités du développement logiciel sont automatisées à l'aide d'outils logiciels, les activités automatisées doivent être documentées, en explicitant en particulier les entrées et les sorties significatives pour la phase concernée.

5.4.7 Les entrées et les sorties de chaque phase doivent être définies et explicitées.

5.4.8 Les sorties de chaque phase doivent être systématiquement vérifiées (point c) de B.1 et point g) de B.4).

5.4.9 Chaque phase doit inclure la constitution d'une documentation appropriée (Annexe F).

5.4.10 Chaque phase doit systématiquement se terminer par une revue incluant l'examen des documents correspondants.

5.4.11 Une liste des documents requis durant le cycle de vie et de sûreté du logiciel doit être établie pendant le développement du logiciel. Un exemple de liste type est donné en annexe F.

5.5 Plan d'assurance qualité logiciel

5.5.1 Un plan d'assurance qualité doit exister ou être établi au plus tôt dans le cycle de vie et de sûreté du logiciel.

Des plans d'assurance qualité spécifiques peuvent être adoptés pour des phases concernant des produits particuliers ou pour des composants logiciels particuliers, en appliquant des normes nationales ou des règles d'entreprise, sous réserve que les principes définis dans cette norme soient pris en compte.

5.5.2 Tout écart par rapport aux exigences de cette norme et de ses annexes normatives doit être identifié et justifié.

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 Si des pratiques différentes de celles figurant dans les annexes normatives sont utilisées, elles doivent être explicitées et pouvoir être vérifiées par rapport aux exigences de cette norme.

5.5.4 En particulier, l'impact de ces pratiques sur le système d'I&C et le logiciel doit être pris en considération.

5.5.5 Les procédures techniques nécessaires à chaque phase du cycle de vie et de sûreté du logiciel doivent être référencées par le plan d'assurance qualité.

5.5.6 Le plan d'assurance qualité doit exiger que la réalisation des activités soit confiée à des personnes compétentes disposant de ressources adéquates.

5.5.7 Le plan d'assurance qualité doit exiger que les modifications de documents déjà approuvés soient identifiées, revues et approuvées par les personnes autorisées.

5.5.8 Le plan d'assurance qualité doit exiger que les méthodes, langages, outils, règles et normes utilisées soient identifiés et documentés, et connus et maîtrisés par les personnes concernées.

5.5.9 Si plusieurs méthodes, langages, outils, règles et/ou normes sont utilisés, le plan d'assurance qualité doit exiger que ce qui a été utilisé pour chaque activité soit clairement établi.

5.5.10 Le plan d'assurance qualité doit exiger que les termes, expressions, abréviations et conventions utilisés dans un sens spécifique au projet soient explicitement définis.

5.5.11 Le plan d'assurance qualité doit exiger que tout problème de qualité rencontré soit suivi et résolu.

5.5.12 Le plan d'assurance qualité doit exiger que des enregistrements résultant de son application soient produits.

5.5.13 Chaque étape de vérification ou revue doit donner lieu à un rapport sur l'analyse réalisée, les conclusions atteintes et les décisions prises. Ce rapport doit être inclus dans la documentation.

5.5.14 Tout écart par rapport au plan d'assurance qualité doit être explicité et justifié.

5.6 Gestion de configuration

Le paragraphe 6.2.1.2 (plan de gestion de configuration du système) de la CEI 61513 énonce des exigences pour la gestion de configuration au niveau du système d'I&C. Le présent paragraphe énonce des exigences supplémentaires spécifiques ou d'une importance particulière pour le logiciel.

5.6.1 La gestion de configuration du logiciel doit être réalisée conformément aux dispositions d'un plan de gestion de configuration ou du plan d'assurance qualité.

5.6.2 Ces dispositions doivent être cohérentes avec celles de la gestion de configuration du niveau système.

5.6.3 Des procédures de gestion de configuration du logiciel doivent être établies et documentées au plus tôt dans le cycle de vie et de sûreté du projet logiciel pour répondre aux exigences suivantes.

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 Chaque version produite de chaque entité logicielle doit être identifiée de façon unique.

5.6.5 Il doit être possible d'identifier la version applicable de tout document associé à chaque entité logicielle.

5.6.6 Tout logiciel en développement doit être clairement séparé des logiciels mis en exploitation ou déjà vérifiés.

5.6.7 Il doit être possible d'identifier la version de toutes les entités logicielles qui collectivement constituent une version complète du produit final.

5.6.8 Il doit être possible de vérifier l'intégrité des entités logicielles.

5.6.9 Il doit être possible d'identifier la version du logiciel incorporé dans le système d'I&C cible.

5.6.10 Il doit être possible d'identifier a posteriori toutes les entités logicielles affectées par la réalisation d'une modification.

5.6.11 L'accès à toute entité logicielle placée sous gestion de configuration doit être soumis à des contrôles appropriés assurant que le logiciel n'est pas modifié par des personnes non autorisées et que la sécurité du logiciel est maintenue.

5.6.12 Il doit être possible d'identifier tous les outils de traduction et les versions d'outils utilisés pour produire chaque entité exécutable (voir 14.3.3).

5.7 Sécurité du logiciel

L'objectif de la sécurité est de protéger le logiciel et les données afin que des personnes et des systèmes non autorisés ne puissent pas les lire ou les modifier, et que l'accès par les personnes et les systèmes autorisés ne leur soit pas refusé.

Les paragraphes 5.4.2 (plan de sécurité globale) et 6.2.2 (plan de sécurité du système) de la CEI 61513 énoncent des exigences pour la sécurité au niveau de l'architecture globale de l'I&C et au niveau des systèmes d'I&C particuliers.

Bien que l'utilisation de logiciels présente des menaces potentielles pour la sécurité, les principales contre-mesures sont généralement réalisées au niveau du système, par exemple des mesures de protection physique ou des dispositifs de verrouillage câblés. Les principales exigences de sécurité sur le logiciel peuvent aider à minimiser la vulnérabilité et peuvent permettre et faciliter les mesures de protection au niveau système.

Ce paragraphe énonce des exigences de sécurité spécifiques ou d'une importante particulière pour le logiciel.

5.7.1 Analyse de sécurité

5.7.1.1 Une analyse des menaces potentielles pour la sécurité du logiciel doit être réalisée. Elle doit prendre en considération les phases concernées des cycles de vie et de sûreté du logiciel et du système. Elle doit déterminer les exigences concernant la protection et l'accessibilité des données et du logiciel, sur la base des exigences de ce paragraphe.

5.7.1.2 L'analyse de sécurité du logiciel doit être prise en compte dans le plan d'assurance qualité du système ou du logiciel, ou dans le plan de sécurité du système ou du logiciel.

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 Si l'analyse montre que les contre-mesures au niveau du système ne sont pas suffisantes, elle doit déterminer des exigences pour des contre-mesures dans la conception du logiciel.

5.7.2 Conception de la sécurité

5.7.2.1 Les exigences pour des contre-mesures dans la conception du logiciel déterminées par l'analyse de sécurité doivent être incluses dans les exigences de conception du logiciel.

5.7.2.2 Il convient que tout nouveau logiciel soit conçu pour minimiser la vulnérabilité du système.

5.7.2.3 Il convient que tout logiciel prédéveloppé soit configuré et paramétré pour minimiser la vulnérabilité du système, par exemple en se limitant aux fonctions strictement nécessaires ou en utilisant les fonctions de sécurité existantes du logiciel.

5.7.2.4 L'opérateur ne doit pas pouvoir dégrader les logiciels résidants.

5.7.2.5 Si un accès direct de l'opérateur aux données est nécessaire pour l'exploitation des fonctions d'I&C, les dispositifs d'interface homme-machine doivent limiter cet accès au strict nécessaire.

5.7.2.6 Lorsqu'il est nécessaire de contrer des menaces de sécurité potentielles, des mesures de protection efficaces doivent être incluses dans la conception, la configuration et/ou le paramétrage du logiciel pour:

– le contrôle d'accès sélectif de l'utilisateur aux fonctions logicielles,

– les liaisons de données avec les systèmes ayant une importance moindre pour la sûreté,

– la traçabilité des modifications du logiciel ou des paramètres.

5.7.2.7 La documentation de la conception doit identifier et décrire les fonctions critiques pour la sécurité et les caractéristiques de sécurité intégrées au logiciel.

5.7.2.8 Pendant la vérification du logiciel, l'efficacité des fonctions de sécurité doit être confirmée.

5.7.2.9 Pendant la validation du système d'I&C, l'efficacité des fonctions de sécurité du logiciel doit être démontrée par des tests adaptés.

5.7.3 Accès de l'utilisateur

5.7.3.1 Lorsque cela est nécessaire, le logiciel doit comporter des mesures techniques pour une authentification efficace de l'utilisateur avant que l'accès lui soit autorisé.

5.7.3.2 Si le contrôle d'accès de l'utilisateur est une fonction critique pour la sécurité et s'il est réalisé par le logiciel, il convient que l'authentification repose sur une combinaison d'identifiants (par exemple mot de passe), de propriétés (par exemple clé, carte à puce) et/ou de caractéristiques personnelles (par exemple empreintes) plutôt que sur seulement un mot de passe.

5.7.3.3 Le logiciel et le contrôle d'accès «basé fonction» doivent être configurés et paramétrés de telle façon que les autorisations d'accès soient limitées aux fonctions et données nécessaires.

5.7.3.4 Les autorisations d'accès des utilisateurs doivent être protégées de façon appropriée contre les occurrences et les conséquences des menaces de sécurité potentielles.

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 **A**ny 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.

Les méthodes cryptographiques (cryptage des données) sont un moyen pour répondre correctement à cette exigence.

5.7.3.5 Aucun accès à distance qui pourrait influer sur les fonctions logicielles ou les données hors des locaux techniques de la centrale (par exemple depuis des bâtiments administratifs ou depuis l'extérieur de l'installation) ne doit être mis en place.

5.7.4 Sécurité pendant le développement

5.7.4.1 Il convient que le cycle de vie et de sûreté du logiciel traite les menaces de sécurité pendant le développement et la maintenance.

5.7.4.2 Des dispositions contre la présence de fonctions cachées dans le logiciel d'application ou dans le logiciel système doivent être prises (par exemple vérification du programme logiciel) parce qu'elles pourraient permettre des accès non autorisés.

5.7.4.3 Si de telles dispositions ne peuvent être mises en œuvre pour un logiciel prédéveloppé, l'utilisation de ce logiciel doit être justifiée, en considérant les menaces de sécurité potentielles, l'importance pour la sûreté des fonctions d'I&C concernées, et les caractéristiques du système et du logiciel.

5.7.4.4 Les moyens qui permettraient une modification délibérée d'un logiciel, pouvant provoquer un comportement erroné déclenché par des conditions temporelles ou de valeurs de données, doivent être identifiés et justifiés pour être détectables pendant les activités de vérification.

6 Exigences du logiciel

6.1 Spécification des exigences du logiciel

6.1.1 Les exigences du logiciel sont issues des exigences des systèmes de sûreté et constituent une partie des spécifications du système programmé.

6.1.2 Les exigences du logiciel doivent décrire ce que le logiciel doit faire et non comment il doit le faire.

6.1.3 Les exigences du logiciel doivent spécifier:

– les fonctions d'application à fournir par le logiciel;

– les différents modes de comportement du logiciel et les conditions correspondantes de transition;

– les interfaces et interactions du logiciel avec son environnement (par exemple les exploitants, le reste du système d'I&C, les éventuels autres systèmes avec lesquels il y a interaction ou partage de ressources), y compris les rôles, types, formats, domaines et contraintes des entrées et des sorties;

– les paramètres du logiciel, lorsqu'il y en a, qui peuvent être modifiés manuellement pendant l'exploitation, leurs rôles, types, formats, domaines et contraintes, et les vérifications à réaliser par le logiciel quand ils sont modifiés;

– les performances exigées, en particulier les exigences de temps de réponse;

– ce que le logiciel ne doit pas faire ou ce qu'il doit éviter, quand cela est approprié;

– les exigences prises en compte, ou les hypothèses faites, par le logiciel vis-à-vis de son environnement, lorsque cela est applicable;

– les éventuelles exigences relatives à des produits logiciels standards.

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 Du fait de l'importance de cette phase de développement du logiciel, le processus d'établissement des exigences du logiciel doit être rigoureux.

6.1.5 La spécification des exigences du logiciel doit être telle que la conformité du système d'I&C aux exigences de la CEI 61513 peut être démontrée.

Des exigences supplémentaires pour la spécification des exigences du logiciel sont données en Annexe A.

6.1.6 Les contraintes entre matériel et logiciel doivent être décrites (A.2.1).

6.1.7 La spécification des exigences du logiciel doit faire référence à la spécification des exigences du matériel qui interagissent avec le logiciel.

6.1.8 Les conditions particulières de fonctionnement, telles que le démarrage de l'installation et le rechargement en combustible, doivent être décrites et déclinées au niveau logiciel pour les fonctions concernées.

6.2 Auto-surveillance

6.2.1 Le logiciel système du système programmé doit surveiller le matériel pendant l'exploitation, à intervalles de temps spécifiés, ainsi que le comportement du logiciel (A.2.2).

Ceci est considéré comme un facteur essentiel pour atteindre l'objectif de fiabilité de l'ensemble du système.

6.2.2 Les zones de la mémoire qui contiennent le code et les données invariables doivent être surveillées pour détecter les changements non prévus.

6.2.3 Il convient que l'auto-surveillance puisse détecter autant que possible:

– les défaillances aléatoires des composants matériels,
– les erreurs de comportement du logiciel (par exemple, écarts de conditions de fonctionnement de l'installation, écarts de traitements logiciels spécifiés ou corruption de données),
– la transmission de données erronées entre différentes unités de traitement.

6.2.4 Si une défaillance est détectée par le logiciel pendant l'exploitation de l'installation, le logiciel doit donner les réponses appropriées en temps voulu. Celles-ci doivent être réalisées en accord avec les réactions du système exigées par la spécification et les règles de conception du système énoncées dans la CEI 61513.

Cela peut impliquer l'étude de dispositions évitant des déclenchements intempestifs.

6.2.5 L'auto-surveillance ne doit pas dégrader les fonctions que le système doit remplir.

6.2.6 Il convient que la collecte automatique des informations utiles au diagnostic, obtenues durant l'auto-surveillance, soit possible.

6.3 Test périodique

6.3.1 Pour les systèmes de sûreté programmés, il convient que les principes généraux de la CEI 60671 soient utilisés pour les composants qui ne sont pas testés de façon adéquate par l'auto-surveillance.

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 Le logiciel doit être conçu pour répondre aux exigences des tests périodiques qui sont effectués à intervalles de temps maximaux spécifiés (par exemple pendant les périodes d'arrêt de l'installation).

1) toutes les fonctions de sûreté doivent être exercées lors d'un test périodique;
2) toute défaillance dans l'exécution des fonctions de sûreté doit être détectée.

6.3.3 Il convient que la collecte automatique de toutes les informations utiles pour le diagnostic, obtenues durant le test périodique, soit possible.

6.3.4 Il convient que la qualité des logiciels des dispositifs auxiliaires de tests corresponde à la qualité des équipements utilisés pour la validation, comme mentionné en 10.2.

Les logiciels des systèmes auxiliaires de test des systèmes de classe 1 ne sont pas tenus de satisfaire à toutes les exigences de sûreté du logiciel de cette norme.

6.4 Documentation

6.4.1 L'objectif principal du document de spécification des exigences du logiciel est d'être la base du développement du logiciel. Cependant, il convient que les aspects relatifs à l'autorisation de fonctionnement d'une installation ne soient pas négligés car ce document peut être soumis à l'autorité de sûreté. En conséquence, il peut contenir des points mineurs pour le développement du logiciel, mais importants pour l'autorisation de fonctionnement.

Ces derniers peuvent être:

– des considérations liées aux risques;
– des recommandations pour les fonctions de sûreté ou les actions de sauvegarde de l'installation;
– des éléments précisant le contexte de certaines exigences;
– des exigences spécifiques de la réglementation sur la structure du logiciel, l'analyse de code, la V&V, etc.

6.4.2 La spécification des exigences du logiciel doit être présentée selon un modèle dont le formalisme ne doit pas nuire à la compréhension (A.2.3).

6.4.3 La spécification des exigences du logiciel doit être non équivoque, testable ou vérifiable, et réalisable.

Un langage formel ou un langage orienté application peut être utilisé pour montrer la cohérence et l'exhaustivité des aspects de la spécification des exigences du logiciel.

Des outils automatiques peuvent être utilisés à cet effet.

6.4.4 La spécification des exigences du logiciel doit être fournie aux acteurs du processus d'ingénierie concernés.

7 Conception et réalisation

Cet article présente de bonnes pratiques pour développer un logiciel ayant des caractéristiques de sûreté appropriées, aussi exempt de défaut que possible et apte à la vérification.

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.

La spécification des exigences du logiciel doit être disponible avant que les phases de conception et de réalisation du développement du logiciel ne commencent.

Les phases de conception et de réalisation concrétisent les exigences de la spécification du logiciel et fournissent la base pour la vérification de la conception et de la réalisation du logiciel.

7.1 Principes pour la conception et la réalisation

7.1.1 Généralités

7.1.1.1 La conception du logiciel doit inclure une auto-surveillance (A.2.2).

7.1.1.2 La détection d'une erreur doit entraîner une action appropriée, conformément à 6.2.

7.1.1.3 Il convient que la structure du logiciel soit basée sur une décomposition en modules.

7.1.1.4 Il convient que la structure du logiciel soit simple et facile à comprendre, à la fois dans sa conception générale et dans ses détails.

7.1.1.5 Il convient de bannir les «astuces», les structures récursives et les compactages de code non nécessaires.

7.1.1.6 Il convient que le programme source soit compréhensible par des ingénieurs compétents non impliqués dans le processus de développement logiciel.

7.1.1.7 Il convient que le programme source soit conforme aux règles établies pour en améliorer la clarté et l'aptitude à être modifié et testé.

7.1.1.8 Il convient de justifier toute non-conformité par rapport aux règles de conception.

7.1.1.9 Une documentation détaillée et lisible doit être fournie.

7.1.1.10 La conception des liens de communication doit être conforme aux exigences sur la communication des données de 5.3.1.3 de la CEI 61513.

7.1.1.11 Les liens de communication utilisés à l'intérieur d'une même voie doivent être déterministes.

7.1.1.12 Les recommandations suivantes découlent de ces principes:

il convient:

1) que les mesures à prendre pour la réalisation des exigences relatives à la sûreté du logiciel, y compris l'auto-surveillance, soient choisies au début de la conception (Article B.3);
2) d'utiliser une approche descendante de préférence à une approche ascendante dans la conception du logiciel (Article B.1);
3) d'adopter un modèle conceptuel de l'architecture du logiciel en début de chaque projet logiciel (Article B.2);
4) que le logiciel soit écrit pour être facilement vérifiable (Articles B.4 et B.5);
5) en cas d'utilisation de logiciel standard d'un fabricant ou fournisseur, d'appliquer les exigences de l'Article 15 en plus du point c) de l'Article B.2);
6) d'utiliser les langages orientés application de préférence à ceux orientés «machine» (point e) de l'Article B.5).

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 Réalisation de nouveaux logiciels en langage généraliste

Ce paragraphe traite le cas du développement d'une partie ou de la totalité des fonctions de sûreté de catégorie A par création de nouveaux composants logiciel avec des langages généralistes.

Les langages généralistes sont les langages de haut niveau tels que Ada, C, Pascal ou les langages assembleurs dédiés à la plate-forme utilisée. Ils peuvent être utilisés pour réaliser tout type de fonction, si des règles de conception et de codage appropriées sont suivies.

7.1.2.1 La phase de conception du logiciel doit identifier les composants du logiciel à développer avec des langages généralistes.

7.1.2.2 Pour ces composants, il convient que le processus de développement définisse une phase de conception détaillée et une phase de codage.

7.1.2.3 Les activités de la phase de conception détaillée doivent préciser les données de sortie de la phase de conception de telle sorte que le codage avec le langage choisi se fasse de façon systématique (par exemple en définissant les algorithmes nécessaires, les structures de données, l'interface des fonctions, les contraintes, etc.).

7.1.2.4 Le niveau de détail de l'information à fournir lors de la phase de conception détaillée dépend du langage généraliste utilisé. Dans le cas de langages assembleurs, la conception doit fournir des algorithmes structurés détaillés et la représentation des données.

7.1.2.5 La phase de codage doit traduire la conception détaillée en un code source, suivant des règles de programmation prédéfinies basées sur les exigences de l'Annexe B.

7.1.3 Réalisation de nouveaux logiciels en langage orienté application

Ce paragraphe traite le cas du développement d'une partie ou de la totalité des fonctions de sûreté de catégorie A par création de nouveaux composants logiciel avec des langages orientés application.

Les langages orientés application s'appuient sur des formalismes (tels que diagrammes logiques ou schémas de blocs fonctionnels, etc.) qui peuvent être utilisés pour décrire tout ou partie des exigences de la spécification du logiciel et/ou de la documentation de conception du logiciel. Ces descriptions peuvent être utilisées comme entrée pour générer un code exécutable par des moyens automatisés.

7.1.3.1 Il convient que les formalismes utilisés aient les propriétés suivantes: faible complexité, clarté et standardisation de la disposition et de la présentation, modularité, présence de commentaires pertinents, absence de caractéristiques contraires à la sûreté. Ces propriétés facilitent généralement la compréhension, la vérification, le test et les modifications ultérieures.

7.1.3.2 Il convient que les langages orientés application aient un format compréhensible par les ingénieurs responsables de la revue des spécifications du logiciel, par exemple les ingénieurs du procédé et les ingénieurs du contrôle-commande qui définissent les systèmes pour lesquels les fonctions de contrôle-commande sont spécifiées.

7.1.3.3 Il convient que le langage orienté application soit associé à une structure de logiciel simple par exemple programmes séquentiels.

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 Il convient que le langage orienté application permette aux développeurs de prendre en compte les spécifications de conception de l'architecture du système de contrôle-commande c'est-à-dire permette la répartition des fonctions de contrôle-commande sur les composants matériels et le support de toute caractéristique de conception de tolérance aux fautes du matériel.

7.1.4 Configuration de logiciels prédéveloppés

Ce paragraphe traite le cas de la réalisation des fonctions de sûreté de catégorie A au moyen d'un logiciel paramétré avec des données de configuration applicatives, c'est-à-dire des données spécifiques à cette application.

Un logiciel prédéveloppé peut être associé à une famille d'équipements ou peut être un produit isolé qui est alors intégré à la plate-forme matérielle choisie.

7.1.4.1 En cas d'utilisation d'un logiciel prédéveloppé, ses capacités doivent être évaluées et démontrées (voir 15.3) pour assurer qu'il est adapté à la fonction prévue.

Les exigences pour l'analyse de l'aptitude à l'usage sont traitées en 15.3.1.2. Si les capacités du logiciel devaient être restreintes, ceci peut être obtenu en ajoutant une couche logicielle pour encapsuler le logiciel prédéveloppé.

Pour les besoins de la configuration des logiciels, une approche outillée est préférable pour limiter le potentiel d'erreurs humaines.

7.1.4.2 Toutes les contraintes concernant les données doivent être documentées, par exemple formats de données autorisés, étendue, règles de calcul.

7.1.4.3 Les données de configuration doivent être documentées.

7.1.4.4 Une justification adéquate doit être fournie pour les valeurs de données utilisées en incluant une référence croisée avec les éléments d'entrée de la conception.

7.1.4.5 Traçabilité – il convient de pouvoir déterminer quand une quelconque modification a été faite sur les données de configuration et par qui.

7.1.4.6 Maintenabilité – le processus de développement doit assurer, du fait d'une approche structurée associée à l'utilisation de commentaires dans les données et/ou d'une documentation d'assistance, que la conception de données est compréhensible et maintenable tout au long de la durée de vie du système dont elles font parties.

Voir 14.3.5 pour d'autres directives sur l'utilisation des outils pour les données d'application.

7.2 Langages, traducteurs et outils associés

7.2.1 Exigences générales

Bien que l'utilisation de langages spécifiques ne puisse être exigée, ce qui suit peut être considéré comme les règles de base communes aux langages utilisés pour la conception et la réalisation de logiciel pour les systèmes de classe 1.

7.2.1.1 Le langage utilisé doit suivre des règles de syntaxe et de sémantique strictes (ou bien définies).

7.2.1.2 La syntaxe du langage doit être complètement et clairement définie et documentée.

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 Il convient que l'utilisation du langage soit limitée à un sous-ensemble «sûr» lorsque cela est approprié; par exemple, être restreint aux primitives aptes à spécifier les fonctions nécessaires.

7.2.1.4 Il convient d'utiliser des langages disposant d'un traducteur complètement testé.

7.2.1.5 Si un traducteur partiellement testé est utilisé, une vérification supplémentaire doit montrer que le résultat de la traduction est correct.

7.2.1.6 Il convient que des outils de test automatique soient disponibles.

Il convient d'utiliser des outils automatiques. Les exigences de l'Article 14 s'appliquent.

7.2.2 Langages généralistes

Il convient que les langages généralistes pour les systèmes de classe 1, et leurs traducteurs, n' interdisent pas l'utilisation de constructions limitant les erreurs telles que:

– la vérification des types lors de la traduction, la vérification des paramètres des sous-programmes;
– la vérification de la limite de validité des pointeurs de tableau à l'exécution.

Des directives relatives au choix des langages, traducteurs, etc. sont données à l'Annexe D.

7.2.3 Langages orientés application et génération automatique du code associé

7.2.3.1 Il convient que les langages orientés application soient transformés en un langage généraliste par des outils automatisés (par exemple générateur de code) avant leur traduction en un format exécutable.

7.2.3.2 La conformité du code généré aux exigences de conception et de codage du logiciel préconisées dans cette norme doit être évaluée et les non-conformités justifiées.

7.2.3.3 La structure du programme doit être définie de façon générique; par exemple, la position des déclarations relatives aux instructions du code.

7.2.3.4 Le code généré ne doit pas être modifié par action manuelle directe sur le code.

7.2.3.5 Le code doit être généré à nouveau en cas de modification de la spécification, par exemple, suite aux résultats d'actions de V&V.

Des recommandations supplémentaires pour la génération automatisée du code sont données au point f) de l'Article B.5.

7.3 Recommandations détaillées

7.3.1 Généralités

L'Annexe B donne un ensemble de recommandations qui spécifient en détail les aspects traités en 7.1.

Les têtes de chapitre des recommandations individuelles de l'Annexe B sont applicables aux deux phases essentielles du développement du logiciel suivant le Tableau 1 ci-après.

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.

Tableau 1– Aspects processus et aspects produit de la conception et de la réalisation

Phase du développement logiciel	Aspects processus	Annexe B	Aspects produit	Annexe B
Conception	- Aptitude à la modification	B1.*a*	- Structures de contrôle et d'accès	B2.*a*
	- Approche descendante	B1.*b*	- Modules	B2.*b*
	- Vérification des produits intermédiaires de la conception	B1.*c*	- Logiciel système opérationnel	B2.*c*
	- Maîtrise des modifications pendant le développement	B1.*d*	- Temps d'exécution	B2.*d*
			- Interruptions	B2.*e*
			- Expressions arithmétiques	B2.*f*
			- Contrôle de vraisemblance	B3.*a*
			- Sûreté des sorties	B3.*b*
			- Branchements et boucles	B4.*a*
			- Sous-programmes	B4.*b*
			- Structures imbriquées	B4.*c*
			- Structures de données	B4.*e*
			- Langages orientés application	B5.*e*
Réalisation	- Vérification des produits intermédiaires de la conception	B1.*c*	- Modules	B2.*b*
	- Contrôle des modifications pendant le développement	B1.*d*	- Temps d'exécution	B2.*d*
	- Tests unitaires et tests d'intégration	B4.*g*	- Interruptions	B2.*e*
	- Règles de programmation	B5.*d*	- Expressions arithmétiques	B2.*f*
			-Contrôle de vraisemblance	B3.*a*
			- Sûreté des sorties	B3.*b*
			- Contenu mémoire	B3.*c*
			- Contrôle des erreurs	B3.*d*
			- Branchements et boucles	B4.*a*
			- Sous-programmes	B4.*b*
			- Structures imbriquées	B4.*c*
			- Adressage et zones de données	B4.*d*
			- Structures de données	B4.*e*
			- Modifications dynamiques	B4.*f*
			- Organisation du programme	B5.*a*
			- Commentaires	B5.*b*
			- Assembleur	B5.*c*
			- Génération automatique de code	B5.*f*

7.3.2 Utilisation des exigences et recommandations

7.3.2.1 Les exigences et recommandations données en Annexe B doivent être respectées pendant le développement du logiciel sauf justification et documentation étayant les écarts.

7.3.2.2 Il convient que la justification intervienne au début de la conception.

Table 1 – Process and product aspects of design and implementation

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

Ce document sert de base à la revue formelle de conception et à la réalisation du logiciel.

7.4.2 Le niveau de détail doit être suffisant pour permettre la réalisation du logiciel sans besoin de clarification supplémentaire de la conception.

7.4.3 Il convient que le document soit structuré selon les niveaux de conception du logiciel.

La documentation de conception du logiciel peut correspondre à un seul document ou à un ensemble intégré de documents.

7.4.4 Dans ce cas, chaque document doit avoir une relation définie avec les autres documents et traiter d'un sujet bien délimité.

7.4.5 Il convient de choisir la présentation de la documentation selon les besoins particuliers, en incluant:

– des descriptions narratives;
– des expressions arithmétiques;
– des représentations graphiques.

7.4.6 Il convient que la documentation inclue les diagrammes et dessins appropriés.

En règle générale, il est préférable de choisir une représentation graphique.

7.4.7 Il convient que la documentation soit conforme aux normes nationales s'il y a lieu.

8 Vérification du logiciel

8.1 Processus de vérification du logiciel

Les activités de vérification menées durant le développement du logiciel sont généralement de la responsabilité du fournisseur et sont réalisées par un personnel indépendant de celui produisant le logiciel, le meilleur moyen étant de constituer une équipe de vérification.

Des activités de vérification supplémentaires peuvent être entreprises par une tierce partie au titre de l'évaluation du logiciel et de son développement, afin de confirmer que le logiciel répond à ses objectifs de qualité. Il y a de nombreuses façons d'organiser et réaliser l'indépendance de la vérification, celle-ci relevant souvent de la réglementation nationale en vigueur.

8.1.1 L'équipe de vérification doit être composée de personnes qui ne sont pas engagées dans la production et qui ont les compétences et les connaissances nécessaires.

Les exigences suivantes définissent explicitement le niveau d'indépendance exigé.

8.1.2 L'encadrement de l'équipe de vérification doit être séparé et indépendant de celui de l'équipe de développement.

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 La communication entre l'équipe de vérification et l'équipe de développement, que ce soit pour demande de clarification ou pour constat d'erreur, doit se faire de manière formelle par écrit à un niveau de précision qui puisse être audité.

8.1.4 Il convient que les interactions entre les deux parties maintiennent l'indépendance de jugement de l'équipe de vérification.

8.1.5 L'équipe de vérification doit disposer de ressources et moyens adéquats. Elle doit aussi disposer du temps nécessaire à la réalisation des activités de vérification.

8.1.6 L'équipe de vérification doit avoir des responsabilités et obligations clairement définies.

8.1.7 L'équipe de vérification doit avoir l'autorité nécessaire pour faire état de ses conclusions.

8.1.8 Les produits de chaque phase de développement du logiciel (Figure 3) doivent être vérifiés.

8.1.9 Les activités de vérification du logiciel doivent confirmer l'adéquation entre la spécification du logiciel et les exigences assignées au logiciel par la spécification du système programmé.

8.1.10 Les activités de vérification du logiciel doivent confirmer l'adéquation de la documentation de conception du logiciel en regard des exigences de la spécification du logiciel.

8.1.11 Les activités de vérification du logiciel doivent confirmer la conformité du code à la documentation de conception du logiciel élaborée lors de la phase de conception. Lorsqu'un outil d'ingénierie assistée par ordinateur avec des caractéristiques telles que la génération automatique de code est utilisé, des exigences dédiées sont données en 8.2.3.2.

8.1.12 Il convient que chaque activité de production débute sur la base de données/ documents d'entrée préalablement vérifiés.

8.1.13 Lorsqu'elle est réalisée conjointement au développement du logiciel, il convient que la vérification du produit d'une phase soit réalisée avant le début de la phase suivante. Si ce n'est pas possible, la vérification du produit d'une phase doit être réalisée avant l'achèvement (c'est-à-dire la vérification) de la phase suivante.

Un travail préparatoire pour une phase à venir peut être réalisé avant que la phase précédente ne soit vérifiée et approuvée.

8.1.14 Si les documents/données d'entrée pour une activité ont été modifiés, cette activité et les suivantes doivent être reprises pour traiter l'impact potentiel.

8.1.15 Toutes les vérifications du logiciel doivent être terminées avant que le système ne soit mis en service.

8.2 Activités de vérification du logiciel

Les activités de vérification suivantes sont exigées.

8.2.1 Plan de vérification

8.2.1.1 Un plan de vérification du logiciel doit être établi avant toute autre activité de vérification du logiciel.

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.

8.2.1.2 Ce plan doit expliciter tous les critères, les techniques et les outils à utiliser dans le processus de vérification.

8.2.1.3 Il doit décrire les activités à réaliser pour évaluer chaque élément du logiciel et chaque outil impliqué dans le processus de développement du logiciel, pour montrer à chaque phase si les exigences de la spécification du logiciel sont respectées.

8.2.1.4 Le niveau de détail doit être tel qu'une équipe indépendante puisse exécuter ce plan de vérification et émettre un jugement objectif sur la capacité ou non du logiciel à satisfaire à ses exigences.

8.2.1.5 Le plan de vérification doit être préparé par une équipe de vérification prenant en charge:

1) la sélection des stratégies de vérification (systématique et/ou aléatoire avec choix de jeux d'essais selon les fonctions requises et/ou les caractéristiques de la structure du programme, voir Annexe E);
2) la sélection et l'utilisation des outils de vérification du logiciel;
3) l'exécution de la vérification;
4) la documentation des activités de vérification;
5) l'évaluation des résultats de la vérification obtenus directement à partir des outils de vérification et des tests, l'évaluation du respect des exigences de sûreté.

8.2.1.6 Il convient que les tests réalisés sollicitent largement le logiciel. Parmi les critères requis par le plan, il convient que les critères de couverture de test soient considérés comme de première importance.

8.2.1.7 Le plan de vérification doit identifier tout élément objectif exigé pour confirmer l'étendue des tests. Pour cela, les critères de couverture de test choisis en accord avec la conception (voir Annexe E) doivent être justifiés et documentés.

8.2.1.8 Des dispositions adéquates doivent être prévues pour le traitement et la résolution de tout problème de sûreté soulevé pendant les activités de vérification réalisées lors du développement du logiciel par le fournisseur ou lors d'une évaluation par une tierce partie.

8.2.1.9 Tout problème de sûreté doit être résolu par des modifications correctives ou des mesures de compensation appropriées.

8.2.2 Vérification de la conception

8.2.2.1 La vérification de la conception doit traiter:

1) la cohérence et l'exhaustivité de la documentation de conception en regard de la spécification des exigences du logiciel, et cela jusqu'au niveau des modules;
2) la décomposition de la conception en modules fonctionnels, et la manière dont ils sont spécifiés du point de vue de:
   - la faisabilité technique;
   - la testabilité en vue de la vérification;
   - la lisibilité par les équipes de développement et de vérification;
   - l'aptitude à la modification;
3) la mise en œuvre correcte des exigences de sûreté.

8.2.1.2 The plan shall document all the criteria, the techniques and tools to be utilized in the verification process.

8.2.1.3 It shall describe the activities to be performed to evaluate each item of software, each tool involved in the software development process, and each phase to show whether the software requirements specification is met.

8.2.1.4 The level of detail shall be such that a verification team can execute the verification plan and reach an objective judgement on whether or not the software meets its requirements.

8.2.1.5 The verification plan shall be prepared by a verification team addressing:

1) selection of verification strategies, either systematic, random or both, with test case selection according to either required functions, special features of program structure, or both (see Annex E);
2) selection and utilisation of the software verification tools;
3) execution of verification;
4) documentation of verification activities;
5) evaluation of verification results gained from verification equipment directly and from tests, evaluation of whether the safety requirements are met.

8.2.1.6 The tests performed should extensively exercise the software. Among the criteria required in the plan, test coverage criteria should be considered of prime importance.

8.2.1.7 The verification plan shall identify any objective evidence required to confirm the extent of testing. For that purpose the test coverage criteria chosen according to the design (see Annex E) shall be justified and documented.

8.2.1.8 There shall be adequate provision for the processing and resolution of all safety issues raised during the verification activities performed either during software development by the supplier or by a third-party assessment.

8.2.1.9 All safety issues shall be resolved through appropriate corrective modifications or mitigating dispositions.

8.2.2 Design verification

8.2.2.1 The design verification shall address:

1) the adequacy of the software design specification for the software requirements with respect to consistency and completeness down to and including the modular level;
2) the decomposition of the design into functional modules and the way they are specified with respect to:
   - technical feasibility of design;
   - testability for further verification;
   - readability by the development and verification teams;
   - modifiability to permit further modification;
3) the correct implementation of safety requirements.

8.2.2.2 Le résultat de la vérification de la conception doit être documenté.

8.2.2.3 La documentation doit inclure les conclusions et identifier clairement les points nécessitant une action, tels que:

– les éléments non conformes aux exigences du logiciel;
– les éléments non conformes aux normes de conception;
– les modules, données, structures et algorithmes mal adaptés au problème.

8.2.3 Vérification de la réalisation

Indépendamment de la manière dont le codage du logiciel est réalisé, il convient de sélectionner d'après le Tableau E.4.2 les méthodes de test à utiliser pour vérifier le produit de la phase de réalisation.

8.2.3.1 Vérification de la réalisation avec des langages généralistes

8.2.3.1.1 Vérification du code

8.2.3.1.1.1 La vérification de la réalisation doit comprendre des activités fondées sur l'analyse du code source et des tests. L'analyse du code source peut être effectuée avec des méthodes de vérification telles que l'inspection de code, en recourant éventuellement à l'assistance d'outils automatiques.

8.2.3.1.1.2 Il convient de commencer les activités de vérification de code par l'analyse du code source au niveau des modules, suivie de leur test.

8.2.3.1.1.3 La vérification de module doit montrer que chaque module réalise exclusivement la fonction attendue.

8.2.3.1.1.4 Un test d'intégration des modules doit alors être effectué pour montrer au plus tôt l'interaction correcte de tous les modules pour une fonction donnée. Si pour cela un outil d'ingénierie assistée par ordinateur est utilisé, il doit également répondre aux exigences de l'Article 14.

8.2.3.1.1.5 Les résultats de la vérification de code doivent être documentés.

8.2.3.1.2 Spécification de test du logiciel

La spécification de test du logiciel est un des principaux documents à produire au titre du plan de vérification.

8.2.3.1.2.1 Ce document doit être fondé sur la documentation de conception du logiciel et sur un examen détaillé des exigences du logiciel.

8.2.3.1.2.2 Il doit fournir des informations détaillées sur les tests à réaliser pour chaque élément du logiciel (les modules et leurs constituants).

8.2.3.1.2.3 La spécification de test du logiciel doit inclure:

1) l'environnement dans lequel s'effectue le test;
2) les procédures de test;
3) les critères d'acceptation, c'est-à-dire la définition détaillée des critères à remplir pour accepter les modules et les composants importants du logiciel aux niveaux sous-système et système;
4) les procédures pour la détection de défaut;
5) une liste de tous les documents à produire.

8.2.2.2 The result of the design verification shall be documented.

8.2.2.3 The documentation shall include the conclusions and identify clearly issues that need actions, such as:

– items which do not conform to the software requirements;

– items which do not conform to the design standards;

– modules, data, structures and algorithms poorly adapted to the problem.

8.2.3 Implementation verification

Regardless how the software code is developed, test methods used to verify the implementation phase output should be selected according to Table E.4.2.

8.2.3.1 Verification of implementation with general-purpose languages

8.2.3.1.1 Code verification

8.2.3.1.1.1 The implementation verification shall include activities based on source code analysis and tests. The source code analysis may be performed using verification methods such as code inspection, possibly with the assistance of automated tools.

8.2.3.1.1.2 Code verification activities should begin with module source code analysis followed by module testing.

8.2.3.1.1.3 Module verification shall show that each module performs its intended function and does not perform unintended functions.

8.2.3.1.1.4 A module integration test shall be performed to show at an early stage of development that all modules interact correctly to perform the intended function. If a CASE tool is used for this, it shall also meet the relevant requirements of Clause 14.

8.2.3.1.1.5 The results of code verification shall be documented.

8.2.3.1.2 Software test specification

The software test specification is one of the principal documents to be addressed by the verification plan.

8.2.3.1.2.1 This document shall be based on the software design specification and a detailed examination of the software requirements.

8.2.3.1.2.2 It shall give detailed information on the tests to be performed addressing each of the components of the software (modules and their constituents).

8.2.3.1.2.3 The software test specification shall include:

1) the environment in which the tests are run;
2) the test procedures;
3) acceptance criteria, i.e. a detailed definition of the criteria to be fulfilled in order to accept modules and major software components on subsystem and system levels;
4) procedures for fault detection;
5) a list of all documents that should be produced.

8.2.3.1.3 Compte rendu des tests du logiciel

8.2.3.1.3.1 Le compte rendu des tests du logiciel doit présenter les résultats de la procédure de test décrite dans la spécification de test du logiciel, en indiquant pour chacun si le logiciel se comporte ou non conformément à la documentation de conception.

8.2.3.1.3.2 Ce document doit rendre compte de toute non-conformité de réalisation ou de conception découverte pendant les tests.

8.2.3.1.3.3 Le compte rendu des tests du logiciel doit inclure les rubriques suivantes, aussi bien au niveau du module que des niveaux de conception plus élevés:

1) la configuration du matériel utilisé pour chaque test, l'identification et la justification de tout matériel et logiciel d'essai utilisé;
2) les moyens de stockage et les conditions d'accès au code final à tester;
3) les valeurs d'entrée de chaque test;
4) les valeurs de sorties attendues et les valeurs effectivement obtenues;
5) les données supplémentaires concernant la chronologie, la séquence d'événements, etc.;
6) la conformité aux critères d'acceptation spécifiés dans la spécification de test;
7) le relevé des défauts décrivant les caractéristiques de chaque défaut.

8.2.3.2 Vérification de la réalisation avec des langages orientés application

L'utilisation de langages orientés application est généralement envisagée pour améliorer la qualité (c'est-à-dire réduire le taux de défauts de conception ou de réalisation introduits pendant le processus de développement) et la maintenabilité du logiciel d'application.

8.2.3.2.1 Pour l'efficacité de la vérification, il convient qu'un logiciel d'application généré automatiquement à partir d'une spécification exprimée en langage orienté application ait une structure systématique.

8.2.3.2.2 Le caractère fonctionnellement correct et cohérent d'un logiciel écrit dans un langage orienté application doit être vérifié, par exemple par le biais d'une inspection manuelle ou par l'utilisation d'outils automatiques simulant l'exécution du logiciel dans un environnement de mise au point.

8.2.3.2.3 Le processus de vérification doit confirmer que:

– tous les éléments de la conception sont correctement réalisés;
– la fonctionnalité du logiciel est cohérente avec les objectifs définis par les exigences de la spécification du logiciel;
– la conception du logiciel est cohérente avec les normes applicables mentionnées dans le plan d'assurance qualité.

8.2.3.2.4 Une sélection justifiée et appropriée de techniques telles que l'animation, les essais, les revues, les relectures, les preuves ou les analyses formelles doit être appliquée pour améliorer la compréhension des spécifications et vérifier leur cohérence et leur adéquation fonctionnelle.

8.2.3.2.5 Les outils logiciels utilisés pour la vérification ou la validation doivent être qualifiés comme exigé par l'Article 14.

8.2.3.1.3 Software test report

8.2.3.1.3.1 The software test report shall present the results of the verification described in the software test specification stating whether or not the software performs in accordance with the software design specification.

8.2.3.1.3.2 This document shall report all software design and implementation discrepancies discovered during the tests.

8.2.3.1.3.3 The software test report shall include the following items both for the module and major design levels:

1) hardware configuration used for the test, identification and justification of any test harness hardware and software used;
2) storage medium used and access requirements of the final code tested;
3) input test values;
4) expected and achieved output values;
5) additional data regarding timing, sequence of events, etc.;
6) conformance with acceptance criteria given in the test specification;
7) fault incident log which describes the characteristics of each fault.

8.2.3.2 Verification of implementation with application-oriented languages

The use of application-oriented languages is generally considered to improve the quality (i.e. reduce the level of design and implementation faults introduced during the engineering process) and maintainability of application software.

8.2.3.2.1 Application software which is automatically generated from a specification using an application-oriented language should have a systematic structure to support effective verification.

8.2.3.2.2 Software written in application-oriented languages shall be verified to be functionally correct and consistent, for example by manual inspection or by the use of automated tools which allow simulated running of the software in a debug environment.

8.2.3.2.3 The verification process shall confirm that:

– all the design features are correctly implemented;
– the software functionality is consistent with the objectives defined in the software requirements specification;
– the software design is compliant with the applicable standards stated in the QA plan.

8.2.3.2.4 An appropriate and justified selection of techniques such as animation, tests, reviews, walkthroughs, formal analyses and proofs shall be applied to improve the understanding of specifications and to verify their functional correctness and consistency.

8.2.3.2.5 Software tools used for verification or validation shall be qualified as required by Clause 14.

8.2.3.3 Vérification de la configuration d'un logiciel prédéveloppé

Ce paragraphe traite du cas où des fonctions de sûreté de catégorie A sont réalisées par un logiciel prédéveloppé configuré avec des données d'application, c'est-à-dire des données spécifiques à cette application. Le logiciel peut être associé à une famille d'équipement ou peut être un produit séparé qui est alors intégré à la plate-forme matérielle choisie.

Un prérequis de ce paragraphe est que le logiciel prédéveloppé aie été qualifié pour l'application prévue (voir Article 15).

8.2.3.3.1 La vérification des données doit être effectuée en utilisant une combinaison d'inspection/d'analyse et/ou de test. Les moyens appropriés pour tester l'impact des données de configuration doivent être analysés et documentés (en prenant en compte le rôle de la simulation, de l'émulation, des équipements de test et des prototypes).

Le test du logiciel configuré est traité dans l'Article 9.

Certains aspects de la fonctionnalité requise du système peuvent s'exprimer en termes de données, par exemple un point de réglage de température. Pour celles-ci, le processus consiste à répliquer les valeurs spécifiées dans les données d'application du système. Pour de telles données, la vérification par inspection ou par comparaison électronique peut confirmer l'absence de défaut, avec un haut niveau de confiance.

D'autres aspects concernant les données peuvent être exprimés dans la spécification du système, comme l'allocation de signaux d'entrée à des cartes d'entrée particulières, ou le contenu des messages.

8.2.3.3.2 La structure documentaire adoptée pour chaque configuration de logiciel prédéveloppé doit assurer l'existence des documents de conception intermédiaire nécessaires à la description détaillée du processus de conception. Par exemple, pour justifier comment une exigence de temps de réponse a été traitée en configurant la fréquence de mesure et la fréquence d'envoi de messages.

8.2.3.3.3 Le processus de vérification doit confirmer que les données d'application sont cohérentes avec la documentation de conception et avec toutes les contraintes et règles établies.

Cela peut être obtenu par inspection manuelle, par une approche outillée, ou une combinaison des deux techniques.

8.2.3.3.4 Si des données produites par un outil sont directement intégrées dans le système sans vérification, alors la justification d'un niveau de confiance suffisant dans l'outil doit être fournie.

A titre de guide sur la vérification des données par des outils d'application, voir 14.3.5.

9 Aspects logiciels de l'intégration du système

Le processus d'intégration du système consiste à assembler du matériel et des modules logiciels vérifiés en sous-systèmes (unités programmées), puis en un système complet.

Ce processus comprend quatre activités:

a) l'assemblage et l'interconnexion des sous-ensembles matériels conformément aux documents de conception;

b) la construction du logiciel cible à partir des modules logiciels;

8.2.3.3 Verification of configuration of pre-developed software

This subclause considers the situation where category A safety functions are provided by pre-developed software configured with application configuration data, i.e. data specific to the intended application. Software may be associated with an equipment family or may be a standalone product which is then integrated with a chosen hardware platform.

A pre-requisite of this subclause is that the pre-developed software has been qualified for its intended application (see Clause 15).

8.2.3.3.1 Verification of data shall be performed using a combination of inspection/analysis and/or test. The appropriate means of testing the impact of the configuration data shall be analysed and documented (giving consideration to the role of simulation, emulation, test rigs and prototypes).

Testing of the configured software is covered in Clause 9.

Some aspects of the required system functionality may be expressed in terms of data, for example a temperature setpoint, and for such items the process is to replicate this data value in the application configuration data for the system. For such data, verification by inspection or by electronic comparison can confirm correctness to very high levels of confidence.

Other aspects of data may have to be developed from the system requirements, for example the allocation of signal inputs to specific input cards, the contents of message buffers.

8.2.3.3.2 The documentation structure adopted for any configuration of pre-developed software shall ensure that intermediate design documents are produced where necessary to document the design process comprehensively, for example to justify how a time response requirement has been addressed by configuring the scan frequency and the frequency of the transmittal of a message buffer.

8.2.3.3.3 The verification process shall confirm that the configuration data is consistent with the design documentation and with any established constraints and rules.

This could be achieved by a manual inspection process, or using a tool based approach, or by a combination of both techniques.

8.2.3.3.4 If data is produced using a tool-based approach and if it is intended to omit the data verification step and proceed directly to system integration, justification shall be provided that sufficient confidence in the correctness of the tool can be achieved.

For further guidance concerning the verification of data produced using application data tools, see 14.3.5.

9 Software aspects of system integration

The process of system integration is the combining of verified hardware and software modules into subsystems (computer units) and finally into the complete system.

This process consists of four activities:

a) assembling and interconnecting hardware modules as defined in design documents;

b) building the target software from software modules;

c) le chargement du logiciel cible dans le matériel cible;

d) la vérification que:

- le logiciel est conforme à la documentation de conception;
- les exigences d'interface matériel/logiciel sont satisfaites;
- le logiciel est capable de fonctionner dans cet environnement matériel particulier.

Les aspects logiciels de l'intégration du système couvrent les activités des points b), c), d) mentionnées ci-dessus.

Cet article complète 6.1.4 de la CEI 61513 en énonçant des exigences supplémentaires pour le plan d'intégration du système, ainsi que des exigences sur les aspects logiciels d'intégration du système.

9.1 Aspects logiciels du plan d'intégration du système

9.1.1 Ce plan doit être préparé et documenté durant les phases de conception, et vérifié par rapport aux exigences s'appliquant à un système de classe 1.

9.1.2 Ce plan doit être préparé suffisamment tôt dans le processus de développement pour permettre la prise en compte des exigences liées à l'intégration, lors de la conception du système et de ses constituants matériels et logiciels.

9.1.3 Ce plan doit spécifier les normes et procédures à suivre lors de l'intégration du système.

9.1.4 Ce plan doit documenter les dispositions du plan d'assurance qualité du système qui sont applicables pour l'intégration du système.

9.1.5 Le plan d'intégration du système doit prendre en compte les contraintes mises sur chaque ensemble de modules logiciels et/ou matériels, par la conception du système, du matériel et du logiciel. Il doit inclure les exigences concernant les procédures et les méthodes de contrôle, en couvrant:

– la gestion de configuration du système (voir 5.6);

– l'intégration du système;

– la vérification du système intégré;

– le traitement des défauts.

9.1.6 Le plan d'intégration du système doit couvrir les deux facettes de la gestion de configuration (identification et contrôle), conformément aux exigences de 6.2.1.2 de la CEI 61513.

9.1.7 Lors de la vérification des modules logiciels et matériels, certains aspects de la conception de ces modules peuvent être vérifiés au niveau des sous-systèmes (unités programmées) ou au niveau du système complet si cela s'avère plus pratique. Si la vérification n'est pas réalisable par des tests à ces niveaux, les exigences de conception relatives à chaque module pris individuellement doivent toutes être vérifiées par d'autres moyens.

9.1.8 Toutes les interdépendances entre la vérification des modules pris individuellement et la vérification du système intégré doivent être documentées dans le plan d'intégration du système.

c) loading the target software into the target hardware;

d) verifying that:

- the software complies with its design specification;
- the hardware/software interface requirements have been satisfied;
- the software is capable of operating in that particular hardware environment.

The software aspects of system integration are the above-mentioned activities b), c) and d).

This clause gives additional requirements for the system integration plan as well as requirements about the software aspects of system integration, in supplement to 6.1.4 of IEC 61513.

9.1 Software aspects of system integration plan

9.1.1 This plan shall be prepared and documented in the design phases and verified against the class 1 system requirements.

9.1.2 This plan shall be prepared sufficiently early in the development process to allow any integration requirements to be included in the design of the system and its hardware and software.

9.1.3 This plan shall specify the standards and procedures to be followed in the system integration.

9.1.4 This plan shall document those provisions of the system quality assurance plan that are applicable to the system integration.

9.1.5 The system integration plan shall take into account the constraints, within any set of hardware and/or software modules, made by the design of the system, of the hardware and of the software. The plan shall include the requirements for procedures and control methods covering:

– system configuration control (see 5.6);

– system integration;

– integrated system verification;

– fault resolution.

9.1.6 The system integration plan shall define both aspects of configuration management (identification and control) according to the requirements of 6.2.1.2 of IEC 61513.

9.1.7 In the process of verifying the individual hardware and software modules, certain aspects of the design of these modules may be verified at the level of subsystems (computer units) or at the level of the complete system if more practical. When verification by testing is not feasible at these levels, then all requirements of the individual module design shall be verified by other means.

9.1.8 All interdependencies between the verification of the individual modules and the verification of the integrated system shall be documented in the system integration plan.

9.2 Intégration du système

Les procédures propres à l'intégration du système dépendent des caractéristiques de l'architecture du système. Elles couvrent les points a) et c) de l'Article 9.

9.2.1 Ces procédures doivent être définies et documentées dans le plan d'intégration du système et doivent couvrir les activités suivantes:

– l'obtention des bons modules, en conformité avec le plan de gestion de configuration du système (6.2.1.2 de la CEI 61513);
– l'intégration des modules matériels dans le système (exemples: positions physiques des modules, adresses des mémoires, adresses des modules, câblage);
– l'édition de liens des modules logiciels et le chargement du logiciel cible dans le matériel cible;
– le test fonctionnel préliminaire des fonctions du système intégré (voir les exigences ci-après);
– la documentation du processus d'intégration et de la configuration du système devant être testée;
– la mise à disposition formelle du système intégré pour le test.

9.2.2 Si la résolution d'un défaut exige une modification du logiciel vérifié ou de tout autre document de conception, ce défaut doit donner lieu à un rapport selon la procédure de 9.4.

Tout défaut détecté durant l'intégration du système, résultant exclusivement du processus d'intégration, et n'affectant aucun document du projet, peut être corrigé sans rapport formel de défaut.

9.3 Vérification du système intégré

La vérification du système détermine si les modules logiciels et matériels préalablement vérifiés ainsi que les sous-systèmes ont été correctement intégrés dans le système, et s'ils sont compatibles et fonctionnent comme prévu.

9.3.1 Le système doit être aussi complet que possible pour ce test.

9.3.2 Les jeux d'essais choisis pour la vérification du système doivent activer toutes les interfaces des modules ainsi que le fonctionnement de base des modules eux-mêmes.

9.3.3 Le plan d'intégration du système doit démontrer que la simulation de toute partie du système ou de ses interfaces est indispensable et équivalente à la partie réelle.

9.3.4 Le plan d'intégration du système doit identifier les tests à réaliser pour chaque exigence relative à l'interface d'une unité programmée.

9.3.5 Le test du système intégré doit faire l'objet d'une revue. Les résultats doivent être évalués par une équipe de vérification. Celle-ci doit avoir une bonne connaissance des spécifications du système.

9.3.6 Les moyens utilisés pour la vérification du système doivent être étalonnés comme nécessaire.

9.3.7 Des mesures d'assurance qualité doivent être établies pour les outils logiciels utilisés pour la vérification, en rapport avec l'importance de ces outils pour cette activité.

9.2 System integration

The specific procedures for the system integration depend on the nature of the system design. They cover items a) to c) stated in Clause 9.

9.2.1 Such procedures shall be established and documented by the system integration plan, and shall cover the following activities:

– the acquisition of the correct modules according to the system configuration management plan (6.2.1.2 of IEC 61513);
– the integration of the hardware modules into the system (e.g. module position, memory address, selection, interconnection wiring);
– the linkage of software modules and the loading of the target software into the target hardware;
– the preliminary functional test of the integrated system functions (see requirements below);
– the documentation of the integration process and the system configuration that will be subjected to test;
– the formal release of the integrated system for testing.

9.2.2 If the resolution of a fault requires a modification to any verified software or any design document, that fault shall be reported according to the procedures established by 9.4.

Any faults detected during the system integration that are strictly mistakes in the integration process itself, and do not affect any project document, may be corrected without formal fault report.

9.3 Integrated system verification

The system verification determines whether or not the verified hardware and software modules and the subsystems have been properly integrated into the system and that they are compatible and perform as specified.

9.3.1 The system shall be as complete as is practical for this testing.

9.3.2 The test cases selected for system verification shall exercise all module interfaces as well as the basic operation of the modules themselves.

9.3.3 In the system integration plan, the simulation of any part of the system or its interfaces shall be demonstrated to be essential and equivalent to the actual part.

9.3.4 The system integration plan shall identify the tests to be performed for each computer unit interface requirement.

9.3.5 The integrated system test shall be reviewed and the test results evaluated by a verification team with a good knowledge of the system specification.

9.3.6 Equipment used for system verification shall be calibrated as required.

9.3.7 Quality assurance measures shall be established for software tools used for verification, commensurate with the importance of those tools for verification.

9.3.8 La vérification du système intégré doit démontrer que tous les composants du système ont des performances appropriées (exemples: les unités de traitement et les équipements de communication).

9.4 Procédures de résolution de défaut

9.4.1 Des procédures pour rendre compte des défauts détectés durant l'intégration du système et de leur correction doivent être établies avant que la vérification du système intégré ne commence.

9.4.2 Ces procédures doivent être appliquées à tous les défauts détectés lors de la vérification du système ou des tests fonctionnels, et nécessitant la modification de modules logiciels ou de documents de conception du système déjà vérifiés.

9.4.3 Elles doivent donner l'assurance que toute revérification de la conception du système, de modules logiciels ou de modules matériels est réalisée suivant le plan de gestion de configuration du système.

9.4.4 Elles doivent donner l'assurance que toute modification de la conception du système, du matériel ou du logiciel est réalisée suivant la procédure de modification de l'Article 11 et le plan de gestion de configuration du système.

9.4.5 Une évaluation de chaque défaut relevé doit être faite afin de déterminer s'il existe une défaillance systématique et aussi pour déterminer si le défaut aurait dû être détecté lors d'une étape antérieure de la vérification.

9.4.6 Si cela est le cas, une investigation de cette étape antérieure doit être conduite pour déterminer s'il existe une défaillance systématique dans la vérification.

9.4.7 Si l'évaluation des défauts révèle une défaillance systématique de la vérification, conduisant à la non-détection de défauts du logiciel, alors cette défaillance doit être identifiée, corrigée ou justifiée.

9.5 Aspects logiciels du compte rendu de vérification du système intégré

9.5.1 Les résultats de la vérification du système intégré doivent être inclus dans un compte rendu (Annexe F).

9.5.2 Celui-ci doit identifier le matériel et le logiciel utilisés, les moyens d'essai utilisés, leur étalonnage et le réglage des paramètres logiciels et matériels, la simulation de parties du système ou des interfaces, les résultats incorrects rencontrés en cours de vérification, et les actions correctives entreprises conformément à 9.4.

9.5.3 Les résultats d'essais doivent être consignés sous une forme permettant l'audit par des personnes non directement impliquées dans le plan de vérification.

9.5.4 La correction de tous les défauts relevés et les résultats de l'évaluation qui en découlent doivent être documentés avec suffisamment de détails et sous une forme permettant l'audit par des personnes non directement impliquées dans le développement du système et le plan de vérification.

9.3.8 The verification of the integrated system shall demonstrate that all system components have appropriate performances (e.g. processing units and communication devices).

9.4 Fault resolution procedures

9.4.1 Procedures for the reporting and resolution of faults found during system integration verification shall be established before integrated system verification begins.

9.4.2 These procedures shall apply to all faults found during the system verification as well as those found during the integration functional test that require modifications to verified software or system design documents.

9.4.3 They shall ensure that any required re-verification of system design, hardware or software modules is performed according to the system configuration management plan.

9.4.4 They shall ensure that any required modification of system design, hardware or software is carried out according to the modification procedure of Clause 11 and to the system configuration management plan.

9.4.5 An evaluation of each fault reported shall be made to determine whether any systematic deficiency exists and also to determine whether the fault was of such a nature that it should have been detected at an earlier phase of the verification.

9.4.6 If this is found to be the case (should have been detected at an earlier phase), then an investigation of that earlier stage of the verification shall be conducted to determine whether any systematic deficiency of the verification exists.

9.4.7 If the evaluation of faults shows that there is a systematic deficiency of the verification, causing faults in software modules to remain undetected, then the deficiency shall be identified, corrected or justified.

9.5 Software aspects of integrated system verification report

9.5.1 The results of the integrated system verification shall be documented in a report (Annex F).

9.5.2 This report shall identify the hardware and software used, the test equipment used, its calibration and software/hardware set-up parameters, the simulation of system or interface components, and any test results discrepancy found along with the corrective actions taken according to 9.4.

9.5.3 The test results shall be retained in a form that is auditable by persons not directly engaged in the verification plan.

9.5.4 The resolution of all reported faults and the results of the subsequent evaluation shall be documented in sufficient detail and in a manner that is auditable by persons not directly engaged in the system development and verification plan.

10 Aspects logiciels du plan de validation

a) Des tests doivent être réalisés pour valider le système et son logiciel en accord avec les exigences des systèmes de sûreté de classe 1 qui doivent être satisfaites par le système intégré.

b) La validation doit comprendre des tests réalisés sur le système dans la configuration d'assemblage final comprenant la version définitive du logiciel.

10.1 Aspects logiciels du plan de validation système

10.1.1 La validation du système doit être conduite en accord avec un plan de validation système formalisé.

10.1.2 Le plan doit identifier des jeux d'essais statiques et dynamiques.

10.1.3 Le plan de validation système doit être développé et les résultats de la validation évalués par des personnes n'ayant pas participé à la conception et à la réalisation.

10.2 Validation du système

10.2.1 Le système doit être sollicité par la simulation statique et dynamique des signaux d'entrée présents durant le fonctionnement normal, ainsi que lors des incidents d'exploitation et des conditions d'accident entraînant une action du système programmé faisant l'objet des tests.

10.2.2 Chaque fonction de sûreté du réacteur classée en catégorie A et réalisée par le système doit être contrôlée par des tests représentatifs de chaque variable intervenant dans l'arrêt d'urgence ou la protection, prise isolément et en combinaison.

10.2.3 Les tests doivent:

– couvrir la plage de variation des signaux et les gammes des variables élaborées ou calculées d'une manière pleinement représentative;
– couvrir les circuits de vote et les autres circuits logiques le plus complètement possible;
– être effectués pour tous les signaux d'arrêt d'urgence ou de protection dans la configuration finale du système;
– donner l'assurance que la précision et les temps de réponse sont respectés et qu'une action correcte est effectuée dans le cas d'une panne quelconque de l'équipement ou d'une combinaison de pannes;
– être effectués pour toute autre fonction qui a un impact direct sur la sûreté du réacteur (par exemple inhibition, verrouillage).

10.2.4 De plus, les valeurs des signaux d'entrée requis, les signaux de sortie attendus et les critères d'acceptation doivent être spécifiés dans le plan de validation du système.

10.2.5 Les équipements utilisés pour la validation doivent être étalonnés et configurés (paramètres matériels et logiciels) de façon appropriée.

10.2.6 Il est recommandé de montrer que les équipements utilisés sont adaptés aux besoins de la validation du système.

10 Software aspects of system validation

a) Testing shall be performed to validate the system and its software in accordance with the class 1 systems requirements to be satisfied by the integrated system.

b) Validation shall comprise tests performed on the system in the final assembly configuration including the final version of the software.

10.1 Software aspects of the system validation plan

10.1.1 The system validation shall be conducted in accordance with a formal system validation plan.

10.1.2 The plan shall identify static and dynamic test cases.

10.1.3 The computer system validation plan shall be developed and the results of the validation evaluated by individuals who did not participate in the design and implementation.

10.2 System validation

10.2.1 The system shall be exercised by static and dynamic simulation of input signals present during normal operation, anticipated operational occurrences and accident conditions requiring action by the computer-based system under test.

10.2.2 Each reactor category A function of the system shall be confirmed by representative tests of each trip or protection parameter singly and for relevant combinations.

10.2.3 The tests shall:

– cover all signal ranges, and the ranges of computed or calculated parameters in a fully representative manner;
– cover the voting and other logic and logic combinations comprehensively;
– be made for all trip or protective signals in the final assembly configuration;
– ensure that accuracy and response times are confirmed, and that correct action is taken for any equipment failure or failure combination;
– be made for all other functions which have a direct impact on reactor safety (e.g. vetoes, interlocks).

10.2.4 In addition, the required input signals and their values, the anticipated output signals and the acceptance criteria shall be stated in the system validation plan.

10.2.5 Equipment used for validation shall be calibrated and configured (hardware and software parameters) as appropriate.

10.2.6 Equipment used for validation should be shown to be suited to the purpose of the system validation.

10.3 Aspects logiciels du compte rendu de validation du système

10.3.1 Le compte rendu de validation du système doit rapporter les résultats des aspects logiciels de la validation du système

10.3.2 Le compte rendu doit identifier le matériel, le logiciel et la configuration du système utilisée, les équipements utilisés et leur étalonnage, les modèles de simulations utilisés.

10.3.3 Ce compte rendu doit aussi identifier les anomalies.

10.3.4 Ce compte rendu doit résumer les résultats de la validation du système.

10.3.5 Ce compte rendu doit montrer comment le système correspond aux spécifications.

10.3.6 Les résultats doivent être présentés de sorte qu'ils puissent être vérifiés par des personnes qui n'ont pas été directement impliquées dans la validation.

10.3.7 Il est recommandé de citer dans le compte rendu les outils logiciels utilisés dans le processus de validation.

10.3.8 Toute simulation de la centrale et de ses systèmes, utilisée pour la validation, doit être mentionnée.

10.4 Procédures de résolution de défaut

10.4.1 Des procédures spécifiques pour rendre compte et résoudre les défauts trouvés pendant la validation du système doivent être établies par le plan de validation du système.

10.4.2 Ces procédures doivent s'appliquer à tous les défauts trouvés pendant la validation du système et qui nécessitent des modifications de conception du système ou des modifications du logiciel.

10.4.3 Elles doivent assurer que toute revérification de la conception du système, du logiciel ou du matériel est réalisée en conformité avec le plan de gestion de configuration du système.

10.4.4 Elles doivent assurer que toute modification de la conception du système et du logiciel est faite en conformité avec la procédure de modification de l'Article 11 et le plan de gestion de configuration du système.

10.4.5 Une analyse de chaque défaut doit être faite pour déterminer si le défaut d'origine était de nature à être détecté plus tôt.

10.4.6 Si cela est le cas, une investigation de cette phase précédente doit être conduite pour déterminer s'il existe une faiblesse systématique.

11 Modification du logiciel

Une modification du logiciel est un changement apporté au logiciel, qui a généralement une incidence à la fois sur le code exécutable et sur la documentation.

Une modification du logiciel peut être demandée pour les raisons suivantes:

– modifications des exigences fonctionnelles,

– modifications de l'environnement du logiciel,

10.3 Software aspects of the system validation report

10.3.1 The system validation report shall document the results of the software aspects of the validation of the system.

10.3.2 The report shall identify the hardware, the software and the system configuration used, the equipment used and its calibration and the simulation models used.

10.3.3 This report shall also identify any discrepancies.

10.3.4 This report shall summarise the results of the system validation.

10.3.5 This report shall assess the system compliance with all requirements.

10.3.6 The results shall be retained in a form that is auditable by persons not directly engaged in the validation.

10.3.7 Software tools used in the validation process should be identified as an item in the validation report.

10.3.8 Simulations of the plant and its systems used for the validation shall be documented.

10.4 Fault resolution procedures

10.4.1 Procedures for the reporting and resolution of faults found during system validation shall be established and referenced by the system validation plan.

10.4.2 These procedures shall apply to all faults found during system validation that require modifications to system design or software.

10.4.3 They shall ensure that any required re-verification of system design, hardware or software is performed according to the system configuration management plan.

10.4.4 They shall ensure that the modification of system design and software is carried out according to the modification procedure of Clause 11 and to the system configuration management plan.

10.4.5 An evaluation of each fault reported shall be made to determine whether the initiating fault was of such a nature that it should have been detected at an earlier phase.

10.4.6 If this is found to be the case (should have been detected at an earlier phase), then an investigation of that earlier stage shall be conducted to determine whether any systematic deficiency exists.

11 Software modification

A software modification is a change made to the software which usually impacts both the executable code and the documentation.

A software modification may be requested for reasons such as:

– changes to functional requirements;

– changes to the software environment;

– modifications du matériel,

– anomalies détectées pendant un test ou en exploitation.

Une modification du logiciel peut être demandée pendant la phase de développement ou après livraison.

a) Avant mise en œuvre de toute modification du logiciel, une procédure formelle de contrôle des modifications du logiciel doit être établie et documentée, incluant des exigences pour la vérification et la validation.

b) Cette procédure doit indiquer comment traiter les exigences de cet article.

11.1 Procédure de demande de modification

11.1.1 Pour qu'une modification du logiciel soit prise en considération, il faut suivre les étapes suivantes:

– émission d'une demande de modification du logiciel;

– évaluation de la demande;

– décision.

11.1.2 Une demande de modification du logiciel doit être établie et identifiée de façon non ambiguë, en indiquant:

– sa raison d'être;

– son but;

– son domaine fonctionnel;

– l'identité du demandeur;

– la date de la demande.

11.1.3 La demande de modification du logiciel doit faire partie de la documentation de modification du logiciel. Si le logiciel est modifié dans le contexte d'une modification de conception du système, alors la documentation de la modification du logiciel doit être une partie de la documentation de la modification de conception du système.

11.1.4 La demande de modification doit être évaluée de façon indépendante.

11.1.5 L'évaluation de la demande de modification doit examiner sa pertinence à donner l'assurance que:

– les modifications proposées sont définies clairement et sans ambiguïté;

– les modifications proposées corrigent les causes de l'anomalie quand les modifications résultent d'un rapport d'anomalie;

– les modifications proposées ne dégradent pas l'aptitude du logiciel à fournir les fonctions de sûreté de catégorie A requises;

– le bénéfice de la mise en œuvre des modifications n'est pas contrecarré par la perturbation occasionnée par leur mise en application (des défaillances du système peuvent résulter d'une mauvaise conception ou mise en oeuvre des modifications).

11.1.6 Les points suivants doivent être examinés pour l'évaluation de la demande de modification:

– la faisabilité;

– changes to the hardware;
– anomalies found during test or operation.

A software modification may be requested during the development phase or after delivery.

a) Prior to the implementation of any software modification, a formal software modification control procedure shall be established and documented, which shall include requirements for verification and validation.
b) This procedure shall state how the requirements of this clause are addressed.

11.1 Modification request procedure

11.1.1 For a software modification to be considered, the following steps shall be followed:

– generation of a software modification request;
– evaluation of the request;
– decision.

11.1.2 A software modification request shall be generated and uniquely identified, stating:

– reason for request;
– aim;
– functional scope;
– originator;
– date of initiation.

11.1.3 The software modification request shall be included as a part of the documentation of the software modification. If the software is modified in the context of a system design modification, then the documentation of the software modification shall be a part of the documentation of the system design modification.

11.1.4 The modification request shall be evaluated independently.

11.1.5 The evaluation of the modification request shall examine its relevance to ensure that:

– the proposed changes have been clearly and unambiguously defined;
– the proposed changes will correct the causes of the anomaly where a change has resulted from an anomaly report;
– the proposed changes do not degrade the ability of the software to provide required category A safety functions;
– the benefits of implementing any changes are not outweighed by the disruption that implementing them may cause (system failures may be caused by incorrectly designed or implemented changes).

11.1.6 The following items shall also be examined in the evaluation of the modification request:

– technical feasibility;

– les conséquences sur le reste du système (par exemple, une extension mémoire) ou sur d'autres équipements (par exemple, les systèmes de test) auquel cas la demande correspondante de modification doit être documentée;

– les effets de modifications éventuelles des méthodes, outils ou normes à appliquer dans l'exécution de la modification (comparés à ceux mis en oeuvre pour le développement de la version du logiciel à modifier);

– les conséquences sur le logiciel lui-même, comprenant une liste des modules affectés;

– les conséquences sur les performances (vitesse, précision, etc.);

– la stratégie et l'effort nécessaire de vérification et de validation pour assurer que le logiciel existant reste correct; l'analyse de la vérification ultérieure nécessaire du logiciel doit être explicitée de manière à permettre un audit;

– la liste des documents à revoir.

Le processus d'évaluation peut être constitué d'un certain nombre de phases.

La demande initiale peut être revue pour sa pertinence et sa faisabilité avant qu'un travail d'évaluation détaillée de l'impact ne soit réalisé.

Une fois l'impact total évalué, une deuxième évaluation plus complète peut être réalisée.

La demande de modification du logiciel reste en suspens jusqu'à ce qu'une décision soit prise, qui peut être:

– le rejet de la demande; dans ce cas, celle-ci est retournée avec des commentaires justificatifs;

– la demande d'une analyse détaillée conduisant à l'élaboration d'un compte rendu d'analyse des incidences sur le logiciel;

– l'acceptation et le traitement de la demande.

11.1.7 Si un compte rendu d'analyse des incidences sur le logiciel est demandé, ce rapport doit être rédigé par un spécialiste du logiciel du système.

11.2 Procédure d'exécution d'une modification du logiciel

11.2.1 Pour les modifications à mettre en œuvre sur un équipement fonctionnel où il n'est pas pratique de réaliser les tests adéquats du fait même de l'exploitation, il convient que le fournisseur de logiciel ait accès à une configuration de tests identique au système réel dans tous ses aspects significatifs (comprenant les machines installées, les traducteurs, les outils de test, les simulateurs du procédé, etc.) pour assurer la validité des modifications.

11.2.2 La procédure de modification dépend du niveau où elle est introduite:

– en cas de modification des exigences de spécification du logiciel, un nouvel examen de toute la procédure de développement du logiciel, en ce qui concerne la partie du système touchée par la modification doit être effectué;

– un changement pendant la phase de développement doit être contrôlé en ce qui concerne les incidences sur les niveaux inférieurs correspondants;

– les modifications doivent être traitées selon les règles figurant dans l'Article 7.

11.2.3 Après réalisation d'une modification, le processus complet ou partiel de vérification et de validation décrit en 8.1 et dans l'Article 10 doit être à nouveau effectué selon l'analyse des incidences de la modification (voir 11.1).

– impact upon the rest of the system (e.g. memory extension) or upon other equipment (e.g. test systems) in which case the request for modification addressing this impact area shall be documented;
– effects of possible changes in the methods, tools or standards to be applied in the execution of the modification (compared to those which were applied for the development of the version of the software to be modified);
– impact upon software itself, including a list of affected modules;
– impact upon performance (including speed, accuracy, etc.);
– strategy and necessary effort for verification and validation to ensure that the correctness of the existing software is maintained; the analysis of the software re-verification needed shall be documented in an auditable form;
– the set of documents to be reviewed.

The evaluation process may consist of a number of phases.

The initial request may be reviewed for relevance and feasibility before any detail impact assessment work has been performed.

When the full impact has been evaluated, a second more thorough evaluation may be performed.

The software modification request is pending until the decision is made, which may be:

– to reject the request; in this case, it is sent back with justification;
– to require a detailed analysis, resulting in a software modification analysis report;
– to accept and process the request.

11.1.7 If a software modification analysis report has been required, this report shall be written by software personnel knowledgeable in the software of the system.

11.2 Procedure for executing a software modification

11.2.1 For modifications which are to be implemented on operational equipment where it is not practical to perform adequate testing due to operational considerations, the software supplier should have access to a test configuration which is identical to the real system in all relevant aspects (including installed machine, translator, testing tools, plant simulator, etc.) to ensure the validity of the modifications.

11.2.2 The modification procedure appropriate for any particular change will depend upon the affected phases of the development process:

– for a change of the software requirements specification, the whole software development process for any part of the I&C system impacted by the change shall be re-examined;
– a change during the development shall be reviewed in terms of its potential impact upon corresponding lower levels;
– the modification shall be carried out according to the rules given in Clause 7.

11.2.3 After implementation of the modification, the whole or part of the verification and validation process described in 8.1 and Clause 10, shall be performed again according to the software modification impact analysis (see 11.1).

11.2.4 Tous les documents concernés par la modification doivent être corrigés et faire référence à l'identification de la demande de modification logicielle.

11.2.5 Un compte rendu de modification du logiciel doit résumer toutes les actions menées pour les besoins de la modification.

11.2.6 Tous ces documents doivent être datés, numérotés et classés dans l'historique de modification du logiciel du projet.

11.2.7 La stratégie de déploiement d'un logiciel modifié dans une centrale en exploitation doit être évaluée d'après son incidence sur la stratégie de vote, la maintenance et les données échangées. Suivant les résultats de cette évaluation, les modifications du logiciel peuvent être déployées graduellement, permettant de tester à tour de rôle les nouveaux logiciels sur un seul canal tandis que l'exploitation se poursuit avec les autres canaux non modifiés.

11.3 Modification du logiciel après livraison

Les causes de telles modifications peuvent être:

– un compte rendu d'anomalie;
– une modification des spécifications fonctionnelles après livraison;
– une évolution technologique;
– un changement dans les conditions d'exploitation.

11.3.1 En cas d'anomalie, un compte rendu d'anomalie doit être établi donnant les symptômes, l'environnement et l'état du système lors de la découverte de l'incident, ainsi que les causes suspectées.

11.3.2 Si un comportement inattendu, apparemment incorrect, non expliqué ou anormal apparaît après livraison, il convient qu'un compte rendu d'anomalie soit établi par le personnel de conduite donnant les détails du comportement, les configurations du matériel et du logiciel et les activités en cours à ce moment.

11.3.3 Il convient que ce compte rendu d'anomalie indique l'identité du demandeur, la localisation, la date, les circonstances et un numéro de série. Il convient que de tels comptes rendus soient revus par l'équipe de développement, qu'il leur soit attribué des catégories d'importance et qu'ils soient conclus par une réponse faite au personnel d'exploitation.

Il est préférable que le compte rendu d'anomalie et la procédure d'élimination des erreurs soient développés en suivant un processus similaire à celui adopté pendant la vérification et la validation.

11.3.4 La correction du défaut à l'origine de l'anomalie requiert l'élaboration d'une demande de modification du logiciel qui doit suivre la procédure décrite en 11.1.

11.3.5 En cas de modifications des spécifications du logiciel, tout le processus de développement du logiciel doit être repris pour ce qui concerne la partie du système touchée par la modification.

11.3.6 De nouvelles exigences et possibilités du matériel doivent être évaluées en ce qui concerne leur impact potentiel sur le logiciel.

11.2.4 All the documents affected by the modification shall be corrected and refer to the identification of the software modification request.

11.2.5 A software modification report shall sum up all the actions made for modification purposes.

11.2.6 All these documents shall be dated, numbered and filed in the software modification control history for the project.

11.2.7 The strategy for deployment of modified software in an operating plant shall be evaluated in respect of its impact on voting strategy, maintenance and data communications. Depending upon the outcome of this evaluation, software modifications may be deployed gradually, allowing testing of new software on each single redundancy train in turn with parallel operation with the non-modified trains.

11.3 Software modification after delivery

The reasons for such modification include:

– an anomaly report;

– a functional requirements change after delivery;

– technological evolution;

– a change in operating conditions.

11.3.1 In case of an anomaly, an anomaly report shall be written giving the symptoms, the system environment and system status at the time at which the anomaly was discovered and the suspected causes.

11.3.2 If unexpected, apparently incorrect, unexplained or abnormal behaviour is experienced after delivery, an anomaly report should be raised by the operating staff giving details of the behaviour, the software and hardware configurations and the activities in hand at the time.

11.3.3 The report should include the originator, location, date, circumstances and a serial number. The reports should be reviewed by the development team, assigned to categories of importance and resolved by a response to the operational staff.

It is an advantage if the anomaly report and clearance procedure is developed from a similar process adopted during verification and validation.

11.3.4 Fault correction requires the generation of a software modification request, the execution of which shall follow the procedure described in 11.1.

11.3.5 In case of a change in the software requirements specification, the whole software development process shall be re-examined for that part of the system impacted by the change.

11.3.6 Any new hardware requirements and capabilities shall be examined with respect to their potential impact on the software.

11.3.7 Pour cette évaluation, il convient de prendre en considération les caractéristiques du matériel examinées lors de la conception initiale du logiciel.

S'il est possible de montrer que la nouvelle configuration matérielle du système n'a pas d'incidence sur les spécifications du logiciel, une procédure simplifiée peut être suivie pour mettre en œuvre la modification soit au niveau de la conception, soit au niveau du codage.

Le point b) de 6.3.6.1 de la CEI 61513 recommande d'inclure la modification du logiciel une fois terminée, dans un lot de modifications. La documentation de ce lot de modifications doit décrire les moyens de mise en œuvre de la modification sur l'équipement fonctionnel ou faire référence à des procédures existantes approuvées.

11.3.8 Dans tous les cas, après implantation de la modification sur l'équipement du site, un document doit être établi, donnant des informations sur la date de la réalisation et les résultats des tests et observations spécifiés en accord avec la procédure de réalisation.

11.3.9 Ce document doit être classé dans l'historique de modification du logiciel relatif au projet.

12 Aspects logiciels de l'installation et de l'exploitation

Cet article donne des exigences pour l'interaction des opérateurs avec les systèmes programmés de classe 1 pendant l'installation et l'exploitation. Ces exigences traitent de:

– l'installation du logiciel sur site;
– la sécurité informatique sur site;
– l'adaptation du logiciel aux conditions sur site;
– la formation.

12.1 Installation du logiciel sur site

Un programme de tests doit être fourni pour vérifier l'intégrité du logiciel installé en ce qui concerne les temps de réponse, les réglages, les opérations fonctionnelles et les interactions avec les autres systèmes.

12.2 Sécurité informatique sur site

12.2.1 Une évaluation de la sécurité informatique de la configuration et de la modification des paramètres sur site doit être réalisée pour vérifier que des contre-mesures adaptées ont été mises en place contre les menaces potentielles.

12.2.2 Quand cela est applicable, le logiciel doit être configuré et paramétré pour rassembler les informations pertinentes pour l'audit et le rapport périodiques sur la sécurité informatique du système d'I&C.

12.2.3 Les activités de maintenance du logiciel doivent être systématiquement préparées en prenant en compte les menaces potentielles sur la sécurité informatique.

12.2.4 A partir du principe de sécurité informatique mis en place pour le fonctionnement normal de la centrale, les écarts nécessaires à la mise en service et à la maintenance du logiciel doivent être identifiés.

Les écarts à considérer peuvent comprendre les options particulières du logiciel bloquées pendant l'exploitation, les fonctions d'alarme bloquées pendant la maintenance, l'utilisation d'interfaces bloquées ou arrêtées pendant l'exploitation, et l'utilisation d'outils et de services, de même que la présence du personnel de maintenance.

11.3.7 This evaluation should include all hardware considerations reviewed in the original software design.

If it can be shown that the modified system does not impact the software requirements specification, a simplified procedure may be used to implement the modification either at the design or coding phase.

Item b) of 6.3.6.1 of IEC 61513 recommends that when the software modification is completed, it is included in a change package. It also requires that the documentation of this package describes the means of implementing the modification upon the operational equipment, or a reference to approved existing procedures may be provided.

11.3.8 In all cases, after implementation of the modification upon the operational equipment, a document shall be issued which gives the date of the implementation and the results of any specified tests or observations required by the implementation procedure.

11.3.9 This document shall be filed in the software modification control history for the project.

12 Software aspects of installation and operation

This clause provides requirements for the interaction of operators with class 1 computer-based systems during installation and operation. These requirements address:

– the on-site installation of software;

– the on-site software security;

– the adaptation of software to on-site conditions;

– training.

12.1 On-site installation of the software

A test procedure shall be provided to verify the integrity of the installed software with respect to response, calibration, functional operation and interaction with other systems.

12.2 On-site software security

12.2.1 A security assessment of the on-site configuration and parameter assignment shall take place to verify that suitable countermeasures have been implemented against potential security threats.

12.2.2 When applicable the software shall be configured and parameterised so as to collect all relevant information for the periodic security auditing and reporting of the I&C system.

12.2.3 Software modification activities shall be systematically prepared taking into account potential security threats.

12.2.4 Proceeding from the security concept for normal plant power operation, the deviations which are necessary to perform the planned software commissioning and modification activities shall be identified.

Deviations to be considered may include special software options which are blocked during power operation, alarm annunciation functions which are blocked during modification, the use of interfaces which are blocked or switched off during power operation and the use of service stations and tools, as well as the required local presence of modification personnel.

12.2.5 Tout écart doit être compensé par des moyens supplémentaires d'assurance qualité, tels que des mesures analytiques et administratives pendant et/ou après les activités de maintenance pour assurer l'intégrité du logiciel.

12.2.6 Il convient que les outils et équipement utilisés pour les activités de maintenance du logiciel soient qualifiés suivant les menaces de sûreté potentielles pour le système.

12.2.7 Les nouveaux fichiers de données ou les nouvelles versions logicielles mettant en place des modifications liées à la sécurité informatique doivent être vérifiés pour confirmer que les exigences relatives à la sécurité informatique ont été prises en compte correctement.

12.2.8 La procédure d'installation de logiciels ou données sur site doit comprendre et spécifier des vérifications de l'intégrité du logiciel à réaliser avant que le système d'I&C soit en exploitation opérationnelle normale.

12.3 Adaptation du logiciel aux conditions sur site

Une procédure appropriée et/ou un verrouillage doivent être établis pour empêcher que l'opérateur modifie par inadvertance ou incorrectement les paramètres qui peuvent affecter les points de consigne ou autres données modifiables du système de sûreté de classe 1.

12.4 Formation des opérateurs

12.4.1 Programme de formation

Afin d'obtenir une exploitation sûre de la centrale, le comportement de l'exploitant est aussi important que la fiabilité de l'équipement.

12.4.1.1 Un programme de formation des opérateurs pour le système de protection et ses logiciels doit donc être fourni à la fois aux exploitants de la centrale et aux spécialistes de l'instrumentation et du contrôle-commande, formation en rapport avec la complexité des fonctions de protection mises en œuvre.

12.4.1.2 Le programme de formation doit traiter des conditions de fonctionnement aussi bien normales qu'anormales.

12.4.1.3 Toutes les interfaces de l'opérateur du système programmé doivent être incluses dans la formation.

12.4.1.4 Il est recommandé qu'une formation spécifique à la reconnaissance des anomalies matérielles et logicielles soit aussi incluse dans le programme.

12.4.2 Plan de formation

12.4.2.1 Un plan de formation homogène avec les principes du programme de formation doit être établi.

12.4.2.2 Un manuel d'utilisation du système d'I&C doit être disponible pour l'exploitant du réacteur.

12.4.2.3 Il convient que ce manuel définisse chaque élément d'interface opérateur. Chaque fonction de chaque dispositif doit être expliquée et illustrée selon sa complexité.

12.4.3 Système de formation

12.4.3.1 La formation des exploitants doit être conduite sur un système de formation qui est équivalent au véritable système matériel/logiciel.

12.4.3.2 La stimulation de ce système doit être réalisée par un système de test capable de simuler les conditions normales et anormales du réacteur.

12.2.5 Any deviations shall be compensated for by additional means of quality assurance, such as additional administrative and analytical measures during and/or after the modification activities to ensure the integrity of the software.

12.2.6 Tools and equipment used for on-site software modification should match a level appropriate to their potential threat to the security of the system.

12.2.7 New data files or new software versions implementing a security-related change shall be verified to confirm that the security requirements have been properly addressed.

12.2.8 The procedure for installing software or data on-site shall include and specify checks of the software integrity to be performed before the I&C system is put into full operational use.

12.3 Adaptation of the software to on-site conditions

An appropriate procedure and/or locking device shall be established to prevent the operator inadvertently or incorrectly changing parameters that can affect the set points or other modifiable data items of the class 1 system.

12.4 Operator training

12.4.1 Training programme

In order to achieve safe plant operation, operator behaviour is as important as equipment reliability.

12.4.1.1 Therefore an operator training programme for the safety systems and its software use shall be provided both for plant operators and instrumentation and control specialists consistent with the complexity of the protective functions implemented.

12.4.1.2 The training programme shall address operations under normal and abnormal plant conditions.

12.4.1.3 The training programme shall address all relevant computer-based system operator interface devices.

12.4.1.4 Specific training in the recognition of hardware and software abnormalities should also be included in the programme.

12.4.2 Training plan

12.4.2.1 A training plan consistent with the principles of the training programme shall be established.

12.4.2.2 A user manual for the I&C system shall be provided for use by the operations and maintenance staff.

12.4.2.3 The user manual should define each operator interface device. Each function of each device shall be explained and illustrated in accordance with its complexity.

12.4.3 Training system

12.4.3.1 Operator training shall be conducted on a training system which is equivalent to the actual hardware/software system.

12.4.3.2 The plant stimulus to this system shall be provided by a test system capable of simulating normal and abnormal reactor conditions.

13 Moyens de défense contre les défaillances logicielles de cause commune

Cet article énonce des exigences en matière de moyens de défense contre les défauts de conception et codage qui peuvent amener à des défaillances de cause commune (CCF) de fonctions classées dans la Catégorie A, conformément à la CEI 61226.

13.1 Généralités

Une CCF peut intervenir dans les systèmes et équipements de l'architecture d'I&C qui réalisent différentes lignes de défense pour le même PIE (voir 5.3.1 de la CEI 61513). Le logiciel par lui-même n'a pas de CCF. Les CCF sont relatives à des défaillances qui ont leur origine dans des erreurs des exigences fonctionnelles, de la conception du système ou du logiciel.

Pour toutes les activités de sûreté, qu'elles soient relatives à l'organisation, au comportement ou à la conception, l'AIEA demande d'appliquer la défense en profondeur (Publication NS-R-1 de l'AIEA, 2.9), pour assurer que ces activités ont des recouvrements, de façon que si une défaillance intervient dans un sous-système, elle puisse être compensée ou corrigée dans le système total.

Le critère de défaillance unique (Publication NS-R-1 de l'AIEA, 5.34 à 5.39) stipule que l'ensemble des systèmes de sûreté doivent pouvoir remplir leur mission malgré une défaillance aléatoire unique pouvant intervenir n'importe où dans cet ensemble.

13.1.1 Les défauts logiciels sont des défauts systématiques et non des défauts aléatoires. Par conséquent, le critère de défaillance unique ne peut pas être appliqué à la conception d'un système de la même manière qu'il a été appliqué pour le matériel. Ainsi, lorsque le concept de défense en profondeur est appliqué, il convient de prendre en compte les éventuels effets des CCF logicielles dans chacune des lignes de défense et entre lignes redondantes. Il convient d'adopter des contre-mesures appropriées pendant le processus de développement et dans le processus d'évaluation (si les CCF logicielles sont une source potentielle de défaillance), par exemple:

1) dans la conception et la réalisation, la vérification et la validation de chacune des lignes de défense; et
2) dans l'évaluation de l'indépendance et de la diversité des lignes de défense redondantes.

L'utilisation de la diversité est un des moyens pour améliorer la fiabilité de certains systèmes et pour réduire le risque de certaines CCF (voir Articles 4.23 à 4.31 du Guide de Sûreté NS-G-1.3 de l'AIEA).

La justification de la défense contre les CCF logicielles est que tout défaut logiciel restera dans le système ou le canal concerné jusqu'à ce qu'il soit détecté et corrigé, et qu'il peut causer une défaillance si une trajectoire de signal particulière la sollicite. Si deux ou plusieurs systèmes ou canaux constituant différentes lignes de défense pour le même PIE (voir 5.3.1.5 de la CEI 61513) contiennent le défaut et sont sollicités par des trajectoires de signal spécifiques au cours d'une période sensible, les deux (ou tous) systèmes ou canaux connaîtront une défaillance, qui sera une CCF. Une description plus détaillée de ces états est donnée à l'Article G.1.

13.1.2 Il convient donc de prendre en compte le risque potentiel de CCF logicielle lors de la conception. Si des conditions de CCF hypothétiques peuvent être prévues, des changements de conception et des fonctionnalités de défense, y compris la diversité logicielle, peuvent être nécessaires pour assurer la protection contre les CCF logicielles.

Le degré d'amélioration des défenses contre les CCF et les améliorations de la fiabilité qui peuvent être obtenues grâce à la diversité ne peuvent pas être quantifiées. Le concepteur utilisera son meilleur jugement sur la base d'une évaluation qualitative de la fiabilité que peut assurer le logiciel.

13 Defences against common cause failure due to software

This clause provides requirements for defences against software design and coding faults which can lead to common cause failures (CCF) of functions classified as category A according to IEC 61226.

13.1 General

CCF may occur in the I&C systems and equipment implementing different lines of defence against the same PIE (see 5.3.1 of IEC 61513). Software by itself does not have a CCF mode. CCF is related to system failures arising from faults in the functional requirements, system design, or in the software.

Defence in depth is required by the IAEA (see 2.9 of IAEA Requirements NS-R-1) to be applied to all safety activities, whether organisational, behavioural or design related, to ensure that there are overlapping defences so that if a failure should occur in a subsystem, it would be compensated for or corrected in the integral system.

The single-failure criterion (see 5.34 to 5.39 of IAEA Requirements NS-R-1) requires that the assembly of safety systems has the ability to meet its purpose despite a single random failure assumed to occur anywhere in the assembly.

13.1.1 Software faults are systematic not random faults and therefore the single-failure criterion cannot be applied to the software design of a system in the same manner as it has been applied for hardware. When the defence-in-depth concept is applied, possible effects of CCF due to software inside each defence layer and between redundant layers should be considered and appropriate countermeasures should be adopted throughout the development process and in the evaluation processes (if software CCF is a potential failure mode), for example

1) in the design and implementation, verification and validation of each individual defence layer; and
2) in the evaluation of the independence and diversity of redundant defence layers.

A means of enhancing the reliability of some systems and reducing the potential for certain CCFs is the use of diversity (see 4.23 to 4.31 of IAEA Safety Guide NS-G-1.3).

The rationale for defence against software faults is that any software fault will remain in the system or channel concerned until detected and corrected, and can cause failure if a specific signal trajectory challenges it. If two or more systems or channels implementing different lines of defence for the same PIE (see 5.3.1.5 of IEC 61513) contain the fault, and are exposed to specific signal trajectories within a sensitive time period, both (or all) systems or channels can fail, which is called a CCF. A more detailed description of these conditions is given in Clause G.1.

13.1.2 The potential for CCF due to software should therefore be considered during design. If postulated conditions of CCF can be foreseen, design changes and defence features, including software diversity, may be needed for protection against CCF due to software.

The degree of improvement of defence against CCF and improvement in reliability that can be achieved by diversity cannot be quantified. Judgement is required based on an evaluation of the qualitative reliability which the software can achieve.

Les erreurs humaines intervenant avant le début de la conception logicielle donnent lieu à des défauts de spécification et à des défaillances potentielles du système que le seul génie logiciel ne peut prévenir. La défense contre ces types de CCF est traitée au niveau système en 5.3.1.5 de la CEI 61513.

Des erreurs humaines intervenant pendant le processus de génie logiciel peuvent donner lieu à des défauts logiciels et à des défaillances potentielles. Lorsque de tels défauts provoquent la défaillance de plus d'une ligne de défense, les défaillances sont considérées comme des CCF logicielles.

13.2 Conception du logiciel pour éviter les CCF

Le moyen de défense élémentaire et le plus important contre les CCF logicielles est de produire des logiciels de la plus haute qualité, c'est-à-dire contenant le moins d'erreurs possible. L'étendue de la couverture des moyens d'autocontrôle, tels que la surveillance de plausibilité des données, le contrôle d'échelle des paramètres, la temporisation cyclique, etc. (voir 6.2, 7.1 et A.2.2), constituent un autre facteur important pour limiter les CCF logicielles potentielles.

L'utilisation de méthodes éprouvées de génie logiciel avec des outils d'aide au développement et à la vérification du logiciel peut réduire le nombre de décisions humaines dans la conception et peut ainsi réduire le nombre de défauts dans le logiciel produit.

13.3 Sources et effets des CCF logicielles

13.3.1 Une analyse des CCF logicielles potentielles doit être effectuée et documentée au niveau du système et/ou au niveau de l'architecture globale des systèmes d'I&C importants pour la sûreté.

NOTE 1 Des exigences sur l'architecture I&C sont données en 5.3.1 de la CEI 61513.

NOTE 2 Des exigences sur l'architecture des systèmes individuels d'I&C sont données en 6.1.2 de la CEI 61513.

13.3.2 Il convient que l'analyse comprenne les étapes suivantes:

1) identification des composants logiciels utilisés dans le système ou dans l'architecture du contrôle commande;
2) analyse des CCF potentielles dues à ces composants dans le cadre du système ou de l'architecture du contrôle commande;
3) analyse des effets possibles de ces CCF.

NOTE L'analyse des effets potentiels des défauts ne supprime pas le besoin d'effectuer les activités de vérification et validation requises par cette norme à l'Article 8. Le but d'une telle analyse est de mettre en évidence toutes les faiblesses dans la conception, afin d'y initier des changements et/ou d'améliorer la confiance dans la conception du logiciel.

13.3.3 Les modules communs utilisés dans plusieurs systèmes doivent être identifiés et l'assurance de la fiabilité de ces modules doit être démontrée. Des méthodes pour mener cette démonstration sont donnés à l'Article G.4.

13.3.4 Les données transmises à l'intérieur du système informatique ou entre différents systèmes informatiques doivent être identifiées. Une analyse doit être effectuée pour déterminer si des données erronées peuvent provoquer une CCF dans les processeurs ou les systèmes en réception.

13.3.5 La possibilité que, dans certaines conditions propres à la centrale, les mêmes logiciels implantés dans différents matériels soient soumis simultanément à des trajectoires de signaux identiques et qu'un même défaut logiciel apparaisse dans plusieurs canaux ou chemins fonctionnels, doit être évaluée.

If human errors are made before software design starts, they may lead to faults of requirements and potential system failures against which software engineering alone cannot provide a defence. Defence against such CCF is discussed at the system level in 5.3.1.5 of IEC 61513.

If human errors are made during the software engineering process, they may lead to software faults and potential system failures. Where such faults lead to the failure of more than one line of protection the failures are considered to be CCFs due to software.

13.2 Design of software against CCF

The basic, and most important, defence against CCF due to software is to produce software of the highest quality, i.e. as error free as possible. The extent of coverage of self-monitoring features, such as for data plausibility, parameter range checking, and loop timing, etc. as addressed by 6.2, 7.1 and A.2.2 is a further important factor in limiting the potential for CCF due to software.

The use of well-developed software engineering methods with software tool support for software development and verification can help to reduce the number of human design decisions and so potentially reduce the number of faults in the developed software.

13.3 Sources and effects of CCF due to software

13.3.1 An analysis of the potential for CCF due to software shall be performed and documented at the system level and/or at the level of the total I&C architecture of the I&C systems important to safety of the NPP.

NOTE 1 Requirements on the I&C architecture are given in 5.3.1 of IEC 61513.

NOTE 2 Requirements on the architecture of the individual I&C systems are given in 6.1.2 of IEC 61513.

13.3.2 The analysis should include the following steps:

1) identification of the software components used in the system or I&C architecture;
2) analysis of the potential CCF due to these components within the system or the I&C architecture;
3) analysis of the possible effects of these CCF.

NOTE Performing an analysis of the potential effects of faults does not obviate the need to perform verification and validation activities as required by this standard in Clause 8. The purpose of such an analysis is to reveal any weaknesses in the design, and hence initiate changes to the design, and/or to improve confidence in the software design.

13.3.3 If common modules are used in more than one system, they shall be identified and assurance of the reliability of such common modules shall be assessed. Methods supporting the demonstration of correctness are given in Clause G.4.

13.3.4 Data transmitted inside a computer-based system or between computer-based systems shall be identified. An analysis shall be performed to determine if faulty data can lead to a CCF in receiving computers or systems.

13.3.5 The potential for plant conditions to subject the same software running in different hardware to identical and simultaneous signal trajectories and hence reveal the same software fault in several channels or functional paths shall be assessed.

NOTE Les défaillances peuvent être provoquées par des trajectoires de signaux qui n'ont pas été pris en compte lors de la conception, de la vérification et de la validation des logiciels des canaux individuels ou systèmes.

13.3.6 Les activités de modification du logiciel (voir l'Article 11) constituent une source potentielle de CCF. Il convient que la procédure de modification du logiciel ou des données donne l'assurance que ces types de défauts ne sont pas introduits.

13.3.7 Une analyse des sources potentielles de CCF logicielles doit être faite et documentée dans le cadre du jugement sur la défense contre les CCF propres à la conception de l'architecture I&C (voir 5.3.3 de la CEI 61513).

13.3.8 Si l'analyse identifie une menace inacceptable résultant de CCF provoquées par le logiciel, la conception du logiciel ou de l'architecture du contrôle commande doit être améliorée. Des méthodes pour la réalisation de défenses contre les CCF sont données à l'Article G.3, et des méthodes pour la mise en œuvre de la diversité sont données à l'Article G.5.

13.4 Mise en oeuvre de la diversité

13.4.1 Il convient que la mise en oeuvre de la diversité utilise des systèmes indépendants avec diversité fonctionnelle. Si la diversité fonctionnelle n'est pas appropriée ou possible, il convient de prendre en considération la diversité des systèmes, la diversité des caractéristiques des logiciels et la diversité des approches de conception. Les caractéristiques importantes sont données à l'Article G.5. Les techniques utilisées pour la défense contre les CCF doivent être documentées et justifiées par rapport à l'analyse qui a été effectuée.

13.4.2 Au niveau du logiciel, il convient que la défense contre les CCF soit basée sur un ensemble de techniques approprié, comme:

1) des garanties sur la diversification des conditions d'exploitation du logiciel;
2) des protections contre les erreurs et la propagation des défaillances;
3) la réduction des effets négatifs des CCF;
4) l'utilisation de logiciels diversifiés répondant à des spécifications différentes, pour des réalisations différentes de la même exigence fonctionnelle.

NOTE 1 Il convient de prendre en considération la possibilité des différencier les méthodes de conception et d'implémentation mais ceci n'est pas une exigence.

NOTE 2 La technique du logiciel N-version n'est pas recommandée.

13.5 Pondération des inconvénients et des avantages liés à l'utilisation de la diversité

Si la diversité logicielle est utilisée et affirmée, il convient que les inconvénients et avantages sur la fiabilité globale du logiciel soient justifiés sur la base de l'analyse ci-dessus, et documentés (voir Guides de Sûreté NS-G-1.3 de l'AIEA, Article 4.27 et NS-G-1.2, Articles 3.81 à 3.85). Les avantages, inconvénients et aspects liés à la justification sont donnés à l'Article G.6.

14 Outils logiciels pour le développement de logiciels

14.1 Généralités

Le présent paragraphe développe les exigences existantes de la présente norme pour les outils logiciels utilisés dans le développement de logiciels destinés aux calculateurs des systèmes de sûreté des centrales nucléaires.

NOTE Failures may be caused by signal trajectories which were not considered during the design, verification and validation of the software of the individual channels or systems.

13.3.6 Software modification activities (see Clause 11) have the potential to cause CCF and the processes used for software or data change assessment should provide assurance that such faults are not introduced.

13.3.7 The analysis of potential CCF due to software shall be performed and documented as part of the assessment of the defence against CCF of the design of the I&C architecture (see 5.3.3 of IEC 61513).

13.3.8 If the analysis identifies an unacceptable threat arising from CCF due to software then the design of the software or of the I&C architecture shall be improved. Methods supporting the implementation of CCF defences are given in Clause G.3 and methods supporting the implementation of diversity features are given in Clause G.5.

13.4 Implementation of diversity

13.4.1 Implementation of diversity should use independent systems with functional diversity. If functional diversity is not appropriate or possible, the use of system diversity, diverse software features and diverse design approaches should be considered. Features of importance are given in Clause G.5. The techniques chosen for defence against CCF shall be documented and justified according to the analysis made.

13.4.2 At the software level, defences against CCF should be based on an appropriate selection of techniques, such as

1) guarantee of diversified operational conditions of the software,
2) defences against error and failure propagation,
3) reduction of the negative effects of CCF,
4) use of software diversified by different specifications for different implementations of the same functional requirement.

NOTE 1 Differences in design and implementation methods should be considered for inclusion but are not required.

NOTE 2 N-version programming is not recommended.

13.5 Balance of drawbacks and benefits connected with the use of diversity

If diverse software is used and claimed, the drawbacks and benefits on the overall reliability of the software should be justified on the basis of the above analysis, and documented (see 4.27 of IAEA Safety Guide NS-G-1.3, and 3.81 to 3.85 of IAEA Safety Guide NS-G-1.2). Potential benefits, drawbacks and justification aspects are given in Clause G.6.

14 Software tools for the development of software

14.1 General

This subclause expands on existing requirements of this standard for software tools used in the development of software for computers in safety systems of nuclear power plants.

L'utilisation des outils logiciels appropriés peut augmenter l'intégrité du processus de développement du logiciel, et donc la fiabilité du produit logiciel, en réduisant le risque d'introduction d'erreurs pendant le processus. L'utilisation d'outils peut également présenter des avantages économiques car le temps et le travail nécessaires à la production du logiciel peuvent être réduits. Les outils peuvent être utilisés pour contrôler automatiquement le respect des règles et normes de construction, générer des enregistrements adéquats et une documentation cohérente aux formats standards, et permettre le contrôle des modifications. Les outils peuvent également réduire les travaux de test et permettre la tenue automatique de journaux. Les outils peuvent aussi être nécessaires car requis par une méthode spécifique de développement.

14.1.1 Les outils sont particulièrement puissants lorsqu'ils sont définis pour fonctionner conjointement. Il convient de prendre garde à ne pas attendre des outils qu'ils réalisent des tâches qui dépassent leurs capacités. En d'autres termes, ils ne peuvent remplacer l'homme lorsqu'il convient de faire appel à son jugement. Dans certains cas, le soutien par des outils est mieux adapté que l'automatisation complète du processus. Lors de la sélection d'un outil, il faut faire la balance entre les avantages et risques liés à son utilisation et les avantages et risques liés à sa non-utilisation. Le principe important est de choisir des outils qui limitent les possibilités d'introduction d'erreurs mais qui maximisent la capacité à les détecter.

Les outils qui entrent dans le cadre de cette norme sont ceux utilisés pour permettre la capture des exigences et ceux utilisés pour permettre la transformation des exigences en codes et en données système finals (il peut y avoir un grand nombre d'étapes intermédiaires). Cette partie recouvre également les outils utilisés pour permettre de procéder directement à la vérification, à la validation et au test, les outils utilisés pour la préparation et le contrôle des données d'application (voir 14.3.5), et enfin les outils utilisés pour la gestion et le contrôle des processus et des produits concernés par le développement des logiciels.

Les outils hors-ligne, employés pour calculer les variables importantes utilisées lors de la conception et de l'analyse des équipements importants pour la sûreté, sortent du cadre de cette norme, de même que les traitements de texte, les outils de gestion de projet et autres outils bureautiques et de soutien administratif utilisés pour des tâches non directement concernées par le développement logiciel.

14.2 Sélection des outils

14.2.1 Les outils utilisés pour le développement de logiciels des systèmes de classe 1 doivent être sélectionnés pour aider le processus d'ingénierie du logiciel. Les critères et le processus de sélection à suivre sont décrits en 14.3.1. Les limites d'application des outils doivent être identifiées et documentées. Les outils et leur sortie ne doivent pas être utilisés en dehors de leur limites déclarées d'application sans justification préalable.

14.2.2 Les outils utilisés pour le développement des logiciels des systèmes de classe 1 doivent être vérifiés et évalués à un niveau adéquat par rapport aux exigences de fiabilité de l'outil, au type d'outil (voir les points 1) à 5) de 14.2.3 et au potentiel de l'outil à introduire des défauts.

14.2.3 Les outils doivent présenter une fiabilité suffisante pour assurer qu'ils ne mettent pas en cause l'intégrité du produit final. Par exemple, un outil peut affecter défavorablement le développement de logiciels en introduisant des erreurs, en produisant une sortie corrompue, ou en ne détectant pas un défaut déjà présent.

Les principes de défense en profondeur et de diversité adoptés pour l'architecture du contrôle commande peuvent être pris en compte lors de la sélection des outils afin de réduire les exigences de fiabilité des outils pris individuellement.

Le niveau de vérification et d'évaluation requis pour un outil dépend également de la classe ou du type d'outil et de la possibilité ou non de vérifier ou de valider complètement la sortie de l'outil. Les classes d'outils sont:

The use of appropriate software tools can increase the integrity of the software development process, and hence software product reliability, by reducing the risk of introducing faults in the process. The use of tools can also have economic benefits as they can reduce the time and human effort required to produce software. Tools can be used to automatically check for adherence to rules of construction and standards, to generate proper records and consistent documentation in standard formats, and to support change control. Tools can also reduce the effort required for testing and to maintain automated logs. Tools can also be necessary because a specific development methodology requires their use.

14.1.1 Tools are most powerful when they are defined to work co-operatively with each other. Care should be taken not to require tools to undertake tasks beyond their capability, for example, they cannot replace humans when judgement is involved. In some cases, tool support is more appropriate than complete automation of the process. When selecting a tool, the benefits and risk of using a tool must be balanced against the benefits and risk of not using a tool. The important principle is to choose tools that limit the opportunity for making errors and introducing faults, but maximise the opportunity for detecting faults.

Tools within the scope of this standard include those used to support the capture of requirements and those used to support the transformation of requirements into the final system code and data (there may be many intermediate steps). This standard also includes those tools used to directly support the performance of verification, validation and testing, tools for the preparation and control of application data (see 14.3.5), and tools for the management and control of the processes and products involved in the software development.

Off-line tools, used to calculate important variables used during the design and analysis of equipment important to safety, are considered beyond the scope of this standard. Word processors, project management tools, and other office administration tools used to support tasks not directly concerned with software development are also not within its scope.

14.2 Selection of tools

14.2.1 The tools used to develop software for class 1 systems shall be selected to support the software engineering process. The criteria and process of tool selection to be followed are described in 14.3.1. The limits of applicability of all tools shall be identified and documented. The tools and their output shall not be used outside their declared limits of application without prior justification.

14.2.2 The tools used in the development of software in class 1 systems in nuclear power plants shall be verified and assessed to a level consistent with the tool reliability requirements, the type of tool (see items 1) to 5) of 14.2.3 and the potential of the tool to introduce faults.

14.2.3 Tools shall have sufficient reliability to ensure that they do not jeopardise the reliability of the end product. For example, a tool could adversely affect the development of software by introducing errors, by producing a corrupted output, by failing to detect a fault that is already present.

Principles of defence in depth and diversity adopted for I&C architecture may be considered when selecting tools, in order to reduce the reliability requirements on individual tools.

The level of verification and assessment required for a tool also depends on the type of tool and whether the output of the tool can be fully verified or validated. Types of tools are:

1) les outils de transformation tels que les générateurs de code, les compilateurs et ceux qui transforment un texte ou un diagramme d'un niveau d'abstraction à un autre, en général inférieur;
2) les outils de vérification et de validation tels que les logiciels d'analyse statique, les contrôleurs de couverture des tests, les logiciels d'aide à la démonstration de théorèmes et les simulateurs;
3) les outils de diagnostic utilisés pour la maintenance et l'observation du logiciel pendant l'exploitation;
4) les outils d'infrastructure tels que les systèmes supports de développement;
5) les outils de contrôle de la configuration tels que les outils de contrôle de version.

14.3 Exigences applicables aux outils

Les exigences applicables aux outils sont présentées comme suit:

a) environnement de génie logiciel;
b) qualification des outils;
c) gestion de la configuration des outils;
d) traducteurs/compilateurs;
e) outils pour données d'application;
f) automatisation des tests.

14.3.1 Environnement de génie logiciel

14.3.1.1 Il convient que les outils soient utilisés pour supporter tous les aspects du cycle de vie du logiciel lorsque leur utilisation comporte des avantages et lorsque les outils existent. L'analyse de l'environnement de génie logiciel et des processus de développement doit être effectuée pour déterminer la stratégie de soutien par les outils. Il convient que les résultats de l'analyse soient documentés. Si les outils n'existent pas, il peut être nécessaire d'envisager le développement d'outils nouveaux.

Exemples de processus et opérations qui peuvent tirer avantage d'un soutien par des outils:

1) la production et la vérification de la spécification, de la conception et du codage (voir Annexe H);
2) les outils fonctionnant sur le langage ou sur un sous-ensemble du langage (voir 14.3.4);
3) la préparation, la vérification, la validation et la gestion des données applicatives (voir 14.3.5);
4) l'automatisation des tests (voir 14.3.6).

14.3.1.2 Il convient que les critères de sélection et d'évaluation de l'environnement de génie logiciel soient développés et classés par ordre de priorité afin de permettre des compromis avant utilisation. Il convient que les critères soient structurés selon les caractéristiques de qualité du logiciel: fonctionnalité, fiabilité, facilité d'emploi, efficacité, maintenabilité, et portabilité, comme définies dans l'ISO/CEI 9126. Les critères peuvent inclure d'autres sujets comme: l'effort nécessaire pour obtenir l'agrément des autorités, les ressources nécessaires pour utiliser un outil, la rigueur du programme qualité sous lequel l'outil a été développé, l'historique du fournisseur de l'outil et les alternatives à l'utilisation de l'outil.

14.3.1.3 Le soutien par les outils de l'environnement de génie logiciel doit être analysé et documenté afin de déterminer:

1) transformation tools such as code generators, compilers, and those that transform text or a diagram at one level of abstraction into another, usually lower, level of abstraction;
2) verification and validation tools such as static code analysers, test coverage monitors, theorem proving assistants, and simulators;
3) diagnostic tools used to maintain and monitor the software under operating conditions;
4) infrastructure tools such as development support systems;
5) configuration control tools such as version control tools.

14.3 Requirements for tools

Requirements for tools are presented by topic:

a) software engineering environment;
b) tool qualification;
c) tool configuration management;
d) translators/compilers;
e) application data tools;
f) automation of testing.

14.3.1 Software engineering environment

14.3.1.1 Tools should be used to support all aspects of the software life cycle where benefits result through their use and where tools are available. Analysis of the software engineering environment and development processes shall be performed to determine the strategy for providing tool support. The results of the analysis should be documented. If tools are not available, the development of new tools may need to be considered.

Examples of processes and operations that can benefit from tool support are:

1) production and checking of specification, design and implementation (see Annex H);
2) tools operating on the language or a subset of the language (see 14.3.4);
3) preparation, verification and validation, and management of application data (see 14.3.5);
4) automation of testing (see 14.3.6).

14.3.1.2 Criteria for the selection and evaluation of tools for the software engineering environment should be developed and prioritised to allow trade-offs prior to use. The criteria should be structured by software quality characteristics as defined in ISO/IEC 9126: functionality, reliability, usability, efficiency, modifiability, and portability. The criteria may include other items like licensing effort and resources required to use a tool, the rigour of the quality plan under which the tool was developed, vendor tool history, and alternatives to tool use.

14.3.1.3 The tool support for the software engineering environment shall be analysed and documented to address:

1) comment chacun des processus est ou n'est pas soutenu par des outils;
2) l'identification précise des outils (par exemple nom et numéro de version), et si possible leur configuration;
3) comment chacun des outils doit être utilisé dans le cadre du projet (c'est-à-dire la classe de l'outil);
4) comment la sortie de chaque outil doit être vérifiée et/ou validée par rapport aux entrées;
5) comment les autres outils ou processus atténuent les conséquences d'une erreur dans l'outil, y compris l'atténuation d'erreurs potentielles pendant la production et la préparation des données pour utilisation en ligne;
6) quelles sont les interfaces des outils avec d'autres outils, à savoir des outils peuvent être nécessaires pour utiliser, traiter et livrer de l'information partagée par d'autres outils ou faisant partie d'une base de données;
7) comment les outils donnent une interface cohérente pour les utilisateurs et pour l'environnement de génie logiciel restant;
8) comment les outils sont adaptés pour les méthodes de génie logiciel sélectionnées;
9) les capacités de détection et de traitement d'erreurs des outils;
10) comment les outils satisfont au contexte d'utilisation comprenant les utilisateurs, le matériel, l'environnement et les tâches des utilisateurs, pour rendre maximale l'efficacité de l'utilisateur et minimiser l'impact des erreurs de l'utilisateur;
11) comment les outils sont protégés contre toute utilisation non autorisée ou non appropriée ou contre des modifications.

14.3.1.4 La modification, l'évolution ou le remplacement des outils doivent être documentés et justifiés. Cette stratégie fait partie de la stratégie de maintenance du logiciel opérationnel qui doit garantir que le logiciel opérationnel peut être adapté et corrigé tout au long de son utilisation dans la centrale nucléaire. Elle doit en particulier garantir que le passage à une nouvelle version d'un outil est justifié et que la nouvelle version est qualifiée de façon appropriée, c'est-à-dire par rapport aux exigences de cette norme.

14.3.1.5 Il convient de démontrer que les outils utilisés pour fournir une diversification, c'est-à-dire les compilateurs utilisés pour le développement de versions multiples de systèmes de logiciel différents, sont différents. Ceci peut être obtenu en montrant que:

1) chaque outil a été fourni par un fournisseur différent (ainsi un outil peut être développé et un autre acheté dans le commerce), ou
2) les outils ont des langages d'entrée et/ou sortie différents, ou
3) les outils ont des exigences et un processus de conception différents.

14.3.2 Qualification des outils

14.3.2.1 Les outils doivent être qualifiés selon une stratégie de qualification documentée. La stratégie doit prendre en compte les exigences de fiabilité de l'outil et le type d'outil.

14.3.2.2 Les exigences qualitatives de fiabilité d'un outil doivent être déterminées compte tenu:

1) des conséquences d'une erreur dans l'outil;
2) de la probabilité qu'un outil provoque ou induise des erreurs dans le logiciel de la fonction de sûreté;
3) des autres outils ou processus qui atténuent les conséquences d'une erreur dans l'outil.

NOTE Les principes de défense en profondeur et de diversité peuvent réduire les exigences de fiabilité des outils.

1) how each process is, or is not, supported by tools;
2) the precise identification of the tools (for example, name, version number) and possibly their configuration;
3) how each tool is to be used within the project;
4) how the output of each tool is to be verified and/or validated against its input;
5) how other tools or processes mitigate the consequences of a fault in the tool, including mitigation of potential errors during production and preparation of data for on-line use;
6) how tools interface with other tools, i.e. tools may be required to use, process, and deliver information shared by other tools or part of a repository;
7) how tools provide a consistent interface to users and to the remainder of the software engineering environment;
8) how tools are suitable for the software engineering methods selected;
9) the error detection and handling capability of tools;
10) how tools satisfy the context of use including users, equipment, environment, and user's tasks, to maximise user effectiveness and minimise the impact of user errors;
11) how tools prevent unauthorised use/misuse or modification.

14.3.1.4 The modification, upgrade or replacement strategy for tools shall be documented and justified. This strategy is a part of the operational software modification strategy which shall ensure that the operational software can be adapted or corrected throughout its use in the NPP. It shall also ensure that moving to a new version of a tool is justified and the new version of the tool is suitably qualified, i.e. assessed against the requirement of this standard.

14.3.1.5 Tools used to provide diversity, i.e. compilers used for the development of multiple-version dissimilar software systems, should be demonstrated to be dissimilar. This may be achieved by showing that:

1) each tool was obtained from a different supplier (for example, one tool could be developed and the other tool could be purchased off the shelf); or
2) the tools have different input and/or output languages; or
3) the tools have dissimilar requirements and design processes.

14.3.2 Tool qualification

14.3.2.1 A tool qualification strategy shall be produced and the tools shall be qualified in accordance with that strategy. The strategy shall consider the reliability requirements of the tool and the type of the tool.

14.3.2.2 The qualitative reliability requirements of a tool shall be determined considering:

1) the consequences of a fault in the tool;
2) the probability that a tool causes or induces faults in the software implementing the safety function;
3) what other tools or processes mitigate the consequences of a fault in the tool.

NOTE Principles of defence in depth and diversity can reduce the reliability requirements on tools.

14.3.2.3 La stratégie de qualification des outils doit prendre en compte:

1) l'analyse du processus de développement des outils et l'historique du fournisseur;
2) l'aptitude de la documentation de l'outil à permettre la vérification de la sortie de l'outil et la facilité d'apprendre;
3) le test ou la validation de l'outil;
4) l'évaluation de l'outil pendant une période d'utilisation;
5) le retour d'expérience sur l'utilisation de l'outil.

NOTE L'Article 15 contient des exigences pour l'utilisation de logiciels prédéveloppés qu'il convient de prendre en considération pour la stratégie de qualification des outils.

14.3.2.4 Il convient que les sorties de l'outil soient vérifiées systématiquement (par exemple par test, analyse ou comparaison avec les sorties d'outils ayant des fonctionnalités similaires), si les sorties doivent être intégrées dans le logiciel final.

14.3.2.5 Les outils doivent faire l'objet d'évaluations, comme indiqué à l'Article 15, ou bien être développés conformément aux exigences des Articles 1 à 12 de cette norme, sauf si:

– l'outil ne peut pas introduire d'erreur dans le logiciel (par exemple, un traitement de texte utilisé pour la documentation) ou;
– il a une atténuation des erreurs potentielles de l'outil (par exemple, par la diversification du processus ou de la conception du système, point 5) de 14.3.1.3, ou;
– la sortie de l'outil est toujours systématiquement vérifiée (point 4) de 14.3.1.3. Le processus de qualification peut prendre en compte l'expérience opérationnelle de l'outil lorsque l'utilisation de l'outil a été justifiée de façon pertinente dans une utilisation liée à la sûreté.

14.3.3 Gestion de la configuration des outils

14.3.3.1 Tous les outils doivent être sous la gestion de la configuration pour assurer l'identification complète des outils sélectionnés (y compris le nom, la version, la variante et éventuellement la configuration) et les paramètres des outils utilisés pour générer les logiciels de référence, voir 5.6.

NOTE Cela est utile non seulement pour la cohérence finale du logiciel, mais aussi aide à évaluer l'origine d'un défaut, qui peut se trouver dans le code source, dans l'outil ou dans les paramètres de celui-ci. Il peut aussi être nécessaire dans l'estimation du risque de CCF dû aux outils logiciels.

14.3.3.2 Des enregistrements documentant l'historique des erreurs et les limitations des outils doivent être conservés pendant toute la vie du logiciel pour les outils dont la sortie peut injecter un défaut dans le logiciel final.

14.3.3.3 Toute modification d'un outil doit être vérifiée et évaluée.

14.3.4 Traducteurs/compilateurs

Le présent paragraphe présente les exigences liées spécifiquement aux traducteurs/ compilateurs. Du fait de la taille et de la complexité de nombreux compilateurs, il peut être extrêmement difficile de démontrer qu'un compilateur fonctionne correctement. Toutefois, une grande expérience d'utilisation peut augmenter l'assurance de bon fonctionnement d'un compilateur.

14.3.4.1 Il convient de sélectionner les traducteurs et compilateurs sur la base des critères indicatifs concernant les traducteurs/compilateurs donnés dans le présent paragraphe. (Ces critères viennent compléter les recommandations énoncées dans l'Annexe D).

14.3.2.3 Tool qualification strategy shall consider:

1) analysis of tool development process and vendor tool history;
2) adequacy of tool documentation to allow verification of tool output and ease of learning;
3) testing or validation of the tool;
4) evaluation of the tool over a period of use;
5) feedback of experience with tool use.

NOTE Clause 15 contains qualification requirements for the use of pre-developed software that should also be considered for tool qualification strategy.

14.3.2.4 Tool outputs should be systematically verified (for example, by test, analysis, or comparison with the output of functionally similar tools), if the output is to be included in the final software.

14.3.2.5 Tools shall be subject to evaluation and assessment as described in Clause 15, or developed according to QA requirements of Clauses 1 to 12 of this standard, unless:

– the tool cannot introduce faults into the software (for example a word processor tool for documentation), or;
– there is mitigation of any potential tool faults (e.g. by process diversity or system design, see item 5) of 14.3.1.3, or;
– the tool output is always systematically verified (see item 4) of 14.3.1.3. The qualification process may take into account experience of prior use of the tools where the tool has previously been demonstrated adequately through use in a relevant safety related application.

14.3.3 Tool configuration management

14.3.3.1 All tools shall be under configuration management to ensure the complete identification of selected tools (including name, version, variant, and possibly configuration) and the tool parameters used to generate baselined software (see 5.6).

NOTE This is useful not only for the final software consistency, it also helps in assessing the origin of a fault, which may lie in the source code, in the tool or in the tool parameters. It may also be necessary in the assessment of the potential for CCF due to software tools.

14.3.3.2 Records documenting the error history and limitations of tools shall be maintained throughout the life of any tool whose output can introduce a fault into the final software.

14.3.3.3 Any modification of a tool shall be verified and assessed.

14.3.4 Translators/compilers

This subclause presents requirements specifically related to translators/compilers. The size and complexity of many compilers can make it extremely difficult to demonstrate that a compiler works correctly. However, extensive experience of use can increase confidence that the compiler works correctly.

14.3.4.1 Translators/compilers should be selected on the basis of guidance criteria relevant to translators/compilers in this subclause (which complement the recommendations in Annex D).

14.3.4.2 Les traducteurs/compilateurs ne doivent pas retirer sans préavis les moyens de programmation défensive ou de contrôle d'erreurs introduits par le programmeur.

14.3.4.3 Il convient d'éviter l'utilisation de l'optimisation par compilateur. Celle-ci ne doit pas être utilisée si elle produit du code objet trop difficile à comprendre, à déboguer, à tester et à valider.

NOTE L'optimisation du code peut être utilisée pour satisfaire à des exigences de performances dues à des contraintes liées à la vitesse du matériel ou aux limites de mémoire. Dans des cas exceptionnels, l'utilisation de code assembleur peut être envisagée en plus du remplacement de la plate-forme matérielle.

14.3.4.4 Lorsque l'optimisation est utilisée, les tests, la vérification et/ou la validation doivent être effectués sur le code optimisé.

14.3.4.5 Les bibliothèques utilisées dans le système cible doivent être considérées comme des ensembles de composants logiciels prédéveloppés. Les composants de la bibliothèque utilisés doivent être évalués, qualifiés et utilisés conformément aux exigences de l'Article 15 relatives à la qualification de logiciels prédéveloppés.

14.3.4.6 Des tests, vérifications et/ou validations doivent être effectués afin de s'assurer que le code supplémentaire (instructions d'assembleur) introduit par le traducteur et non directement lié aux instructions du code source (code de contrôle d'erreurs, code de traitement d'erreurs et d'exceptions, code d'initialisation, par exemple) est correct.

14.3.5 Outils pour les données d'application

Les systèmes informatiques de sûreté nécessitent en général des données applicatives pour définir les signaux, les adresses et les paramètres des fonctions d'application et de service. Les données peuvent être importantes et se composent en général d'informations telles que:

– références des étiquettes des signaux, description des signaux, implantation des sources et numéros des câbles, types de mesures, plages ou états électriques, unités techniques, définitions des états d'alarme, niveaux d'alarme et de déclenchement;
– les points de raccordement des signaux, les adresses et pointeurs de bases de données, les adresses et pointeurs d'informations, les adresses et caractéristiques matérielles, le format des affichages, les informations liées aux symboles et couleurs d'affichage, l'identification du contenu des signaux d'affichage, les formats des journaux et des messages internes et le détail de leur contenu;
– les codes d'action de protection, la priorité ou la logique des alarmes, les signaux d'actionnement, l'identification des opérations logiques et temporisateurs, les états de sortie à adopter en cas de défaillance.

Les données peuvent être extraites de plans, calendriers de conception et des spécifications d'exploitation de la centrale et d'instrumentation des processus. Elles seront traduites pour chargement dans les processeurs cibles du système, puis utilisées pour commander l'action du logiciel en ligne.

Les exigences liées à la préparation, la vérification, la validation et la gestion des données pour utilisation en ligne sont présentées ci-dessous.

14.3.5.1 La conception des données d'application du système à partir des données d'application de la centrale doit être définie et documentée.

14.3.5.2 Les paramètres d'application qui peuvent être modifiés par les opérateurs pendant l'exploitation ainsi que les méthodes pour contrôler ces modifications doivent être identifiés.

14.3.4.2 Translators/compilers shall not remove without warning defensive programming or error-checking features introduced by the programmer.

14.3.4.3 The use of compiler optimisation should be avoided. It shall not be used if it produces object code that is excessively difficult to understand, debug, test and validate.

NOTE Code optimisation can be used to meet performance requirements due to constraints in hardware speed and storage limits. In exceptional cases, the alternative use of assembly code can be considered in addition to changing the hardware platform.

14.3.4.4 Where optimisation is used, tests, verification and/or validation shall be performed on the optimised code.

14.3.4.5 Libraries which are used in the target system shall be considered as sets of pre-developed software components. Those components of the library used shall be evaluated, qualified and used in accordance with the requirements in Clause 15 on qualification of pre-developed software.

14.3.4.6 Tests, verification and/or validation shall be carried out to ensure that any additional code (assembly instructions) introduced by the translator which is not directly traceable to source line statements (for example, error checking code, error and exception handling code, initialisation code) is correct.

14.3.5 Application data tools

Computer-based safety systems usually require application data to define signals, addresses and function parameters of the application functions and service functions. The data can be extensive and normally consists of information such as:

– signal tag references, signal descriptions, source locations and cable numbers, measurement types, electrical ranges or states, engineering units, alarm state definitions, alarm and trip levels;
– signal termination points, data base addresses and pointers, information addresses and pointers, hardware addresses and characteristics, display layouts, display symbol and colour information, display signal content identification, log and internal message formats and details of contents;
– protection action codes, alarm priority or logic, outputs for action, identification of logic operations and timers, output states to be adopted at failure.

The data may be taken from design drawings, lists and specifications of plant operations and process instrumentation. It will be translated for loading to target processors of the system, and then used to control the action of the on-line software.

Requirements related to the preparation, verification and validation and management of data for on-line use are presented below.

14.3.5.1 The design of the software system application data and its method of derivation from the plant application data shall be defined and documented.

14.3.5.2 Application parameters that can be changed during operation by the operator shall be identified, together with the methods to be used to control changes of such parameters.

14.3.5.3 Il convient que les modifications de données d'application ne corrompent ni les autres données, ni le code exécutable du système.

14.3.5.4 Le formalisme des procédures pour la vérification et la validation des données doit être semblable aux procédures de vérification et de validation du logiciel, y compris pour ce qui concerne l'identification et l'élimination des erreurs. Des contrôles d'un bout à l'autre doivent être réalisés et doivent inclure chaque étape de la transformation des données, de l'extraction de données sur la base d'informations sur la conception de la centrale jusqu'à l'intégration des structures de données dans le logiciel en ligne, y compris l'utilisation des moyens de transfert des données.

14.3.5.5 Il convient que les données à charger dans le logiciel en ligne soient dans une forme adaptée pour pouvoir être imprimées et vérifiées. Autrement, un outil doit être utilisé pour reprendre les données et les remettre dans une forme compréhensible pour la vérification.

14.3.5.6 Des moyens doivent exister pour permettre la vérification de toutes les configurations de données chargées sur site.

14.3.5.7 Si des données définissent l'interface entre deux systèmes, il convient que les données pour chaque système soient générées automatiquement à partir de la même base de données (voir 5.3.1.4 de la CEI 61513).

14.3.5.8 Dans certains cas, lorsque la fonctionnalité du logiciel comprenant le traitement, le flux des données et les liaisons d'entrée et sortie est contrôlée ou modifiée par des «données de configuration», un justificatif spécifique est requis pour confirmer que les données ont fait l'objet d'un niveau adéquat d'évaluation et successivement de tests. Il peut s'avérer nécessaire d'effectuer à nouveau un volume important de test en cas de changement de telles données.

14.3.6 Automatisation des tests

L'automatisation augmente le volume de tests qui peut être effectué dans une période déterminée. Ceci peut être obtenu lorsque les exigences suivantes sont remplies.

14.3.6.1 Il convient que les outils automatisant la validation qui génèrent des données de tests, qui transportent ou transforment des données et des résultats d'essais et qui évaluent les résultats d'essais, enregistrent un journal complet de tests. Ceci s'applique au test de modules aussi bien qu'aux simulations de la centrale.

14.3.6.2 Il convient que des outils appropriés soient utilisés pour tester et/ou simuler le comportement du code chargé dans le système cible.

14.3.6.3 Des outils appropriés doivent être utilisés pour assurer ou vérifier que le bon code exécutable est chargé correctement dans le système cible.

14.3.6.4 Il convient de prendre en considération l'utilisation des outils supplémentaires suivants:

1) générateurs de test, analyseurs de la couverture des tests et bancs de test;
2) programmes de diagnostic en ligne avec des moyens de traçabilité et d'inspection;
3) débogueurs avec des moyens pour déboguer au niveau du code source; et
4) suites de tests automatiques pour faciliter les tests de régression.

14.3.5.3 Changes made to modifiable application data should not corrupt other data and code on the runtime system.

14.3.5.4 The formality of procedures for data verification shall be similar to the formality of procedures for software verification and validation, including identification and clearance of errors. End-to-end verification checks shall be performed, and these shall include each stage of data transformation starting from data extraction from plant design information through to incorporation of data structures in the on-line software, including the use of transfer media.

14.3.5.5 The data to be loaded to the on-line software should be in a form, which can be printed and verified, or a tool shall be used to take that data and restore it to an intelligible form for verification.

14.3.5.6 A facility shall be provided to allow verification of all loaded configuration data on site.

14.3.5.7 Where data define the interface between two systems, then the data provided for each system should be automatically generated from the same data base (see 5.3.1.4 of IEC 61513).

14.3.5.8 In some cases, where software functionality including processes, data flow, and input and output connections are controlled or modified by configuration data, specific documented justification shall confirm an adequate level of assessment and subsequent testing has been applied to the data. Extensive system re-testing may be required following changes to such data.

14.3.6 Automation of testing

Automation increases the amount of testing that can be performed in a given period. This can be achieved by meeting the following requirements.

14.3.6.1 Automated validation tools that generate test data, transport or transform test data and test results and evaluate test results should record a complete test log. This is applicable to module tests as well as to plant simulations.

14.3.6.2 Appropriate tools should be used to test and/or simulate the behaviour of the executable code loaded in the target system.

14.3.6.3 Appropriate tools shall be used to ensure or verify that the right executable code is loaded correctly in the target system.

14.3.6.4 The use of the following additional tools should be considered:

1) test generators, test coverage analysers and test drivers;
2) on-line diagnostic programs with dump inspect and trace facilities;
3) debuggers with debugging facilities at the source code level; and
4) automated test suites to facilitate regression testing.

15 Qualification de logiciels prédéveloppés

15.1 Généralités

Le présent article présente les exigences applicables à l'utilisation de logiciels prédéveloppés (PDS) dans les systèmes de contrôle-commande programmés. Ces exigences sont établies comme faisant partie des exigences de qualification du système dans lequel le PDS est intégré (voir 6.4 de la CEI 61513).

Les PDS pour les systèmes d'I&C peuvent comprendre des petits composants logiciels (par exemple un module d'une bibliothèque de fonctions d'application) ainsi que des gros produits logiciels complexes (par exemple une partie d'un système d'exploitation ou de communication). Les PDS peuvent être classés en deux catégories, selon le type de matériel:

a) PDS polyvalents non spécifiquement développés pour un environnement matériel spécifique, et
b) PDS intégrés dans des composants du matériel, qui sont à utiliser en association avec ces composants.

Des composants PDS sont dits réutilisables quand ils peuvent être utilisés dans différents programmes informatiques ou systèmes, par exemple comme partie d'une famille d'équipements (plate-forme d'équipements). Il est possible que des composants, qui sont indépendants des particularités propres aux applications de la centrale, aient déjà été qualifiés pour l'utilisation dans des systèmes qui réalisent des fonctions de catégorie A.

Les spécifications de nouveaux systèmes de sûreté souvent tendent à utiliser des équipements prédéveloppés pour réaliser une partie ou la totalité d'un «système nouveau» (voir 6.1.2.1 de CEI 61513). L'utilisation des équipements prédéveloppés peut être intéressante pour la productivité et la fiabilité du système lorsque ces équipements sont introduits de manière appropriée et que leur qualité est correcte. Lorsque des PDS ont été utilisés dans de nombreuses applications similaires, les gains résultant de cette expérience en exploitation peuvent être mis en évidence dans le processus de qualification. En particulier, la réutilisation de PDS validés peut accroître la confiance dans la fiabilité du système.

15.2 Exigences générales

15.2.1 Les PDS susceptibles d'être utilisés dans un système doivent satisfaire à toutes les exigences de cette norme, comme tout composant logiciel réalisant des fonctions de catégorie A.

15.2.2 Le processus d'évaluation du PDS doit:

1) déterminer la capacité des PDS à satisfaire aux exigences fonctionnelles, de performances et architecturales du cahier des charges du système informatique (voir 6.1.1 de la CEI 61513), et par conséquent leur aptitude à l'usage;
2) déterminer les modifications requises pour corriger ou adapter le PDS;
3) agréer la qualité du PDS; et
4) évaluer l'expérience acquise en exploitation du PDS, quand cela est nécessaire pour les évaluations mentionnées ci-dessus.

15.2.3 Les conclusions du processus d'évaluation doivent être documentées.

NOTE Dans la présente norme, le terme agrément est utilisé pour décrire une action conduite par l'organisation responsable du développement du système informatique (ou au nom de cette organisation). Il n'est ni impliqué, ni exigé, que cet agrément soit conduit par l'organisme de sûreté, mais celui-ci peut toutefois décider de le faire.

15 Qualification of pre-developed software

15.1 General

This Clause gives requirements for the use of pre-developed software (PDS) in I&C computer-based systems. These requirements are established as part of the qualification requirements of the system in which the PDS is integrated (see 6.4 of IEC 61513).

PDS for I&C systems may range from small software components (for example, an application function library module), to large and complex software products (for example, parts of an operating system, or communication drivers). PDS may be divided into two types with respect to hardware:

a) general-purpose PDS that has not been specifically developed for a specific hardware environment; and

b) PDS integrated in hardware components that has to be used in association with this hardware.

PDS components are called reusable when they can be used in different computer programs or systems, for example, as part of an equipment family (equipment platform). Those components which are independent of specific plant application details may have already been qualified for use in systems performing category A functions.

The specifications of new safety systems often identify the use of pre-developed equipment including PDS to implement part or the whole of a “new system” (see 6.1.2.1 of IEC 61513). Use of pre-developed equipment can be beneficial to productivity and the reliability of the system when these items are of suitable quality and introduced in a proper manner. When PDS items have been used in many applications similar to the intended use, a benefit from this operating experience can be claimed in their evaluation. In particular, the reuse of validated PDS can increase confidence in the reliability of the system.

15.2 General requirements

15.2.1 A PDS that is a candidate for use as part of a system shall comply with all the requirements developed in this standard, as any software component performing category A functions.

15.2.2 The evaluation of the PDS shall:

1) determine the capability of the PDS to meet the functional, performance and architectural requirements of the system requirements specification (see 6.1.1 of IEC 61513,), and its resulting suitability;

2) identify any modifications needed to correct or adapt the PDS;

3) assess the quality of the PDS; and

4) evaluate the operating experience of the PDS, when required for the above evaluations.

15.2.3 The conclusions of this evaluation process shall be documented.

NOTE In this standard, assessment is used to describe an action made by the organisation in charge of the computer-based system development (or on behalf of this organisation); it is neither implied nor required that assessment be made by licensing authorities, although they may also choose to do so.

15.3 Processus d'évaluation et d'agrément

Le processus d'évaluation et d'agrément du PDS doit comprendre:

a) une évaluation des caractéristiques fonctionnelles et de performances du PDS et de la documentation existante de qualification (voir 15.3.1);

NOTE Pour les PDS intégrés dans un produit, ces caractéristiques peuvent être exprimées comme des propriétés du produit en accord avec la CEI 61069-2.

b) une évaluation de la qualité de la conception et du développement du logiciel (voir 15.3.2);

NOTE Dans le cas des logiciels validés réutilisables, seule l'évaluation de l'appropriation à l'usage est nécessaire, l'évaluation de la qualité étant impliquée par sa validation.

c) une évaluation de l'expérience acquise en exploitation lorsqu'elle est nécessaire pour compenser des faiblesses dans la démonstration obtenue par a) et b), (voir 15.3.3); et

d) un agrément détaillé et documenté de la qualité des preuves obtenues à la suite de l'évaluation détaillée et des travaux supplémentaires associés, qui permettra de déclarer le PDS bon pour utilisation dans le système.

La Figure 4 montre les relations entre les différentes étapes du processus d'évaluation et d'agrément du PDS.

La Figure 5 montre les relations entre ce processus et la qualification du système.

NOTE Le processus décrit dans le présent paragraphe est une vision simplifiée de la réalité et, en tant que tel, ne fait pas état de toutes les itérations ou chevauchements entre activités d'évaluation ainsi qu'entre ces processus et les activités de spécification et de développement du système informatique.

15.3 Evaluation and assessment process

The PDS evaluation and assessment process shall include:

a) an evaluation of the functional and performance features of the PDS and existing qualification documentation (see 15.3.1);

NOTE For PDS integrated in a product these features may be expressed as product properties according to IEC 61069-2.

b) a quality evaluation of the software development process (see 15.3.2);

NOTE For reusable pre-assessed software, only the evaluation of suitability needs to be made; the evaluation of quality is implied by its validation.

c) an evaluation of operating experience if needed to compensate for weaknesses in demonstration gained from both a) and b) (see 15.3.3); and

d) a comprehensive documented assessment of the evidence from the above evaluations, and associated complementary work, which will enable the PDS to be accepted for use in the system.

Figure 4 shows the relations between the different stages of the evaluation and assessment process of the PDS.

Figure 5 shows the relationship between this process and the qualification of the system.

NOTE The process described in this subclause is a simplified view of reality and, as such, does not show all the iterations or overlap between the evaluation activities as well as between these processes and the computer-based system development activities.

1 Evaluation de l'aptitude à l'usage (15.3.1)

Documention d'entrée requise (15.3.1.1)		
Documentation spécification système		Spécification du PDS et documentation utilisateur
Exigences d'évaluation (15.3.1.2)		
Comparaison des spécifications du système et du PDS		Identification des modifications et points manquants
Conclusions		
Le PDS est apte à l'usage	Des actions complémentaires sont nécessaires	Le PDS doit être rejeté

2 Evaluation de la qualité (15.3.2)

Documentation d'entrée requise (15.3.2.1)		
Documentation de conception	Documentation du cycle de vie	(Documentation de l'historique d'exploitation)
Exigences d'évaluation (15.3.2.2)		
Analyse de la conception	Analyse de l'AQ	Identification des points manquants
Conclusions		
La qualité du cycle de vie du PDS est appropriée ou les modifications nécessaires sont faisables	Des tests/documentation supplémentaires sont requis ou l'évaluation de l'expérience acquise en exploitation est nécessaire	Le PDS doit être rejeté

3 Evaluation de l'expérience acquise en exploitation (15.3.3)

Documentation d'entrée requise (15.3.3.1)		
Collecte des données	Temps de fonctionnement	Historique des défauts
Exigences d'évaluation (15.3.3.2)		
Conclusions		
L'expérience acquise en exploitation est suffisante	L'expérience acquise en exploitation n'est pas encore suffisante	Le PDS doit être rejeté

4 Estimation globale (15.3.4)

La qualité du PDS est appropriée	Les modifications nécessaires sont faites

5 Intégration dans le système et maintenance (15.4)

IEC 718/06

Figure 4 – Processus de qualification des logiciels prédéveloppés

1 Suitability evaluation (15.3.1)

Required input documentation (15.3.1.1)	
System specification documentation	PDS specification and user's documentation

Evaluation requirements (15.3.1.2)	
Comparison of the system and PDS specifications	Identification of modifications and missing points

Conclusions		
The PDS is suitable	Complementary work is needed	Ought to be rejected

2 Quality evaluation (15.3.2)

Req. Input documentation (15.3.2.1)		
Design documentation	Life cycle documentation	(Operating history documentation)

Evaluation requirements (15.3.2.2)		
Analysis of design	Analysis of the QA	Identification of missing points

Conclusions		
The quality of the PDS lifecycle is appropriate or the needed modifications of the PDS are feasible	Additional test and documentation is required or operating experience evaluation required	The PDS ought to be rejected

3 Evaluation of operating experience (15.3.3)

Req. input documentation (15.3.3.1)		
Collection of data	Operating time	History of defects

Evaluation requirements (15.3.3.2)

Conclusions		
Sufficient operating experience	Operating experience not sufficient yet	The PDS ought to be rejected

4 Comprehensive assessment (15.3.4)

The quality of the PDS is appropriate	The needed modifications are done

5 Integration in the system and maintenance (15.4)

IEC 718/06

Figure 4 – Outline of the qualification process of pre-developed software

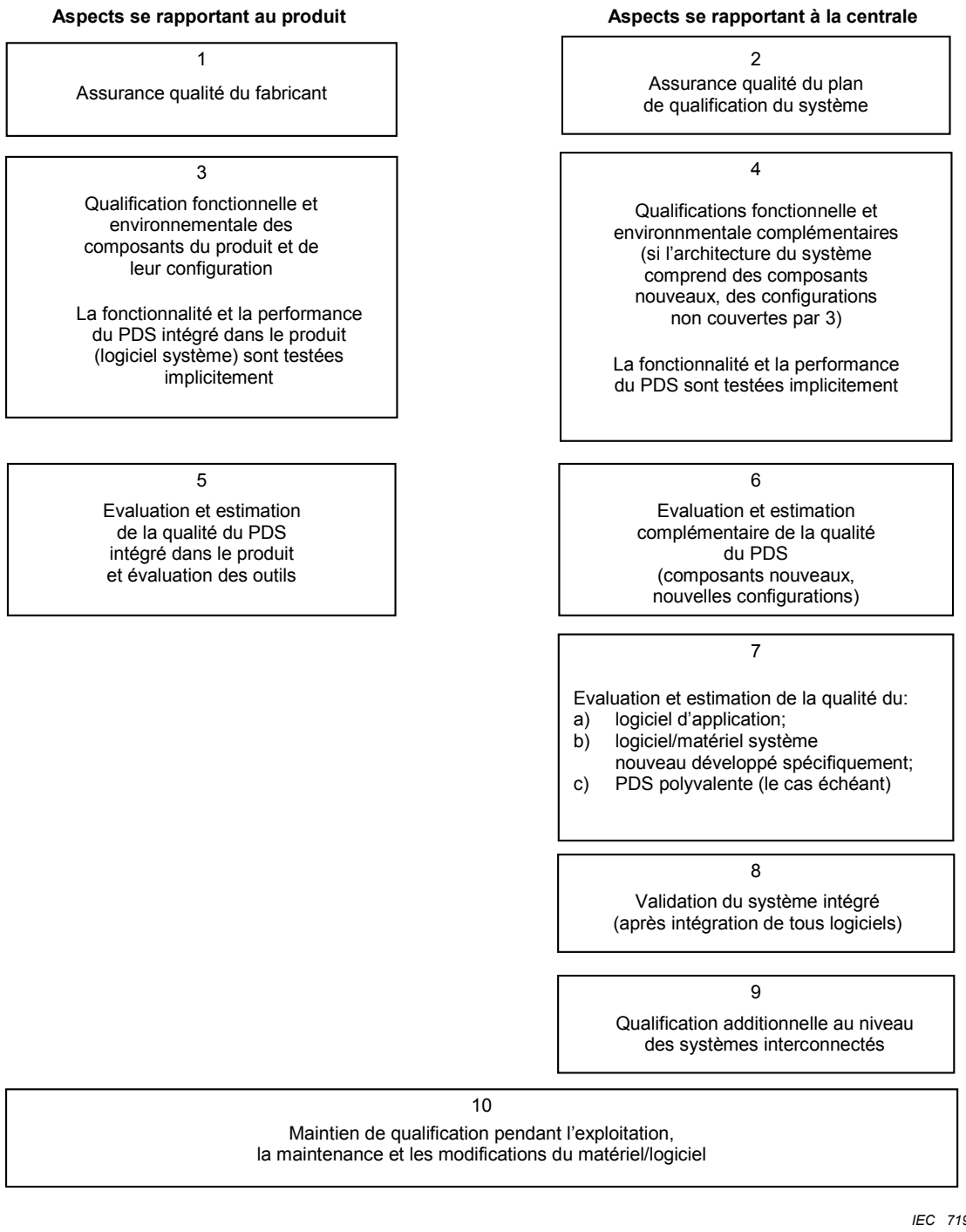


Figure 5 – Relations de l'évaluation et de l'estimation du PDS avec le plan de qualification du système dans lequel il est intégré

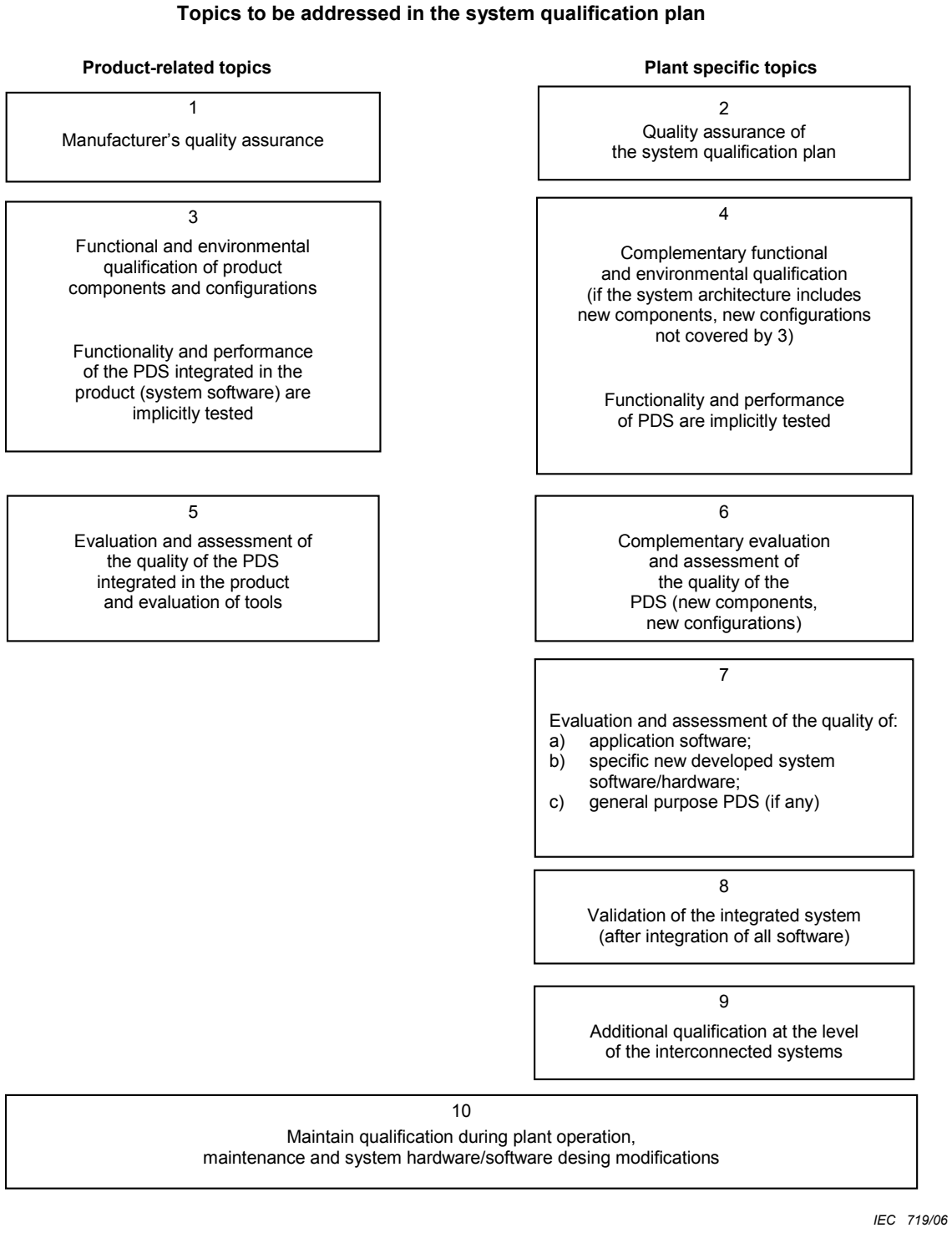


Figure 5 – Relation of PDS evaluation and assessment with the qualification plan of the system in which it is integrated

15.3.1 Evaluation de l'aptitude à l'usage

L'objectif de ce processus est de confirmer que les spécifications fonctionnelles, de performances et architecturales du PDS sont conformes aux exigences relatives au PDS spécifiées dans le cahier de charges. Le processus identifie les composants directement aptes à l'usage dans le système de la centrale et aussi les composants qui nécessitent des modifications.

NOTE L'évaluation est basée sur l'analyse des spécifications et de la documentation fonctionnelle.

Il convient que l'évaluation de l'aptitude à l'usage soit initiée dans un stade précoce de la spécification du système (voir 6.2 de la CEI 61513). Cette évaluation doit être complétée afin:

– d'aider les concepteurs dans la sélection de la conception architecturale du système;
– d'obtenir confirmation vérifiable que les spécifications fonctionnelles et de performances du PDS sont conformes aux spécifications des exigences du système.

15.3.1.1 Documentation d'entrée requise

15.3.1.1.1 La documentation suivante doit être disponible:

– la documentation de spécification du système, qui définit les exigences fonctionnelles, d'interface et de performances qui doivent être remplies par le PDS dans le cadre de l'architecture du système (voir 6.1.1 et 6.1.2 de la CEI 61513);
– les documents de spécification et les documents utilisateur. Il convient que ces documents définissent explicitement toutes les caractéristiques liées au respect des spécifications fonctionnelles et de performances du système. Des analyses ou tests doivent être réalisés afin de rendre explicites les caractéristiques mises en œuvre si elles ne sont pas explicitement définies.

15.3.1.1.2 Le PDS doit être sous le contrôle de configuration; sa version et sa configuration doivent être connues de façon précise.

15.3.1.2 Exigences pour l'évaluation de l'aptitude à l'usage

15.3.1.2.1 Les spécifications du PDS doivent être évaluées par rapport aux spécifications des exigences du système (voir 6.1.1 de la CEI 61513). Si des non-conformités sont identifiées, le PDS doit être refusé ou modifié, ou les spécifications d'exigences doivent être adaptées pour résoudre les non-conformités, sous réserve que les fonctions importantes pour la sûreté soient préservées.

15.3.1.2.2 S'il est nécessaire de modifier le PDS, il doit être procédé à une évaluation, sur la base de la documentation relative à la conception du système, afin de déterminer si ces modifications sont réalisables conformément à cette norme. Si la modification ne peut pas être réalisée de manière conforme, le PDS doit être refusé.

NOTE L'évaluation de qualité indique si ces modifications sont faisables (voir 15.3.2.2). La réalisation correspondante est effectuée dans le cadre du cycle de vie du système (voir l'Article 6 de la CEI 61513).

15.3.1.2.3 Dans le cas des PDS contenus dans une bibliothèque, sauf dans le cas où la totalité de la bibliothèque doit être agréée, il devrait être possible de découper la bibliothèque pour réaliser une bibliothèque restreinte correspondant aux besoins du logiciel et de lier le programme avec les modules de cette bibliothèque réduite, qui doit être constituée de modules validés.

15.3.1.2.4 L'évaluation d'aptitude à l'usage doit identifier les fonctions incluses dans le PDS qui ne sont ni voulues ni nécessaires dans le système et les mesures prises pour s'assurer que ces fonctions ne vont pas interférer avec les fonctions de sûreté.

15.3.1 Evaluation of suitability

The objective of this process is to confirm that the functional, performance and architectural specifications of the PDS comply with the requirement specification. This process identifies components directly suitable for use in the plant system and also identifies those that will require modification.

NOTE The evaluation is based on the analysis of the specifications and functional documentation.

The evaluation of the suitability of the PDS should be initiated in an early stage of the system specification (see 6.2 of IEC 61513) and it shall be completed in order to

– assist the designers in the architectural design of the system;
– gain auditable evidence that the functionality and performance of the PDS meets the requirements of the system.

15.3.1.1 Required input documentation

15.3.1.1.1 The following documentation shall be available:

– system specification documentation which identifies the functional, interface and performance requirements, to be fulfilled by the PDS in the framework of the system architecture (see 6.1.1 and 6.1.2 of IEC 61513);
– the PDS description and user documentation. These documents should explicitly define all the characteristics that are relevant in fulfilling the system functional and performance specifications. Analysis or test shall be performed to make the implemented characteristics explicit if they are not explicitly defined.

15.3.1.1.2 The PDS shall be under configuration management; the version and configuration of the PDS shall be known precisely.

15.3.1.2 Evaluation requirements for suitability

15.3.1.2.1 The specifications of the PDS shall be evaluated with respect to the system requirements specification (see 6.1.1 of IEC 61513). If discrepancies exist, the PDS shall either be rejected or modified or the requirements specification shall be adapted to resolve them, provided that the functions important to safety are preserved.

15.3.1.2.2 If it is necessary to modify the PDS an evaluation shall be completed, based on the PDS design documentation, to determine if the change can be performed in a manner compliant with this standard. If the change cannot be performed in a compliant manner the use of the PDS shall be rejected.

NOTE The quality evaluation indicates if these modifications are feasible (see 15.3.2.2). The corresponding implementation is handled in the frame of the system lifecycle (see Clause 6 of IEC 61513).

15.3.1.2.3 For a PDS that is contained in a library, except when the whole library has to be assessed, it should be possible to tailor the library to build a restricted library meeting the software needs and to link the program with this restricted library which shall be composed of assessed components.

15.3.1.2.4 The suitability evaluation shall identify the functions that are included in the PDS that are unintended and unneeded by the system and the measures to ensure that these functions do not interfere with safety functions.

15.3.1.2.5 Lorsque l'évaluation a été effectuée, un document doit être produit pour consigner:

1) si les spécifications fonctionnelles et de performances du PDS sont conformes aux exigences de spécification du logiciel du système; ou
2) si le PDS n'est pas approprié à l'usage, les raisons du rejet.

15.3.2 Evaluation de qualité

L'objectif de cette évaluation est de démontrer que les caractéristiques du PDS sont en accord avec les exigences pour un système qui réalise des fonctions de catégorie A et qu'une AQ adéquate a été exercée le long du cycle de vie du PDS. Cette évaluation est basée sur la documentation de la conception et du plan qualité logiciel du PDS mais peut nécessiter l'analyse de l'expérience acquise en exploitation.

15.3.2.1 Documentation d'entrée

15.3.2.1.1 En supplément de ce qui est requis en 15.3.1.1, la documentation suivante doit être disponible:

– la documentation de spécification du système, qui définit l'importance pour la sûreté des fonctions remplies par le PDS dans le cadre de l'architecture du système (voir 6.1.2 de la CEI 61513 et Annexe A de la présente norme); et

NOTE Dans l'évaluation de qualité, le niveau de confiance à atteindre sur le fait que le PDS fonctionnera en accord avec les spécifications sera différent pour les trois catégories, le niveau de confiance le plus élevé étant requis pour la catégorie A (voir 8.2.1 de la CEI 61226).

– les documents de qualification du PDS, y compris les certifications précédentes ou les agréments indépendants, si ceux-ci sont utilisés pour l'agrément.

15.3.2.1.2 La documentation suivante relative au PDS doit être fournie ou l'utilisation d'informations alternatives doit être justifiée:

– le plan qualité logiciel (division en tâches élémentaires et activités associées) utilisé dans le cycle de vie logiciel du PDS (voir Article 5) et les tâches et procédures d'assurance de la qualité correspondantes (en particulier le plan de vérification);
– les documents de spécification, de conception, de codage et de modification, et les documents de vérification correspondants;
– le plan d'intégration logiciel/matériel et la vérification associée;
– la validation et les essais effectués sur le PDS par le fournisseur ou le client.

NOTE Pour les deux derniers points: cette documentation est nécessaire seulement si le PDS est intégré dans des composants matériels.

15.3.2.1.3 La documentation de l'expérience acquise en exploitation du PDS doit être disponible si elle est utilisée dans l'évaluation pour compenser des manques dans la documentation ci-dessus ou pour justifier l'utilisation de pratiques différentes de celles décrites dans cette norme.

15.3.2.1.4 Il convient que la documentation donne des informations sur des facteurs industriels tels que la distribution du PDS et l'assistance clients.

15.3.2.2 Exigences de l'évaluation de la qualité

15.3.2.2.1 Les exigences du plan qualité logiciel du PDS et la vérification et documentation correspondantes doivent être évaluées par rapport aux exigences de cette norme. Cette analyse de conformité nécessite des interprétations pour identifier les exigences qui sont applicables dans le contexte d'utilisation du PDS dans le système.

15.3.1.2.5 When the evaluation is concluded, a document shall be produced to record:

1) whether the functional and performance specifications of the PDS comply with the software requirements specification of the system; and
2) where the PDS is not adequate, the grounds for rejection.

15.3.2 Quality evaluation

The objective of this evaluation is to provide evidence that the features of the PDS design are appropriate for a system performing a category A function, and that adequate QA has been exercised throughout the lifecycle of the PDS. This evaluation is based on the design and software quality plan documentation of the PDS but may also require analysis of its operating history.

15.3.2.1 Input documentation

15.3.2.1.1 The following documentation shall be available, in addition to that required in 15.3.1.1:

– the system specification documentation which identifies the importance to safety of the functions implemented with the PDS in the architectural design of the system (see 6.1.2 of IEC 61513 and Annex A of this standard);

NOTE The level of assurance to be achieved by the quality evaluation that the PDS will perform as specified will be different for the three categories, with category A requiring the highest assurance (see 8.2.1 of IEC 61226).

– the qualification documentation of the PDS, including the documentation on previous certifications or independent assessments of the PDS, if it is to be used for the assessment.

15.3.2.1.2 The following documentation related to the PDS shall be provided or the use of alternative information shall be justified:

– the software quality plan (division into elementary tasks and associated activities) used in the software life cycle of the PDS (see Clause 5), and the corresponding quality assurance tasks and procedures records (notably verification planning);
– the specification, design, implementation and modification documents, and the corresponding verification documents;
– the software/hardware integration plan and associated verification;
– the validation plan and tests performed on the product by the vendor or customer.

NOTE For the latter two points: this documentation is needed only if the PDS is integrated in hardware components.

15.3.2.1.3 Documentation of operating experience of the PDS shall be available if it is to be used in the evaluation to compensate for lack of the above documentation or to justify use of practices differing from those of this standard.

15.3.2.1.4 The documentation should provide information on industrial factors such as the distribution of the PDS and the customer support.

15.3.2.2 Evaluation requirements for quality

15.3.2.2.1 The requirements of the PDS software quality plan and the corresponding verification and documentation shall be evaluated for conformance with the requirements of this standard. This analysis of conformance needs interpretation to identify the requirements which are applicable in the context of the use of the PDS in the system.

15.3.2.2.2 La conception du PDS doit être en accord avec les contraintes sur l'architecture et le comportement déterministe du système.

15.3.2.2.3 Lorsque des procédures différentes de celles décrites dans les annexes de cette norme ont été utilisées pour le développement du PDS, leur adéquation doit être analysée et justifiée en accord avec 5.5 de la présente norme. Leur importance dans l'assurance des caractéristiques de qualité logicielle doit être évaluée conjointement avec les exigences du système. Les résultats de l'évaluation et de l'analyse doivent être consignés pour un examen indépendant.

15.3.2.2.4 Les non-conformités par rapport aux exigences de la présente norme, les propriétés qui ne peuvent pas être vérifiées et les faiblesses ou les étapes manquantes dans le processus de vérification ou de documentation doivent être identifiées. Chacune doit être graduée selon son importance pour l'assurance des caractéristiques de qualité logicielle et l'importance pour la sûreté des fonctions remplies dans le système. (L'Article I.1 donne des directives pour la graduation des non-conformités).

15.3.2.2.5 Si un système qui réalise des fonctions de catégorie A comprend aussi des fonctions de catégorie inférieure devant être réalisées par un PDS, et si la conception architecturale du système est telle que ce PDS peut mettre en danger les fonctions de catégorie A (voir 6.2 de la CEI 61513), alors les critères d'évaluation pour la mise en œuvre du PDS réalisant des fonctions de catégorie A doivent être appliqués à ce PDS.

15.3.2.2.6 La documentation de qualification doit prouver que le PDS intégré dans le matériel a été validé pour démontrer qu'il satisfait à ses spécifications fonctionnelles et de performances.

NOTE Le comportement fonctionnel et la performance des composants PDS peuvent être qualifiés implicitement lors de la qualification fonctionnelle des équipements individuels dans lesquels ils sont intégrés (voir 2 de la Figure 5). Toutefois il y a des propriétés qui peuvent être qualifiées seulement en utilisant des configurations d'équipements.

15.3.2.2.7 Lorsque des composants du PDS ont des caractéristiques qui ne peuvent être validées que dans le cadre de la configuration finale du système, alors la validation de ces caractéristiques doit être réalisée sur cette configuration finale du système.

15.3.2.2.8 La qualité et le taux de couverture des tests de validation effectués sur le PDS doivent être évalués par rapport aux exigences des Articles 9 et 10 de cette norme et des tests de validation supplémentaires doivent être réalisés si nécessaire.

15.3.2.2.9 Lorsque l'évaluation de la conception et du cycle de vie est terminée, un document doit être produit pour consigner que:

1) la qualité du PDS a été démontrée et qu'aucun test supplémentaire, ni aucune analyse de l'expérience acquise en exploitation n'est requise;
2) une qualification complémentaire doit être effectuée lorsque le système configuré sera disponible;
3) des manques d'information ont été détectés durant l'évaluation, mais que cela peut être compensé par des compléments de vérification et validation, d'analyse de code et/ou de test, et de documentations;
4) des manques d'information ont été détectés dans l'évaluation, mais que cela peut être compensés sur la base de l'expérience acquise en exploitation;
5) le PDS (ou des parties du PDS) nécessite des modifications pour son utilisation prévue dans le système (voir 15.3.3), et qu'il a le niveau de qualité approprié; il est alors préférable mais pas essentiel que les modifications soient effectuées en conformité avec cette norme;

   NOTE Après réalisation des modifications définies, une évaluation et un agrément complémentaires sont nécessaires dans le cadre de la qualification du système (voir 5 de la Figure 5).

15.3.2.2.2 The PDS design shall be consistent with the constraints on the architecture and deterministic internal behaviour of the system.

15.3.2.2.3 If practices differing from those of the annexes of this standard have been used for the development of the PDS, their adequacy shall be analysed and justified according to 5.5 of this standard. Their importance in the assurance of the software quality characteristics shall be evaluated in conjunction with the system requirements. The results of the evaluation and analysis shall be recorded for independent review.

15.3.2.2.4 Non-conformities against requirements of this standard, properties that cannot be verified, weakness or missing steps in the verification or documentation process shall be identified. Each shall be ranked according to its importance in the assurance of the software quality characteristics, and the importance to safety of the functions implemented in the system. Clause I.1 provides guidance for the ranking of non-conformities.

15.3.2.2.5 If systems implementing category A functions include functions of a lower category that are to be performed by a PDS, and the system architectural design is such that this PDS could potentially jeopardise the category A functions (see 6.2 of IEC 61513), then the evaluation criteria for PDS software implementing category A functions shall be applied to such PDS.

15.3.2.2.6 The qualification documentation shall provide evidence that PDS integrated in hardware components, has been validated to demonstrate that it meets its functional and performance specifications.

NOTE The functional and performance behaviour of the PDS components may be implicitly qualified by the functional qualification of the individual equipment in which they are integrated (see 2 of Figure 5). However, there are properties that may only be qualified using configurations of equipment.

15.3.2.2.7 Where PDS components contain features that cannot be validated other than in the final system configuration, then validation of these features shall be performed in the final system configuration.

15.3.2.2.8 The quality and degree of coverage of the validation tests performed on the PDS shall be evaluated with reference to the requirements of Clauses 9 and 10 of this standard and additional validation tests performed if necessary.

15.3.2.2.9 When the evaluation of the design and of the lifecycle is concluded, a document shall be produced to record that at least one of the following conclusions is true:

1) the PDS quality has been proved and no additional tests or analysis of operating experience is required;
2) complementary qualification shall be performed when the system configuration is available;
3) lack of information has been detected during the evaluation, but this can be compensated by the completion of additional verification and validation, testing or code analysis and documentation;
4) lack of information has been detected during the evaluation, which can be compensated for by use of operating experience;
5) the PDS (or part of the PDS) requires modification for the intended use in the system (see 15.3.3), and that it has the appropriate level of quality so it is desirable, but not essential, that the modifications be performed in accordance with this standard;

NOTE After satisfactory completion of the defined modifications, a complementary evaluation and assessment is needed in the frame of the system qualification (see 5 of Figure 5).

6) il faut s'attendre à des problèmes significatifs lors du transfert du PDS dans le nouveau matériel;
7) la qualité du PDS n'est pas adéquate et le PDS doit être rejeté au motif que les faiblesses sont trop importantes ou que les informations sont trop insuffisantes pour pouvoir être compensées; et
8) l'indépendance des fonctions ou propriétés du PDS qualifiées par rapport à celles non qualifiées a ou n'a pas été établie.

15.3.3 Evaluation de l'expérience acquise en exploitation

L'objectif de cette évaluation est de démontrer qu'une expérience appropriée acquise en exploitation du PDS peut compenser des faiblesses et des manques d'information détectés dans l'évaluation de la qualité.

Parmi la totalité des fonctions/propriétés du PDS, les fonctions/propriétés à qualifier sur la base du retour d'expérience doivent être identifiées et ce qui suit doit être évalué pour le PDS:

1) les méthodes pour le recueil des données sur l'expérience en exploitation;
2) les méthodes pour l'enregistrement du temps d'exploitation de la version du PDS et pour produire l'historique d'exploitation;
3) l'historique opérationnel des faits techniques, défauts et comptes rendus d'erreurs; et
4) l'historique opérationnel des modifications suite à des défauts ou pour d'autres raisons.

15.3.3.1 Validation des données d'entrée et des méthodes pour produire l'historique d'exploitation

15.3.3.1.1 Les données liées à l'évaluation de l'expérience acquise en exploitation du PDS sont obtenues auprès du fournisseur et si possible des utilisateurs de systèmes utilisant le PDS. Pour que l'expérience acquise en exploitation soit considérée apte à l'évaluation du PDS, il faut évaluer les méthodes utilisées pour recueillir les données et pour produire l'historique d'exploitation.

15.3.3.1.2 Seules des informations issues d'un processus de recueil de données bien défini et contrôlé doivent être utilisées.

15.3.3.1.3 Les procédures de recueil doivent être évaluées afin de valider l'exhaustivité et la crédibilité des données. (L'Article I.2 donne des directives pour le recueil et la validation des données.)

15.3.3.1.4 Seule l'expérience acquise en exploitation qui a été collectée dans des conditions similaires à celles de l'utilisation prévue doit être considérée comme valide.

15.3.3.1.5 Le temps d'exploitation cumulé du PDS objet de l'évaluation doit être établi. Il peut être calculé en additionnant le temps d'exploitation de chaque installation pour lequel un retour d'expérience a été recueilli et validé. Il convient d'exclure les temps pendant lesquels le PDS n'a pas été utilisé de manière représentative.

15.3.3.1.6 Les temps d'exploitation à prendre en compte doivent faire référence à des PDS de la même version que celle dont l'utilisation est prévue. Lorsque les temps d'exploitation d'autres versions sont inclus, une analyse des différences et de l'historique de ces versions doit être effectuée. Cette analyse doit identifier les parties et les fonctions du PDS qui diffèrent, et les parties et fonctions qui ne sont pas affectées par les modifications afin de démontrer que cet historique est valide.

6) significant problems can be expected because of the transfer of the PDS to new hardware;
7) the PDS quality is not adequate and the PDS shall be rejected on grounds that the weaknesses are too great or the information inadequate for effective compensation; and
8) the independence of the qualified functions/properties of the PDS from those not qualified has/has not been established.

15.3.3 Evaluation of operating experience

The objective of this evaluation is to provide evidence that suitable operating experience of the PDS may complement the confidence in the PDS in case of deficiencies detected in the quality evaluation.

Those functions/properties of the PDS whose feedback of experience has to be evaluated, shall be identified and the following shall be evaluated:

1) the methods for collection of data on operating experience;
2) the methods for recording the PDS version operating time and for producing the operating history;
3) the operational history of findings, defects and error reports; and
4) the operational history of modifications made for defects or other reasons.

15.3.3.1 Validation of input data and methods for producing the operating history

15.3.3.1.1 The evaluation of operating experience of the PDS is based on data available from the vendor and, if possible, from users of systems running the PDS. In order to consider the operating experience suitable for the evaluation of the PDS, the methods for collection of data and producing the operating history shall be assessed.

15.3.3.1.2 Only information that is derived from a well-defined and controlled data collection process shall be used.

15.3.3.1.3 Collection procedures shall be assessed to validate the data for completeness and credibility. Clause I.2 gives guidance for collection and validation of data.

15.3.3.1.4 Operating experience shall be considered valid only if it is observed under conditions similar to the conditions during intended operation.

15.3.3.1.5 The accumulated operating time for the PDS under evaluation shall be established. It may be calculated by adding together the operating time of each installation for which operational experience has been collected and validated. Time in which the PDS was not operating in a representative manner should be excluded.

15.3.3.1.6 The operating times shall be for the same version of the PDS that is intended to be used. When operating time of other versions is included, an analysis of the differences and history of these versions shall be made. This analysis shall identify the parts and functions of the PDS which differ, and the parts and functions which are not affected by the modifications to demonstrate that the history is valid.

15.3.3.1.7 Les problèmes et défaillances survenus ainsi que leurs corrections dans les différentes versions du PDS doivent être analysés et classés en fonction de leur sévérité. Leurs conséquences sur les fonctions désirées doivent être évaluées.

15.3.3.1.8 Aucune des fonctions qualifiées ne doit être affectée par les erreurs rencontrées ou les modifications apportées sur d'autres fonctions d'un même PDS.

15.3.3.1.9 Il convient que l'évaluation des fonctions de communication prenne en compte l'expérience acquise en exploitation. Il convient que les limites de service soient identifiées et comparées aux prévisions pour les conditions normales, de charge de pointe et de défaillance d'équipements.

15.3.3.1.10 Il convient que l'expérience acquise en exploitation satisfasse aux critères d'acceptation suivants:

1) le PDS a accumulé un temps d'exploitation suffisant (voir Annexe I);

   NOTE Il convient que le temps d'exploitation suffisant soit déterminé au cas par cas sur la base du jugement technique. Il convient que ce jugement prenne notamment en compte le niveau de fiabilité prévisionnel requis au niveau système pour les fonctions pour lesquelles le PDS est utilisé.

2) aucune modification importante n'a été faite, ni aucune erreur importante détectée pendant une période d'exploitation significative sur plusieurs sites ou applications;

3) le PDS a de préférence été exploité sur plusieurs installations.

15.3.3.1.11 Un document d'évaluation doit fournir les conclusions de l'évaluation de l'expérience acquise en exploitation et indiquer:

1) si l'expérience acquise en exploitation pour les fonctions/propriétés du PDS est appropriée et son utilisation pour soutenir l'évaluation de la qualité du PDS est justifiée; ou

2) si l'expérience acquise en exploitation n'est pas appropriée ou suffisamment éprouvée.

15.3.3.2 Critères d'acceptation à utiliser pour la prise en compte de l'expérience acquise en exploitation comme facteur de compensation

L'expérience acquise en exploitation peut être utilisée comme un facteur de compensation pour l'acceptation du PDS, si les critères suivants sont remplis.

15.3.3.2.1 L'évaluation du retour d'expérience ne doit jamais complètement remplacer l'évaluation de la conception du produit lui-même et de sa documentation (voir 15.3.2.2).

15.3.3.2.2 L'expérience acquise en exploitation doit être acceptée comme une partie de la justification seulement si elle est utilisée pour compenser des lacunes dans l'évaluation du PDS par rapport aux recommandations sur la conception du point c) de l'Article B.2 sur les systèmes d'exploitation et les programmes standards.

15.3.3.2.3 L'expérience acquise en exploitation doit être acceptée comme une partie de la justification seulement si elle démontre qu'aucun défaut suspecté ou connu n'est susceptible d'empêcher le fonctionnement d'une fonction de catégorie A ou de la faire fonctionner incorrectement dans le système.

15.3.3.2.4 La rigueur de l'analyse du retour d'expérience doit être cohérente avec la fiabilité du système et il convient que les preuves d'absence de défauts techniques fournies par cette analyse soient cohérentes avec celles obtenues lorsque les Articles 1 à 12 de la présente norme sont appliqués.

15.3.3.1.7 Problems, failures and their correction in the different versions of the PDS shall be analysed and classified according to their severity. Their impact on the intended functions shall be evaluated.

15.3.3.1.8 No qualified function shall be affected by errors found or modifications made to any other functions of the same PDS.

15.3.3.1.9 Evaluation of communication functions should take account of operating experience. The service limits should be identified and compared with the forecasts for normal, peak load and equipment failure conditions.

15.3.3.1.10 Operating experience should be considered as suitable when the following criteria are met:

1) the PDS has achieved a sufficient accumulated operating time (see Annex I);

   NOTE The sufficient operating time should be determined on a case-by-case decision based on engineering judgement. This judgement should take into account notably the anticipated reliability level required at system level for the functions in which the PDS is used.

2) no significant modifications have been made and no errors are detected over a significant operating time on several sites or applications; and
3) the PDS has preferably operated on several installations.

15.3.3.1.11 When the evaluation of the operating experience is completed, a document shall be produced to record either:

1) that the operating experience is suitable for the identified functions/properties of the PDS and the justifications why the operating experience may be used to support assessment of the quality; or
2) that the operating experience is not suitable or is not sufficiently proved.

15.3.3.2 Acceptance criteria to be applied when using operating experience as a compensating factor

Suitable operating experience may be used as a compensating factor for acceptance of the PDS, when the following criteria are met.

15.3.3.2.1 The evaluation of the feedback of operating experience shall never completely replace the evaluation of the design of the product itself and of its documentation (see 15.3.2.2).

15.3.3.2.2 Suitable operating experience shall be accepted as part of the justification only if it is used to compensate for weaknesses in the assessment of a PDS against the design recommendations of item c) of Clause B.2 of this standard on operating systems and standardised programs.

15.3.3.2.3 The operational history shall be accepted as part of the justification only if it demonstrates that no suspected or known defects are able to prevent operation of a category A function or cause it to act incorrectly in the system.

15.3.3.2.4 The rigour of the analysis on feedback of experience shall be consistent with the safety category of the system functions, and the evidence of technical correctness provided by this analysis should be consistent with that achievable when applying Clauses 1 to 12 of this standard.

15.3.3.2.5 Un document d'évaluation final doit être rédigé et indiquer:

1) qu'une expérience d'exploitation satisfaisante compense les faiblesses détectées dans l'évaluation de la conception et du cycle de vie du PDS; ou
2) que l'utilisation du PDS est rejetée parce que les résultats de l'évaluation sont négatifs, ou parce que l'expérience acquise en exploitation n'est pas encore suffisante pour compenser les faiblesses identifiées au cours du développement.

15.3.4 Evaluation globale

15.3.4.1 Lorsque les évaluations et le travail complémentaire nécessaire (modifications du PDS, tests complémentaires, documentation complémentaire) ont été achevés, un document de justification global pour l'utilisation du PDS dans la mise en oeuvre du système informatique doit être rédigé.

15.3.4.2 Le document, basé sur les conclusions des évaluations décrites en 15.3.1, 15.3.2 et 15.3.3, doit enregistrer l'agrément qui démontre que le PDS (ou une partie du PDS) est adapté et a le niveau de qualité approprié pour l'usage prévu dans le système et qu'aucune autre modification n'est nécessaire.

15.4 Exigences liées à l'intégration dans le système et à la maintenance des PDS

15.4.1 Après le processus d'évaluation et d'estimation, la décision d'utiliser le PDS doit être prise formellement et documentée dans le cadre du projet à la suite d'une revue formelle de conception.

15.4.2 Les procédures d'intégration du PDS doivent être décrites dans le plan d'assurance qualité et dans le plan d'intégration du système informatique (voir 6.2.1 et 6.2.3 de la CEI 61513).

15.4.3 Après acceptation, le PDS doit être placé sous la gestion de configuration du système (voir 6.2.1.2 de la CEI 61513) et seules la version soumise à la qualification décrite et les éventuelles modifications nécessaires identifiées par le processus doivent être utilisées.

15.4.4 Le plan qualité du système doit contenir des procédures pour la mise à jour du PDS lorsque l'utilisation d'une nouvelle version devient nécessaire.

15.4.5 Il convient que les informations sur les erreurs et les défaillances provoquées par le PDS sur d'autres sites et applications, et sur les modifications correspondantes du PDS continuent d'être fournies et analysées formellement pendant la durée de vie.

15.3.3.2.5 An evaluation document shall be produced to record either:

1) how satisfactory feedback of experience compensates for any weakness identified in the evaluation of the design and life cycle of the PDS; or
2) that the use of the PDS is rejected because the results of the evaluation are negative, or because the operating experience is not yet sufficient to compensate the weaknesses identified in the development.

15.3.4 Comprehensive assessment

15.3.4.1 When the evaluations and all the necessary complementary work (modifications of the PDS, complementary tests, complementary documentation) have been completed, a comprehensive justification document for the use of the PDS in the implementation of the system shall be prepared.

15.3.4.2 This document, based on the conclusions derived from the evaluations described in 15.3.1, 15.3.2 and 15.3.3, shall record the assessment that demonstrates that the PDS (or part of the PDS) is suitable and has the level of quality appropriate for its intended use in the system and no further modification is needed.

15.4 Requirements for integration in the system and modification of PDS

15.4.1 After the comprehensive assessment, the decision to use the PDS shall be formally made and documented within the project following a formal design review.

15.4.2 The procedures for integration of the PDS shall be described in the system quality assurance plan and system integration plan (see 6.2.1 and 6.2.3 of IEC 61513).

15.4.3 After acceptance, the PDS shall be placed under the system configuration management (see 6.2.1.2 of IEC 61513) and only the release subjected to the qualification described, and any necessary modifications identified by the process shall be used.

15.4.4 The quality plan of the system shall provide procedures for upgrading the PDS when the use of a new release becomes necessary.

15.4.5 Information on errors and failures due to the PDS on other sites and applications, and on corresponding modifications of the PDS, should continue to be accessed and formally analysed during the subsequent life.

Annexe A
(normative)

Cycle de vie et de sûreté du logiciel et détails des exigences du logiciel

A.1 Cycle de vie et de sûreté du logiciel

Le processus de développement du logiciel est illustré par la Figure 3, celle-ci montre les relations existant entre les activités du cycle de vie, de la spécification à la vérification et validation, en passant par la conception et la réalisation.

Les détails portant sur les exigences du logiciel sont donnés dans l'Article A.2.

A.2 Détails des exigences du logiciel

A.2.1 Description des contraintes entre matériel et logiciel

A.2.1.1 Les éléments suivants doivent être décrits:

– caractéristiques fonctionnelles générales (longueur du mot calculateur, types d'échange, vitesse, etc.), dans la plupart des cas une référence au manuel du fabricant du matériel est suffisante;
– caractéristiques fonctionnelles spéciales du matériel (pilotes particuliers, matériels de transmission de données, etc.);
– contraintes arithmétiques;
– exigences du logiciel standard;
– exigences d'auto-contrôle du matériel;
– il doit être fait au moins une référence au document d'exigences matériel.

A.2.1.2 Les exigences réciproques entre matériel et logiciel doivent être définies en ce qui concerne la détection de pannes et la réaction sur l'indication de pannes.

Les critères établis dans le guide NS-G-1.3 de l'AIEA n'exigent pas de développement complémentaire relatif au matériel du système programmé. Cependant, la documentation de toutes les exigences matériel qui ont un impact sur le logiciel est nécessaire pour servir de base à l'intégration du matériel et du logiciel et à la validation du système programmé.

A.2.1.3 L'interface entre fonctions de différents niveaux d'importance pour la sûreté (par exemple fonctions de catégorie A ou fonctions d'autres catégories) réalisées par le logiciel doit être décrite.

A.2.2 Auto-surveillance

A.2.2.1 En cas de pannes, il convient de réagir automatiquement et de façon appropriée en prenant en compte les facteurs suivants:

– les défaillances doivent être identifiées avec un degré de détail raisonnable;
– dans la mesure du possible, on doit garantir que les sorties sont orientées vers la sûreté, en cas de panne;
– si une telle garantie ne peut être fournie, la sortie du système ne doit transgresser que les exigences de sûreté les moins essentielles;

Annex A
(normative)

Software safety life cycle and details of software requirements

A.1 Software safety life cycle

The software development process is illustrated by Figure 3, showing the relationship of the life cycle activities from specification through design and implementation to verification and validation.

The details of software requirements are described in Clause A.2.

A.2 Details of software requirements

A.2.1 Description of constraints between hardware and software

A.2.1.1 The following items shall be described:

– operating characteristics in general (word length, exchange types, speed, etc.); in many cases a reference to the equipment manufacturer manuals is sufficient;
– special equipment operating characteristics (particular drivers, data-communications equipment, etc.);
– arithmetic constraints;
– requirements of standard software packages;
– requirements of hardware self-supervision;
– at least a reference to the hardware requirements document shall be made.

A.2.1.2 The reciprocal requirements between hardware and software shall be determined with respect to failure detection and reaction on failure indications.

The criteria established in the IAEA guide NS-G-1.3 require no additional amplification for computer-based system hardware. However, documentation of all hardware requirements which impact software is necessary to provide the basis for hardware/software integration and computer-based system validation.

A.2.1.3 The interface between functions of different levels of safety significance (e.g. category A and non-category A functions) performed by the software shall be described.

A.2.2 Self-supervision

A.2.2.1 Appropriate automated actions should be taken at failures considering the following factors:

– failures shall be identified to a reasonable degree of detail;
– fail-safe output shall be guaranteed as far as possible;
– if such a guarantee cannot be given, system output shall violate only less essential safety requirements;

– les conséquences de telles pannes doivent être réduites au minimum;
– il convient d'inclure des sous-programmes correctifs, tels que des mises en position de repli, des reprises système;
– des informations suffisamment détaillées sur les défaillances doivent être fournies au personnel de conduite.

A.2.2.2 La conception du système doit garantir qu'un autotest approprié est réalisable. De tels principes de conception sont:

– la modularité;
– les tests intermédiaires de vraisemblance;
– l'emploi de la redondance et de la diversité; la diversité peut être mise en œuvre au niveau fonctionnel ou logiciel;
– les réserves de temps d'exécution et d'espace mémoire pour les besoins d'auto-surveillance;
– la simulation des défaillances peut être utilisée pour confirmer la pertinence des mécanismes d'auto-surveillance.

A.2.2.3 En aucune circonstance, l'auto-surveillance ne doit empêcher le système de réagir dans le temps imparti.

A.2.3 Présentation des spécifications du logiciel

A.2.3.1 Les spécifications du logiciel doivent être présentées de façon à être facilement comprises par tous les groupes d'utilisateurs.

A.2.3.2 La présentation doit être suffisamment détaillée, ne doit pas comporter de contradiction ni, autant que possible, de duplication.

A.2.3.3 Le document doit être exempt de détail de réalisation, complet, cohérent et à jour.

A.2.3.4 Le document de spécifications logiciel doit distinguer clairement les exigences essentielles, des objectifs de moindre importance.

Le document de spécifications logiciel est destiné à être utilisé par:

– ses auteurs;
– le client ou l'utilisateur final;
– l'équipe de développement du logiciel;
– l'équipe de vérification du logiciel;
– les contrôleurs et les autorités de sûreté.

– the consequences of failures shall be minimised;
– remedial routines, such as fall back, system recovery, should be considered for inclusion;
– sufficiently detailed information on failures shall be provided for the operating staff.

A.2.2.2 The system shall be designed in such a way that appropriate self-supervision is feasible. Design principles used to assist this include:

– modularisation;
– intermediate plausibility checks;
– use of redundancy and diversity; diversity may be implemented as functional diversity or software diversity;
– provision of sufficient execution time and memory space for self-checking purposes;
– failure simulation may be used to confirm the adequacy of self-supervision features.

A.2.2.3 Self-supervision shall not prevent timely system reactions in any circumstance.

A.2.3 Presentation of software specifications

A.2.3.1 The software specifications shall be presented in a manner which is easy to understand for all user groups.

A.2.3.2 The presentation shall be detailed sufficiently, free from contradiction, and without duplication as far as possible.

A.2.3.3 The document shall be free of implementation details, complete, consistent and up-to-date.

A.2.3.4 The software specifications document shall distinguish clearly between essential requirements and less stringent targets.

The software specifications document is intended to be used by

– its authors;
– the customer, client or final user;
– the software development team;
– the software verification team;
– assessors and licensing personnel.

Annexe B
(normative)

Exigences et recommandations détaillées relatives à la conception et à la réalisation

L'Annexe B est constituée d'un ensemble de tableaux, chaque tableau débutant par une exigence générale (doit) telle que B1.a, suivie d'une liste de recommandations associées (il convient de), telles que B1.aa.

Pour chaque exigence, il convient qu'un projet choisisse les recommandations devant être satisfaites au niveau de la conception et de la réalisation. Il convient que les recommandations non retenues soient justifiées (par exemple non applicable ou couvert par une autre disposition,etc.).

Les exigences et les recommandations sont données dans les tableaux suivants.

B.1 Processus de conception

B1.a – Aptitude à la modification

Art.	Recommandation	Bon contre/Bon pour
B1.*a*	Le processus de conception doit faciliter la modification du logiciel.	Risque d'introduire des erreurs lors de la réalisation des modifications./
B1.*aa*	Il convient d'identifier au début des études les caractéristiques du logiciel à développer ainsi que les exigences fonctionnelles susceptibles d'évoluer pendant le cycle de vie.	/ Obtention d'une flexibilité pour un coût justifié sans mise cause de la sûreté.
B1.*ab*	Il convient de choisir les modules pour que les modifications probables aient un impact limité sur le logiciel lors des phases d'étude suivantes.	/ Minimiser le risque d'introduction d'erreur lors des modifications.
B1.*ac*	Il convient que l'aptitude à la modification soit soigneusement mise en balance avec l'accroissement de complexité, du temps d'exécution et de l'espace mémoire.	Augmentation de la complexité des systèmes /

B1.b – Approche descendante par affinements successifs

Art.	Recommandation	Bon contre/Bon pour
B1.*b*	Une approche descendante doit être utilisée lors de la conception du logiciel.	Erreurs de conception/Complétude de la conception
B1.*ba*	Il convient que les aspects généraux soient traités avant les aspects de détails	Risques d'incohérence/Permet aux concepteurs de travailler logiquement en partant des exigences pour arriver à la conception finale
B1.*bb*	Il convient de décrire et de vérifier l'ensemble du système à chaque niveau de la conception.	/ Permet de garantir la cohérence et la complétude de la conception, pour découvrir les problèmes au plus tôt.

Annex B
(normative)

Detailed requirements and recommendations for design and implementation

Annex B provides a set of tables, each table starting with a general requirement (shall) such as B1.a, followed by a list of associated recommendations (should), such as B1.aa.

For each requirement, projects should select the recommendations to be met by the design and implementation. Recommendations not chosen should be justified (e.g not applicable, or covered by another measure, etc.).

The requirements and recommendations are listed in the following tables.

B.1 Design process

B1.a Modifiability

Item	Recommendation	Good against/Good for
B1.*a*	The design process shall make the software easily modifiable	Risk of introducing faults when implementing changes /
B1.*aa*	Characteristics of the software to be developed and its functional requirements, which are likely to change during its life cycle, should be identified at an early design stage	/ Achieving safe and cost effective flexibility
B1.*ab*	Modules should be chosen so that impact on the software of anticipated changes is limited for further design stages	/ Minimise the risk of faults due to changes
B1.*ac*	Modifiability should be carefully counter- balanced with resulting overhead in complexity, run time and memory space	Systems being too complex/

B1.b Top-down approach

Item	Recommendation	Good against/Good for
B1.*b*	A top-down approach shall be used in software design	Design errors/Completeness of the design
B1.*ba*	General aspects should precede specific ones	Risks of inconsistency/Allows developers to work logically from requirements through to final design
B1.*bb*	At each level of design, the whole system should be completely described and verified	/ To ensure consistency and completeness of the design, to find areas of difficulty early

Art.	Recommandation	Bon contre/Bon pour
B1.*bc*	Il convient d'identifier les points durs autant que possible au début des études.	/ Les points durs constituent des informations d'entrée pour les décisions concernant la conception.
B1.*bd*	Il convient de revoir et de documenter les principales décisions de conception au début des études.	/ Evaluer la faisabilité de la conception le plus tôt possible. Réduire les nécessités de modification au cours des étapes ultérieures de développement du logiciel.
B1.*be*	En cas de décision majeure ayant des conséquences sur d'autres systèmes, il convient de considérer les solutions alternatives et de documenter leurs risques associés.	Duplication du travail/Conception limitant les risques
B1.*bf*	Il convient d'identifier les conséquences de décisions individuelles sur les autres parties du système.	Duplication du travail/Conception limitant les risques
B1.*bg*	Il convient que les écarts entre les niveaux de conception du logiciel soient tels que les personnes chargées des revues puissent comprendre chaque niveau de conception par rapport au précédent.	Difficultés pour comprendre la conception/Garantir que la conception du logiciel est claire et permettre de vérifier qu'elle est cohérente.
B1.*bh*	Il convient que la conception du logiciel se fasse en utilisant un ou plusieurs formalisme de description de haut niveau (lorsque cela est adapté et efficace), tels que ceux utilisés en logique mathématique, en théorie des ensembles, pour l'algorithmique, pour les tables de décisions, pour les diagrammes logiques ou autres graphiques ou pour les langages orientés application	Erreurs d'interprétation ou imprécisions/
B1.*bi*	Il convient d'utiliser des aides automatisées pour le développement.	/ Réduire le champ des activités sujettes aux erreurs humaines .
B1.*bj*	Il convient que la documentation soit telle que l'auteur des spécifications puisse comprendre et contrôler la réalisation des fonctions au cours de la conception.	/ Modifications et cohérence avec les spécifications.

B1.c – Vérification des produits intermédiaires de la conception

Art.	Recommandation	Bon contre/Bon pour
B1.*c*	Les produits intermédiaires de conception doivent être vérifiés.	/ Découverte des erreurs le plus tôt possible, complétude de la conception
B1.*ca*	Il convient de démontrer que chaque niveau de conception forme un tout complet et cohérent.	Nécessité de modifications ultérieurement /
B1.*cb*	Il convient de démontrer que chaque niveau de conception est cohérent avec le précédent et avec la spécification des exigences du logiciel pertinente pour ce niveau de conception.	Oubli de certains aspects/Garantir qu'aucune exigence n'a été omise
B1.*cc*	Il convient que les vérifications relatives à la cohérence soient réalisées par des personnes qui n'ont pas été impliquées dans le processus de conception.	
B1.*cd*	Il convient que ces personnes indiquent les erreurs possibles mais ne recommandent aucune action.	Identification personnelle avec le programme/Maintenir une attitude critique.
B1.*ce*	S'il y a lieu, il convient qu'elles fournissent aux concepteurs les raisons explicites et pertinentes pour qu'ils comprennent leurs remarques.	/ Augmenter l'intérêt de la vérification ainsi que l'efficacité de l'ensemble de l'équipe.

Item	Recommendation	Good against/Good for
B1.*bc*	Areas of difficulty should be identified as far as possible at an early stage in the design process	/ Potential issues to be an input to design decisions
B1.*bd*	Key design decisions should be documented and reviewed at an early stage of the design process	/ To assess he feasibility of the design as early as possible. To reduce the possibility of changes being required later in the software development.
B1.*be*	In case of a major decision affecting other system parts, the alternatives should be considered and their risks documented	Duplicating work/Careful design
B1.*bf*	Consequences of individual decisions for other system parts should be identified.	Duplicating work/Careful design
B1.*bg*	The gap between software design levels should be such that reviewers can understand each design level in reference to the former level.	Designs difficult to understand/To ensure software design clarity and allows the consistency of the design to be verified.
B1.*bh*	The software design should proceed using one or more higher level descriptive formalisations (where appropriate and effective), such as used in mathematical logic, set theory, pseudo code, decision tables, logic diagrams or other graphic aids or application oriented languages	Misinterpretation or inaccuracy /
B1.*bi*	Automated development aids should be used	/ To reduce the scope for human error
B1.*bj*	Documentation should be such that the specifier is able to understand and check the implementation of the functions in the design.	/ Modification and consistency with specifications

B1.c Verification of intermediate design products

Item	Recommendation	Good against/Good for
B1.*c*	The intermediate design products shall be verified	/ Finding errors as soon as possible, completeness of the design
B1.*ca*	It should be shown that each level of design is complete and consistent in itself	Necessity for later changes /
B1.*cb*	It should be shown that each level of design is consistent with the previous level and with the software requirements specification that is relevant for that design level	Forgetting aspects/To ensure that no requirement has been omitted
B1.*cc*	Consistency checks should be made by personnel not involved in the design process	
B1.*cd*	These personnel should only mark possible faults and not recommend any action	Personal identification with the program/Maintaining critical attitude.
B1.*ce*	They should provide explicit rationale when relevant to help designers correctly understand their findings	/ Increase interest in verification activities and global team efficiency

B1.d – Maîtrise des modifications pendant le développement

Art.	Recommandation	Bon contre/Bon pour
B1.*d*	Les modifications qui sont nécessaires durant le développement du logiciel doivent commencer à l'étape de conception la plus amont encore en rapport avec la modification.	Introduction de nouvelles erreurs dues aux modifications/Eviter les effets induits à distance.
B1.*da*	Il convient que les modifications soient réalisées de l'étape la plus générale vers les étapes plus spécifiques.	/ Identifier tous les effets dus à la modification.
B1.*db*	Lorsqu'un module est modifié, il convient de le tester à nouveau conformément aux principes décrits précédemment (voir 11.2) et avant que celui-ci soit réintégré dans le système.	Erreurs cachées dans le module liées à la modification /
B1.*dc*	De plus, il convient de tester à nouveau les modules appelants et appelés, si ceux-ci sont affectés par la modification.	Erreurs cachées dans l'environnement du module liées à la modification du module /
B1.*dd*	Il convient que la documentation des principales modifications inclue les exigences, le code, les zones de données, les caractéristiques du flot de contrôle et les aspects temporels affectés par la modification.	/ Trace des effets des modifications.
B1.*de*	Il convient que les modifications affectant des parties déjà testées ou les travaux d'autres personnes soient évaluées et revues avant leur incorporation.	/ Effets cachés indirects.
B1.*de.1*	NOTE Cette procédure est valable pour les modifications affectant le travail d'une seule personne ainsi que pour les modifications affectant l'ensemble du système. Dans ce dernier cas les recommandations de l'Article 11 s'appliquent en plus.	

B.2 Structure du logiciel

B2.a – Structures de contrôle et d'accès

Art.	Recommandation	Bon contre/Bon pour
B2.*a*	Les programmes et les sous-programmes doivent être regroupés de façon systématique.	/ Aide à l'évaluation et à la modification.
B2.*aa*	Il convient de réaliser les opérations propres au système dans des parties spécifiques.	/ Aptitude aux tests

B1.d Modification control during the development

Item	Recommendation	Good against/Good for
B1.*d*	Changes which are necessary during program development shall begin at the earliest design stage which is still relevant to the change	Introducing new faults due to changes/Avoiding hidden far distant effects
B1.*da*	Changes should proceed from the general stage to the more specific stages	/ Recognizing all effects of the change
B1.*db*	If any module is changed, it should be re-tested according to the principles described earlier (see subclause 11.2), before it is reintegrated into the system	Hidden faults in module due to change /
B1.*dc*	In addition, related calling and called modules should also be re-tested, as far as they are affected by the change.	Hidden faults in module environment due to change /
B1.*dd*	Documentation of major changes should include the requirements, code parts, data areas, control flow characteristics and time aspect that were affected	/ Traceability of change effects
B1.*de*	Changes affecting already tested parts or the work of other persons should be evaluated and reviewed prior to incorporation	/ Hidden far distant effects
B1.*de. 1*	NOTE This procedure is valid for changes that affect the work of only one person and for modifications that affect the whole system. In the latter case the recommendations of Clause 11 apply additionally.	

B.2 Software structure

B2.a Control and access structures

Item	Recommendation	Good against/Good for
B2.*a*	Programs and program parts shall be grouped systematically	/ Aiding assessment and modification
B2.*aa*	Specific system operations should be performed by specific parts	/ Testability

Art.	Recommandation	Bon contre/Bon pour
B2.*ab*	Il convient d'organiser le logiciel de façon à ce que les parties qui gèrent des fonctions telles que: – interfaces externes au calculateur (par exemple, gestion de périphériques ou d'interruptions); – signaux temps réel (par exemple, horloge); – gestion du parallélisme (par exemple, ordonnanceur); – allocation mémoire; – fonctions particulières (par exemple, utilitaires); – adaptation de «fonctions» standards à la configuration matérielle spécifique du calculateur; soient séparées des programmes d'application avec des interfaces bien définies.	/ Aide pour les tests, clarté de la conception.
B2.*ac*	Il convient que la structure du programme permette la réalisation de modifications à venir avec un minimum d'effort (voir aussi B1.a).	/ Aptitude du système à l'adaptation.
B2.*ad*	Il convient que les méthodes employées pour la structuration du programme soient clairement définies.	/ Compréhension.
B2.*ae*	Autant que possible, il convient que sur un processeur le programme se déroule séquentiellement.	Confusions dues aux problèmes de synchronisation ou aux différentes séquences d'interruption /
B2.*af*	Il convient que les programmes aient une structure modulaire, claire et explicite.	/ Compréhension et aptitude aux tests.

B2.b – Modules

Art.	Recommandation	Bon contre/Bon pour
B2.*b*	Les modules doivent être clairs et intelligibles.	/ Compréhension.
B2.*ba*	Il convient que chaque module corresponde à une fonction spécifique.	/ Aptitude aux tests.
B2.*bb*	Il convient qu'un module ait un seul point d'entrée. Bien que parfois plusieurs points de sortie soient nécessaires, un point de sortie unique est recommandé.	/ Facilité de vérification
B2.*bc*	Il convient que la taille des modules ne dépasse pas un nombre d'instructions donné. Cette limite doit être spécifiée pour un système donné et il convient que les modules plus longs ne soient permis que dans des circonstances spéciales.	Modules trop gros /
B2.*bd*	Il convient que les interfaces entre modules soient aussi simples que possible, uniformisées dans le système et complètement documentées.	Erreurs aux interfaces /
B2.*be*	Il convient de minimiser le nombre de paramètres d'interface des modules.	Erreurs aux interfaces /
B2.*bf*	Il convient de définir clairement le type des paramètres aux interfaces des modules (par exemple entrée, sortie, entrée/sortie).	Erreurs aux interfaces /

Item	Recommendation	Good against/Good for
B2.*ab*	The software should be partitioned so that aspects which handle such functions as: – computer external interfaces (e.g. devices driving, interrupt handling); – real time signals (e.g. clock); – parallel processing (e.g. scheduler); – store allocation; – special functions (e.g. utilities); – mapping standard "functions" onto the particular computer hardware; are separated from application programs with well-defined interfaces between them	/ Aiding testability, clarity of design
B2.*ac*	The program structure should permit the implementation of anticipated changes with a minimum of effort (see also Table B1.a)	/ System adaptability
B2.*ad*	The structuring methods being used should be clearly stated	/ Understandability
B2.*ae*	In one processor, the computer program should work sequentially, as far as possible	Confusion due to timing problems or different interrupt sequences/
B2.*af*	Computer programs should have an explicit and clear modular structure	/ Understandability, testability

B2.b Modules

Item	Recommendation	Good against/Good for
B2.*b*	Modules shall be clear and intelligible	/ Understandability
B2.*ba*	Each module should correspond to a specific function	/ Testability
B2.*bb*	A module should have only one entry. Although multiple exits may be sometimes necessary, single exits are recommended	/ Ease of verification
B2.*bc*	The size of modules should not exceed a limit of executable statements. This limit is to be specified for the particular system and longer modules should be allowed only under special circumstances.	Bulky modules /
B2.*bd*	The interfaces between modules should be as simple as possible, uniform throughout the system and fully documented	Interface faults /
B2.*be*	The number of module interface parameters should be minimized	Interface faults /
B2.*bf*	The status of module interface parameters (i.e. input, output or input/output) should be clearly stated.	Interface faults /

B2.c – Logiciel système opérationnel

Art.	Recommandation	Bon contre/Bon pour
B2.*c*	L'utilisation du logiciel système opérationnel doit être limitée.	Défaillances dues aux erreurs du logiciel système opérationnel/Facilité de vérification du système.
B2.*ca*	Il convient d'utiliser seulement des logiciels système opérationnel soigneusement testés accompagnés de leur documentation de vérification; si le logiciel système opérationnel est un PDS, se référer à l'Article 15.	Erreurs cachées. /
B2.*cb*	Il convient d'éviter l'utilisation de logiciels système opérationnels généralistes (systèmes d'exploitation).	Utilisation de produits logiciels trop complexes. /
B2.*cc*	Si l'utilisation d'un logiciel système opérationnel généraliste est jugée nécessaire, alors il convient de réduire celle-ci à quelques fonctions simples.	/ Pour démontrer l'utilisation d'un logiciel système opérationnel soigneusement testé.
B2.*cd*	Il convient que le logiciel système opérationnel ne comporte que les fonctions nécessaires.	Code mort /
B2.*ce*	Il convient d'appeler les fonctions du logiciel système opérationnel toujours de la même façon.	/ Vérification facilitée.
B2.*cf*	Il convient que les fonctions utilisées pour contrôler le matériel soient des fonctions du logiciel système opérationnel ou soient développées et vérifiées au sein de celui-ci.	Programmation supplémentaire et effort de test /
B2.*cg*	Il convient que les fonctions du logiciel système opérationnel soient définies rigoureusement et qu'elles aient des interfaces bien définies.	Erreurs dans l'utilisation des fonctions /
B2.*ch*	Il convient de connaître et de vérifier les conditions d'utilisation et d'interdépendance des fonctions du logiciel système opérationnel.	Erreurs dans l'utilisation des fonctions /
B2.*ci*	L'ensemble de cette norme doit être appliquée lorsqu'un logiciel système opérationnel dédié est développé totalement ou partiellement pour une application de sûreté particulière.	/ Vérification facilitée.

B2.d –Temps d'exécution

Art.	Recommandation	Bon contre/Bon pour
B2.*d*	L'influence du comportement du procédé technique sur le temps d'exécution doit rester faible.	Problèmes de synchronisation non compréhensibles. /
B2.*da*	Il convient que le temps d'exécution de tout système ou partie de système durant une pointe de charge soit court comparé au temps d'exécution au-delà duquel les exigences de sûreté du système sont transgressées.	Nécessité de modifier tardivement la conception. /
B2.*db*	Les résultats produits par le programme ne doivent pas dépendre: – ni du temps nécessaire pour l'exécution du programme, – ni de l'heure (référence à une «horloge» indépendante) de lancement de l'exécution du programme	Problèmes de synchronisation non compréhensibles/Déterminisme.
B2.*dc*	Il convient de concevoir les programmes pour que les actions s'effectuent dans un ordre correct indépendant de leur vitesse d'exécution.	Problèmes de synchronisation et de temps d'exécution/Analyse.

B2.c Operational system software

Item	Recommendation	Good against/Good for
B2.*c*	Operational system software use shall be restricted	Failures due to operating system errors/Ease of system verification
B2.*ca*	Only thoroughly verified operational system software with its associated verification documentation should be used; if operational system software is a PDS, refer to Clause 15	Hidden errors /
B2.*cb*	The use of general purpose operational system software (operating system) should be avoided	Use of overly complex software products /
B2.*cc*	If a general purpose operational system software is considered necessary, its use should be restricted to a few simple functions	/ Demonstrating use of a thoroughly tested operational system software
B2.*cd*	It should contain only the necessary functions	Dead code /
B2.*ce*	Operational system software functions should be called always in a similar way	/ Ease of verification
B2.*cf*	Functions used to control hardware should be taken from the operational system software or developed and verified within it	Additional programming and test effort /
B2.*cg*	Operational system software functions should be rigorously defined and should have well-defined interfaces	Errors in using the functions /
B2.*ch*	The conditions of use and the interdependences of operational system software functions should be known and checked.	Errors in using the functions /
B2.*ci*	If a dedicated operational system software or a dedicated part is developed for a special safety application, the whole standard shall apply	/ Ease of verification

B2.d Execution time

Item	Recommendation	Good against/Good for
B2.*d*	Physical process behaviour influence on execution time shall be kept low	Unintelligible timing problems /
B2.*da*	The execution time of any system or part of a system under peak load conditions should be short compared to the execution time beyond which the system safety requirements are violated	Necessity of late design changes /
B2.*db*	The results produced by the computer program shall not be dependent on either: – the time taken to execute the program, or – the time (referenced to an independent "clock") at which execution of the programs is initiated	Unintelligible timing problems/Determinism
B2.*dc*	The computer programs should be designed so that operations are performed in a correct sequence independent of their speed of execution	Synchronization problems and run time problems/Analysis

Art.	Recommandation	Bon contre/Bon pour
B2.*dd*	Il convient que le temps d'exécution ne varie pas de façon significative lorsque les données d'entrée changent.	/ Estimation et vérification du temps d'exécution facilitées
B2.*de*	Il convient de documenter l'étendue de variation du temps d'exécution liée aux données d'entrée.	/ Estimation et vérification du temps d'exécution facilitées
B2.*df*	Il convient que les parties de code dont le temps d'exécution dépend des données d'entrée soient courtes.	/ Satisfaction de B2.*de*
B2.*dg*	Il convient que la quantité de données lue à chaque cycle de traitement soit constante.	/ Maintenir les différences de temps d'exécution à un niveau faible.
B2.*dh*	Il convient que le temps d'exécution des programmes ne soit pas dépendant de la réception des données.	Problèmes de synchronisation et de temps d'exécution / Analyse

B2.e – Interruptions

Art.	Recommandation	Bon contre/Bon pour
B2.*e*	L'utilisation des interruptions doit être limitée.	Problèmes de synchronisation et de temps d'exécution/Analyse.
B2.*ea*	Des interruptions peuvent être utilisées si elles simplifient la conception du logiciel et ne rendent pas la vérification excessivement complexe.	/ Facilité de compréhension de configurations spéciales.
B2.*eb*	Il convient d'identifier et de protéger les sections critiques du logiciel (par exemple, par rapport au temps d'exécution, à la modifications de données).	Problèmes de synchronisation et de temps d'exécution/Analyse.
B2.ec	Le logiciel peut inhiber la prise en compte des interruptions durant les sections critiques. Il convient de justifier de telles inhibitions.	Problèmes de synchronisation et de temps d'exécution/Analyse.
B2.*ed*	Lorsque des interruptions sont utilisées, il convient que le temps d'exécution maximal correspondant aux sections non interruptibles soit défini, de telle façon que le temps maximum pendant lequel une interruption est masquée puisse être calculé.	Problèmes de temps d'exécution. /
B2.*ee*	Il convient de documenter avec précision l'usage et le masquage des interruptions.	/ Validation du système.

B2.f – Expressions arithmétiques

Art.	Recommandation	Bon contre/Bon pour
B2.*f*	Lorsque c'est possible il convient d'utiliser des expressions arithmétiques simples plutôt que des expressions complexes.	Effets de bord/Tests facilités.
B2.*fa*	Il convient que les décisions ne dépendent pas de calculs arithmétiques volumineux.	/ Trouver la fonction inverse en calculant des expressions de chemins.
B2.*fb*	Il convient d'utiliser autant que possible des expressions arithmétiques simplifiées et déjà vérifiées.	/ Montrer la relation entre la fonction spécifiée et le code.
B2.*fc*	Lorsque des fonctions arithmétiques volumineuses sont utilisées, il convient de les coder de façon à pouvoir démontrer facilement la cohérence du code par rapport à l'expression arithmétique spécifiée.	/ Analyse du programme.

Item	Recommendation	Good against/Good for
B2.*dd*	Runtime should not vary significantly in response to changes in input data	/ Ease of run time estimation and run time verification
B2.*de*	The extent of runtime variation which may be caused by the inputs should be documented.	/ Ease of run time estimation and run time verification
B2.*df*	Code parts for which execution time depends on input data should be short	/ Reaching B2.*de*
B2.*dg*	The amount of data read during one computation cycle should be constant	/ Keeping run time differences short
B2.*dh*	Programme execution time should not be coupled to receipt of data	Synchronization problems and run time problems/Analysis

B2.e Interrupts

Item	Recommendation	Good against/Good for
B2.*e*	The use of interrupts shall be restricted	Synchronization problems and run time problems/Analysis
B2.*ea*	Interrupts may be used if they simplify the design of the software and do not make the verification overly complex.	/ Ease of understanding of special configurations
B2.*eb*	Critical software parts (e.g. time critical, critical to data changes) should be identified and protected.	Synchronization problems and run time problems/Analysis
B2.*ec*	The software may inhibit the handling of interrupts during critical parts. Such inhibitions should be justified.	Synchronization problems and run time problems/Analysis
B2.*ed*	If interrupts are used, parts not interruptible should have a defined maximum computation time, so that the maximum time for which an interrupt is inhibited can be calculated	Run time problems /
B2.*ee*	Interrupts usage and masking shall be thoroughly documented	/ System validation

B2.f Arithmetic expressions

Item	Recommendation	Good against/Good for
B2.*f*	Where possible simple arithmetic expressions should be used instead of complex ones.	Side effects/Easy testing
B2.*fa*	Decisions should not depend on voluminous arithmetic calculations	/ Finding reverse function calculating path expressions
B2.*fb*	As far as possible, simplified previously verified arithmetic expressions should be used	/ Showing relationship between intended function and code
B2.*fc*	If voluminous arithmetic expressions are used, they should be coded so that their consistency with the specified arithmetic expression can be shown easily from the code.	/ Program analysis

B.3 Autosurveillance

B3.a – Contrôles de vraisemblance

Art.	Recommandation	Bon contre/Bon pour
B3.*a*	Des contrôles de vraisemblance doivent être réalisés (programmation défensive).	Erreurs résiduelles potentielles./
B3.*aa*	Il convient que la correction ou la vraisemblance des résultats intermédiaires soit contrôlée périodiquement.	Erreurs résiduelles potentielles./
B3.*ab*	Il convient que les plages: – des variables d'entrée, – des variables de sortie, – des paramètres intermédiaires, soient contrôlées et en particulier les limites des tableaux.	Erreurs résiduelles potentielles./
B3.*ac*	Il convient que les échecs des contrôles de vraisemblance fassent l'objet de comptes-rendus.	/ Diagnostique des erreurs résiduelles potentielles.

B3.b – Sûreté des sorties

Art.	Recommandation	Bon contre/Bon pour
B3.*b*	Lorsqu'une défaillance est détectée, le système doit produire une sortie bien définie.	Propagation de faute/Comportement orienté vers la sûreté.
B3.*ba*	Lorsque cela est possible, il convient que des techniques de récupération correctes et complètes après erreur soient utilisées.	Perte complète du système due à des défaillances mineures /
B3.*bb*	Lorsque des techniques de récupération d'erreur sont utilisées, toute occurrence d'erreur doit être signalée.	Accumulation de fautes/Correction précoce des erreurs.
B3.*bc*	L'occurrence d'une erreur persistante affectant les fonctions du système doit faire l'objet d'un compte rendu.	Accumulation de fautes/Correction précoce des erreurs.

B.3 Self-supervision

B3.a Plausibility checks

Item	Recommendation	Good against/Good for
B3.*a*	Plausibility checks shall be performed (defensive programming)	Potential residual faults/
B3.*aa*	The correctness or plausibility for intermediate results should be checked periodically.	Potential residual faults /
B3.*ab*	The ranges of: - input variables; - output variables; - intermediate parameters should be checked, including array bound checking	Potential residual faults /
B3.*ac*	Failed plausibility checks should be reported.	/ Diagnostic of potential residual faults

B3.b Safe output

Item	Recommendation	Good against/Good for
B3.*b*	If a failure is detected, the system shall produce a well-defined output	Fault propagation/Failsafe behaviour
B3.*ba*	If possible, complete and correct error recovery techniques should be used	System breakdown due to minor failures /
B3.*bb*	If error recovery techniques are used the occurrence of any error shall be reported	Accumulation of faults/Early fault removal
B3.*bc*	The occurrence of a persistent error affecting the system function shall be recorded	Accumulation of faults/Early fault removal

B3.c – Contenu mémoire

Art.	Recommandation	Bon contre/Bon pour
B3.*c*	Le contenu des mémoires doit être protégé ou surveillé. NOTE La protection peut être considérée sous l'angle prévention (empêcher les modifications intempestives des mémoires) ou sous l'angle détection (repérer les états non appropriés – corruptions de la mémoire) avec un temps de réaction rapide et adaptée (confinement des erreurs ou correction). La détection des erreurs suppose la surveillance (ou son synonyme, la supervision).	Modifications non autorisées, erreurs résiduelles potentielles/
B3.*ca*	L'espace mémoire utilisé par les constantes ou par les instructions doit être protégé en écriture ou surveillé contre toutes modifications.	Propagation des erreurs d'adressage ou des erreurs matérielles, dont les pannes intermittentes. /
B3.*cb*	Il convient que les accès en lecture ou en écriture non autorisés soient empêchés.	
B3.*cc*	Il convient que le système soit sécurisé contre les modifications non autorisées de code ou de données.	/Maintenir le logiciel dans sa forme certifiée.

B3.d – Contrôle des erreurs

Art.	Recommandation	Bon contre/Bon pour
B3.*d*	Un contrôle des erreurs doit être réalisé par le code.	Propagation des défaillances /
B3.*da*	Il convient que des compteurs et des trappes de vraisemblance soient utilisés pour garantir que le programme s'est déroulé correctement.	Erreurs spécifiques du flot de contrôle, pannes matériel intermittentes /
B3.*db*	Il convient de contrôler que toutes les sortes de transfert de paramètres sont correctes, incluant la vérification du type des paramètres.	Erreurs dans la conception des interfaces et le flot de contrôle/
B3.*dc*	Lorsqu'on accède à un tableau, il convient d'en contrôler les limites.	Erreurs dans le flot de données, nombre de répétition trop grand dans une boucle. /
B3.*dd*	Il convient de surveiller le temps d'exécution des parties critiques (par exemple, par une horloge chien de garde).	Erreurs dans le flot de contrôle, nombre de répétition trop grand dans une boucle. /
B3.*de*	Il convient d'utiliser des assertions (par exemple, dans un triangle, Si NON (a+b>c) alors Erreur)	/ Vraisemblance des résultats intermédiaires.

B3.c Memory contents

Item	Recommendation	Good against/Good for
B3.*c*	Memory contents shall be protected or monitored NOTE Protection may be viewed as being either prevention (prevent untimely modification from occurring) or detection (detect an inappropriate state – memory corruption) with appropriate and quick reaction (error confinement or correction). Detection presupposes monitoring (or its synonym: supervision)	Unauthorized changes, potential residual faults /
B3.*ca*	Memory space for constants and instructions shall be write protected or supervised against changes	Propagation of addressing faults or hardware faults, including intermittent faults /
B3.*cb*	Unauthorised reading and writing should be prevented	
B3.*cc*	The system should be secure against code or unauthorised data changes by the plant operator	/Maintaining the software in its licensed form

B3.d Error checking

Item	Recommendation	Good against/Good for
B3.*d*	Error checking shall be performed by the code	Failure propagation /
B3.*da*	Counters and reasonableness traps should ensure that the program structure has been run through correctly	Specific control flow, intermittent hardware faults /
B3.*db*	The correctness of any kind of parameter transfer should be checked, including parameters type verification	Faults in the design of interface and data flow/
B3.*dc*	When addressing an array its bounds should be checked	Data flow faults, too high loop repetition numbers /
B3.*dd*	The run time of critical parts should be monitored (e.g. by a watchdog timer)	Faults in the design of control flow, too high loop repetition numbers /
B3.*de*	Assertions should be used (e.g. in a triangle, if NOT (a+b > c) then Error)	/ Plausibility of intermediate results

B.4 Conception détaillée et codage

B4.a – Branchements et boucles

Art.	Recommandation	Bon contre/Bon pour
B4.*a*	Il convient d'employer les branchements et les boucles avec précaution.	/ Compréhension et vérification du flot de contrôle.
B4.*aa*	Il convient d'utiliser des instructions de boucle plutôt que des branchements arrière qu'il faut éviter (applicable seulement pour les langages de haut niveau).	Difficultés d'analyse du flot de contrôle/Lisibilité.
B4.*ab*	Il convient d'interdire les branchements vers l'intérieur de boucles, de modules ou de sous-programmes.	Difficultés d'analyse du flot de contrôle/Lisibilité.
B4.*ac*	Il convient d'éviter les branchements sortant d'une boucle, si ces branchements n'aboutissent pas précisément à la fin de la boucle. Exception: sortie en cas d'erreur.	Difficultés d'analyse du flot de contrôle/Lisibilité.
B4.*ad*	Dans les modules structurellement complexes, il convient d'utiliser des macros ou des sous-programmes pour que la structure du module apparaisse clairement.	Difficultés d'analyse du flot de contrôle/Lisibilité.
B4.*ae*	Comme mesure spéciale afin d'améliorer le contrôle et la vérification des programmes, il convient d'éviter les instructions de GOTO calculés et les étiquettes variables.	
B4.*af*	Lorsqu'un ensemble de condition de branchements ou d'instructions contrôlées par un «CASE» est utilisé, il convient que l'ensemble des conditions de branchement ou des conditions du «CASE» soit une liste de possibilités exhaustives. Il convient que le concept de «branchement par défaut» soit réservé au traitement des défaillances.	/ Clarifier le «ou» exclusif.
B4.*ag*	Il convient d'utiliser les boucles uniquement avec des limites constantes d'évolution d'indice de boucle.	Problème d'exécution, violation des limites de tableaux/Nombre de chemins contrôlables.

B4.b – Sous-programmes

Art.	Recommandation	Bon contre/Bon pour
B4.*b*	Il convient que les sous-programmes soient organisés le plus simplement possible.	Complexité inutile /
B4.*ba*	Il convient que les sous-programmes aient un nombre maximum prédéfini de paramètres.	/ Conserver des sous-programmes et des interfaces courts et simples.
B4.*bb*	Il convient que les sous-programmes communiquent avec leur environnement uniquement par leurs paramètres.	/ Compréhension et analyse du flot de données.
B4.*bc*	Il convient que les sous-programmes n'aient qu'un point d'entrée.	/ Compréhension et analyse du flot de données.
B4.*bd*	Il convient que le retour des sous-programmes se fasse en un point unique. Exception: sortie en cas d'erreur.	/ Compréhension et analyse du flot de contrôle.
B4.*be*	Il convient que le point de retour suive immédiatement le point d'appel.	/ Compréhension et analyse du flot de contrôle.

B.4 Detailed design and coding

B4.a Branches and loops

Item	Recommendation	Good against/Good for
B4.*a*	Branches and loops should be handled cautiously	/ Understandable and verifiable control flow
B4.*aa*	Backward going branches should be avoided, loop statements should be used instead (only for higher level languages)	Difficulties with control flow analysis/readability
B4.*ab*	Branches into loops, modules or subroutines should be barred	Difficulties with control flow analysis/readability
B4.*ac*	Branches out of loops should be avoided, if they do not lead exactly to the end of the loop. Exception: failure exit	Difficulties with control flow analysis/readability
B4.*ad*	In modules with complex structure, macros, or subroutines should be used so that structure stands out clearly	Difficulties with control flow analysis/readability
B4.*ae*	As a special measure to support program proving and verification computed GOTO statements as well as label variables should be avoided.	
B4.*af*	Where a list of alternative branches or case controlled statements are used, the list of branch or case conditions should be an exhaustive list of possibilities. The concept of a "default branch" should be reserved for failure handling	/ Clarifying exclusive "or"
B4.*ag*	Loops should only be used with constant maximum loop variable ranges	Run time problems, violating array boundaries/Surveyable path number

B4.b Subroutines

Item	Recommendation	Good against/Good for
B4.*b*	Subroutines should be organised as simply as possible	Unnecessary complexity /
B4.*ba*	They should have only a predefined maximum number of parameters	/ Keeping routines and interfaces short and simple
B4.*bb*	They should communicate exclusively via their parameters to their environment	/ Understandability of data flow, data flow analysis
B4.*bc*	Subroutines should have only one entry point	/ Understandability of control flow, control flow analysis
B4.*bd*	Subroutines should return to only one point for each subroutine call. Exception: default exit	/ Understandability of control flow, control flow analysis
B4.*be*	The return point should immediately follow the point of call	/ Understandability of control flow, control flow analysis

B4.c – Structures imbriquées

Art.	Recommandation	Bon contre/Bon pour
B4.*c*	Les structures imbriquées doivent être employées avec soin.	/ Intelligibilité
B4.*ca*	Il convient d'éviter l'emploi de macros imbriquées.	Production de code trop complexe. /
B4.*cb*	Il convient d'éviter le regroupement de différents types d'action du programme par le biais d'instructions de boucles imbriquées ou par le biais de sous-programmes imbriqués, si cela obscurcit la relation entre la structure du problème et celle du programme.	/ Favoriser les structures séquentielles.
B4.*cc*	Il convient de hiérarchiser les sous-programmes et les boucles si cela clarifie la structure du programme	/ Matérialiser les différents niveaux d'abstraction de l'approche descendante de conception.

B4.d – Adressage et zones de données

Art.	Recommandation	Bon contre/Bon pour
B4.*d*	Des techniques simples d'adressage doivent être utilisées.	/ Faciliter l'analyse du flot de données.
B4.*da*	Il convient d'utiliser une seule technique d'adressage pour chaque type de données.	/ Interface uniformisée de la base de données.
B4.*db*	Il convient d'éviter les calculs d'index compliqués.	/ Compréhension du flot de données.
B4.*dc*	Il convient que la longueur des tableaux soit prédéfinie et fixe.	Difficultés d'exécution, flot de contrôle complexe/Faciliter l'analyse du flot de données.
B4.*dd*	Il convient que le nombre de dimensions dans toutes références à un tableau soit le même que le nombre déclaré.	/ Compréhension du processus d'adressage.

B4.e – Structures de données

Art.	Recommandation	Bon contre/Bon pour
B4.*e*	Les structures de données et les conventions de nom doivent être utilisées uniformément dans tout le système.	/ Analyse du flot de données, compréhension intuitive de la signification des données élémentaires.
B4.*ea*	Il convient que les variables, tableaux et cellules mémoire aient un seul objet et une seule structure. Il convient d'éviter l'emploi de techniques d'équivalence.	Erreurs dans l'utilisation des données, problèmes artificiel de temps/Trace du flot de données.
B4.*eb*	Il convient que le nom de chaque variable rappelle son objet.	/ Compréhension intuitive des données élémentaires.
B4.*ec*	Il convient d'implanter les constantes et les variables dans des zones différentes de la mémoire.	Pollution du code et des données/Auto-surveillance du matériel.
B4.*ed*	Lorsqu'une base de donnée partagée ou un moyen similaire est utilisé comme structure globale de données, il convient que cette structure soit accessible au moyen de sous-programmes standards ou au travers de communications par des tâches standards.	
B4.ee	Il convient que les programmes qui reçoivent ou émettent des données en provenance/vers d'autres programmes échangent des paquets de données cohérents.	Incohérence des données /

B4.c Nested structures

Item	Recommendation	Good against/Good for
B4.*c*	Nested structures shall be handled with care	/ Intelligibility
B4.*ca*	Nested macros should be avoided	Producing overly complex code /
B4.*cb*	Joining of different types of program action by means of nested loop statement or nested subroutines should be avoided, if they obscure the relationship between problem structure and program structure	/ Favouring sequential structures
B4.*cc*	Hierarchies of subroutines and loops should be used, if they clarify the program structure	/ Indicating different levels of abstraction during top-down design

B4.d Addressing and arrays

Item	Recommendation	Good against/Good for
B4.*d*	Simple addressing techniques shall be used	/ Ease of data flow analysis
B4.*da*	Only one addressing technique should be used for each data type	/ Uniform interface to data base
B4.*db*	Bulky computations of indexes should be avoided	/ Understandable data flow
B4.*dc*	Arrays should have a fixed, predefined length	Run time difficulties, complex control flow/Ease of data flow analysis
B4.*dd*	The number of dimensions in every array reference should equal the number of dimension in its corresponding declaration	/ Understanding addressing process

B4.e Data structures

Item	Recommendation	Good against/Good for
B4.*e*	Data structures and naming conventions shall be used uniformly throughout the system	/ Data flow analysis, intuitive comprehension of the significance of data elements
B4.*ea*	Variables, arrays and memory cells should have a single purpose and structure. The use of equivalence techniques should be avoided	Errors in data use, artificial timing problems/Traceability of data flow
B4.*eb*	Each variable's name should reflect its use	/ Intuitive comprehension of data element
B4.*ec*	Constants and variables should be located in different parts of the memory	Poisoning of data and code/Hardware self-+ supervision
B4.*ed*	When a universally accessible “data base” or similar resource is used as global data structures, they should be accessed via standard resource handling subroutines or via communication with standard resource manipulating tasks.	
B4.*ee*	Programs which receive or transmit data from/to other programs should exchange consistent data sets	Incoherence of data /

B4.f – Modifications dynamiques

Art.	Recommandation	Bon contre/Bon pour
B4.*f*	On doit éviter les modifications dynamiques du code exécutable.	/ Analyse du flot de contrôle.

B4.g – Tests unitaires et tests d'intégration

Art.	Recommandation	Bon contre/Bon pour
B4.*g*	Des tests unitaires et d'intégration doivent être réalisés durant le développement du programme.	/ Découverte des erreurs le plus tôt possible.
B4.*ga*	Il convient que l'approche de test suive l'approche de conception (par exemple, durant la conception descendante, il convient que les tests soient réalisés en utilisant des simulations des parties du système qui n'existent pas encore. Une fois la conception et la réalisation achevées, il convient que celle-ci soient suivies de tests d'intégration ascendants).	/ Découverte des erreurs le plus tôt possible.
B4.*gb*	Il convient que chaque module soit testé soigneusement avant son intégration dans le système et que les résultats des tests soient documentés.	Difficultés après l'intégration/Gestion des modifications.
B4.*gc*	Il convient de formaliser la description des entrées et des résultats des tests (protocole de test).	Duplication du travail/Accélération de la certification.
B4.*gd*	Il convient d'enregistrer et d'analyser les erreurs rencontrées durant le programme de test.	/ Détection de certaines erreurs de conception.
B4.*ge*	Il convient d'enregistrer les tests incomplets.	/ Clarté.
B4.*gf*	De façon à faciliter l'utilisation des résultats des tests unitaires et d'intégration lors de la validation finale du système, il convient d'enregistrer le niveau de test précédemment atteint (par exemple, l'ensemble des chemins testés dans le module).	Duplication du travail /

B.5 Recommandations dépendant du langage

B5.a – Organisation du programme

Art.	Recommandation	Bon contre/Bon pour
B5.*a*	Des règles détaillées d'organisation des diverses constructions des langages utilisés doivent être élaborées	/ Obtenir une forme commune et intelligible des listages de programme.
B5.*aa*	Il convient que les recommandations incluent la séquence des déclarations, y compris le type des paramètres,	
B5.*ab*	La séquence d'initialisation	
B5.*ac*	La séquence de code non exécutable/ exécutable,	
B5.*ad*	La séquence de déclaration des formats (par exemple pour des langages tel que FORTRAN)	

B4.f Dynamic changes

Item	Recommendation	Good against/Good for
B4.*f*	Dynamic changes to executable code shall be avoided	/ Control flow analysis

B4.g Unit and integration tests

Item	Recommendation	Good against/Good for
B4.*g*	Unit and integration tests shall be performed during the program development	/ Finding faults as soon as possible
B4.*ga*	The approach to testing should follow the approach to design (e.g. during top down design testing should be made by using simulation of not yet existing system parts – stubs –; after completion of system design and implementation this should be followed by bottom up integration testing)	/ Finding faults as soon as possible
B4.*gb*	Each module should be tested thoroughly before it is integrated into the system and the test results documented	Difficulties after integration/Change management
B4.*gc*	A formal description of the test inputs and results (test protocol) should be produced	Duplication of work/Speeding up licensing
B4.*gd*	Faults which are detected during program testing should be recorded and analysed	/ Detection of some design faults
B4.*ge*	Incomplete testing should be recorded	/ Clarity
B4.*gf*	In order to facilitate the use of unit and integration test results during final validation, the former degree of testing achieved should be recorded (e.g. all paths through the module tested)	Duplication of work /

B.5 Language dependent recommendations

B5.a Sequences and arrangements

Item	Recommendation	Good against/Good for
B5.*a*	Detailed rules shall be elaborated for the arrangement of various language constructs	/ Getting a uniform and intelligible shape of program listings
B5.*aa*	The recommendations should include sequence of declarations, including parameter types	
B5.*ab*	Sequence of initialisations	
B5.*ac*	Sequence of non-executable code/executable code	
B5.*ad*	Sequence of formats declaration (e.g for languages such as FORTRAN)	

B5.b – Commentaires

Art.	Recommandation	Bon contre/Bon pour
B5.*b*	Les relations entre les commentaires et le code doivent relever de règles détaillées et fixes.	Difficultés d'écriture et de compréhension des commentaires./Obtenir des commentaires significatifs.
B5.*ba*	Il convient de préciser ce qui doit faire l'objet de commentaires.	
B5.*bb*	Il convient d'uniformiser la position des commentaires.	
B5.*bc*	Il convient d'uniformiser la forme et le style des commentaires.	

B5.c – Assembleur

Art.	Recommandation	Bon contre/Bon pour
B5.*c*	Si un langage assembleur est utilisé, des règles de programmation détaillées et documentées doivent être suivies.	Difficultés de l'assembleur, astuces/Simplicité et compréhension.
B5.*ca*	Il convient de ne pas utiliser les instructions de branchement faisant appel à une substitution d'adresse. Il convient que le contenu des tables de branchement soit constant.	Branchements dont la destination ne peut être identifiée à partir du code au niveau du point de branchement/Améliorer la testabilité et la clarté.
B5.*cb*	Il convient que tous les adressages indirects suivent le même schéma.	/ Clarté de l'adresse finale des zones mémoire.
B5.*cc*	Il convient d'éviter les décalages indirects.	/ Visualiser immédiatement l'amplitude d'un décalage.
B5.*cd*	Il convient d'éviter les substitutions multiples ou indexages multiples à l'intérieur d'une même instruction machine.	/ Compréhension de l'emplacement adressée.
B5.*ce*	Il convient d'utiliser toujours le même nombre de paramètres pour l'appel d'une même macro.	/ Compréhension de la fonction de la macro.
B5.*cf*	Il convient d'utiliser des étiquettes (nom d'emplacements) pour faire des références dans le code. Il convient d'éviter les valeurs numériques (adresses absolues ou relatives).	/ Association d'une signification à la destination du branchement.
B5.*cg*	Il convient que les conventions d'appel des sous-programmes soient uniformisées dans le code et spécifiées par des règles complémentaires.	Zones et types de paramètres arbitraires. /

B5.b Comments

Item	Recommendation	Good against/Good for
B5.*b*	Relations between comments and the code shall be fixed by detailed rules	Difficulties with both writing and understanding comments/Getting meaningful comments
B5.*ba*	It should be made clear what has to be commented	
B5.*bb*	The position of comments should be uniform	
B5.*bc*	Form and style of comments should be uniform	

B5.c Assembler

Item	Recommendation	Good against/Good for
B5.*c*	If an assembly language is used, extended and documented coding rules shall be followed	Difficulties of assembler programming, tricks/Simplicity and understandability
B5.*ca*	Branching instructions using address substitution should not be used. Branch table contents should be constant	Branches whose goal cannot be identified from the code at the branching point/Improve testability and clarity
B5.*cb*	All indirect addressing should follow the same scheme	/ Clarity of ultimately addressed memory locations
B5.*cc*	Indirect shifting should be avoided	/ Seeing immediately how far shift goes
B5.*cd*	Multiple substitutions or multiple indexing within a single machine instruction should be avoided	/ Understanding addressed location
B5.*ce*	The same macro should always be called with the same number of parameters	/ Understanding the macro's function
B5.*cf*	Labels (named locations) should be used to make references within the code. Numerical values (either absolute addresses or relative offsets) should be avoided	/ Associating a meaning with the branching goal
B5.*cg*	Subroutine call conventions should be uniform throughout the coding and specified by further rules	Arbitrary parameters types and locations /

B5.d – Règles de programmation

Art.	Recommandation	Bon contre/Bon pour
B5.*d*	Des règles de programmation détaillées doivent être publiées.	/ Améliorer la clarté et la cohérence du code.
B5.*da*	Il convient de préciser où et comment le code doit être indenté.	/ Identification des blocs.
B5.*db*	Il convient d'uniformiser l'organisation des modules.	/ Compréhension de la structure des modules.
B5.*dc*	Il convient de régler les détails apparaissant ultérieurement en fonction des besoins.	

B5.e – Langages orientés application

Art.	Recommandation	Bon contre/Bon pour
B5.*e*	Il convient d'utiliser des langages orientés application plutôt que des langages orientés machine.	/ Compréhension, simplicité.
B5.*ea*	Il convient de développer des modules indépendants pour les fonctions ou les parties de fonctions pour lesquelles le langage orienté application ne convient pas.	
B5.*eb*	Il convient que les langages orientés application possédant une syntaxe graphique présentent aussi un langage textuel associé.	/ Utilisation d'outils d'analyse automatiques.
B5.*ec*	Il convient d'identifier, de documenter et finalement d'éviter l'emploi de toutes les caractéristiques du langage orienté application qui ne sont pas appropriées au développement des systèmes de classe 1.	

B5.f – Génération automatique de code

Art.	Recommandation	Bon contre/Bon pour
B5.*fa*	Il convient de pouvoir mettre en correspondance les sorties du générateur de code avec les entrées.	/ Aptitude à vérifier les sorties du générateur de code durant la qualification ou l'utilisation.
B5.*fb*	Il convient que le code généré soit lisible.	/ Découverte des erreurs dues au générateur de code.
B5.*fc*	Aucune modification ne doit être faite sur le code généré. Si des modifications sont nécessaires, elles doivent être faites au niveau des entrées.	/ Maintenir la cohérence entre les sorties et les entrées. Assurance qualité du code généré.
B5.*fd*	Il convient que le langage employé pour la génération du code soit conforme aux recommandations applicables de l'Annexe D.	/ Disponibilité d'outils logiciels pour la compilation, et la V&V. Lisibilité.

B5.d Coding rules

Item	Recommendation	Good against/Good for
B5.*d*	Detailed coding rules shall be issued	/ Improve clarity and consistency of code
B5.*da*	It should be made clear, where and how code lines are to be indented	/ Identification of blocks
B5.*db*	Module layout should be fixed uniformly	/ Understanding module structure
B5.*dc*	Further details should be regulated according to need	

B5.e Application oriented languages

Item	Recommendation	Good against/Good for
B5.*e*	Application oriented languages should be used rather than machine-oriented languages	/ Understandability, simplicity
B5.*ea*	Functions or part of functions not achievable with application oriented languages should be developed as independent modules	
B5.*eb*	Application oriented languages with a graphical syntax should offer an associated literal language	/ Use of automated analysis tools
B5.*ec*	Any features of application oriented languages not appropriate for a class 1 system development should be identified and documented and finally avoided	

B5.f Automated code generation

Item	Recommendation	Good against/Good for
B5.*fa*	The output of the code generator should be traceable to the input	/ Ability to verify the code generator output during qualification or use
B5.*fb*	The generated code should be readable	/ Finding faults due to the code generator
B5.*fc*	No changes shall be made at generated code level. If changes are necessary, they shall be introduced and documented at input level	/ Keeping consistency of output with input /Assurance of software quality of the generated code
B5.*fd*	The language used for the generated code should comply with the recommendations of Annex D, where applicable	/ Availability of software tools for compilation and V&V, readability

Annexe C
(informative)

Exemple d'ingénierie à base de logiciel orienté application (développement de logiciel avec un langage orienté application)

L'ingénierie à base de logiciel orienté application s'est développée rapidement depuis la publication de la première édition de la CEI 60880 en 1986.

Les familles d'équipement (système programmé basé sur une plate-forme d'équipements) utilisables pour l'automatisation de tâches industrielles sont maintenant très largement disponibles sur le marché.

De puissants outils d'ingénierie, développés pour des raisons économiques aussi bien que pour satisfaire aux exigences d'assurance qualité, font partie intégrante de ces plates-formes.

La réalisation automatique des parties les plus problématiques du processus de développement du logiciel caractérise ce type d'ingénierie.

Cette annexe fournit un exemple de mise en œuvre des exigences de cette norme dans le contexte spécifique de développement logiciel avec un langage orienté application.

C.1 Principe d'application des exigences

La qualité du logiciel est un élément essentiel pour garantir la sûreté et la qualité globales de systèmes de contrôle-commande numériques importants pour la sûreté.

Les éléments de base permettant d'assurer la conformité aux exigences de cette norme sont:

a) un processus de développement logiciel structuré en phases auxquelles sont associées des définitions précises des entrées et des sorties (voir 5.4);
b) une structure du logiciel compréhensible, développée durant la phase de conception logiciel qui est la base de réalisation du code et de l'évaluation du logiciel (voir 7.1);
c) des règles et des pratiques de programmation conformes aux exigences et aux recommandations de l'Annexe B (voir 7.3.2);
d) la possibilité d'établir la trace des liens de filiation depuis les exigences jusqu'au code du programme final (voir 7.4).

Le processus de développement logiciel, objet de cet exemple, a été fortement influencé par les outils logiciels. Néanmoins, il est structuré en phases possédant des entrées et des sorties clairement définies.

Le générateur de code a été conçu pour satisfaire aux exigences d'ingénierie propres à la conception du logiciel. En conséquence, la structure du code généré garantit un découpage clair du code par rapport aux fonctions spécifiées.

De la même façon les règles de programmation ont été mises en œuvre au niveau du générateur de code pour que le code généré satisfasse aux exigences et aux recommandations de l'Annexe B de cette norme.

L'ensemble d'outils utilisé permet d'établir la trace entre les exigences en entrée de toutes les phases du processus de développement jusqu'au code intégré dans le système cible.

Annex C
(informative)

Example of application oriented software engineering (software development with application-oriented language)

Application oriented software engineering has taken a rapid development since the first edition of IEC 60880 in 1986.

Equipment families (system platforms) for industrial automation tasks are now widely available on the market.

Powerful engineering tools, which are developed for economical reasons as well as for quality assurance, are an integral part of these platforms.

A typical aspect of the application oriented software is that the most challenging parts of the traditional software development process are performed automatically.

This annex gives an example of how the requirements of this standard may be implemented in the context of a typical software development with an application-oriented language.

C.1 Principle of application of requirements

Software quality is an essential part to ensure the overall quality and safety of computer-based I&C systems important to safety.

Basic factors to ensure compliance with the requirements of this standard are:

a) a software development process structured into phases with clear input and output definitions (see 5.4);

b) an understandable software structure developed during the software design phase which forms the basis for the coding as well as for software assessment (see 7.1);

c) coding rules and practices in accordance with the requirements and recommendations given in Annex B (see 7.3.2);

d) traceability of input requirements down to the final software code (see 7.4).

In the case of this example, the software development process is strongly determined by software tools. Nevertheless, it has been structured into phases with clear input and output definitions.

The code generator has been designed to comply with engineering requirements related to the software design. In consequence, the generated code has a structure which ensures clear assignment of the code to the specified functions.

In the same way, coding rules have been implemented in the code generator so that the generated code meet the requirements and recommendations given in Annex B of this standard.

The tool set applied supports traceability of input requirements during all phases of the development process up to the executable code integrated into the target system.

C.2 Mise en œuvre des exigences dans le cycle de vie du logiciel

L'utilisation de langages orientés application reposant sur des outils pour la génération automatique du code a un impact significatif sur le cycle de vie du logiciel.

La Figure C.1 ci-dessous présente un exemple de cycle de vie pour le développement de logiciel orienté application. Par rapport au cycle de vie du logiciel traditionnel, les phases de conception détaillée, codage, intégration des modules et test sont intégrées dans des processus automatisés.

Au cours du projet, les activités relatives à la spécification et une partie de celles relatives à la validation sont généralement assurées par des ingénieurs spécialistes du procédé, tandis que les activités liées à la conception du système de contrôle-commande, à la spécification fonctionnelle et au test du système sont réalisées par des ingénieurs spécialisés en contrôle-commande.

La génération du logiciel d'application peut commencer immédiatement après que les spécifications du système de contrôle-commande et des fonctions associées aient été établies.

Ainsi, il est possible d'analyser les fonctions spécifiées en se servant du code généré sollicité par simulateur ou à l'aide de trajectoires des données d'entrée.

Cette évaluation fonctionnelle pouvant être réalisée avant même que le matériel du système cible ne soit fabriqué peut permettre d'améliorer la qualité de la conception par une détection précoce des défauts entachant celle-ci.

L'outillage couvre le cycle de vie logiciel dans sa globalité, à savoir: la spécification, la conception, y compris la vérification et la validation, l'exploitation du système et les modifications correspondant à des évolutions ultérieures des exigences initiales.

Les formats de documentation adoptés pour les spécifications relatives à la conception du système et des fonctions associées sont la base d'une vérification efficace. Cette caractéristique de l'ingénierie à base de logiciel orienté application satisfait aux exigences des Articles 8 et 10 de cette norme.

C.3 Application des exigences à la génération automatique de code

La génération de code automatique est une caractéristique courante de l'état de l'art relatif aux familles d'équipement (système programmé basé sur une plate-forme d'équipements), celle-ci est garante d'une conception efficace des systèmes de contrôle-commande comme d'une production de logiciels du niveau de qualité le plus élevé.

De telles familles d'équipement comportent un ensemble d'outils support du processus d'ingénierie logiciel.

Lorsqu'elle est conforme à cette norme, la méthodologie de génération automatique de code assure la production de logiciel de haute qualité et ainsi réduit notablement la probabilité d'introduction d'erreurs humaines.

C.3.1 Outils de conception et de gestion de projet

Dans l'exemple présenté ici, les outils logiciels génèrent le code source de l'application à partir de spécifications formalisées (à savoir, des spécifications graphiques) dans un langage soigneusement choisi:

C.2 Application of requirements for software life cycle

The use of application-oriented languages supported by tools for automated code generation influences significantly the software life cycle.

Figure C.1 below shows an example of life cycle for application oriented software development. In comparison to the classical software life cycle, the phases for detailed software design, software coding, module integration and testing are integrated into automated processes.

During project execution the activities related to specification and parts of the system validation are typically performed by process engineers whereas the activities for I&C system design, functional specification and system testing are performed by I&C engineers.

Application software can be generated immediately after the specification of I&C system and related functions has been set up.

Hence, the possibility exists to assess the specified functions by means of the generated code against a simulator or specific input data trajectories.

This functional assessment which can be performed even before the hardware of the target system is manufactured may improve the design quality by allowing early detection of design faults.

The software tool set supports the whole software life cycle i.e. specification, design including verification and validation, system operation and modifications in case of later changes of the input requirements.

The formats to document the specification of the system design and the related functions support effective verification. This feature of the application-oriented software engineering complies with the requirements of Clauses 8 to 10 of this standard.

C.3 Application of requirements for automated code generation

Automated code generation is a typical feature of state of the art equipment families (system platforms) to ensure efficient design of I&C systems as well as software production on the highest quality level.

Such equipment families include a tool set to support the software engineering process.

The methodology of automated code generation provides software of high quality when complying with this standard and reduces respectably the potential for the introduction of human error.

C.3.1 Tools for design and project management

In the example denoted here, the software-based tools generate the application source code from a formalised specification (e.g. graphical specification) in a language which has been carefully chosen:

a) l'utilisation de notations orientées application pour la spécification de l'architecture des systèmes de contrôle-commande et des fonctions associées, facilite les relations entre les concepteurs du système et ceux du contrôle-commande;

b) les notations en entrée utilisent des représentations graphiques spécifiques des fonctions de contrôle-commande à réaliser. Ces représentations sont comparables aux diagrammes fonctionnels traditionnels et éprouvés; ceci permet une vérification complète des fonctions spécifiées;

c) la documentation produite permet une corrélation rigoureuse avec résultats du développement.

L'ensemble des outils est une aide pour les activités d'ingénierie logiciel suivantes:

– possibilité de spécifier l'architecture matérielle en conformité avec les exigences relatives à la tolérance aux fautes, ce qui conduit à la structure de redondance nécessaire;

– démonstration de l'exactitude de la syntaxe de la spécification du logiciel par utilisation d'outils de vérification intégrés;

– aide pour une gestion de configuration efficace par l'identification rigoureuse de tous les composants logiciel (par exemple, par vérification de CRC);

– aide à la vérification pour garantir le déroulement correct des activités d'ingénierie associées à chaque phase de conception;

– aide à la vérification de la conception au moyen de simulations;

– aide au diagnostic et à l'inspection du logiciel installé sur le système cible au moyen d'outils de service;

– aide à la conception des systèmes de contrôle-commande au moyen d'outils évaluant les cas de charges des processeurs et des bus les plus pénalisants;

– aide à l'administration des données de conception au moyen de bases de données pour enregistrer toutes les informations de conception pertinentes.

C.3.2 Outils de génération automatique

Les outils logiciels de génération automatique de code présentés dans cet exemple offrent les caractéristiques suivantes:

a) la génération automatique du code couvre l'ensemble du domaine des fonctions d'application, des données d'application et des communications entre toutes les unités de traitement d'un système de classe 1;

b) les notations utilisées en entrée de la génération automatique de code ont été spécifiées (syntaxe et sémantique);

c) les fonctions d'application sont spécifiées à l'aide de diagrammes fonctionnels;

d) les outils permettent de répartir les fonctions conçues dans des unités de traitement dédiées;

e) des règles de conception logiciel éprouvées sont intégrées dans l'outil et mises en œuvre pour assurer que:

   – la structure du code source généré est claire et satisfait aux exigences générales de cette norme;

   – le flot de contrôle du programme est indépendant de la séquence d'entrée de la spécification graphique;

   – une vérification rigoureuse de faisabilité algébrique évite les erreurs d'exécution;

   – les ressources du système sont allouées statiquement.

f) le code est généré en langage de haut niveau qui permet l'utilisation de compilateurs standards pour produire le code objet associé au système cible. Cette génération permet d'établir directement les liens documentés avec les codes de simulation de l'installation, pour l'analyse des transitoires ou des perturbations.

a) the use of application-oriented notations for specification of I&C systems architecture and associated functions help solving interface between system designers and I&C designers;

b) the input notations have been based on specified graphical representations for the intended I&C functions comparable to the classical proven function diagrams and enable comprehensive checking of the functions specified;

c) the documentation produced allows strict correlation to the development results.

The set of software tools provides assistance for the following software engineering activities:

– possibility to specify the hardware architecture in accordance with the required fault tolerance which leads to the necessary redundancy structure;

– demonstration of syntactic correctness for the software specification by the use of integrated check tools;

– assistance for an effective configuration management by unmistakable identification of all software components (e. g. by CRC check sum);

– assistance for verification to ensure correct engineering related to each design phase;

– assistance for verification of the design by means of simulation;

– assistance for diagnosis and inspection of the software processed on the target system by means of service tools;

– assistance for I&C systems design by means of tools to evaluate worst case processor load and worst case bus load;

– assistance for the administration of design data by a database for all relevant design data.

C.3.2 Tools for automated generation

The software tools for automated code generation of the example include the following features:

a) the automated code generation covers the full scope of application functions, application data and the communication between all the processing units of the class 1 system;

b) the input notations for the automated code generation have been specified (syntax and semantics);

c) the application functions are specified by functional diagrams;

d) the tools allow the assignment of the designed functions to dedicated processing units;

e) proven software design rules integrated in the tool have been implemented to ensure:

   – clear structure of the generated source code to meet the general requirements stated in this standard;

   – program control flow is independent from the sequence of inputs to the graphical specification;

   – consequent checks on algebraic performability to avoid software exceptions have been put in;

   – static allocation of system resources is used.

f) the code is generated in a high level language which enables the application of standard compilers to produce the executable object code for the target system. It is produced in a traceable way directly linked to plant simulation codes for transient or disturbance analysis.

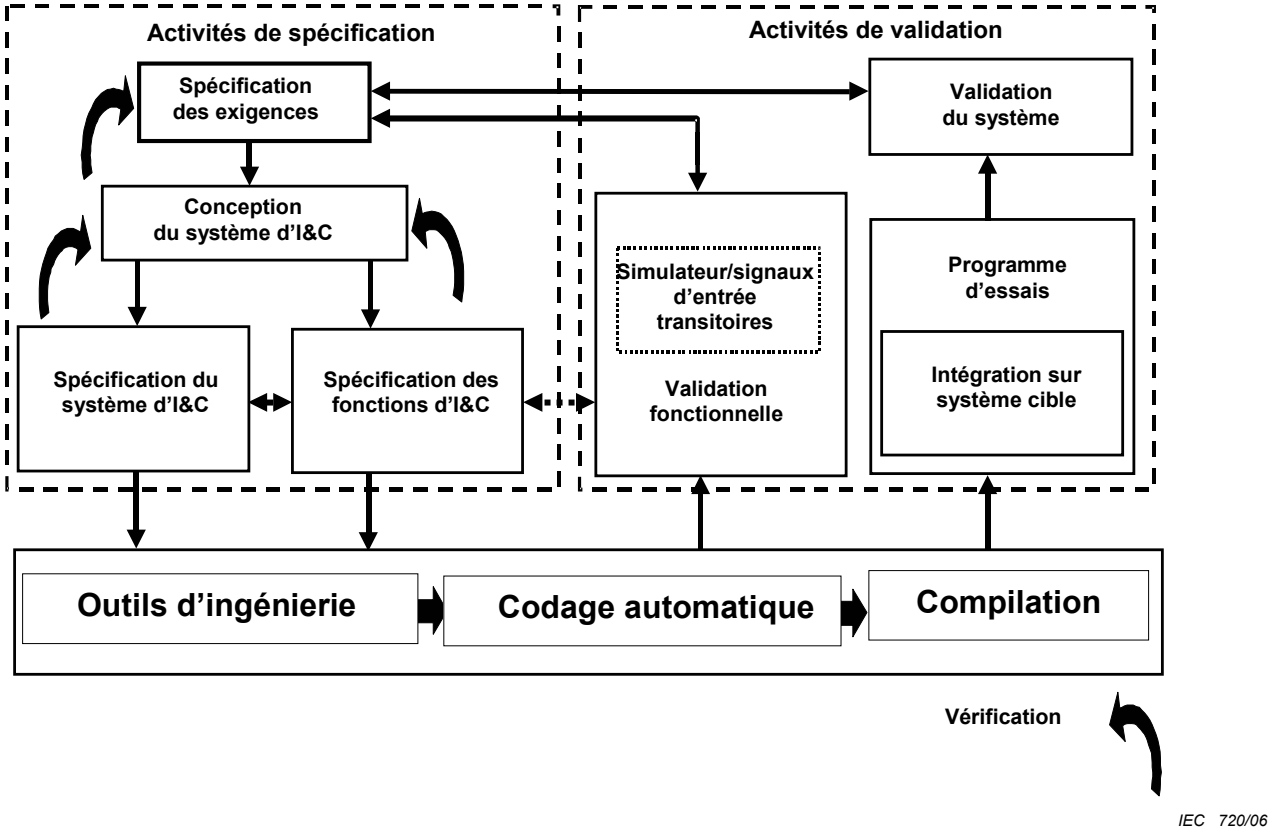


Figure C.1 – Cycle de vie pour l'ingénierie logiciel orienté application

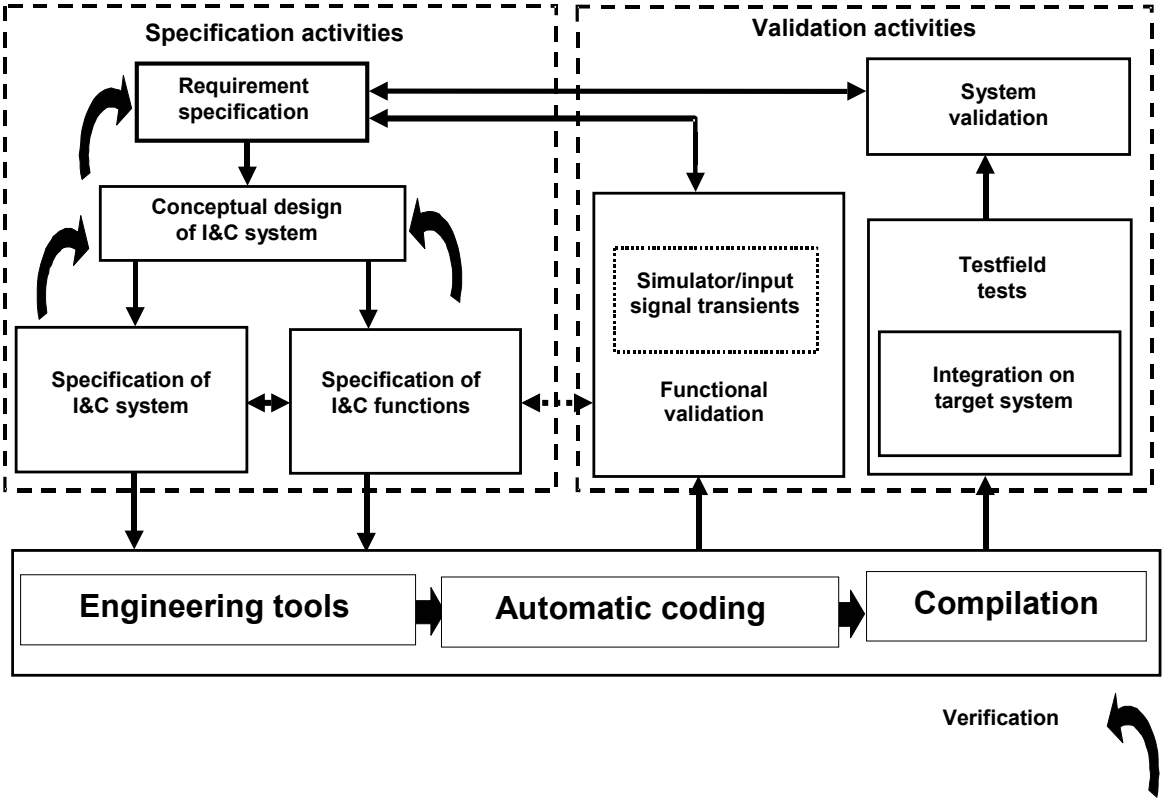


Figure C.1 – Life cycle for application oriented software engineering

Annexe D
(informative)

Langage, traducteur, éditeur de liens

Les recommandations détaillées suivantes qui concernent l'utilisation sûre de langages, de traducteurs et d'éditeurs de liens, sont données en complément de celles qui font partie de cette norme. De la même façon, ces recommandations s'appliquent aux autres programmes additionnels. Les recommandations relatives aux traducteurs s'appliquent de la même manière aux interpréteurs, aux compilateurs croisés et aux émulateurs. De façon similaire, il convient de prendre en compte les aspects qui s'appliquent aux traducteurs et aux éditeurs de liens lors de la sélection et de la spécification formalisée des outils de conception ainsi que durant leur emploi. Au niveau du projet, il convient de choisir les critères recommandés en fonction des priorités données.

D.1 Généralités

Art.	Recommandations	Priorité
a	Il convient de tester soigneusement les traducteurs, les éditeurs de liens et les chargeurs avant de les employer; leur fonctionnement est considéré comme très important.	1
b	Il convient que des données de fiabilité de qualité suffisante concernant les traducteurs, les éditeurs de liens et les chargeurs soient disponibles.	2
c	Lorsque des programmes additionnels, tels que des aides, des systèmes documentaires ou équivalents sont employés, il convient qu'ils soient testés de façon appropriée avant d'être employés.	1
d	Il convient que la syntaxe des langages soit définie de façon complète et non ambiguë.	2
e	Il convient que la sémantique soit spécifiée de façon correcte et complète et qu'elle soit compréhensible.	1
f	Il convient d'utiliser plutôt des langages de haut niveau que des langages orientés machine.	2
g	Les questions de diffusion d'un langage et de son adéquation au problème traité sont toutes deux considérées comme importantes.	2
h	Il convient qu'en général, les recommandations de l'Annexe B soient suivies autant que possible.	1
i	La facilité à relire le code produit est plus importante que la facilité à l'écrire lors de la programmation du système.	2
j	Il convient que les notations syntaxiques soient uniformisées; il convient qu'une seule notation soit autorisée par concept.	2
k	Il convient que l'emploi des caractéristiques de langage susceptibles d'introduire des erreurs soit évité.	2
l	Il convient que les programmes produits puissent être aisément modifiés.	2
m	Il convient que les paramètres d'entrée, les paramètres de sortie et les paramètres intermédiaires soient distincts en matière de syntaxe.	2
n	Il convient d'avoir une sortie supplémentaire pouvant être examinée à tous les niveaux du processus de traduction.	3

Annex D
(informative)

Language, translator, linkage editor

For a safe application of a language, its translator and linkage editor, the following more detailed recommendations are given in addition to those from the main part of this standard. These recommendations also apply to any other auxiliary system program. Recommendations for translators apply likewise to interpreters, cross compilers and emulators. Similarly, the aspects that apply to translators and linkage editors should be recognised during the selection and formal specification of design tools and their use. A project should select recommended criteria according to the priority indicated.

D.1 General

Item	Recommendations	Priority
a	Translator, linkage editor and loaders should be thoroughly tested prior to use; operation is considered very important	1
b	Reliability data of sufficient quality about translator, linkage editor and loader should be available	2
c	In the case where auxiliary system programs are used such as aids, documentation systems and the like, they should be appropriately tested before employed	1
d	Language syntax should be completely and unambiguously defined	2
e	Semantics should be well and completely specified and understandable	1
f	High level languages should be used rather than machine-oriented ones	2
g	Both the spread of a language and its problems adequacy are considered important	2
h	The recommendations of Annex B should be supported in general and as far as possible	1
i	Readability of produced code is more important than writeability during programming	2
j	Syntactic notations should be uniform; for the same concept not more than one notation should be allowed	2
k	The language should avoid error prone features	2
l	Produced programs should be easy to modify	2
m	Input parameters, output parameters and transient parameters should be syntactically distinct	2
n	An additional output that is reviewable should be provided at all stages of the translation process	3

D.2 Traitement des erreurs

Art.	Recommandations	Priorité
a	Il convient que les traducteurs de langage et les éditeurs de liens puissent assurer la détection d'autant d'erreurs de programmation que possible durant les phases de traduction ou d'exécution.	2
b	Il convient que le traitement d'erreur soit possible durant l'exécution.	2
c	Le langage peut permettre de formuler des assertions.	3
d	Il convient que les erreurs provoquant une exception durant l'exécution incluent:	
	– les dépassements des limites des tableaux;	1
	– les dépassements des gammes de valeur;	1
	– les accès à des variables non initialisées;	3
	– les non-respects des assertions;	2
	– les troncatures de chiffres significatifs de valeurs numériques;	2
	– les passages de paramètres d'un mauvais type.	1
e	Lorsqu'une erreur est découverte au niveau de la traduction ou de l'édition de liens, il convient qu'elle soit signalée sans tentative de correction.	2
f	Il convient d'émettre un message d'avertissement lorsqu'il existe un doute à propos de la transgression d'une règle.	3
g	Durant la phase de traduction il convient de vérifier les types de paramètres.	3

D.3 Traitement des variables et des données

Art.	Recommandations	Priorité
a	Il convient de pouvoir déterminer la gamme de chaque variable lors de la traduction.	1
b	Il convient de pouvoir déterminer la précision de chaque variable à virgule flottante et de chaque expression lors de la traduction.	2
c	Il convient de n'avoir aucune conversion implicite entre types.	2
d	Il convient de pouvoir déterminer le type de chaque variable, chaque tableau, chaque composant d'un enregistrement, chaque expression, chaque fonction, chaque paramètre au moment de la traduction.	2
e	Il convient de déclarer explicitement les variables, les tableaux, les paramètres, etc., y compris leurs types.	1
f	Il convient que les types de variable permettent de distinguer les paramètres d'entrée, de sortie, intermédiaires et ceux des sous-programmes.	2
g	Il convient que les noms de variable d'une longueur arbitraire soient autorisés.	2
h	Il convient, autant que possible, que la vérification de type ait lieu durant la traduction plutôt que durant l'exécution.	3
i	Il convient de vérifier au moment de la traduction si une affectation est permise pour un élément particulier de donnée.	2

D.4 Aspects temps réel

Art.	Recommandations	Priorité
a	Durant l'évaluation d'une expression, il convient que l'assignation externe de variable accessible à l'intérieur de l'expression ne soit pas autorisée.	1
b	Il convient de pouvoir examiner en direct les temps de traitement.	3
c	Il convient de pouvoir détecter les erreurs en temps réel (D.2.d).	1

D.2 Error handling

Item	Recommendations	Priority
a	Language translator and linkage editor should provide for detection of as many programming errors as possible during translation time or on-line execution	2
b	During on-line execution, exception handling should be possible	2
c	The language may provide assertions	3
d	Errors likely to cause an exception during execution time should include:	
	- exceeding of array boundaries;	1
	- exceeding of value ranges;	1
	- access to variables that are not initialised;	3
	- failing to satisfy assertions;	2
	- truncation of significant digits of numerical values;	2
	- passing of parameters of wrong type.	1
e	If any error is detected during translation or linkage time, it should be reported without any attempt at correction	2
f	If it is not clear whether any rules have been violated, a warning should be issued	3
g	During translation time, parameter types should be checked	3

D.3 Data and variable handling

Item	Recommendations	Priority
a	The range of each variable should be determinable at translation time	1
b	The precision of each floating point variable and expression should be determinable at translation time	2
c	No implicit conversion between types should take place	2
d	The type of each variable, array, record component, expression, function and parameter should be determinable at translation time	2
e	Variables, arrays, parameters, etc. should be explicitly declared, including their types	1
f	Variable types should distinguish between input, output, transient and sub-routine parameters	2
g	Variable names of arbitrary length should be allowed	2
h	As far as possible, type checking should take place during translation time rather than execution time	3
i	At translation time, it should be checked whether an assignment is allowed to any particular data item	2

D.4 On-line aspects

Item	Recommendations	Priority
a	During expression evaluation, external assignment should not be allowed to any variable that is accessible in the scope of the expression	1
b	Used computing time should be examinable on-line	3
c	On-line error capture should be provided (D.2*d*)	1

Annexe E
(informative)

Vérification et test du logiciel

E.1 Activités de vérification et de test du logiciel

Le but de cet article est de fournir des directives pour les activités de vérification et de test du logiciel. Les différentes méthodes utilisées pour cela sont plus ou moins efficaces pour trouver les erreurs de programmation suivant les programmes considérés. Les principales méthodes suivantes, employées pour la vérification et le test du logiciel, sont complémentaires: analyse statique de code assistée par des outils (par exemple vérification de modèle), inspection de code (relecture) et exécution dynamique (par exemple simulation ou exécution réelle du logiciel). La plupart de ces méthodes sont des approches systématiques recommandées. Un ensemble de méthodes de test complémentaires peut être employé, y compris les tests statistiques qui peuvent être utilisés pour rechercher des cas non sollicités par l'approche systématique.

Le paragraphe E.4.1 présente une revue des méthodes qu'il est possible d'employer. Lorsque cela est nécessaire, il convient de choisir différentes approches et différents critères de données de test pour différentes parties de programme, de façon à s'assurer qu'en particulier une partie de programme ne comporte pas d'erreur ou que la confiance dont elle fait l'objet est suffisante. La sélection dépend de la structure interne du programme, du niveau de fiabilité exigé, des demandes imposées au cours de l'exploitation de l'installation et des outils de test disponibles.

Le test du logiciel considère les différents niveaux de conception du logiciel (par exemple niveaux module, sous-système et système).

E.2 Approches systématiques

Il convient de tester et de vérifier chaque module de façon systématique en fonction des objectifs et du critère de couverture associés. En plus des activités de test et de vérification manuelles, il convient d'utiliser autant que possible des outils de test et de vérification automatiques. Il convient de vérifier les résultats des tests en les comparant aux résultats attendus tirés des spécifications du module du programme. Il convient que les entrées et les sorties de module soient traitées de la même manière que dans l'ensemble du système de sûreté de classe 1.

Le paragraphe E.4.2 est une check-list pouvant être interprétée suivant les besoins de chaque cas particulier. Prenant en compte la grande diversité des cas pratiques, il n'est pas possible de recommander une quelconque combinaison de tests particuliers pour une classe d'applications. Il est cependant possible de lister les tests qu'il convient de réaliser en toutes circonstances. D'autre part, il est évident que pour une application particulière, ni toutes les combinaisons de tous les tests, ni tous les tests individuels, peuvent être réalisés.

Au niveau sous-système, le logiciel est partiellement intégré dans le système. Les tests du niveau sous-système assurent que l'intégration des modules logiciel est correcte. Il convient de distinguer le cas où une dépendance du flot de données existe entre les modules logiciel, du cas où celle-ci n'existe pas. Il convient de réaliser les tests pour assurer que tous les arcs indépendants du sous-système sont correctement suivis. Il convient de réaliser des tests aux frontières des domaines d'entrée et aux limites opérationnelles des modules.

Annex E
(informative)

Software verification and testing

E.1 Software verification and testing activities

This clause is intended to provide guidance for software verification and testing activities. Depending on the program, the different methods for this purpose are more or less effective in finding program fault. The following main methods for software verification and testing are complementary: tool-based static analysis of the code (e.g. model-checking), code inspection (walkthrough) and dynamic execution (e. g. simulation or actual software execution). Most of these methods are recommended systematic approaches. A set of complementary testing approaches may be used, including statistical testing that could be applied to search for cases not exercised using the systematic approach.

A review of possible methods is given in E.4.1. When necessary, different approaches and different criteria for test data should be chosen for different program parts in order to determine whether a particular part is free from fault or determine a level of confidence. The selection depends on the internal structure of a program part, the level of reliability required, the demands imposed on it during plant operation and the available testing tools.

Software testing gives consideration to different levels of software design (e.g. module, subsystem and system levels).

E.2 Systematic approaches

Each module should be verified and tested in a systematic way according to objectives and their associated coverage criteria. In addition to manual verification and testing activities, automated verifier and testing tools should be used as much as possible. Test results should be checked with expected results derived from the program module specifications. Input to and output from the module should be handled in the same manner as in the safety system.

Subclause E.4.2 is a checklist, which can be interpreted according to the needs of each special case. Due to the enormous variety of possible practical cases, it is not possible to recommend any combination of the tests indicated for a particular class of applications. It is, however, indicated which tests should be performed under all circumstances. On the other hand, it is evident that neither all combinations of all tests nor all individual tests can be executed in a particular application.

At the subsystem level, the software is partially integrated into the system. The subsystem level tests assure the proper integration of the software modules. A distinction should be made between the case in which data flow dependency exists among the software modules and the case in which it does not. Testing should be done to provide assurance that all independent branches in the subsystem are correctly followed. Test cases at the borders of input domains and at the limits of module operational region should be used.

Il convient de réutiliser les ensembles de données employés au niveau module. Il convient de vérifier les résultats à l'aide de valeurs calculées à l'avance. Il convient que les paramètres du sous-système soient définis en entrée et en sortie, de la même manière que dans le système de sûreté. Il convient, lorsque cela est réalisable, que les tests de niveau sous-système soient réalisés sur la plate-forme matériel cible.

Au niveau système, le logiciel est totalement intégré au matériel. Les tests système garantissent que l'intégration des sous-systèmes est correcte. Il convient que les tests soient réalisés en exécutant les programmes avec un simulateur du contexte ou avec une version réaliste du système à superviser ou à commander par les systèmes de classe 1.

Il convient de réaliser le test complet du système conformément aux exigences de performances données par la spécification d'exigences du logiciel. Au-delà des activités de test décrites ci-dessus, il convient de réaliser une vérification systématique du comportement temporel.

E.3 Approches statistiques

Les approches statistiques peuvent être utilisées en complément des approches systématiques.

Les tests statistiques présentent les caractéristiques suivantes:

– les tests sont des échantillons indépendants, suivant une distribution de probabilité et représentatifs du fonctionnement opérationnel du système;
– la séquence et le nombre de tests réalisés n'influencent pas le résultat obtenu lors de la réalisation isolée d'un test;
– chaque défaillance survenant est détectée;
– le nombre de tests réalisés est important (voir les formules ci-dessous);
– les défaillances sont rares.

Généralement, les tests statistiques sont réalisés pour confirmer la confiance placée dans le bon fonctionnement du système de classe 1 qui a déjà été obtenue sur la base du déroulement du programme de test étendu constitué d'essais fonctionnels classiques.

Les formules développées pour la vérification probabiliste du logiciel peuvent être employées pour l'estimation de la probabilité de défaillance du système à la demande dans certaines hypothèses vérifiées. Les formules fournissent une estimation de la limite supérieure de la probabilité de défaillance à la demande de la façon suivante.

Supposons que n tests statistiques soient réalisés sur le système et qu'aucune défaillance ne soit observée. Alors pour que la probabilité de détection de défaillance (pfd) soit inférieure ou égale à la valeur de pfd visée avec une probabilité α, il faut que la condition suivante soit satisfaite:

$$pfd \leq -\frac{\ln(1-\alpha)}{n}.$$

Ainsi pour α = 0,95, nous obtenons approximativement après n tests statistiques sans défaillance:

$pfd \leq \frac{2{,}99}{n}$ avec une probabilité de 0,95; par exemple pour obtenir une pfd de 10^{-4} avec une probabilité of 95 %, il convient qu'un ensemble de 29 900 tests soit réalisé sans échec.

The same test data set utilised at module level should be used. Results should be checked with pre-calculated values. Parameters should be input and output from the subsystem in the same manner as in the safety system. Where practical, the subsystem level testing should use a hardware configuration identical to the target hardware.

At the system level, the software is completely integrated in the hardware. The system test assures proper integration of subsystems. Testing should be performed by running programs together with a realistic simulator, or with a realistic version of the system to be monitored or controlled by the class 1 systems.

This test of the complete system should be performed in accordance with the performance statements given in the software requirements specification. Beyond the testing activities described above, a systematic timing check should be performed.

E.3 Statistical approaches

Statistical approaches may be utilised to complement systematic approaches.

Statistical testing has the following characteristics:

– tests are independent samples from a probability distribution representing operational use of the system;
– sequence and number of tests do not influence the outcome of a single test-run;
– each occurring failure is detected;
– the number of tests is large (cf. formulae below);
– failures are rare.

Typically, statistical testing may be performed to supplement the confidence in correct operation of class 1 systems that has been gained through an extensive programme of conventional functional-based testing.

The formulae developed for a probabilistic software verification can be used for the estimation of a system's probability of failure on demand under verified assumptions. The formulae provides an estimate of the upper bound of the probability of failure on demand as follows.

Assuming that n statistical tests are run on a system and 0 failures are observed. Then, for the probability of failure detection (pfd) to be smaller or equal to some target value pfd with probability α, the following condition needs to be fulfilled:

$$pfd \leq -\frac{\ln(1-\alpha)}{n}.$$

Thus for α = 0,95, we obtain after n failure-free statistical tests approximately:

$pfd \leq \frac{2{,}99}{n}$ with probability 0,95; for example to obtain a pfd of 10^{-4} to a probability of 95 %, a total of 29 900 tests should be performed without failure.

Pour α = 0,99, nous obtenons approximativement:

$pfd \leq \frac{4{,}6}{n}$ avec une probabilité de 0,99; par exemple pour obtenir une pfd de10^{-4} avec une probabilité de 99 %, il convient qu'un ensemble de 46 000 tests soit réalisé sans échec.

La validité de la pfd calculée dépend de la représentativité du profil d'entrée des tests par rapport au profil d'entrée réel observable sur le système en exploitation. Si l'équation ci-dessus est utilisée avec un profil d'entrée opérationnel non réaliste, cela fournira une estimation de la pfd pour un profil d'entrée irréaliste, à savoir la pfd estimée pourra être très différente de la disponibilité réelle du système qui sera observée en exploitation. C'est la principale faiblesse de l'approche par les tests statistiques, du fait qu'il est généralement très difficile de déterminer précisément le profil opérationnel qui sera observable sur le système en exploitation, ceci étant particulièrement vrai pour les systèmes possédant un grand nombre d'entrées.

For α = 0,99, we obtain approximately:

$pfd \leq \frac{4{,}6}{n}$ with probability 0,99; for example to obtain a pfd of 10^{-4} to a probability of 99 %, a total of 46 000 tests should be performed without failure.

The validity of the calculated pfd depends upon the similarity of the profile of the test inputs to the profile of the actual inputs experienced by the system in operation. If the above equation is used on an unrealistic operational profile, it will provide an estimate for the pfd under an imaginary profile of use, i.e. a pfd will be estimated that may be very different to the actual system availability that would be obtained in active use. This is a fundamental weakness of the statistical testing approach as it is generally very difficult to accurately determine the operational profile that a system will experience in use, and this is particularly true for systems with large numbers of inputs.

E.4 Méthodes de vérification et de test

E.4.1 Sélection de méthodes de vérification

Synoptique								
No	**Méthode**	**But**	**Avantages principaux**	**Problèmes d'application, inconvénients principaux**	**Coût et effort comparés aux coûts de développement**	**Résultat**	**Relations avec les autres méthodes**	**Evaluation**
1	Surveillance de la procédure de test	Obtenir un profil de fiabilité sans effort supplémentaire.	- Effort supplémentaire réduit. - Pas de connaissance détaillée de l'objet soumis au test nécessaire.	- Profil de fiabilité obtenu insuffisant. - Les situations pratiques ne suivent que grossièrement la théorie.	Faible	Informations probabilistes, par exemple MTBF.	Utilise la théorie des probabilités comme N° 2	Difficile à utiliser pour les applications de sûreté.
2	Tests statistiques							
2.1	Par rapport à la demande	Obtenir un profil de fiabilité sans examiner le code en détail.	Pas de connaissance détaillée de l'objet soumis au test exigée.	- Fournir un profil de test qui représente les états d'entrée avec la même probabilité qu'en exploitation.	- Nombre de cas test très élevé, si des résultats significatifs sont demandés.	Informations probabilistes, par exemple disponibilité en fonction de la demande ou risque en fonction de la durée de vie du programme.	Même type de raisonnement que pour le N° 1	Peut être raisonnablement appliqué en complément à l'analyse du programme.
2.2	Par rapport à la justesse			- Fournir un profil de test qui prend en compte tous les aspects du programme avec la même probabilité.	- Le coût des tests peut être très supérieur.	Informations probabilistes, par exemple pour fixer une limite à la probabilité de défaillance à cause d'erreurs résiduelles.		

E.4 Verification and testing methods

E.4.1 Selected verification methods

Overview

No.	Method	Aim	Major advantages	Problems during application, major disadvantages	Cost and effort compared with development cost	Result	Relationship to other methods	Assessment
1	Supervision of testing procedure	To derive a reliability figure without additional effort	- Small additional effort for application - No detailed knowledge from test object required	- Reliability figures to be gained not sufficient - Practical cases behave only roughly according to theory	Small	Probabilistic figure, e.g. MTBF	Uses probability theory as No.2	Difficult to use for safety applications
2	Statistical testing							
2.1	According to demand	To derive a reliability figure without looking into the code's details	No detailed know-ledge from test object required	- To provide a test profile that represents input states with the same probability as the on-line operation	- Number of required test cases very high, if meaningful results are to be obtained	Probabilistic figure, e.g. availability per demand or risk per program life time	Same reasoning as for No.1	Can be reasonably applied in a short form complementary to program analysis
2.2	According to correctness			- To provide a test profile that hits any program property with equal probability	- Cost for test can be much higher	Probabilistic figure, e.g. to place a limit on the probability of failure due to possibly remaining software errors		

Synoptique

No	Méthode	But	Avantages principaux	Problèmes d'application, inconvénients principaux	Coût et effort comparés aux coûts de développement	Résultat	Relations avec les autres méthodes	Evaluation
3	Preuve de programme	Obtenir une base de comparaison par rapport aux spécifications.	Données rigoureuses sur la qualité de réalisation.	Pas de formalisme disponible permettant d'établir les assertions ou les invariants de boucle; risque d'erreur.	Du même ordre de grandeur ou supérieur.	Programme correct ou non correct dans la mesure où la preuve est correcte.	Certains points peuvent être analysés.	Sûrement la seule façon raisonnable de traiter les boucles générales «TANT QUE».
4	Analyse du programme		Donne de bons résultats sur des programmes simples.	Certaines structures de programme sont difficiles à analyser.		Exempts de certains types d'erreur spécifiques dans la mesure où l'analyse a été suffisamment poussée.	Une démarche rappelant celle des preuves de programme est utilisée.	
4.1	Analyse manuelle	Obtenir une base pour un test significatif ou pour une comparaison avec les spécifications.	Peut être réalisée sans outil supplémentaire.	Opération manuelle soumise à risque d'erreur.	Légèrement inférieur au coût de développement.			Recommandée si aucun outil automatique n'est disponible.
4.2	Analyse automatique	Comme ci-dessus parfois restreinte à ne fournir que quelques vagues indications sur la qualité.	- Travail manuel limité; - Résultats rapidement obtenus.	- De nombreux outils, demande du travail manuel supplémentaire; - L'interprétation des résultats fournis par certains outils est difficile.	Dépend des outils qui sont disponibles, beaucoup plus faible que le coût de développement.			L'utilisation la plus large possible est recommandée. *L'approche la plus prometteuse*.

Overview

No.	Method	Aim	Major advantages	Problems during application, major disadvantages	Cost and effort compared with development cost	Result	Relationship to other methods	Assessment
3	Program proving	Provide a basis for comparison with specifications	Rigorous statement on correctness achievable	No formalism available for finding loop assertions or loop invariants; error prone	In the same order of magnitude or higher	Correct/not correct, as far as proof if correct	Some aspects to be used with analysis	Possibly the only reasonable way for general "WHILE" loops
4	Program analysis		Brings good results with simple programs	Some program constructs are difficult to analyse		Free from some specific error types as far as analysis was deep enough	Same kind of thinking from program proving is used	
4.1	Manual analysis	To provide a basis for a meaningful test or for a comparison with specifications	Can be performed without additional tools	Error prone	Slightly less than development cost			Should be used if no automated tool available
4.2	Automated analysis	As above sometimes restricted to some vaguer quality measurement	- Only limited hand work - Results quickly obtained	- Many tools requires additional hand work - Some results of such tools are difficult to interpret	Depending on the available tool, much lower than development cost			Should be used as widely as possible *Most promising approach*

E.4.2 Méthodes de tests

E.4.2.1 Généralités

No	Type de test	Types d'erreurs détectées	Test à réaliser	Remarques
1	Tests représentatifs du comportement du programme en général, de ses calculs, de sa durée.	Toutes mais sans garantie d'exhaustivité.	Toujours.	On suppose que le système fonctionne correctement si les tests sont exécutés correctement.
2	Toutes les exigences spécifiées spécifiquement et individuellement.	Toutes les fonctions oubliées sont détectées.	Toujours dans un premier temps, si les fonctions sont spécifiées en détail.	- Peut être exhaustif, si la séparation des fonctions est complète; - Ne donne pas beaucoup d'information sur les problèmes de durée d'exécution.
3	Toutes les variables d'entrée à leur valeur extrême (test aux limites).	Problèmes de durée d'exécution, mais sans garantie. Overflow, underflow.	Toujours.	
4	Mise en œuvre de tous les périphériques extérieurs.	Erreurs de conception des interfaces matérielle et logicielle.	Toujours.	
5	Tests statiques et dynamiques des chemins représentatifs du comportement du procédé technique.	Toutes, mais sans garantie.	Toujours.	Particulièrement, efficace si un simulateur de procédé est disponible.
6	Démonstration du bon fonctionnement à l'arrêt et au démarrage de chaque sous-système redondant/dispositifs externes (il convient de tester aussi certaines combinaisons lorsque c'est pertinent).	Gestion des erreurs de l'interface avec le matériel	Toujours	Garantie de la robustesse du système.

E.4.2 Testing methods

E.4.2.1 General

No	Kind of test	Kind of detected errors	To be performed	Remarks
1	Cases representative for program behaviour in general, its arithmetics, timing	All, but with no guarantee for being exhaustive	Always	It is assumed that the system works correctly, if the cases are executed correctly
2	All individually and explicitly specified requirements	Forgotten functions detected completely	Always primarily, if required functions are specified in detail	- Can be an exhaustive test, if functions are kept completely separate - Does not say much on timing problems
3	All input variables in extreme positions (crash test)	Timing errors, but with no guarantee. Overflow, underflow	Always	
4	Operation of all external devices	Hardware and software interface design errors	Always	
5	Static cases and dynamic paths which are representative for the behaviour of the technical process	All, but with no guarantee	Always	Especially valuable if simulator of technical process is available
6	Correct operation shown by turning off and on each redundant subsystem/external device (some combinations should be also tested where relevant)	Hardware interface handling errors	Always	Ensures system robustness

E.4.2.2 Tests des chemins

No	Type de test	Type d'erreurs détectées	Test à réaliser	Remarques
7	Exécution de chaque instruction au moins une fois.	Code inaccessible.	Toujours.	
8	Exécution au moins une fois de chaque sortie de chaque instruction de branchement.	Erreurs dans le flot de contrôle, sans garantie d'exhaustivité.	Toujours.	Contient 5; peut être exhaustif en l'absence de boucle et de problème de durée d'exécution. Problème purement combinatoire calqué sur la structure du programme.
9	Chaque condition exécutée pour chaque branche.	Erreurs dans le flot de contrôle et le flot de données.	Si la combinaison des tests 8 et 12 n'est pas faite.	Contient 8; est inclus dans la combinaison de 9 et de 14.
10	Chaque boucle est exécutée une fois avec le minimum, le maximum et au moins une valeur intermédiaire du nombre d'itérations.	Erreurs dans le contrôle des boucles et dans le traitement des données des tableaux.	Toujours si le programme contient des boucles.	Non applicable aux constructions de type « boucle sans fin »
11	Chaque chemin est exécuté au moins une fois.	Toutes les erreurs du flot de contrôle.	Pour s'assurer que les choix de conception et de codage ne rendent pas la vérification excessivement complexe.	Réalisable uniquement pour des modules. Contient 8, noter que chaque nouvelle exécution d'une boucle mène au minimum à un nouveau chemin.

E.4.2.3 Test du transfert de données

No	Type de test	Type d'erreurs détectées	Test à réaliser	Remarques
12	Toute affectation à un emplacement mémoire est exécuté au moins une fois.	Erreurs dans le flot de données mais sans garantie.	Lorsque des tableaux sont utilisés.	
13	Chaque référence à chaque emplacement mémoire est exécuté au moins une fois.	Erreurs dans le flot de données, exhaustivité dans certains cas.	Lorsque des tableaux sont utilisés.	Contient 12 dans la plupart des cas. N'est raisonnable qu'en relation avec 12.
14	Toutes les correspondances des entrées vers les sorties exécutées au moins une fois.	Toutes les erreurs du flot de données.	Toujours pour des segments individuels.	Réalisable seulement pour les modules. Peut être couvert par 7, 8 ou 11.

E.4.2.2 Path testing

No	Kind of test	Kind of detected errors	To be performed	Remarks
7	Every statement executed at least once	Inaccessible code	Always	
8	Every outcome of every branch executed at least once	Errors in control flow, with no guarantee of completeness	Always	Contains 5; can be exhaustive, if no loops no timing problem Purely combinatorial problem mapped on program structure
9	Every predicate term exercised to each branch	Errors in control flow and data flow	If combination of tests 8 and 12 is not applied	Contains 8; included in combination of 9 and 14
10	Each loop executed with minimum, maximum and at least one intermediate number of repetitions	Errors in loop control and array data handling	Always, if program contains loops	Not applicable to 'loop forever' constructs
11	Every path executed at least once	All the erroneous control flow	To validate that design and coding choices do not make the verification overly complex	Achievable only for modules. Contains 8, note that any new loop running implies at least a new path.

E.4.2.3 Data movement testing

No	Kind of test	Kind of detected errors	To be performed	Remarks
12	Every assignment to each memory place executed at least once	Errors in data flow, but with no guarantee	Arrays used	
13	Every reference to each memory place executed at least once	Errors in data flow, exhaustive in special cases	Arrays used	Contains 12 in most cases Only reasonable in connection with 12
14	All mappings from input to output executed at least once each	All data flow errors	Always for individual segments	Only feasible for modules Possibly covered by 7, 8 or 11

E.4.2.4 Test de la durée d'exécution

No	Type de test	Type d'erreurs détectées	Test à réaliser	Remarques
15	Vérification de toutes les contraintes liées au temps.	Erreurs de synchronisation, temps de calcul trop long.	Toujours	
16	Combinaison maximale de séquences d'interruptions.	Erreurs d'organisation.	Si le nombre n'est pas trop grand.	
17	Toutes les combinaisons significatives de séquences d'interruption.	Erreurs d'organisation.	Toujours	

E.4.2.5 Divers

No	Type de test	Type d'erreurs détectées	Test à réaliser	Remarques
18	Contrôle de la position de toutes les frontières dans l'espace des données d'entrée.	Erreurs dans le découpage de l'espace des données d'entrée.	Lorsque des entrées analogiques sont utilisées.	Déterminer le nombre et le type des zones de données par analyse.
19	Contrôle de la précision des calculs arithmétiques pour tous les points critiques.	Erreurs numériques, erreurs dans les algorithmes, erreurs d'arrondis.	Si la longueur du mot calculateur est courte; si des calculs arithmétiques compliqués sont faits.	
20	Uniquement pour les programmes; tests d'interfaces entre les modules et d'interaction entre modules.	Transferts de données incorrects entre modules.	Toujours.	Une aide au test est souhaitable
21	Chaque module appelé au moins une fois.	Flot de contrôle et flot de données entre module incorrects, sans garantie.	Toujours.	
22	Tout appel de module mis en œuvre au moins une fois.		Toujours.	
23	Fonctionnement avec une charge élevée.	Erreurs de synchronisation et de réponse du système.	Toujours.	Garantie de la robustesse du système.

E.4.2.4 Timing testing

No	Kind of test	Kind of detected errors	To be performed	Remarks
15	Checking of all time constraints	Timing errors, computing time too long	Always	
16	Maximum possible combinations of interrupt sequences	Organisational errors	If number not too large	
17	All significant combinations of interrupt sequences	Organisational errors	Always	

E.4.2.5 Miscellaneous

No	Kind of test	Kind of detected errors	To be performed	Remarks
18	Check for correct position of boundaries of data inputs	Erroneous subdivision of input data space	If analogue input is used	Number and kind of input data subareas to be found by analysis
19	Check for sufficient accuracy of arithmetical calculations at all critical points	Numerical errors, errors in algorithms, rounding errors	If word length of computer is short; complicated arithmetic used	
20	Only for programs; test of module interfaces and module interaction	Incorrect data transfer between modules	Always	Test aid welcome
21	Every module invoked at least once	Incorrect control flow and incorrect data flow between modules, with no guarantee	Always	
22	Every invocation to a module exercised at least once		Always	
23	Operation at high load	System timing and response errors	Always	Ensures system robustness

Annexe F
(informative)

Liste typique des documents relatifs au logiciel

Références aux paragraphes de cette norme	Référence principale	Autres références
Documents relatifs au développement logiciel		
Spécification des exigences système	15.2	15.3
Spécification système	5.3	6.1, 8.1, 9.3, 15.3
Spécification des exigences logiciel	6.1	3.33 (3.35 et 3.37), 5.3, 6.4, 7, 7.1.3, 8.1, 8.2, 8.2.3.2, 11.2, 11.3, 15.3.1.2
Plan d'assurance qualité logiciel	5.5	
Recommandations détaillées pour le développement du logiciel	7.3	Annexe D
Plan de vérification logiciel	8.2.1	
Aspects logiciel du plan d'intégration	9.1	9.3
Spécification de conception logiciel	7.4	7.1.3, 8.1, 8.2.2, 8.2.3
Compte rendu de vérification de conception logiciel	8.2.2	7
Spécification de test logiciel	8.2.3.1.2	8.2.3.1.3
Compte rendu de vérification du code	8.2.3.1.1	
Compte rendu de test logiciel	8.2.3.1.3	
Vérification de l'intégration du système, compte rendu de test concernant les aspects logiciel	9.5	9.1, 9.2, 9.3, 9.4
Aspects logiciel du plan de validation système	10.1	10.2, 10.4
Aspects logiciel du compte rendu de validation du système	10.3	
Manuel utilisateur logiciel	12.4.2	
Aspects logiciel du plan de test de certification		
Aspects logiciel du compte rendu de test de certification		
Documents relatifs aux modifications du logiciel		
Compte rendu d'anomalie	11.1	11.3
Demande de modification logiciel	11.1	11.2, 11.3
Compte rendu de modification logiciel	11.2	
Historique de la gestion des modifications logiciel	11.2	11.3

Annex F
(informative)

Typical list of software documentation

References to the subclauses of this standard	Main reference	Other references
Documents relating to software production		
System requirements specification	15.2	15.3
System specification	5.3	6.1, 8.1, 9.3, 15.3
Software requirements specification	6.1	3.33 (3.35 and 3.37), 5.3, 6.4, 7, 7.1.3, 8.1, 8.2, 8.2.3.2, 11.2, 11.3, 15.3.1.2
Software quality assurance plan	5.5	
Detailed recommendations (program development)	7.3	Annex D
Software verification plan	8.2.1	
Software aspects of system integration plan	9.1	9.3
Software design specification	7.4	7.1.3, 8.1, 8.2.2, 8.2.3
Software design verification report	8.2.2	7
Software test specification	8.2.3.1.2	8.2.3.1.3
Software code verification report	8.2.3.1.1	
Software test report	8.2.3.1.3	
Software aspects of integrated system verification report	9.5	9.1, 9.2, 9.3, 9.4
Software aspects of the system validation plan	10.1	10.2, 10.4
Software aspects of the system validation report	10.3	
Software user manual	12.4.2	
Software aspects of the commissioning test plan		
Software aspects of the commissioning test report		
Documents relating to software modifications		
Anomaly report	11.1	11.3
Software modification request	11.1	11.2, 11.3
Software modification report	11.2	
Software modification control history	11.2	11.3

Annexe G
(informative)

Considérations sur les CCF et la diversification

Les CCF peuvent par exemple se produire:

– si une faute latente est implémentée dans deux ou plusieurs composants ou systèmes, et que tous ces composants ou systèmes fonctionnent dans des conditions identiques ou similaires de sorte qu'une défaillance puisse être déclenchée de manière corrélée dans le temps ou,
– si des conditions de défaillance se propagent via les communications de données.

G.1 CCF du fait du logiciel

Pour qu'il y ait CCF logicielle, des trajectoires de signal doivent activer un défaut du logiciel ayant pour effet une défaillance affectant deux systèmes ou voies ou plus (par exemple deux voies de protection, deux régulateurs en boucle fermée ou deux sous-systèmes logiques de commande). La réduction de l'éventualité de CCF de plusieurs systèmes due au logiciel peut être réalisée en réduisant les possibilités que le logiciel de ces systèmes contienne des défauts communs et/ou en faisant fonctionner ces systèmes avec des trajectoires de signaux différentes. Les défauts logiciels peuvent provenir de la spécification du système d'I&C ou peuvent être introduits lors du développement du logiciel.

Pour que les CCF logicielles constituent une préoccupation du point de vue de la sûreté, elles doivent affecter une fonction de sûreté et se produire à un moment où elles pourraient causer un risque pour la sûreté, ou doivent elles-mêmes entraîner un risque pour la sûreté comme par exemple une perte de protection ou de commande.

Si des logiciels, des méthodes de mise en oeuvre ou des algorithmes identiques ou similaires sont utilisés dans des systèmes redondants ou différents, il existe un élément commun significatif.

Il n'existe actuellement aucune méthode convenue d'estimation de la probabilité de défaillance ou du taux de défaillance des logiciels.

G.2 Causes et effets des CCF potentielles

G.2.1 Potentiel de CCF

Il y a un potentiel de CCF logiciel entre différents systèmes ou entre des voies différentes d'un système lorsque des modules logiciels communs sont utilisés. Les sources potentielles de défauts latents sont les erreurs de conception commises lors de la spécification du système d'I&C, l'architecture, les algorithmes, les méthodes de développement, les outils, les méthodes de mise en oeuvre, la maintenance.

Les exigences mal comprises ou transformées de manière incorrecte peuvent entraîner des défauts des spécifications du logiciel débouchant sur des risques de CCF logicielles. Les déficiences du logiciel peuvent être dues à des exigences logicielles et à des spécifications logicielles incorrectes, incomplètes, imprécises ou mal comprises. Des erreurs de conception ou des défauts logiciels peuvent être présents dans les programmes diversifiés en raison de facteurs humains communs tels que la formation, l'organisation, les processus de pensée et les approches de conception.

Annex G
(informative)

Considerations of CCF and diversity

CCF can occur for example:

– if a latent fault is implemented in two or more components or systems and, all these components or systems are operated under the same or similar conditions so that a failure can be triggered in a timely correlated way or,

– if failure conditions propagate via data communication.

G.1 CCF due to software

For a software induced CCF, signal trajectories must trigger a software fault causing a failure that affects two or more systems or channels (for example two protection channels, two closed loop controllers, or two logic control subsystems). The reduction of the likelihood of CCF of different systems due to software can be achieved by reducing the potential that the software of these systems contains common faults and/or by operating these systems with different signal trajectories. Software faults may originate from the requirement specification for the I&C system or may be introduced during software development.

For the CCF to be of a safety concern, these failures must disable a safety function and happen within a time period in which a safety challenge could result, or it must itself cause a safety challenge such as loss of protection or control.

If the same or similar software, implementation methods or algorithms are used in redundant or in different systems, then a significant common element exists.

No agreed method of estimation currently exists for estimating the probability of failure or for failure rate arising from software faults.

G.2 Potential CCF causes and effects

G.2.1 CCF potential

A potential for a software induced CCF of different systems or between different channels in one system exists if common software or software modules are used. Potential sources of latent faults are design errors from the I&C system requirement specification, the architecture, algorithms, development methods, tools, implementation methods, or maintenance.

Requirements that are not properly understood or not correctly transformed can result in faults in the software specification resulting in risks of CCF due to exercising the resulting software fault. Deficiencies in software can be due to incorrect, incomplete, inaccurate or misunderstood software requirements and software specifications. Design errors leading to software faults can be introduced into diverse programs, due to common human factors such as training, organisation, thinking processes and design approaches.

Une autre cause potentielle de CCF peut être la connexion de systèmes avec d'autres systèmes contenant des logiciels de moindre qualité.

G.2.2 Déclenchement de CCF

Un transitoire spécifique des signaux d'entrée peut déclencher une CCF s'il affecte:

– deux voies redondantes ou plus d'un système utilisant du logiciel commun;
– deux systèmes dont les fonctions sont diversifiées et qui ont des modules logiciel commun.

Un défaut logiciel peut entraîner une défaillance logicielle lorsqu'un transitoire spécifique des signaux d'entrée apparaît. Si ce transitoire est identique pour deux ou plusieurs voies ou systèmes, cela peut entraîner une CCF qui compromettra une ou plusieurs couches de défense lorsqu'une qualité, une indépendance et une diversification suffisantes n'auront pas été assurées.

G.2.3 Anomalies et événements anormaux

Des défaillances matérielles, des anomalies de la centrale et des événements anormaux peuvent entraîner des états logiciels non prévus, des transitoires ou des états de surcharge imprévus, non couverts par les exigences initiales ou par la conception du logiciel.

Les événements potentiels qui entraînent des CCF sont:

– les activités de maintenance;
– les défaillances de signal de temporisation commun, entraînant la perte des actions temporisées;
– les transitoires d'alimentation entraînant l'arrêt ou le redémarrage automatique du logiciel;
– les arrêts d'urgence de la centrale causant la surcharge des voies de communication;
– la saturation des capacités de l'opérateur, le conduisant à réagir incorrectement;
– les demandes de l'opérateur saturant la capacité du système en cas d'arrêt d'urgence de la centrale;
– toutes les fonctions des régulateurs engagées et en fonctionnement, et
– les situations anormales pendant les arrêts et la mise en service.

G.3 Défense contre les CCF

Les moyens de défense possibles sont:

– les méthodes utilisées pendant toute la durée de vie dans le but de produire un logiciel exempt de défauts (voir 13.1);
– la démonstration et l'amélioration de la qualité du logiciel commun (voir 13.2);
– l'utilisation du logiciel commun dans des conditions d'exploitation très restreintes et garanties;
– les limitations des effets des défaillances logicielles (voir 13.3);
– la conception des canaux ou systèmes telle qu'une défaillance simultanée de deux canaux ou systèmes est très improbable car il est démontré que les trajectoires des signaux des systèmes sont différentes;
– la conception des canaux ou systèmes utilisant un fonctionnement asynchrone. Ceci peut être utilisé pour montrer une défense vers les même processeurs dans des canaux différents soumis à des trajectoires identiques en même temps; et
– la diversification pour certaines ou l'ensemble des fonctions, et l'amélioration des concepts d'indépendance pendant toute la durée de vie (voir 13.4).

Another potential cause of CCF could result from connection of systems to ones with lower quality software.

G.2.2 Triggering of CCF

A specific transient of input signals may trigger a CCF if it affects:

– two or more redundant channels of a system using common software;
– two systems whose functions are diverse but which use some common software modules.

A software fault may result in a software failure when a specific transient of input signals appears. If this transient is identical for two or more channels or systems, this may result in a CCF which will jeopardise one or more defence layers when sufficient quality, independence and diversity are not provided.

G.2.3 Abnormal conditions and events

Abnormal hardware failures, plant conditions and events can cause unforeseen signal trajectories, unexpected software states, transients or overload conditions that were not covered by the initial requirements or by the software design.

Potential events which may cause CCF include:

– maintenance activities;
– common timing signal failure, causing loss of timed actions;
– power supply transients causing software stop or auto-restart;
– plant trips causing communication channels to overload;
– saturation of operator capacity, causing an incorrect action;
– operator demands saturating system capacity during plant trips and transients;
– all automated controller functions engaged and operating; and
– abnormal conditions during outages and commissioning.

G.3 CCF defences

Possible defence features include:

– methods used throughout the life-cycle which aim to produce fault-free software (see 13.1);
– demonstration and enhancement of the quality of common software (see 13.2);
– use of common software in a very narrow and guaranteed set of operating conditions;
– limitations of the effects of software failures (see 13.3);
– design of the channels or systems so that a coincident failure of two channels or systems is very unlikely due to there being a demonstrable difference in the signal trajectories for the systems;
– design of the channels or systems using asynchronous operation; this may be used to show defence against the same processors in different channels being subjected to identical trajectories at the same time; and
– diverse features for some or all functions and enhancement of independence concepts during the whole life-cycle (see 13.4).

G.4 Preuve de conformité

Les méthodes permettant la preuve de la conformité sont les suivantes:

– la réutilisation de modules logiciels standards validés avec des interfaces claires et validées (voir Article 15); les fonctions typiques comprennent par exemple les modules pour le pilotage des périphériques, la surveillance des processus et l'acquisition des signaux d'entrée, les algorithmes de base de régulation (comme par exemple proportionnelle-intégrale-dérivée, bande morte, hystérésis);

– l'utilisation d'outils et de procédures indépendants des processus de décodage du code chargé en mémoire et la démonstration que le code chargé est conforme à la spécification;

– l'utilisation de l'analyse dynamique pour tester le comportement correct du logiciel dans un environnement simulé représentant les parties pertinentes de la centrale (voir 8.2.3.2.3);

– l'utilisation de l'analyse statique du code afin d'identifier les flux de commandes et de données, et de démontrer que les processus de prise de décision et les processus logiques sont corrects;

– l'utilisation de deux versions logicielles testées dos-à-dos, en soumettant le logiciel à des trajectoires de signal aléatoires. Cette méthode peut être utilisée en complément aux tests systématiques pour la détection des défauts de conception et codage;

– la réalisation d'un programme de tests poussé et progressif, où le fonctionnement correct de chaque composant du système est rigoureusement validé avant d'être intégré dans le système.

G.5 Caractéristiques de la diversité

a) Les caractéristiques importantes de la diversité logicielle sont:
   – la diversité fonctionnelle;
   – des spécifications de conception différentes pour le même besoin fonctionnel.

b) La diversité au niveau système peut comprendre:
   – l'utilisation de systèmes indépendants pour différents critères d'actionnement;
   – l'utilisation de technologies de base différentes, comme par exemple des calculateurs par rapport à des conceptions câblées;
   – l'utilisation de différents types de calculateurs, modules matériels et concepts principaux de conception;
   – l'utilisation de différentes classes de techniques informatiques telles que les automates, les microprocesseurs ou les mini-ordinateurs.

c) Les caractéristiques de l'approche de conception et les solutions aux problèmes qui améliorent la diversité considèrent des différences en matière de:
   – algorithmes de traitement;
   – données de configuration, d'étalonnage et fonctionnelles;
   – matériel d'acquisition de signaux;
   – interfaces matérielles et communications;
   – processus d'échantillonnage des entrées;
   – chronologie d'opérations;
   – processus de temporisation;
   – l'utilisation d'informations mémorisées, de verrouillages et de taux de variation.

G.4 Demonstration of correctness

Methods of supporting demonstration of correctness include:

– reuse of proven software standard modules with a clear and proven interface (see Clause 15); typical functions include for example modules for device driving, process monitoring and input gathering, basic control algorithms (such as proportional-integral-derivative, deadband, hysteresis);

– use of tools and procedures independent of the design processes for decoding of the code loaded in memory and demonstration that the loaded code matches the specification;

– use of dynamic analysis for testing the correct software behaviour in a simulation environment which represents relevant parts of the plant (see 8.2.3.2.3);

– the use of static analysis of code to identify control and data flow, and to demonstrate correct decision and logic processes;

– the use of two software versions tested back-to-back, exercising the software by random signal trajectories. This method can be used in addition to systematic testing for detection of design and coding faults;

– performing a comprehensive bottom-up testing programme, where the correct operation of each system component is thoroughly validated before being integrated into the system.

G.5 Diversity features

a) Software diversity features of importance include:
   – functional diversity;
   – different design specifications for the same functional requirements.

b) Diversity at the system level can include:
   – use of independent systems for different actuation criteria;
   – use of different basic technology, such as computers versus hardwired design;
   – use of different types of computers, hardware modules and major design concepts;
   – use of different classes of computer technique such as PLCs, microprocessors or mini-computers.

c) The design approach features and problem solutions which enhance diversity include differences of:
   – processing algorithms;
   – data for configuration, calibration and functionality;
   – signal input hardware;
   – hardware interfaces and communications;
   – input sampling processes;
   – time sequences of operations;
   – timing processes;
   – use of historical information, latches and rates of change.

d) Les différences de méthodes de conception et de mise en oeuvre comprennent:
   – les langages;
   – les systèmes de compilation;
   – les bibliothèques supports;
   – les outils logiciels;
   – les techniques de programmation;
   – les logiciels systèmes et applicatifs;
   – les structures logicielles;
   – l'utilisation différente des mêmes modules logiciels;
   – les données et les structures de données.
e) Diversité pendant les tests (tests dos à dos)
f) Les divers aspects de l'approche de la gestion incluent:
   – deux conceptions selon des méthodes de développement volontairement dissemblables (forcées);
   – la séparation des équipes de conception;
   – la restriction de la communication entres les équipes;
   – la communication formelle de la levée des ambiguïtés dans les exigences ou les spécifications;
   – l'utilisation de processus de définition logique différents;
   – les différences dans les méthodes de documentation;
   – l'utilisation de personnel différent.

G.6 Inconvénients, avantages et justification de la diversité

G.6.1 Inconvénients

Les inconvénients introduits par la diversité peuvent être les suivants:

– complexité globale plus grande;
– risque de mise en marche intempestive accru;
– spécifications et conception plus complexes;
– contrôle de deux fournisseurs;
– problèmes de modification, par exemple pour assurer que la diversité n'est pas perdue lors d'une modification;
– documentation plus importante;
– besoins en espace, en fournitures et de contrôle de l'environnement plus importants;
– le coût de plusieurs versions du logiciel peut réduire son potentiel commercial, sauf en tant que méthode de test;
– chaque version produite peut être de qualité inférieure.

G.6.2 Avantages

Lorsque la diversité fonctionnelle ou logicielle est utilisée, des versions à diversité adéquate assurent une plus grande protection contre les CCF provoquées par le logiciel.

G.6.3 Justification

La justification peut porter sur l'amélioration de la fiabilité des fonctions de sûreté obtenues par l'utilisation de la diversité.

d) Differences in design and implementation methods include:
   – languages;
   – compilation systems;
   – support libraries;
   – software tools;
   – programming techniques;
   – system and application software;
   – software structures;
   – different use of the same software modules;
   – data and data structures.
e) Diversity during tests (back-to-back testing)
f) Diverse aspects of management approach include:
   – two designs following deliberately dissimilar development methods (forced);
   – separation of the design teams;
   – restriction of communication between the teams;
   – formal communication of resolution of ambiguities in requirements or specifications;
   – use of different logic definition processes;
   – differences in documentation methods;
   – use of different staff.

G.6 Drawbacks, benefits and justification of diversity

G.6.1 Drawbacks

The disadvantages introduced by diversity may include:

– greater overall complexity;
– increased risk of spurious actuation;
– more complex specifications and design;
– control of two suppliers;
– modification problems, for example ensuring diversity is not lost during modification;
– increased documentation;
– increased space, supplies, environmental control requirements;
– the cost of several versions of the software can reduce its commercial potential, except as a testing method;
– each version produced may be of lower quality.

G.6.2 Benefits

When functional or software diversity are used, adequately diverse versions give increased protection against CCF due to software.

G.6.3 Justification

The justification can consider the improvement in reliability of the safety functions achieved by use of diversity.

Annexe H
(informative)

Outils pour la production et la vérification des spécifications, de la conception et du code

Les outils constituent une partie essentielle du développement des logiciels qui réalisent des fonctions de catégorie A. Des méthodes qui sont appliquées strictement manuellement sont fortement sujettes aux erreurs et exigent l'intervention de personnes très bien formées. De ce fait, il convient qu'elles soient aidées par des outils utilisant des techniques permettant de révéler la structure et les relations fonctionnelles internes du logiciel et de vérifier la cohérence interne, la cohérence avec un éventuel modèle antérieur, les propriétés désirables/ indésirables, etc.

La démonstration finale de conformité du code à la spécification peut être effectuée au moyen d'un analyseur de conformité. Lorsqu'un générateur de code éprouvé assure que le code exécutable est entièrement cohérent avec la description de conception, les analyses statique et dynamique du code permettent un contrôle diversifié de la conformité de cette description.

Les outils destinés aux méthodes formelles de spécification et de conception se décomposent en outils constructifs et en outils analytiques.

H.1 Outils constructifs

Les outils constructifs sont utilisés pour supporter les phases de spécification, conception et réalisation et peuvent comprendre ce qui suit:

H.1.1 Editeur de texte

Du fait que les méthodes formelles basées sur la théorie des ensembles, le calcul des prédicats et des propositions, exigent des symboles mathématiques spéciaux, il est important qu'un éditeur de texte approprié soit disponible, capable de les afficher sur un écran à haute définition et de les imprimer de manière lisible.

H.1.2 Interface graphique

Une capacité graphique appropriée est requise lorsque une méthode formelle implique l'utilisation de graphiques.

H.1.3 Générateur automatique de code

Une fois qu'une spécification formelle a été validée, l'intégrité du processus de conception peut être grandement améliorée par l'utilisation d'un générateur automatique de code validé. Un tel générateur de code permet de transformer la spécification en code exécutable et donc de réduire les possibilités d'introduction d'erreurs. En outre, un sous-ensemble fiable d'un langage peut être mis en oeuvre par conception du générateur de code.

Il convient que des modules logiciels certifiés soient utilisés pour les fonctions standards.

Il convient que le code généré de manière automatique soit lisible. Il convient que les commentaires permettent l'identification des parties associées de la spécification. Il convient que la structure du code généré de manière automatique permette la vérification automatique.

Annex H
(informative)

Tools for production and checking of specification, design and implementation

Tools now form an essential part of the development environment for software performing category A functions. Methods which are applied purely manually are highly error-prone and require the involvement of very well-trained humans. Therefore, they should be supported by tools which use techniques to reveal the structure and internal functional relationships of the software and to check for internal consistency, consistency with some prior model, desirable/undesirable properties, etc.

Final demonstration that the code meets its specification can be performed by means of a compliance analyser. Where a proven code generator ensures that the executable code is fully consistent with its design description, then static and dynamic analyses of the code provide a diverse check on the correctness of that description.

Tools for formal specification and design methods can be classified as constructive or analytical tools.

H.1 Constructive tools

Constructive tools are used to support the specification, design and implementation phases of development, and may include:

H.1.1 Text editor

Because formal methods based on set theory and on predicate and propositional calculus require special mathematical symbols, it is important that a suitable text editor is available capable of both displaying these on a high definition screen and printing them out legibly.

H.1.2 Graphical interface

A suitable graphics capability is required where a formal method involves the use of graphics.

H.1.3 Automated code generator

Once a formal specification has been proved, the integrity of the design process can be greatly enhanced by the use of a validated automated code generator. Such a code generator will transform the specification into executable code and thus reduce the likelihood of introducing errors. Additionally, a safe sub-set of a language may be enforced through the code generator design.

Certified software modules should be used for standard functions.

Automatically generated code should be readable. Comments should support the identification of the associated parts of the specification. The structure of automatically generated code should support automated verification.

H.1.4 Générateur d'obligations de preuve

Les méthodes formelles basées sur le raisonnement logique exigent un générateur d'obligations de preuve qui enregistre automatiquement les obligations de preuve inhérentes aux étapes de conception.

H.2 Outils analytiques

Les outils analytiques permettent le contrôle de la spécification, de la conception et du code. Ils peuvent comprendre ce qui suit:

H.2.1 Contrôleur syntaxique

Un contrôleur de syntaxe donne des informations sur la structure du programme, l'utilisation des données du programme, la dépendance des variables de sortie vis-à-vis des variables d'entrée, et le flux de contrôle dans le programme, ce qui permet:

a) d'identifier les défauts de structure comme par exemple les démarrages multiples, les fins multiples, le code inaccessible, le code redondant, la non-utilisation des résultats de fonctions;

b) d'identifier la hiérarchie des modules/sous-programmes;

c) d'identifier la violation des normes et des conventions de programmation, y compris les vérifications de branchements inconditionnels dans des boucles;

d) d'identifier les données qui sont lues avant d'être écrites, les données écrites avant d'être lues, les données écrites deux fois sans lecture intermédiaire;

e) de contrôler le flux des informations par rapport à la spécification;

f) d'aider à la conception d'un plan de test dynamique;

g) la gestion des données de test et la génération éventuelle des données de test.

H.2.2 Contrôleur sémantique

Un contrôleur sémantique décrit les relations mathématiques entre les variables de sortie et d'entrée pour chacun des chemins possibles du point de vue sémantique dans les régions sans branchements du programme. Cela permet de vérifier ce que fera le programme dans toutes les circonstances et de détecter des défauts tels que les valeurs de sorties inattendues affectées par les valeurs d'entrée, la réponse incorrecte à des valeurs inattendues d'entrée, le signe incorrect de fonctions et d'opérateurs, etc.

H.2.3 Générateur de preuves formelles

Les preuves formelles d'une conception exigent l'utilisation d'un programme interactif qui effectue les manipulations de symboles nécessaires sous le contrôle d'un opérateur humain afin de remplir les obligations de preuve. Un tel programme s'appelle un assistant démonstrateur de théorèmes (TPA). Les TPA sont de gros programmes dont l'absence d'erreur ne peut pas être démontrée, et qui exigent donc l'emploi d'un moyen de preuve diversifié qu'il convient d'utiliser. Cela implique en général l'application d'un contrôleur de preuve dont l'entrée est la sortie du TPA. Il convient que le générateur de preuves soit basé sur une théorie de preuve formelle et il convient qu'il soit vérifié par rapport à cette théorie de preuve.

H.2.4 Animateur

Il convient que chaque fois que possible les spécifications soient animées de manière que l'utilisateur final du système puisse examiner des aspects de la spécification ou de la conception afin de valider (dans toute la mesure du possible) la conception proposée. Il convient que l'animation soit aussi représentative que possible de la conception et il se peut qu'elle exige l'utilisation de prototypes pour démontrer les aspects non fonctionnels.

H.1.4 Proof obligation generator

Formal methods based on logical reasoning require a proof obligation generator that automatically records the proof obligations arising during the design steps.

H.2 Analytical tools

Analytical tools enable checking of the specification, design and implementation. They may include the following.

H.2.1 Syntax checker

A syntax checker provides information on the program structure, the use of program data, the dependency of output variables on input variables, and the control flow through the program allowing:

a) identifying structure defects like multiple starts, multiple ends, unreachable code, redundant code, non-usage of function results;

b) identifying module/subroutine hierarchy;

c) identifying violation of standards and programming conventions, including checks for unconditional branches into loops;

d) identifying data that is read before written, data written before read, data written twice without an intervening read;

e) checking information flow against specification;

f) assisting the design of a dynamic test plan;

g) test data management and possible test data generation.

H.2.2 Semantic checker

A semantic checker describes the mathematical relationships between output and input variables for every semantically feasible path through the branch free regions of the program. This allows checking what the program will do under all circumstances and detection of faults such as: unexpected output values affected by input values, incorrect response to unexpected input values, incorrect polarity of functions and operators, etc.

H.2.3 Formal proof generator

Formal proofs of a design require the use of an interactive program which carries out the necessary symbol manipulations under the guidance of a human operator in order to discharge the proof obligations. Such a program is known as a theorem proving assistant (TPA). This usually means the application of a proof checker whose input is the output of the TPA. TPA are large programs for which one cannot prove the absence of faults. Therefore, a diversified verification is needed and should be performed. This usually means the application of a proof checker whose input is the output of the TPA. The proof checker should be based on a formalised proof theory and it should be verified against this proof theory.

H.2.4 Animator

Where possible the specifications should be animated so that the end user of the system can examine aspects of the specification or design in order to validate (as far as is practicable) the requirements specification and proposed design. The animation should be as representative as possible of the design and may require the use of prototypes to demonstrate the non-

Cette évaluation est faite par rapport aux critères de l'utilisateur et les exigences du système peuvent être modifiées à la lumière de cette évaluation.

H.2.5 Analyseur de conformité

Un analyseur de conformité peut démontrer que le code met correctement en oeuvre la spécification. Pour cette démonstration, cet outil utilise des préconditions et des post-conditions plus des invariants de boucle. L'outil confirme systématiquement que chaque condition est remplie par le code.

functional aspects. This evaluation is done against the user's criteria, and the system requirements may be modified in light of this evaluation.

H.2.5 Compliance analyser

A compliance analyser can demonstrate that the code correctly implements the specification. Such a tool uses pre- and post-conditions plus loop invariables in such a demonstration. The tool confirms systematically that each condition is fulfilled in the code.

Annexe I
(informative)

Exigences concernant les logiciels prédéveloppés (PDS)

I.1 Directives pour le classement des non-conformités et des facteurs compensateurs

Les faiblesses et les non-conformités dans le respect des exigences applicables de la CEI 60880 peuvent appartenir à différents types, à savoir:

– exigences concernant la compréhensibilité et l'exhaustivité de la documentation du logiciel;
– exigences concernant la lisibilité des programmes;
– exigences relatives à la conception du logiciel insistant sur le comportement déterministe des performances en temps réel;
– exigences relatives au logiciel assurant l'auto-supervision du matériel programmé;
– exigences relatives à l'exhaustivité des comptes rendus de validation documentés.

Les faiblesses et les non-conformités sont rarement un problème de type «tout ou rien» et il convient qu'elles soient classées en fonction du degré de conformité et en fonction des conséquences sur la qualité finale du système programmé. Par exemple:

– les exigences de la CEI 60880 concernant la compréhensibilité sont critiques pour le développement et les modifications et sont moins importantes pour un produit stable et éprouvé.

Il convient que la possibilité de se prévaloir de l'expérience acquise en exploitation ou de tests supplémentaires comme facteurs compensateurs soit modulée en fonction du type de PDS et du rôle joué par le PDS dans le système programmé. Par exemple:

– le logiciel système opérationnel, normalement disponible sous forme de micro-code ou de code binaire, peut être difficile à analyser et l'accès à la documentation du processus de développement peut être limité. Dans certains cas, une expérience de fonctionnement considérable peut être disponible et peut constituer un facteur d'acceptation important en particulier pour les logiciels exécutant des fonctions répétitives définies, comme par exemple des pilotes de communication ou des parties de systèmes d'exploitation. Dans d'autres cas, comme par exemple les logiciels qui mettent en oeuvre des fonctions de surveillance et de recouvrement du système programmé, une expérience de fonctionnement importante peut ne pas être disponible du fait que ces fonctions sont rarement activées pendant l'exploitation des systèmes programmés;
– les bibliothèques de logiciels d'application peuvent être soigneusement documentées sur la base des informations accessibles sur la conception des modules, le processus de développement et les tests de validation. D'autre part, le retour d'expérience d'applications de centrales similaires ou des tests supplémentaires peuvent permettre la qualification de sorte qu'un degré de confiance raisonnablement élevé puisse être obtenu dans la conformité aux spécifications de ce type de logiciel.

Annex I
(informative)

Requirements concerning pre-developed software (PDS)

I.1 Guidance for graduating non-conformities and compensating factors

Weakness and non-conformities with respect to IEC 60880 requirements, may be of different types, for example:

– requirements concerning the understandability and the completeness of the software documentation;

– requirements concerning the readability of the programs;

– requirements related to software design that enhance the deterministic behaviour of real time performance;

– requirements related to software that provide self-supervision of the computer-based hardware; or

– requirements related to the completeness of documented validation reports.

Weakness and non-conformities are rarely a "black and white" issue and should be graduated in terms of degree of fulfilment and according to the impact on the final quality of the computer-based system. For instance:

– IEC 60880 requirements concerning understandability are critical for development and for modifications and less significant for a stable and proven product.

The possibility of taking credit of operating experience or additional testing as compensating factors ought to be judged according to the type of the PDS and its role in the computer-based system. For instance:

– operational system software, normally available as micro-code or binary-code, may be difficult to analyse and there may be limited access to the documentation of the development process. In some cases, a large amount of operating experience may be available and may be an important acceptance factor particularly for software performing defined repetitive functions, for example communication drivers or parts of operating systems. In other cases, for example software that implements surveillance and recovery functions of the computer-based system, significant operational experience may be not available because these functions are seldom activated during the operation of the computer-based systems;

– application software libraries may be thoroughly documented on the basis of accessible information on module design, development process and validation tests. Feedback of operating experience from similar plant applications or additional testing may also be used to support the qualification so that a high degree of confidence may be reasonably achieved in the correctness of the execution of this type of software.

I.2 Recueil et validation des données relatives à l'historique d'exploitation

I.2.1 Recueil des données

Il convient que les données à recueillir comprennent:

– informations sur le site: configuration et conditions opérationnelles du PDS dans le système programmé, fonctions utilisées, nombre de PDS utilisés;

– temps d'exploitation du site: temps écoulé depuis le démarrage initial, temps écoulé depuis la dernière évolution du PDS, temps écoulé depuis la dernière erreur grave (le cas échéant), temps écoulé depuis le dernier compte rendu d'erreur (le cas échéant);

– compte rendu d'erreur: date de l'erreur, sévérité, mesures correctives;

– historique des évolutions: date et identification des évolutions et de la configuration associée, erreurs corrigées, modifications ou extensions fonctionnelles, problèmes en cours.

I.2.2 Vérification des données

Il convient que la pertinence statistique des données recueillies soit vérifiée en insistant sur différents facteurs:

– les données relatives au temps d'exploitation ne sont acceptables que lorsqu'un temps minimal préétabli s'est écoulé depuis la mise en service du PDS;

– les temps d'exploitation du PDS ne peuvent être pris en compte que lorsqu'ils se réfèrent aux PDS effectivement utilisés pendant l'exploitation du système programmé sur lequel est installé le PDS;

NOTE Par exemple, certains modules des bibliothèques d'applications, des logiciels relatifs aux fonctions auxiliaires du système, peuvent être installés et jamais ou rarement utilisés sur un site particulier.

– il convient que l'exhaustivité des données décrivant l'historique d'exploitation (par exemple, tous les dysfonctionnements sont-ils réellement correctement documentés?) soit estimée en fonction de l'organisation du site ou du personnel de maintenance;

– seules des données issues de sources d'informations authentiques et non biaisées sont acceptables.

I.2 Collection and validation of data on the operational history

I.2.1 Data collection

Data to be collected should include:

– site information: including configuration and operational conditions of the PDS in the computer-based system, functions used, number of PDS running;

– site operating time: including elapsed time since first start-up, elapsed time since last release of the PDS, elapsed time since last severe error (if any), elapsed time since last error report (if any);

– error report: including date of error, severity, fixes; and

– release history: including date and identification of releases and associated configuration, errors fixed, functional modifications or extensions, pending problems.

I.2.2 Data verification

The statistical relevance of collected data should be verified keeping in mind different factors:

– data on operating time are acceptable only when a minimum pre-established time has elapsed since the commissioning of the PDS;

– the PDS operating time may be taken into account only when it refers to features of the PDS which are actually used during operation of the computer-based system where the PDS is installed;

NOTE For instance, some modules of the application libraries, e.g. software related to auxiliary system functions, can be installed but never or seldom used on a particular site.

– completeness of data describing operation history (e.g. are all malfunctions really documented correctly?) should be estimated depending on the site organisation of the modification staff;

– only data from genuine and unbiased information sources are acceptable.

Annexe J
(informative)

Correspondance entre la CEI 61513 et cette norme

J.1 Généralités

La CEI 60880 (1986) a été élaborée plusieurs années avant la CEI 61513 et comprend des articles traitant du niveau système en plus de ceux traitant du niveau logiciel.

Au cours du processus de révision, les articles se référant au niveau système et à présent traités de façon appropriée dans la CEI 61513 ont été retirés du texte révisé. Les articles explicitement référencés par la CEI 61513 ou qui ne sont pas traités de façon appropriée ont été conservés dans le texte révisé et se trouvent dans cette annexe, avec la référence des articles dans la CEI 60880 (1986) et dans la révision de la CEI 60880.

J.2 Articles de la CEI 60880 (1986) traités par la CEI 61513

La CEI 61513 est un document de niveau système couvrant toutes les catégories des fonctions I&C de sûreté.

Il y est écrit en introduction:

«Lorsqu'il s'agit d'un système informatique de classe 1, la présente norme s'utilise conjointement avec la CEI 60880, la CEI 60880-2 et la CEI 60987 afin d'assurer l'exhaustivité des exigences relatives au logiciel et au matériels du système.»

La CEI 61513 fait explicitement référence aux exigences de la CEI 60880 pour plusieurs sujets (intégration, validation et modification).

Les articles concernés sont maintenus dans cette révision de la CEI 60880 et révisés quand nécessaire.

Le Tableau J.1 donne la correspondance entre les articles de la CEI 60880 (1986) et de la CEI 60880-2 (désignée ici par «Suppl»), référencés par la CEI 61513 et les articles révisés dans le présent document.

Tableau J.1 – Correspondance entre des articles de la CEI 61513 et cette norme

Article de la CEI 61513 faisant référence à	Article de la CEI 60880 (1986)/ Suppl	Article de la présente norme
5.3.1.5.4 Défense contre les DCC provenant de défauts systématiques NOTE 2 Le présent paragraphe a pour objet de donner une vue d'ensemble. Le détail des exigences relatives aux défenses contre les DCC provenant d'erreurs de logiciels concernant des fonctions de catégorie A figure en 4.1 de la CEI 60880-2.	4.1 Suppl	13
5.3.3.1 Estimation de la fiabilité et des défenses contre les DCC NOTE 2 Les exigences relatives à l'analyse des DCC logicielles sont données en 4.1.1 de la CEI 60880-2.	4.1.3 Suppl	13.3

Annex J
(informative)

Correspondence between IEC 61513 and this standard

J.1 General

IEC 60880 (1986) has been developed several years before IEC 61513 and includes system level clauses in addition to software level clauses.

During the revision process, the clauses relevant to system level and now adequately addressed in IEC 61513 have been removed from the revised text. Clauses explicitly referenced by IEC 61513 or not adequately addressed have been kept in the revised text and are provided in this annex, with IEC 60880 (1986) and IEC 60880 revision clause numbers.

J.2 IEC 60880 (1986) clauses referenced by IEC 61513

IEC 61513 is a system level document covering all categories of safety I&C functions.

It states in the Introduction:

“When class 1 computer-based systems are addressed, this standard has to be used in conjunction with IEC 60880, IEC 60880-2 and IEC 60987 to assure the completeness of the system requirements for software and hardware aspects.”

IEC 61513 makes explicit references to IEC 60880 requirements for several subjects (integration, validation, and modification).

The related clauses have been kept in this revision of IEC 60880, and revised where appropriate.

This table gives the correspondence between the clauses referenced by IEC 61513 in IEC 60880 (1986), IEC 60880-2 here designated by ‘Suppl’ and the revised clauses in this document.

Table J.1 – Correspondence between clauses of IEC 61513 and this standard

IEC 61513 clauses referring to	IEC 60880 (1986)/Suppl. clause	Clause in this standard
5.3.1.5.4 Defence against CCF due to systematic faults NOTE 2 This subclause is intended to give an overview. Detailed requirements for defences against CCFs due to software faults for category A functions are given in 4.1 IEC 60880-2.	4.1 Suppl	13
5.3.3.1 Assessment of reliability and defences against CCF NOTE 2 Requirements for the analysis of CCF due to software are given in 4.1.1 of IEC 60880-2.	4.1.3 Suppl	13.3

Article de la CEI 61513 faisant référence à	Article de la CEI 60880 (1986)/ Suppl	Article de la présente norme
5.5.1 Documentation de la conception de l'architecture	4.2 Suppl	14
NOTE Les exigences relatives aux méthodes et outils logiciels pour les systèmes de classe 1 figurent en 4.2 et 4.3 de la CEI 60880-2.	4.3 Suppl	15
6 Cycle de vie et de sûreté du système	6.1	8.1
NOTE 1 Cette exigence diffère de celles de 6.1 de la CEI 60880. Pour les logiciels, il est souhaitable, sans que ce soit une nécessité, de terminer chaque phase d'un développement avant de commencer la suivante, à condition que les exigences susmentionnées aient été prises en compte.		
NOTE 2 Pour les systèmes de classe 1, les exigences relatives au logiciel et à cette activité sont définies dans la CEI 60880.	Aucun	
NOTE 3 Pour les systèmes de classe 1, les exigences relatives aux logiciels prédéveloppés sont définies dans la CEI 60880-2.	Aucun	
6.1.1.1.1 Fonctions d'application	A.2.2	Aucun
… Une estimation quantitative de la fiabilité des fonctions d'application peut être nécessaire pour vérifier la conception du système et de la centrale (voir A.2.2 de la CEI 60880). Cette estimation est normalement effectuée pour la conception matérielle du système (des méthodes bien rodées existent) mais aucune méthode unanimement reconnue n'existe pour l'évaluation quantitative de la fiabilité du logiciel (voir 6.1.3.1.2). …		
6.1.1.2.1 Architecture du système	4.1.1	13.1
NOTE Les défaillances dues au logiciel sont des défaillances systématiques et non aléatoires. Ainsi, le critère de défaillance unique ne peut pas être appliqué à la conception du logiciel comme il l'est pour la conception du matériel. Les effets possibles de DCC dues au logiciel à l'intérieur de chaque ligne de défense et entre des lignes de défense redondantes sont à prendre en compte au niveau de chaque système et de l'architecture de l'I&C (voir 4.1.1 de la CEI 60880-2).	Suppl	
6.1.1.2.2 Comportement interne du système	B2.d	B2.d
c) (Exigence spécifique) Afin de garantir avec un haut niveau de confiance le comportement déterministe , il convient que les systèmes de classe 1 soient développés avec des techniques telles que celles mentionnées à l'Annexe B de la CEI 60880 (notamment B2.*d* sur le temps d'exécution et B2.*e* sur les interruptions)...	B2.e	B2.e
NOTE 3 Voir l'Article 1 de la CEI 60880 concernant le rôle des annexes à la norme et les exigences applicables si les pratiques diffèrent de celles mentionnées dans les annexes.	1	5.5.3
6.1.1.2.3 Auto-surveillance et tolérance aux défaillances	4.8	6.2
a) Il convient que les systèmes soient conçus de telle manière que les erreurs et défaillances soient détectées suffisamment tôt pour assurer la disponibilité requise du système, et que la détection des défaillances par les dispositifs d'auto-contrôle soit mise en regard de la complexité supplémentaire qui est introduite. Il convient que les exigences de 4.8 et de A.2.8 de la CEI 60880 relatives à l'auto-surveillance soient remplies autant que possible pour chaque classe de système.	A.2.8	A.2.2
6.1.2 Spécification du système	A.1	5.3
Conformément à l'Article A.1 de la CEI 60880, cette phase inclut les activités nécessaires à la définition des exigences relatives au logiciel, au matériel et à l'intégration du système.		
6.1.2.1 Sélection des composants préexistants	4.3	15
NOTE 2 Le paragraphe 4.3 de la CEI 60880-2 traite des critères d'acceptation de logiciels préexistants réutilisables pour les fonctions de catégorie A.		
6.1.2.2.3 Défense contre la propagation et les effets secondaires des défaillances	Aucun	Aucun
NOTE Les exigences détaillées pour éviter les structures logicielles prédisposées aux erreurs et pour vérifier et tester les modules logiciels figurent dans la CEI 60880 et la CEI 60880-2.		

IEC 61513 clauses referring to	IEC 60880 (1986)/Suppl. clause	Clause in this standard
5.5.1 Architectural design documentation	4.2 Suppl	14
NOTE Requirements on software engineering methods and tools for class 1 systems are given in 4.2 and 4.3 of IEC 60880-2	4.3 Suppl	15
6. System safety life cycle	6.1	8.1
NOTE 1 This requirement differs from that stated in 6.1 of IEC 60880. For software it is desirable, but not considered necessary, to complete each phase of a development before starting the next phase providing the requirements defined above have been addressed		
NOTE 2 For class 1 systems, software requirements on this activity are defined in IEC 60880.	None	
NOTE 3 For class 1 systems, requirements for predeveloped software are defined in IEC 60880-2.	None	
6.1.1.1.1 Application function	A.2.2	None
… An estimate of the quantitative reliability of the application functions may be required for verification of the system design and of the plant design basis (see A.2.2 of IEC 60880). This estimation is normally performed for the system hardware design where well-established practices exist but there is no generally recognised method available for the quantitative evaluation of software reliability (see 6.1.3.1.2). …		
6.1.1.2.1 System architecture	4.1.1	13.1
NOTE Failures due to software are systematic and not random failures. Therefore, the single-failure criterion cannot be applied to the software design of a system in the same manner as it can be applied for hardware design. Possible effects of CCF due to software inside each defence layer and between redundant layers have to be considered at the level of each system and of the I&C architecture (see 4.1.1 of IEC 60880-2)	Suppl	
6.1.1.2.2 Internal behaviour of the system	B2.d	B2.d
c) (Specific requirement) In order to provide a high degree of assurance on deterministic behaviour, class 1 systems should be developed by techniques such as those of appendix B of IEC 60880 (notably B2.*d* on execution time and B2.*e* on interrupts). …	B2.e	B2.e
NOTE 3 See Clause 1 of IEC 60880 concerning the role of the appendices to the standard and what is required if practices differing from those of the appendices are used.	1	5.5.3
6.1.1.2.3 Self-monitoring and tolerance to failure	4.8	6.2
a) Systems should be designed so that errors and failures are detected sufficiently early to maintain the required system availability. The detection of failures by self-test facilities should be balanced with the additional complexity that is introduced. The requirements of 4.8 and A.2.8 of IEC 60880 on self-supervision should be supported in general and as far as possible for each class of system	A.2.8	A.2.2
6.1.2 System specification	A.1	5.3
In accordance with Clause A.1 of IEC 60880, this phase includes the activities necessary to produce the software requirements, the hardware requirements and the integration requirements of the system		
6.1.2.1 Selection of pre-existing component	4.3	15
NOTE 2 Subclause 4.3 of IEC 60880-2 addresses acceptance criteria for reusable pre-existing software for category A functions.		
6.1.2.2.3 Defence against propagation and side-effects of failure	None	None
NOTE Detailed requirements for avoidance of error-prone software structures and for verification and tests of software modules are given in IEC 60880 and IEC 60880-2.		

Article de la CEI 61513 faisant référence à	Article de la CEI 60880 (1986)/ Suppl	Article de la présente norme
6.1.2.3 Spécification du logiciel NOTE 1 L'architecture du logiciel définit les principaux composants et sous-systèmes du logiciel, leur interconnexion et la manière dont les caractéristiques exigées seront atteintes. Les exigences relatives à l'architecture du logiciel sortent du cadre de la présente norme (pour les systèmes de classe 1, se référer à la CEI 60880 et à la CEI 60880-2). a) Afin de faciliter la spécification, la vérification et la validation des fonctions d'application, il convient que l'architecture du logiciel prévoie une séparation claire entre le logiciel d'application et le logiciel système (voir B2.*a* de la CEI 60880). Dans ce cas, la vérification et la validation de la spécification du logiciel d'application peuvent être effectuées indépendamment.	Aucun B2.a	Aucun B2.a
6.1.3 Conception détaillée et réalisation du système Pour les systèmes de classe 1, les exigences relatives au développement du logiciel figurent dans la CEI 60880 et la CEI 60880-2, et les exigences relatives au matériel dans la CEI 60987.	Aucun	Aucun
6.1.4 Intégration du système L'objectif de cette phase est d'assembler les modules matériels et logiciels et de vérifier la compatibilité du logiciel chargé dans le matériel (voir Article 7 de la CEI 60880).	7	9
6.1.5 Validation du système L'objectif de cette phase est de tester le système intégré afin de démontrer la conformité aux spécifications de fonctionnalité, de performance et d'interface (voir Article 8 de la CEI 60880). b) (Exigence spécifique) Les exigences de l'Article 8 de la CEI 60880 doivent être appliquées pour valider les fonctions de catégorie A.	8	10
6.1.7 Modification de la conception e) (Exigence spécifique) Pour les systèmes de classe 1, les processus de modification du logiciel et du matériel doivent être conformes respectivement à l'Article 9 de la CEI 60880 et à l'Article 11 de la CEI 60987.	9	11
6.2.1 Plan d'assurance qualité du système NOTE Les exigences relatives au plan d'assurance qualité du logiciel des systèmes de sûreté sont définies à l'Article 3 de la CEI 60880. e) Le plan d'assurance qualité doit être établi tôt dans le cycle de vie du système, et doit être prévu dans le programme général des autres activités de cycle de vie de la sûreté de l'I&C. Le plan peut soit faire partie de la spécification du système, soit constituer un document associé (voir 3.2 de la CEI 60880).	3 3.2	5.5 5.5
6.2.1.1 Plan de vérification du système h) (Exigence spécifique) Pour les systèmes de classe 1, le plan de vérification doit être réalisé par des personnes indépendantes de celles ayant conçu le système (selon 6.2.1 de la CEI 60880).	6.2.1	8.2.1
6.2.3 Plan d'intégration du système Le plan d'intégration du système traite des mesures administratives et techniques pour intégrer les sous-systèmes au système et pour intégrer le matériel et le logiciel. Il couvre les activités décrites en 7.4 de la CEI 60880.	7.4	9.1
6.2.4 Plan de validation du système b) (Exigence spécifique) Pour les fonctions de catégories A, le plan de validation du système doit être établi et les activités de validation réalisées par des équipes indépendantes de celles qui ont conçu, réalisé et/ou modifié le système (voir Article 8 de la CEI 60880).	8	10
6.2.5 Plan d'installation du système b) (Exigence spécifique) Pour les systèmes de classe 1, le plan d'installation doit être conforme aux exigences de l'Article 9 de la CEI 60987 et de 10.1.1 de la CEI 60880.	10.1.1	Aucun

IEC 61513 clauses referring to	IEC 60880 (1986)/Suppl. clause	Clause in this standard
6.1.2.3 Software specification NOTE 1 The software architecture defines the major components and subsystems of the software, how they are interconnected, and how the required attributes will be achieved. The requirements for software architecture are outwith the scope of this standard. (For class 1 systems refer to IEC 60880 and IEC 60880-2.) a) In order to ease the specification, verification and validation of the application functions, the software architecture should provide a clear separation between the application software and the system software (see B2.a of IEC 60880). In such cases, the verification and validation of the application software may be performed independently.	None B2.a	None B2.a
6.1.3 System detailed design and implementation For class 1 systems, requirements on software development are established in IEC 60880 and IEC 60880-2, and hardware requirements in IEC 60987.	None	None
6.1.4 System integration The objective of this phase is to assemble hardware and software modules and verify compatibility of the software loaded into the hardware (see Clause 7 of IEC 60880).	7	9
6.1.5 System validation The objective of this phase is to test the integrated system to ensure compliance with the functional, performance and interface specifications (see Clause 8 of IEC 60880). b) (Specific requirement) The requirements of Clause 8 of IEC 60880 shall apply to validation of category A functions.	8	10
6.1.7 System design modification e) (Specific requirement) For class 1 systems, the modification process of software shall be in accordance with Clause 9 of IEC 60880 and the modification process of hardware in accordance with Clause 11 of IEC 60987.	9	11
6.2.1 System quality assurance plan NOTE The requirements for software quality assurance plan of safety systems are defined in Clause 3 of IEC 60880. e) The quality assurance plan shall be established at an early stage of the system life cycle and shall be planned within the general schedule of the other activities of the I&C safety life cycle. The plan may be either a part of the system specification or a companion document (see 3.2 of IEC 60880).	3 3.2	5.5 5.5
6.2.1.1 System verification plan h) (Specific requirement) For class 1 systems, the verification plan shall be executed by individuals independent of the designers of the system (according to 6.2.1 of IEC 60880).	6.2.1	8.2.1
6.2.3 System integration plan System integration covers the activities to integrate subsystems into the system and to integrate hardware and software for CB systems. It covers notably the activities described in 7.4 of IEC 60880.	7.4	9.1
6.2.4 System validation plan b) (Specific requirement) For category A functions, the system validation plan shall be developed and validation activities performed by teams independent from the ones who designed, implemented, and or modified the system (see Clause 8 of IEC 60880).	8	10
6.2.5 System installation plan b) (Specific requirement) For class 1 systems, the installation plan shall meet the requirements of Clause 9 of IEC 60987 and 10.1.1 of IEC 60880.	10.1.1	None

Article de la CEI 61513 faisant référence à	Article de la CEI 60880 (1986)/ Suppl	Article de la présente norme
6.3.1.2 Caractéristiques NOTE 3 Les exigences détaillées relatives aux outils logiciels pour les systèmes de classe 1 figurent en 4.2 de la CEI 60880-2.	4.2 Suppl	14
6.3.3 Documentation de la conception et réalisation détaillées du système NOTE Pour les systèmes de classe 1, les exigences relatives à la documentation du logiciel figurent dans la CEI 60880 et la CEI 60880-2, et les exigences relatives à la documentation du matériel dans la CEI 60987.	Aucun	Aucun
6.3.4.2 Caractéristiques b) (Exigence spécifique) Pour les systèmes de classe 1, les exigences de 7.7 de la CEI 60880 s'appliquent.	7.7	9.5
6.3.5.2 Caractéristiques (Exigence spécifique) Pour les systèmes de classe 1, les exigences de 8.1 de la CEI 60880 s'appliquent.	8.1	10.3
6.4.1.2 Evaluation et estimation des logiciels b) (Exigence spécifique) Pour les systèmes de classe 1, les nouveaux logiciels doivent être évalués et estimés conformément aux exigences de la CEI 60880. c) (Exigence spécifique) Il convient que les logiciels des équipements préexistants sélectionnés pour les systèmes de classe 1 aient été élaborés conformément à des directives et normes reconnues et appropriées au niveau de qualité élevé requis pour les fonctions de catégorie A (voir 8.1.2 de la CEI 61226). En particulier, les exigences de la CEI 60880-2 relatives aux logiciels et outils prédéveloppés et celles de la CEI 60987 doivent être respectées.	 Aucun Aucun Suppl	
Tableau 5 Exigences relatives à la spécification et à la mise en oeuvre des FSE Validation: CEI 60880 (6.2.4)	Clause inexistante	

Il convient que les références de la CEI 61513 à la CEI 60880 soient mises à jour lors de la prochaine révision de la CEI 61513.

J.3 Articles de la CEI 60880 (1986) non référencés dans la CEI 61513 mais pertinents au niveau du système

Ces articles sont maintenus dans cette révision de la CEI 60880 et révisés quand nécessaire. Il convient qu'ils soient considérés comme des entrées pour la prochaine révision de la CEI 61513 et donc de cette norme.

Tableau J.2 – Articles de la CEI 60880 à considérer lors de la prochaine révision de la CEI 61513

6.3 Test périodique
9.2 Intégration du système
10.2 Validation du système
12.4 Formation des opérateurs

IEC 61513 clauses referring to	IEC 60880 (1986)/Suppl. clause	Clause in this standard
6.3.1.2 Characteristics NOTE 3 Detailed requirements regarding software tools for class 1 systems are given in 4.2 of IEC 60880-2.	4.2 Suppl	14
6.3.3 System detailed design documentation NOTE For class 1 systems, requirements on software documentation are established in IEC 60880 and IEC 60880-2, and requirements on hardware documentation in IEC 60987.	None	None
6.3.4.2 Characteristics b) (Specific requirement) For class 1 systems, the requirements of 7.7 of IEC 60880 apply.	7.7	9.5
6.3.5.2 Characteristics (Specific requirement) For class 1 systems, the requirements of 8.1 of IEC 60880 apply.	8.1	10.3
6.4.1.2 Software evaluation and assessment b) (Specific requirement) For systems of class 1, newly developed software shall be evaluated and assessed in accordance with the requirements of IEC 60880. c) (Specific requirement) Pre-existing equipment selected for class 1 systems should have been developed according to recognized guides and standards appropriate to the high level of quality required for class A functions (see 8.1.2 of IEC 61226). In particular, the requirements of IEC 60880-2 on pre-developed software and tools and the requirements of IEC 60987 shall be met.	None None Suppl	
Table 5 Requirements for the specification and implementation of the FSE Validation: IEC 60880 (6.2.4)	Clause not defined	

References from IEC 61513 to IEC 60880 should be updated during the next revision of IEC 61513.

J.3 IEC 60880 (1986) clauses not referenced by IEC 61513 but relevant to the system level

These clauses have been kept in this revision of IEC 60880 and revised where appropriate. They should be considered as input during the next revision of IEC 61513 and then of this standard.

Table J.2 – IEC 60880 clauses to be considered during the next revision of IEC 61513

6.3 Periodic testing
9.2 System integration
10.2 System validation
12.4 Operator training

Standards Survey

The IEC would like to offer you the best quality standards possible. To make sure that we continue to meet your needs, your feedback is essential. Would you please take a minute to answer the questions overleaf and fax them to us at +41 22 919 03 00 or mail them to the address below. Thank you!

Customer Service Centre (CSC)

International Electrotechnical Commission
3, rue de Varembé
1211 Genève 20
Switzerland

or

Fax to: **IEC**/CSC at +41 22 919 03 00

Thank you for your contribution to the standards-making process.

Nicht frankieren
Ne pas affranchir

Non affrancare
No stamp required

A Prioritaire

RÉPONSE PAYÉE

SUISSE

Customer Service Centre (CSC)
International Electrotechnical Commission
3, rue de Varembé
1211 GENEVA 20
Switzerland

Q1 Please report on **ONE STANDARD** and **ONE STANDARD ONLY**. Enter the exact number of the standard: *(e.g. 60601-1-1)*

.....

Q2 Please tell us in what capacity(ies) you bought the standard *(tick all that apply).* I am the/a:

purchasing agent ❑
librarian ❑
researcher ❑
design engineer ❑
safety engineer ❑
testing engineer ❑
marketing specialist ❑
other.....

Q3 I work for/in/as a: *(tick all that apply)*

manufacturing ❑
consultant ❑
government ❑
test/certification facility ❑
public utility ❑
education ❑
military ❑
other.....

Q4 This standard will be used for: *(tick all that apply)*

general reference ❑
product research ❑
product design/development ❑
specifications ❑
tenders ❑
quality assessment ❑
certification ❑
technical documentation ❑
thesis ❑
manufacturing ❑
other.....

Q5 This standard meets my needs: *(tick one)*

not at all ❑
nearly ❑
fairly well ❑
exactly ❑

Q6 If you ticked NOT AT ALL in Question 5 the reason is: *(tick all that apply)*

standard is out of date ❑
standard is incomplete ❑
standard is too academic ❑
standard is too superficial ❑
title is misleading ❑
I made the wrong choice ❑
other .....

Q7 Please assess the standard in the following categories, using the numbers:
(1) unacceptable,
(2) below average,
(3) average,
(4) above average,
(5) exceptional,
(6) not applicable

timeliness.....
quality of writing.....
technical contents.....
logic of arrangement of contents .....
tables, charts, graphs, figures.....
other .....

Q8 I read/use the: *(tick one)*

French text only ❑
English text only ❑
both English and French texts ❑

Q9 Please share any comment on any aspect of the IEC that you would like us to know:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Enquête sur les normes

La CEI ambitionne de vous offrir les meilleures normes possibles. Pour nous assurer que nous continuons à répondre à votre attente, nous avons besoin de quelques renseignements de votre part. Nous vous demandons simplement de consacrer un instant pour répondre au questionnaire ci-après et de nous le retourner par fax au +41 22 919 03 00 ou par courrier à l'adresse ci-dessous. Merci !

Centre du Service Clientèle (CSC)

Commission Electrotechnique Internationale
3, rue de Varembé
1211 Genève 20
Suisse

ou

Télécopie: **CEI**/CSC +41 22 919 03 00

Nous vous remercions de la contribution que vous voudrez bien apporter ainsi à la Normalisation Internationale.

Nicht frankieren
Ne pas affranchir

Non affrancare
No stamp required

A Prioritaire

RÉPONSE PAYÉE

SUISSE

Centre du Service Clientèle (CSC)
Commission Electrotechnique Internationale
3, rue de Varembé
1211 GENÈVE 20
Suisse

Q1 Veuillez ne mentionner qu'**UNE SEULE NORME** et indiquer son numéro exact: (*ex. 60601-1-1)*

.....

Q2 En tant qu'acheteur de cette norme, quelle est votre fonction?
(cochez tout ce qui convient)
Je suis le/un:

agent d'un service d'achat ❑
bibliothécaire ❑
chercheur ❑
ingénieur concepteur ❑
ingénieur sécurité ❑
ingénieur d'essais ❑
spécialiste en marketing ❑
autre(s).....

Q3 Je travaille:
(cochez tout ce qui convient)

dans l'industrie ❑
comme consultant ❑
pour un gouvernement ❑
pour un organisme d'essais/ certification ❑
dans un service public ❑
dans l'enseignement ❑
comme militaire ❑
autre(s).....

Q4 Cette norme sera utilisée pour/comme
(cochez tout ce qui convient)

ouvrage de référence ❑
une recherche de produit ❑
une étude/développement de produit ❑
des spécifications ❑
des soumissions ❑
une évaluation de la qualité ❑
une certification ❑
une documentation technique ❑
une thèse ❑
la fabrication ❑
autre(s).....

Q5 Cette norme répond-elle à vos besoins:
(une seule réponse)

pas du tout ❑
à peu près ❑
assez bien ❑
parfaitement ❑

Q6 Si vous avez répondu PAS DU TOUT à Q5, c'est pour la/les raison(s) suivantes:
(cochez tout ce qui convient)

la norme a besoin d'être révisée ❑
la norme est incomplète ❑
la norme est trop théorique ❑
la norme est trop superficielle ❑
le titre est équivoque ❑
je n'ai pas fait le bon choix ❑
autre(s) .....

Q7 Veuillez évaluer chacun des critères ci-dessous en utilisant les chiffres
(1) inacceptable,
(2) au-dessous de la moyenne,
(3) moyen,
(4) au-dessus de la moyenne,
(5) exceptionnel,
(6) sans objet

publication en temps opportun .....
qualité de la rédaction.....
contenu technique .....
disposition logique du contenu .....
tableaux, diagrammes, graphiques, figures.....
autre(s) .....

Q8 Je lis/utilise: *(une seule réponse)*

uniquement le texte français ❑
uniquement le texte anglais ❑
les textes anglais et français ❑

Q9 Veuillez nous faire part de vos observations éventuelles sur la CEI:

.....
.....
.....
.....
.....
.....

ISBN 2-8318-8636-8

ICS 27.120.20

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND