NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
61508-7

Première édition
First edition
2000-03

Sécurité fonctionnelle des systèmes électriques/ électroniques/électroniques programmables relatifs à la sécurité –

Partie 7:
Présentation de techniques et mesures

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

Part 7:
Overview of techniques and measures

Numéro de référence
Reference number
CEI/IEC 61508-7:2000

Numéros des publications

Depuis le 1er janvier 1997, les publications de la CEI sont numérotées à partir de 60000.

Publications 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.

Validité de la présente publication

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 la date de reconfirmation de la publication sont disponibles dans le Catalogue de la CEI.

Les renseignements relatifs à des questions à l'étude et des travaux en cours entrepris par le comité technique qui a établi cette publication, ainsi que la liste des publications établies, se trouvent dans les documents ci-dessous:

- **«Site web» de la CEI***
- **Catalogue des publications de la CEI**
  Publié annuellement et mis à jour régulièrement
  (Catalogue en ligne)*
- **Bulletin de la CEI**
  Disponible à la fois au «site web» de la CEI* et comme périodique imprimé

Terminologie, symboles graphiques et littéraux

En ce qui concerne la terminologie générale, le lecteur se reportera à la CEI 60050: *Vocabulaire Electrotechnique International* (VEI).

Pour les symboles graphiques, les symboles littéraux et les signes d'usage général approuvés par la CEI, le lecteur consultera la CEI 60027: *Symboles littéraux à utiliser en électrotechnique,* la CEI 60417: S*ymboles graphiques utilisables sur le matériel. Index, relevé et compilation des feuilles individuelles,* et la CEI 60617: *Symboles graphiques pour schémas.*

* Voir adresse «site web» sur la page de titre.

Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

Consolidated publications

Consolidated versions of some IEC publications including amendments are available. 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.

Validity of this publication

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 the date of the reconfirmation of the publication is available in the IEC catalogue.

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 to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
  Published yearly with regular updates
  (On-line catalogue)*
- **IEC Bulletin**
  Available both at the IEC web site* and as a printed periodical

Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV).

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams.*

* See web site address on title page.

NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
61508-7

Première édition
First edition
2000-03

Sécurité fonctionnelle des systèmes électriques/ électroniques/électroniques programmables relatifs à la sécurité –

Partie 7: Présentation de techniques et mesures

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

Part 7: Overview of techniques and measures

International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland
Telefax: +41 22 919 0300 e-mail: inmail@iec.ch IEC web site http://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

Pages

CONTENTS

Articles Pages

Clause Page

Articles Pages

Clause Page

Articles Pages

Clause Page

Articles Pages

Clause Page

Articles Pages

Clause Page

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

SÉCURITÉ FONCTIONNELLE DES SYSTÈMES ÉLECTRIQUES/ ÉLECTRONIQUES/ÉLECTRONIQUES PROGRAMMABLES RELATIFS À LA SÉCURITÉ –

Partie 7: Présentation de techniques et mesures

AVANT-PROPOS

1) La CEI (Commission Électrotechnique Internationale) 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. 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 intéressés sont représentés dans chaque comité d'études.
3) Les documents produits se présentent sous la forme de recommandations internationales. Ils sont publiés comme normes, spécifications techniques, rapports techniques ou guides et agréés comme tels par les Comités nationaux.
4) Dans le but d'encourager l'unification internationale, les Comités nationaux de la CEI s'engagent à appliquer de façon transparente, dans toute la mesure possible, les Normes internationales de la CEI dans leurs normes nationales et régionales. Toute divergence entre la norme de la CEI et la norme nationale ou régionale correspondante doit être indiquée en termes clairs dans cette dernière.
5) La CEI n'a fixé aucune procédure concernant le marquage comme indication d'approbation et sa responsabilité n'est pas engagée quand un matériel est déclaré conforme à l'une de ses normes.
6) L'attention est attirée sur le fait que certains des éléments de la présente Norme internationale 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 61508-7 a été établie par le sous-comité 65A: Aspects systèmes, du comité d'études 65 de la CEI: Mesure et commande dans les processus industriels.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
65A/293/FDIS	65A/299/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 3.

Les annexes A, B, C et D sont données uniquement à titre d'information.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

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

Part 7: Overview of techniques and measures

FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the 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, the IEC publishes International Standards. 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. The 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 the 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 National Committees.
3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.
6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61508-7 has been prepared by subcommittee 65A: System aspects, of IEC technical committee 65: Industrial-process measurement and control.

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

FDIS	Report on voting
65A/293/FDIS	65A/299/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 3.

Annexes A, B, C and D are for information only.

La CEI 61508 est composée des parties suivantes, regroupées sous le titre général *Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité:*

Partie 1: Prescriptions générales

Partie 2: Prescriptions pour les systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité

Partie 3: Prescriptions concernant les logiciels

Partie 4: Définitions et abréviations

Partie 5: Exemples de méthodes pour la détermination des niveaux d'intégrité de sécurité

Partie 6: Lignes directrices pour l'application de la CEI 61508-2 et de la CEI 61508-3

Partie 7: Présentation de techniques et mesures

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant 2006. A cette date, la publication sera

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

IEC 61508 consists of the following parts, under the general title *Functional safety of electrical/electronic/programmable electronic safety-related systems:*

– Part 1: General requirements
– Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems
– Part 3: Software requirements
– Part 4: Definitions and abbreviations
– Part 5: Examples of methods for the determination of safety integrity levels
– Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3
– Part 7: Overview of techniques and measures

The committee has decided that the contents of this publication will remain unchanged until 2006. At this date, the publication will be

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

INTRODUCTION

Les systèmes électriques/électroniques sont utilisés depuis des années pour exécuter des fonctions liées à la sécurité dans la plupart des secteurs d'application. Des systèmes à base d'informatique (que l'on nommera de façon générique systèmes électroniques programmables (PES)) sont utilisés dans tous les secteurs d'application pour exécuter des fonctions non liées à la sécurité, mais aussi de plus en plus souvent liées à la sécurité. Si l'on veut exploiter efficacement, et en toute sécurité, la technologie des systèmes informatiques, il est indispensable de fournir à tous les responsables suffisamment d'éléments liés à la sécurité pour les guider dans leurs prises de décisions.

La présente Norme internationale présente une approche générique de toutes les activités liées au cycle de vie de sécurité de systèmes comprenant des composants électriques et/ou électroniques et/ou électroniques programmables (systèmes électriques/électroniques/électroniques programmables (E/E/PES)) qui sont utilisés pour réaliser des fonctions de sécurité. Cette approche unifiée a été adoptée afin de développer une politique technique rationnelle et cohérente concernant tous les appareils électriques liés à la sécurité. L'un des principaux objectifs poursuivis consiste à faciliter l'élaboration de normes par secteur d'application.

Dans la plupart des cas, la sécurité est obtenue par un certain nombre de systèmes de protection fondés sur diverses technologies (par exemple mécanique, hydraulique, pneumatique, électrique, électronique, électronique programmable). En conséquence, il faut que toute stratégie de sécurité prenne non seulement en compte tous les éléments d'un système individuel (par exemple les capteurs, les appareils de commande, les actionneurs), mais qu'elle considère aussi tous les systèmes relatifs à la sécurité comme des éléments individuels d'un ensemble complexe. C'est pourquoi la présente Norme internationale, bien que traitant essentiellement des systèmes E/E/PES relatifs à la sécurité, fournit néanmoins un cadre de sécurité susceptible de concerner les systèmes relatifs à la sécurité basés sur d'autres technologies.

Personne n'ignore la grande variété des applications E/E/PES. Celles-ci recouvrent, à des degrés de complexité très divers, un fort potentiel de danger et de risques dans tous les secteurs d'application. Pour chaque application, la nature exacte des mesures de sécurité envisagées dépendra de plusieurs facteurs propres à l'application. La présente Norme internationale, de par son caractère général, rendra désormais possible la prescription de ces mesures dans des Normes internationales par secteur d'application.

La présente Norme internationale

– concerne toutes les phases appropriées du cycle de vie de sécurité global des E/E/PES et du logiciel (depuis la conceptualisation initiale, en passant par la conception, l'installation, l'exploitation et la maintenance, jusqu'à la mise hors service) lorsque les E/E/PES exécutent des fonctions de sécurité;

– a été élaborée dans le soucis de l'évolution rapide des technologies; le cadre est suffisamment solide et étendu pour pourvoir aux évolutions futures;

– permet l'élaboration de normes internationales par secteur d'application concernant les E/E/PES relatifs à la sécurité; l'élaboration de normes internationales par secteur d'application à partir de la présente norme devrait permettre d'atteindre un haut niveau de cohérence (par exemple pour ce qui est des principes sous-jacents, de la terminologie, etc.) à la fois au sein de chaque secteur d'application, et d'un secteur à l'autre. La conséquence en est une amélioration en termes de sécurité et de bénéfices économiques;

– fournit une méthode de développement des prescriptions de sécurité nécessaires pour réaliser la sécurité fonctionnelle des systèmes E/E/PE relatifs à la sécurité;

INTRODUCTION

Systems comprised of electrical and/or electronic components have been used for many years to perform safety functions in most application sectors. Computer-based systems (generically referred to as programmable electronic systems (PESs)) are being used in all application sectors to perform non-safety functions and, increasingly, to perform safety functions. If computer system technology is to be effectively and safely exploited, it is essential that those responsible for making decisions have sufficient guidance on the safety aspects on which to make those decisions.

This International Standard sets out a generic approach for all safety lifecycle activities for systems comprised of electrical and/or electronic and/or programmable electronic components (electrical/electronic/programmable electronic systems (E/E/PESs)) that are used to perform safety functions. This unified approach has been adopted in order that a rational and consistent technical policy be developed for all electrically based safety-related systems. A major objective is to facilitate the development of application sector standards.

In most situations, safety is achieved by a number of protective systems which rely on many technologies (for example mechanical, hydraulic, pneumatic, electrical, electronic, programmable electronic). Any safety strategy must therefore consider not only all the elements within an individual system (for example sensors, controlling devices and actuators) but also all the safety-related systems making up the total combination of safety-related systems. Therefore, while this International Standard is concerned with electrical/electronic/programmable electronic (E/E/PE) safety-related systems, it may also provide a framework within which safety-related systems based on other technologies may be considered.

It is recognised that there is a great variety of E/E/PES applications in a variety of application sectors and covering a wide range of complexity, hazard and risk potentials. In any particular application, the exact prescription of safety measures will be dependent on many factors specific to the application. This International Standard, by being generic, will enable such a prescription to be formulated in future application sector International Standards.

This International Standard

– considers all relevant overall, E/E/PES and software safety lifecycle phases (for example, from initial concept, through design, implementation, operation and maintenance to decommissioning) when E/E/PESs are used to perform safety functions;
– has been conceived with a rapidly developing technology in mind; the framework is sufficiently robust and comprehensive to cater for future developments;
– enables application sector International Standards, dealing with safety-related E/E/PESs, to be developed; the development of application sector international standards, within the framework of this standard, should lead to a high level of consistency (for example, of underlying principles, terminology, etc.) both within application sectors and across application sectors; this will have both safety and economic benefits;
– provides a method for the development of the safety requirements specification necessary to achieve the required functional safety for E/E/PE safety-related systems;

– utilise des niveaux d'intégrité de sécurité afin de spécifier les niveaux cibles d'intégrité de sécurité des fonctions de sécurité à réaliser par les systèmes E/E/PE relatifs à la sécurité;

– adopte une approche basée sur le risque encouru pour déterminer les niveaux d'intégrité de sécurité prescrits;

– fixe des objectifs quantitatifs pour les mesures de défaillances des systèmes E/E/PE relatifs à la sécurité qui sont en rapport avec les niveaux d'intégrité de sécurité;

– fixe une limite inférieure pour les mesures de défaillances, dans le cas d'un mode de défaillance dangereux, cette limite pouvant être exigée pour un système E/E/PE relatif à la sécurité unique. Dans le cas d'un système E/E/PE relatif à la sécurité fonctionnant

  – dans un mode de faible sollicitation, la limite inférieure est fixée à une probabilité moyenne de défaillance de 10^{-5} afin que les fonctions pour lesquelles le système a été conçu soient exécutées lorsqu'elles sont requises;

  – dans un mode de fonctionnement continu ou de forte sollicitation, la limite inférieure est fixée à une probabilité de défaillance dangereuse de 10^{-9} par heure;

  NOTE Un système E/E/PE relatif à la sécurité unique n'implique pas nécessairement une architecture à une seule voie.

– adopte une large gamme de principes, techniques et mesures pour la réalisation de la sécurité fonctionnelle des systèmes E/E/PE relatifs à la sécurité, mais n'est pas fondée sur le concept de sécurité intrinsèque qui peut être intéressant lorsque les modes de défaillances sont bien définis et que le niveau de complexité est relativement faible; le concept de sécurité intrinsèque a été considéré comme inadéquat en raison de l'immense gamme de complexité des systèmes E/E/PE relatifs à la sécurité qui entrent dans le domaine d'application de la présente norme.

- uses safety integrity levels for specifying the target level of safety integrity for the safety functions to be implemented by the E/E/PE safety-related systems;
- adopts a risk-based approach for the determination of the safety integrity level requirements;
- sets numerical target failure measures for E/E/PE safety-related systems which are linked to the safety integrity levels;
- sets a lower limit on the target failure measures, in a dangerous mode of failure, that can be claimed for a single E/E/PE safety-related system; for E/E/PE safety-related systems operating in
  - a low demand mode of operation, the lower limit is set at an average probability of failure of 10^{-5} to perform its design function on demand;
  - a high demand or continuous mode of operation, the lower limit is set at a probability of a dangerous failure of 10^{-9} per hour;

  NOTE A single E/E/PE safety-related system does not necessarily mean a single-channel architecture.
- adopts a broad range of principles, techniques and measures to achieve functional safety for E/E/PE safety-related systems, but does not rely on the concept of fail-safe, which may be of value when the failure modes are well defined and the level of complexity is relatively low – the concept of fail-safe was considered inappropriate because of the full range of complexity of E/E/PE safety-related systems that are within the scope of the standard.

SÉCURITÉ FONCTIONNELLE DES SYSTÈMES ÉLECTRIQUES/ ÉLECTRONIQUES/ÉLECTRONIQUES PROGRAMMABLES RELATIFS À LA SÉCURITÉ –

Partie 7: Présentation de techniques et mesures

1 Domaine d'application

1.1 La présente partie de la CEI 61508 contient une présentation de différentes techniques et mesures de sécurité pertinentes pour la CEI 61508-2 et la CEI 61508-3.

NOTE Il convient que les références citées soient considérées comme des références fondamentales des méthodes et outils, ou comme des exemples; elles peuvent ne pas représenter l'état de l'art.

1.2 La CEI 61508-1, la CEI 61508-2, la CEI 61508-3 et la CEI 61508-4 sont des publications fondamentales de sécurité, bien que ce statut ne s'applique pas dans le cas de systèmes E/E/PE de sécurité de faible complexité (voir 3.4.4 de la CEI 61508-4). En tant que publications fondamentales de sécurité, elles sont destinées à être utilisées par tous les comités d'études pour la mise au point de leurs normes, conformément aux principes décrits dans le Guide 104 de la CEI et dans le Guide 51 ISO/CEI. La CEI 61508 est également prévue pour une utilisation en tant que norme autonome.

L'une des responsabilités d'un comité d'études est, chaque fois que cela peut s'appliquer, d'utiliser les publications fondamentales de sécurité pour préparer ses propres publications. Dans ce contexte, les prescriptions, les méthodes d'essais ou les conditions d'essais de la présente publication fondamentale de sécurité ne sont pas applicables, sauf s'il y est spécifiquement fait référence, ou si elles sont incorporées dans les publications préparées par ces comités d'études.

NOTE 1 La sécurité fonctionnelle d'un système E/E/PE relatif à la sécurité ne peut être réalisée que lorsque toutes les prescriptions pertinentes sont remplies. En conséquence, il est important que toutes les prescriptions pertinentes soient prises en considération avec soin et référencées de façon appropriée.

NOTE 2 Aux Etats-Unis et au Canada, dans l'attente de la publication de la future CEI 61511 (la version de CEI 61508 pour le processus) les normes nationales existantes pour la sécurité des processus industriels basés sur la CEI 61508 (c'est-à-dire ANSI/ISA 584.01-1996) peuvent être appliquées au domaine des processus industriels à la place de la CEI 61508.

1.3 La figure 1 montre la structure générale des parties 1 à 7 de la présente norme et indique le rôle que joue la CEI 61508-7 dans la réalisation de la sécurité fonctionnelle pour les systèmes E/E/PE relatifs à la sécurité.

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

Part 7: Overview of techniques and measures

1 Scope

1.1 This part of IEC 61508 contains an overview of various safety techniques and measures relevant to IEC 61508-2 and IEC 61508-3.

NOTE The references should be considered as basic references to methods and tools or as examples, and may not represent the state of the art.

1.2 IEC 61508-1, IEC 61508-2, IEC 61508-3 and IEC 61508-4 are basic safety publications, although this status does not apply in the context of low-complexity E/E/PE safety-related systems (see 3.4.4 of IEC 61508-4). As basic safety publications, they are intended for use by technical committees in the preparation of standards in accordance with the principles contained in IEC Guide 104 and ISO/IEC Guide 51. IEC 61508 is also intended for use as a stand-alone standard.

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

NOTE 1 The functional safety of an E/E/PE safety-related system can only be achieved when all related requirements are met. Therefore it is important that all related requirements are carefully considered and adequately referenced.

NOTE 2 In the USA and Canada, until the proposed process sector implementation of IEC 61508 (i.e. IEC 61511) is published as an international standard in the USA and Canada, existing national process safety standards based on IEC 61508 (i.e. ANSI/ISA S84.01-1996) can be applied to the process sector instead of IEC 61508.

1.3 Figure 1 shows the overall framework for parts 1 to 7 of this standard and indicates the role that IEC 61508-7 plays in the achievement of functional safety for E/E/PE safety-related systems.

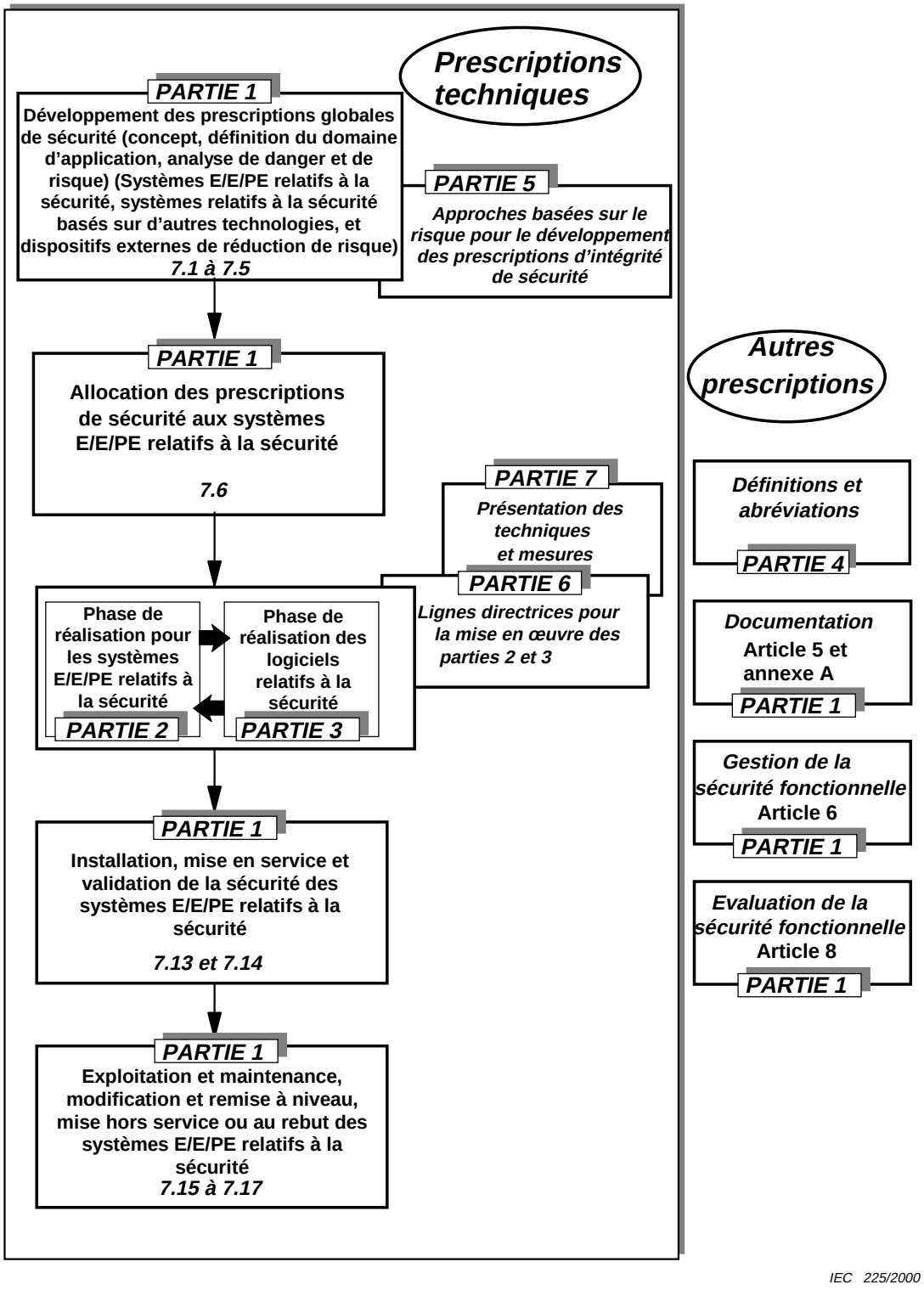

Figure 1 – Structure générale de la CEI 61508

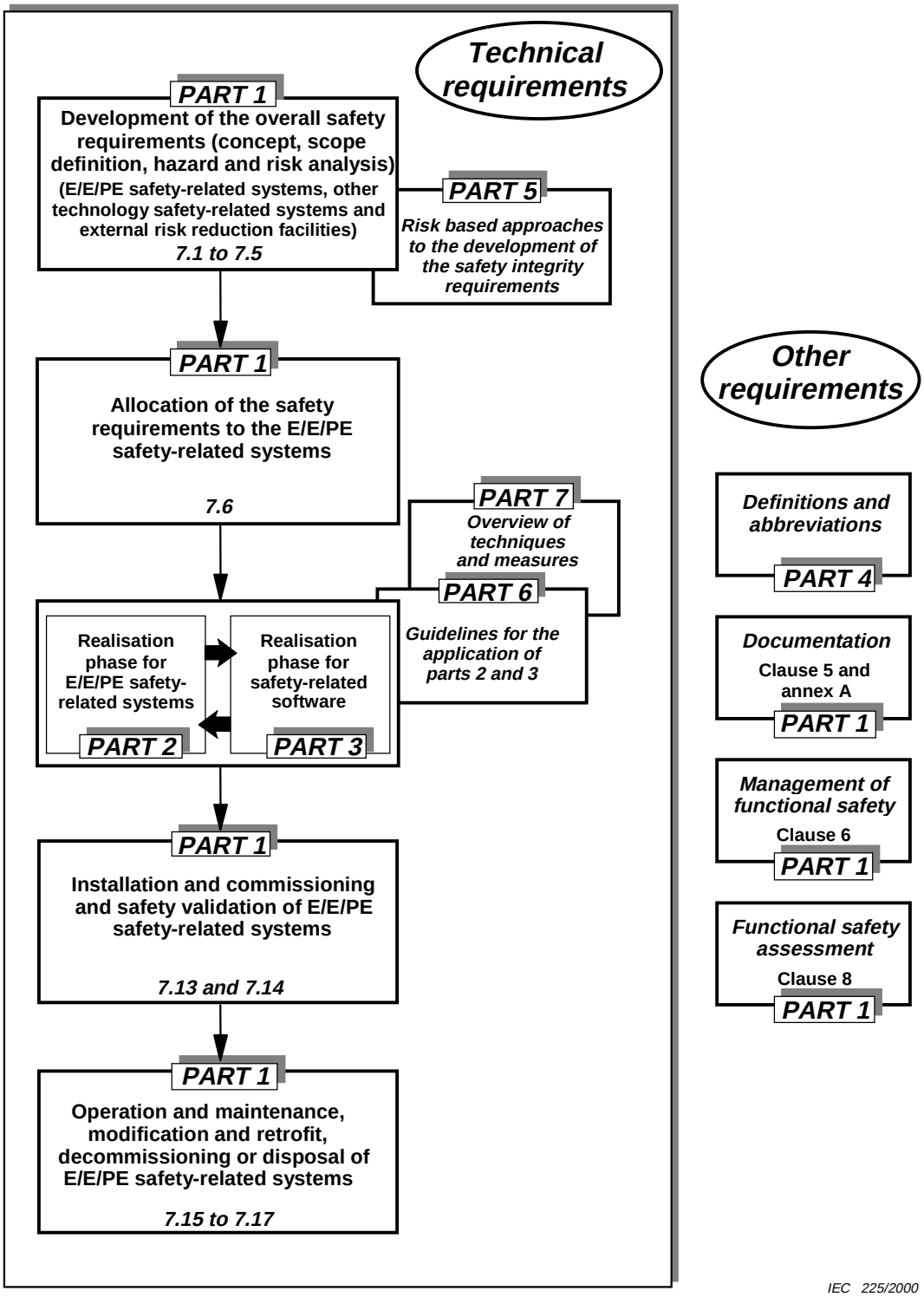

Figure 1 – Overall framework of IEC 61508

2 Références normatives

Les documents normatifs suivants contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente partie de la CEI 61508. Pour les références datées, les amendements ultérieurs ou les révisions de ces publications ne s'appliquent pas. Toutefois, les parties prenantes aux accords fondés sur la présente partie de la CEI 61508 sont invitées à rechercher la possibilité d'appliquer les éditions les plus récentes des documents normatifs indiqués ci-après. Pour les références non datées, la dernière édition du document normatif en référence s'applique. Les membres de la CEI et de l'ISO possèdent le registre des Normes internationales en vigueur.

CEI 61508-1:1998, *Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 1: Prescriptions générales*

CEI 61508-2, *Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 2: Prescriptions pour les systèmes électriques/ électroniques/électroniques programmables relatifs à la sécurité* [1)]

CEI 61508-3:1998, *Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 3: Prescriptions concernant les logiciels*

CEI 61508-4:1998, *Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 4: Définitions et abréviations*

CEI 61508-5:1998, *Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 5: Exemples de méthodes pour la détermination des niveaux d'intégrité de sécurité*

CEI 61508-6, *Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité – Partie 6: Lignes directrices pour l'application de la CEI 61508-2 et la CEI 61508-3* [1)]

Guide CEI 104:1997, *Elaboration des publications de sécurité et utilisation des publications fondamentales de sécurité et publications groupées de sécurité*

Guide ISO/CEI 51:1990, *Principes directeurs pour inclure dans les normes les aspects liés à la sécurité*

3 Définitions et abréviations

Pour les besoins de la présente partie de la CEI 61508, les définitions et abréviations données dans la CEI 61508-4 s'appliquent.

---

1) A publier.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 61508. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 61508 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

IEC 61508-1:1998, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements*

IEC 61508-2, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems* [1)]

IEC 61508-3:1998, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements*

IEC 61508-4:1998, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 4: Definitions and abbreviations of terms*

IEC 61508-5:1998, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 5 Examples of methods for the determination of safety integrity levels*

IEC 61508-6, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3* [1)]

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

IEC/ISO Guide 51:1990, *Guidelines for the inclusion of safety aspects in standards*

3 Definitions and abbreviations

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

[1)] To be published.

Annexe A
(informative)

Présentation de techniques et mesures pour les E/E/PES: maîtrise des défaillances aléatoires du matériel (voir la CEI 61508-2)

A.1 Electriques

Objectif général: Maîtriser les défaillances des composants électromécaniques.

A.1.1 Détection des défaillances par surveillance en ligne

NOTE Cette technique/mesure est mentionnée dans les tableaux A.2, A.3, A.7 et A.14 à A.19 de la CEI 61508-2.

But: Détecter les défaillances en surveillant le comportement du système E/E/PE relatif à la sécurité en réponse à l'exploitation normale (en ligne) de l'équipement commandé (EUC).

Description: Dans certaines conditions, les défaillances peuvent être détectées grâce aux informations concernant, par exemple, le comportement temporel de l'EUC. Par exemple, si un commutateur faisant partie du système E/E/PE relatif à la sécurité est normalement commandé par l'EUC et qu'il ne change pas d'état à l'instant voulu, une défaillance est détectée. Il n'est généralement pas possible de localiser la défaillance.

A.1.2 Surveillance des contacts de relais

NOTE Cette technique/mesure est mentionnée dans les tableaux A.2 et A.15 de la CEI 61508-2.

But: Détecter les défaillances, de soudure par exemple, des contacts de relais.

Description: Les relais à contact forcé (ou à contact dirigé positivement) sont conçus de façon que leurs contacts soient rigidement liés ensemble. En supposant qu'il y ait deux ensembles de contacts inverseurs *a* et *b*, si le contact normalement ouvert, *a*, se soude, le contact normalement fermé, *b*, ne peut pas se fermer quand la bobine du relais n'est plus commandée. Par conséquent, la vérification de la fermeture du contact *b* normalement fermé quand la bobine du relais n'est plus commandée peut servir à prouver que le contact *a* normalement ouvert s'est ouvert. Le défaut de fermeture du contact *b* normalement fermé indique une défaillance du contact *a*; il convient alors que le circuit de surveillance garantisse un arrêt en sécurité ou que l'arrêt en sécurité soit maintenu pour toute machine commandée par le contact *a*.

Références:

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Verriegelungseinrichtungen. F. Kreutzkampf, W. Hertel, Sicherheitstechnisches Informations- und Arbeitsblatt 330212, BIA-Handbuch. 17. Lfg. X/91, Erich Schmidt Verlag, Bielefeld.

Anlagensicherung mit Mitteln der MSR-Technik. G. Strohrman, Oldenburg, 1983.

A.1.3 Comparateur

NOTE Cette technique/mesure est mentionnée dans les tableaux A.2, A.3, A.4 de la CEI 61508-2.

But: Détecter, le plus tôt possible, les défaillances (non simultanées) dans une unité de traitement autonome ou dans le comparateur.

Annex A
(informative)

Overview of techniques and measures for E/E/PES: control of random hardware failures (see IEC 61508-2)

A.1 Electrical

Global objective: To control failures in electromechanical components.

A.1.1 Failure detection by on-line monitoring

NOTE This technique/measure is referenced in tables A.2, A.3, A.7 and A.14 to A.19 of IEC 61508-2.

Aim: To detect failures by monitoring the behaviour of the E/E/PE safety-related system in response to the normal (on-line) operation of the equipment under control (EUC).

Description: Under certain conditions, failures can be detected using information about (for example) the time behaviour of the EUC. For example, if a switch, which is part of the E/E/PE safety-related system, is normally actuated by the EUC, then if the switch does not change state at the expected time, a failure will have been detected. It is not usually possible to localise the failure.

A.1.2 Monitoring of relay contacts

NOTE This technique/measure is referenced in tables A.2 and A.15 of IEC 61508-2.

Aim: To detect failures (for example welding) of relay contacts.

Description: Forced contact (or positively guided contact) relays are designed so that their contacts are rigidly linked together. Assuming there are two sets of changeover contacts, *a* and *b*, if the normally open contact, *a*, welds, the normally closed contact, *b*, cannot close when the relay coil is next de-energised. Therefore, the monitoring of the closure of the normally closed contact *b* when the relay coil is de-energised may be used to prove that the normally open contact *a* has opened. Failure of normally closed contact *b* to close indicates a failure of contact *a*, so the monitoring circuit should ensure a safe shut-down, or ensure that shut-down is continued, for any machinery controlled by contact *a*.

References:

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Verriegelungseinrichtungen. F. Kreutzkampf, W. Hertel, Sicherheitstechnisches Informations- und Arbeitsblatt 330212, BIA-Handbuch. 17. Lfg. X/91, Erich Schmidt Verlag, Bielefeld.

Anlagensicherung mit Mitteln der MSR-Technik. G. Strohrman, Oldenburg, 1983.

A.1.3 Comparator

NOTE This technique/measure is referenced in tables A.2, A.3, A.4 of IEC 61508-2.

Aim: To detect, as early as possible, (non-simultaneous) failures in an independent processing unit or in the comparator.

Description: Les signaux d'unités de traitement autonomes sont comparés de façon cyclique ou continue par un comparateur matériel. Le comparateur peut lui-même être testé par un appareil externe ou utiliser une technologie d'auto-surveillance automatique. Les différences détectées dans le comportement des processeurs génèrent un message de défaillance.

A.1.4 Voteur majoritaire

NOTE Cette technique/mesure est mentionnée dans les tableaux A.2, A.3 et A.4 de la CEI 61508-2.

But: Détecter et masquer les défaillances dans au moins l'un des n canaux matériels.

Description: Un dispositif de vote majoritaire (2 sur 3, 3 sur 3, ou *m* sur *n*) est utilisé pour détecter et masquer les défaillances. Le voteur peut lui-même être testé par un appareil externe ou utiliser une technologie d'auto-surveillance automatique.

Références:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Anlagensicherung mit Mitteln der MSR-Technik. Praxis der Sicherheitstechnik, Vol. 1, Dechema, 1988.

Sicherung von Anlagen der Verfahrenstechnik mit Mitteln der Mess-, Steuerungs- und Regelungstechnik. VDI/VDE Blatt 1 à 5, 1984 à 1988.

A.1.5 Principe du courant au repos

NOTE Cette technique/mesure est mentionnée dans les tableaux A.2, A.9, A.14 et A.15 de la CEI 61508-2.

But: Assurer la fonction de sécurité lorsque l'alimentation est coupée ou perdue.

Description: La fonction de sécurité est assurée si les contacts sont ouverts et le courant ne passe pas. Par exemple, si les freins sont utilisés pour arrêter un mouvement dangereux d'un moteur, les freins sont desserrés en fermant des contacts dans le système relatif à la sécurité et serrés en ouvrant des contacts dans le système relatif à la sécurité.

Référence: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

A.2 Electroniques

Objectif général: Maîtriser les défaillances dans les composants à semi-conducteurs.

A.2.1 Tests par du matériel redondant

NOTE Cette technique/mesure est mentionnée dans les tableaux A.3, A.16, A.17 et A.19 de la CEI 61508-2.

But: Détecter les défaillances à l'aide de matériel redondant, c'est-à-dire en utilisant du matériel supplémentaire non nécessaire pour l'exécution des fonctions du processus.

Description: Le matériel redondant peut être utilisé pour tester à une fréquence adéquate les fonctions de sécurité spécifiées. Cette approche est normalement nécessaire pour réaliser A.1.1 ou A.2.2.

Référence: DIN V VDE 0801: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principes pour les ordinateurs dans les Systèmes de sécurité), Beuth-Verlag, Berlin, 1990.

Description: The signals of independent processing units are compared cyclically or continuously by a hardware comparator. The comparator may itself be externally tested, or it may use self-monitoring technology. Detected differences in the behaviour of the processors lead to a failure message.

A.1.4 Majority voter

NOTE This technique/measure is referenced in tables A.2, A.3 and A.4 of IEC 61508-2.

Aim: To detect and mask failures in one of at least three hardware channels.

Description: A voting unit using the majority principle (2 out of 3, 3 out of 3, or *m* out of *n*) is used to detect and mask failures. The voter may itself be externally tested, or it may use self-monitoring technology.

References:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Anlagensicherung mit Mitteln der MSR-Technik. Praxis der Sicherheitstechnik, Vol 1, Dechema, 1988.

Sicherung von Anlagen der Verfahrenstechnik mit Mitteln der Mess-, Steuerungs- und Regelungstechnik. VDI/VDE Blatt 1 to 5, 1984 to 1988.

A.1.5 Idle current principle (de-energised to trip)

NOTE This technique/measure is referenced in tables A.2, A.9, A.14 and A.15 of IEC 61508-2.

Aim: To execute the safety function if power is cut or lost.

Description: The safety function is executed if the contacts are open and no current flows. For example, if brakes are used to stop a dangerous movement of a motor, the brakes are opened by closing contacts in the safety-related system and are closed by opening the contacts in the safety-related system.

Reference: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

A.2 Electronic

Global objective: To control failure in solid-state components.

A.2.1 Tests by redundant hardware

NOTE This technique/measure is referenced in tables A.3, A.16, A.17 and A.19 of IEC 61508-2.

Aim: To detect failures using hardware redundancy, i.e. using additional hardware not required to implement the process functions.

Description: Redundant hardware can be used to test at an appropriate frequency the specified safety functions. This approach is normally necessary for realising A.1.1 or A.2.2.

Reference: DIN V VDE 0801: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems), Beuth-Verlag, Berlin, 1990.

A.2.2 Principes dynamiques

NOTE Cette technique/mesure est mentionnée dans le tableau A.3 de la CEI 61508-2.

But: Détecter les défaillances statiques par un traitement dynamique des signaux.

Description: Un changement forcé de signaux normalement statiques (générés à l'intérieur ou à l'extérieur) aide à la détection les défaillances statiques des composants. Cette technique est souvent associée aux composants électromécaniques.

Référence: Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

A.2.3 Port d'accès de test normalisé et architecture de test du type «scrutation aux frontières»

NOTE Cette technique/mesure est mentionnée dans les tableaux A.3, A.16 et A.19 de la CEI 61508-2.

But: Surveiller et observer ce qu'il se passe à chaque broche d'un circuit intégré.

Description: Le test du type scrutation aux frontières est une technique de conception de CI qui accroît la possibilité de tester le CI en résolvant le problème d'accès aux points de test situés à l'intérieur de celui-ci. Dans un CI typique avec scrutation aux frontières, composé d'un noyau logique et de registres tampons d'entrée/sortie, un registre à décalage est placé entre le noyau logique et les registres tampons d'entrée/sortie adjacent à chaque broche du CI. Chaque registre à décalage est situé dans une cellule de scrutation. La cellule de scrutation peut surveiller et observer ce qui se passe à chaque broche d'entrée et de sortie d'un CI via le port d'accès pour les tests normalisés. La surveillance interne du noyau logique du CI est effectuée en isolant le noyau logique des stimuli reçus des composants alentours et en effectuant ensuite un test automatique interne. Ces tests peuvent être utilisés pour détecter les défaillances dans le CI.

Référence: IEEE 1149.1:1990, Standard Test Access Port and Boundary-Scan Architecture.

A.2.4 Matériel à sécurité intégrée

NOTE Cette technique/mesure est mentionnée dans le tableau A.3 de la CEI 61508-2.

But: Mettre un système dans un état sûr si une défaillance survient.

Description: Dans les systèmes câblés, une unité est dite à sécurité intégrée:

– si un ensemble défini d'anomalies induit un état de sécurité, et
– si ces anomalies sont détectées.

EXEMPLE L'ensemble défini d'anomalies pourrait comprendre les anomalies liées à un blocage, à un blocage à l'ouverture, à des courts-circuits à l'intérieur des composants et entre les composants et à des courts-circuits dirigés.

Références:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

A.2.2 Dynamic principles

NOTE This technique/measure is referenced in table A.3 of IEC 61508-2.

Aim: To detect static failures by dynamic signal processing.

Description: A forced change of otherwise static signals (internally or externally generated) helps to detect static failures in components. This technique is often associated with electromechanical components.

Reference: Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

A.2.3 Standard test access port and boundary-scan architecture

NOTE This technique/measure is referenced in tables A.3, A.16 and A.19 of IEC 61508-2.

Aim: To control and observe what happens at each pin of an IC.

Description: Boundary-scan test is an IC design technique which increases the testability of the IC by resolving the problem of how to gain access to the circuit test points within it. In a typical boundary-scan IC, comprised of core logic and input and output buffers, a shift-register stage is placed between the core logic and the input and output buffers adjacent to each IC pin. Each shift-register stage is contained in a boundary-scan cell. The boundary-scan cell can control and observe what happens at each input and output pin of an IC, via the standard test access port. Internal testing of the IC core logic is accomplished by isolating the on-chip core logic from stimuli received from surrounding components, and then performing an internal self-test. These tests can be used to detect failures in the IC.

Reference: IEEE 1149.1:1990, Standard Test Access Port and Boundary-Scan Architecture.

A.2.4 Fail-safe hardware

NOTE This technique/measure is referenced in table A.3 of IEC 61508-2.

Aim: To put a system into a safe state if a failure occurs.

Description: In hard-wired systems, a unit is said to operate in a fail-safe manner if

– a defined set of faults will lead to a safe condition, and

– they are detected.

EXAMPLE The defined set of faults could include stuck-at faults, stuck-open, short circuits within and between components and directed short circuits.

References:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

A.2.5 Redondance surveillée

NOTE Cette technique/mesure est mentionnée dans le tableau A.3 de la CEI 61508-2.

But: Détecter les défaillances, en fournissant plusieurs unités fonctionnelles, en surveillant le comportement de chacune de celles-ci pour en détecter les défaillances et initier une transition vers un état de sécurité si un écart de comportement est détecté.

Description: La fonction de sécurité est assurée par au moins deux canaux matériels. Les sorties de ces canaux sont surveillées et un état de sécurité est initié si une anomalie est détectée (c'est-à-dire si les signaux de sortie de tous les canaux ne sont pas les mêmes).

Références:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

A.2.6 Composants électriques/électroniques avec contrôle automatique

NOTE Cette technique/mesure est mentionnée dans le tableau A.3 de la CEI 61508-2.

But: Détecter les défaillances par des contrôles périodiques des fonctions de sécurité.

Description: Le matériel est testé avant de lancer le processus et est testé de façon répétée à des intervalles réguliers. L'EUC ne continue à fonctionner que si chaque test est couronné de succès.

Références:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

A.2.7 Surveillance du signal analogique

NOTE Cette technique/mesure est mentionnée dans les tableaux A.3 et A.14 de la CEI 61508-2.

But: Améliorer la fiabilité des signaux mesurés.

Description: Toutes les fois qu'il y a un choix, les signaux analogiques sont utilisés de préférence aux états tout ou rien numériques. Par exemple, la disjonction ou les états de sécurité sont représentés par des niveaux de signaux analogiques, généralement avec une surveillance de tolérance du niveau du signal. La technique offre la continuité de la surveillance et un meilleur niveau de confiance dans le transmetteur, réduisant ainsi la fréquence du test périodique de la fonction du transmetteur. Les interfaces externes, par exemple les lignes d'impulsions, nécessitent également un test.

Référence: UKOOA Guidelines for Instrument-Based Systems, UK Offshore Operators Association Limited, December 1995.

A.2.5 Monitored redundancy

NOTE This technique/measure is referenced in table A.3 of IEC 61508-2.

Aim: To detect failure, by providing several functional units, by monitoring the behaviour of each of these to detect failures, and by initiating a transition to a safe condition if any discrepancy in behaviour is detected.

Description: The safety function is executed by at least two hardware channels. The outputs of these channels are monitored and a safe condition is initiated if a fault is detected (i.e. if the output signals from all channels are not identical).

References:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

A.2.6 Electrical/electronic components with automatic check

NOTE This technique/measure is referenced in table A.3 of IEC 61508-2.

Aim: To detect faults by periodic checking of the safety functions.

Description: The hardware is tested before starting the process, and is tested repeatedly at suitable intervals. The EUC continues to operate only if each test is successful.

References:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

A.2.7 Analogue signal monitoring

NOTE This technique/measure is referenced in tables A.3 and A.14 of IEC 61508-2.

Aim: To improve confidence in measured signals.

Description: Wherever there is a choice, analogue signals are used in preference to digital on/off states. For example, trip or safe states are represented by analogue signal levels, usually with signal level tolerance monitoring. The technique provides continuity monitoring and a higher level of confidence in the transmitter, reducing the necessary proof-test frequency of the transmitter sensing function. External interfaces, for example impulse lines, will also require testing.

Reference: UKOOA Guidelines for Instrument-Based Systems, UK Offshore Operators Association Limited, December 1995.

A.2.8 Dévaluation

But: Augmenter la fiabilité des composants matériels.

Description: Les composants matériels sont utilisés à des niveaux dont la conception du système garantit qu'ils sont bien au-dessous des caractéristiques maximales des spécifications. La dévaluation est la pratique qui garantit que, dans toutes les conditions de fonctionnement normales, les composants sont utilisés bien en dessous de leurs niveaux de contrainte maximale.

A.3 Unités de traitement

Objectif général: Reconnaître les défaillances qui sont à l'origine des résultats incorrects dans les unités de traitement.

A.3.1 Autotest logiciel: nombre limité de configurations (un canal)

NOTE Cette technique/mesure est mentionnée dans le tableau A.4 de la CEI 61508-2.

But: Détecter le plus tôt possible les défaillances de l'unité de traitement.

Description: Le matériel est construit avec les techniques standard qui ne prennent en compte aucune exigence spéciale relative à la sécurité. La détection de défaillances est entièrement effectuée par des fonctions de logiciel supplémentaires qui effectuent des autotests en utilisant au moins deux configurations de données complémentaires (par exemple 55hex et AAhex).

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.3.2 Autotest logiciel: bit glissant (un canal)

NOTE Cette technique/mesure est mentionnée dans le tableau A.4 de la CEI 61508-2.

But: Détecter le plus tôt possible les défaillances de la mémoire physique (par exemple les registres) et le décodeur d'instructions de l'unité de traitement.

Description: La détection de défaillances est entièrement réalisée par des fonctions de logiciel supplémentaires qui effectuent des autotests en utilisant une configuration de données (par exemple la configuration de bit glissant) qui teste le moyen de stockage physique (registres de données et d'adresse) et le décodeur d'instructions. Toutefois, la couverture du diagnostic n'est que de 90 %.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.3.3 Autotest pris en charge par le matériel (un canal)

NOTE Cette technique/mesure est mentionnée dans le tableau A.4 de la CEI 61508-2.

But: Détecter le plus tôt possible les défaillances de l'unité de traitement avec un matériel spécial qui augmente la vitesse et élargit le champ de la détection de défaillances.

Description: Des installations de matériel spécial supplémentaire prennent en charge les fonctions d'autotest pour la détection de défaillances. Cela pourrait être par exemple une unité de matériel qui surveille de façon cyclique la sortie d'une certaine configuration de bit selon le principe du chien de garde.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.2.8 De-rating

Aim: To increase the reliability of hardware components.

Description: Hardware components are operated at levels which are guaranteed by the design of the system to be well below the maximum specification ratings. De-rating is the practice of ensuring that under all normal operating circumstances, components are operated well below their maximum stress levels.

A.3 Processing units

Global objective: To recognise failures which lead to incorrect results in processing units.

A.3.1 Self-test by software: limited number of patterns (one-channel)

NOTE This technique/measure is referenced in table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the processing unit.

Description: The hardware is built using standard techniques which do not take any special safety requirements into account. The failure detection is realised entirely by additional software functions which perform self-tests using at least two complementary data patterns (for example 55hex and AAhex).

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.3.2 Self-test by software: walking bit (one-channel)

NOTE This technique/measure is referenced in table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the physical storage (for example registers) and instruction decoder of the processing unit.

Description: The failure detection is realised entirely by additional software functions which perform self-tests using a data pattern (for example walking-bit pattern) which tests the physical storage (data and address registers) and the instruction decoder. However, the diagnostic coverage is only 90 %.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.3.3 Self-test supported by hardware (one-channel)

NOTE This technique/measure is referenced in table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the processing unit, using special hardware that increases the speed and extends the scope of failure detection.

Description: Additional special hardware facilities support self-test functions to detect failure. For example, this could be a hardware unit which cyclically monitors the output of a certain bit pattern according to the watch-dog principle.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.3.4 Traitement codé (un canal)

NOTE Cette technique/mesure est mentionnée dans le tableau A.4 de la CEI 61508-2.

But: Détecter le plus tôt possible les défaillances de l'unité de traitement.

Description: Les unités de traitement peuvent être conçues avec des techniques de circuits spéciales reconnaissant ou corrigeant les défaillances. A ce jour, ces techniques ne sont appliquées qu'à des circuits relativement simples et ne sont pas très répandues; il convient toutefois de ne pas exclure des développements futurs.

Références:

The Coded Microprocessor Certification. P. Ozello, Proc. SAFECOMP '92, 185-190, 1992.

Vital Coded Microprocessor Principles and Application for Various Transit Systems. P. Forin, IFAC Control Computers Communications in Transportation, 79-84, 1989.

Le Processeur Codé: un nouveau concept appliqué à la sécurité des systèmes de transports. Gabriel, Martin, Wartski, Revue Générale des chemins de fer, n° 6, juin 1990.

A.3.5 Comparaison réciproque par logiciel

NOTE Cette technique/mesure est mentionnée dans le tableau A.4 de la CEI 61508-2.

But: Détecter le plus tôt possible les défaillances de l'unité de traitement par comparaison de logiciel dynamique.

Description: Deux unités de traitement se transmettent réciproquement les données (y compris les résultats, les résultats intermédiaires et les données de test). Une comparaison des données est effectuée à l'aide d'un logiciel dans chaque unité et les différences détectées génèrent un message de défaillance.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.4 Gamme de mémoire invariable

Objectif général: La détection des modifications des informations dans la mémoire invariable.

A.4.1 Redondance multi-bits à sauvegarde de mot (par exemple, surveillance de la ROM avec un code de Hamming modifié)

NOTE Voir également A.5.6 et C.3.2. Cette technique/mesure est mentionnée dans le tableau A.5 de la CEI 61508-2.

But: Détecter toutes les défaillances d'un seul bit, toutes les défaillances de deux bits, quelques défaillances de trois bits et quelques défaillances de tous les bits dans un mot de 16 bits.

Description: Chaque mot de mémoire est allongé par plusieurs bits redondants pour produire un code de Hamming modifié avec une distance de Hamming d'au moins 4. Chaque fois qu'un mot est lu, la vérification des bits redondants peut déterminer si une altération a eu lieu ou non. Si une différence est détectée, un message de défaillance est produit. La procédure peut aussi servir à détecter les défaillances d'adressage en calculant les bits redondants pour la concaténation du mot porteur d'information et son adresse.

A.3.4 Coded processing (one-channel)

NOTE This technique/measure is referenced in table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the processing unit.

Description: Processing units can be designed with special failure-recognising or failure-correcting circuit techniques. So far, these techniques have been applied only to relatively simple circuits and are not widespread; however, future developments should not be excluded.

References:

The Coded Microprocessor Certification. P. Ozello, Proc. SAFECOMP '92, 185-190, 1992.

Vital Coded Microprocessor Principles and Application for Various Transit Systems. P. Forin, IFAC Control Computers Communications in Transportation, 79-84, 1989.

Le Processeur Codé: un nouveau concept appliqué à la sécurité des systèmes de transports. Gabriel, Martin, Wartski, Revue Générale des chemins de fer, No. 6, June 1990.

A.3.5 Reciprocal comparison by software

NOTE This technique/measure is referenced in table A.4 of IEC 61508-2.

Aim: To detect, as early as possible, failures in the processing unit, by dynamic software comparison.

Description: Two processing units exchange data (including results, intermediate results and test data) reciprocally. A comparison of the data is carried out using software in each unit and detected differences lead to a failure message.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.4 Invariable memory ranges

Global objective: The detection of information modifications in the invariable memory.

A.4.1 Word-saving multi-bit redundancy (for example ROM monitoring with a modified Hamming code)

NOTE See also A.5.6 and C.3.2. This technique/measure is referenced in table A.5 of IEC 61508-2.

Aim: To detect all single-bit failures, all two-bit failures, some three-bit failures, and some all-bit failures in a 16-bit word.

Description: Every word of memory is extended by several redundant bits to produce a modified Hamming code with a Hamming distance of at least 4. Every time a word is read, checking of the redundant bits can determine whether or not a corruption has taken place. If a difference is found, a failure message is produced. The procedure can also be used to detect addressing failures, by calculating the redundant bits for the concatenation of the data word and its address.

Références:

Error detecting and error correcting codes. R. W. Hamming, The Bell System Technical Journal 29 (2), 147-160, 1950.

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967.

A.4.2 Somme de contrôle modifiée

NOTE Cette technique/mesure est mentionnée dans le tableau A.5 de la CEI 61508-2.

But: Détecter toutes les défaillances des bits impairs, soit environ 50 % de toutes les défaillances de bits possibles.

Description: Une somme de contrôle est créée par un algorithme adéquat qui utilise tous les mots dans un bloc de mémoire. La somme de contrôle peut être stockée comme un mot supplémentaire dans la ROM ou alors un mot supplémentaire peut être ajouté au bloc de mémoire pour être sûr que l'algorithme de la somme de contrôle produise une valeur prédéterminée. Dans un test de mémoire ultérieur, une somme de contrôle est recréée avec le même algorithme et le résultat est comparé avec la valeur stockée ou définie. Si une différence est détectée, un message de défaillance est produit.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.4.3 Signature d'un seul mot (8 bits)

NOTE Cette technique/mesure est mentionnée dans le tableau A.5 de la CEI 61508-2.

But: Détecter toutes les défaillances d'un bit et toutes les défaillances multi-bits dans un mot, ainsi qu'environ 99,6 % de toutes les défaillances de bits possibles.

Description: Le contenu d'un bloc de mémoire est comprimé (soit avec du matériel soit avec un logiciel) à l'aide d'un algorithme de contrôle cyclique par redondance (CRC) en un seul mot mémoire. Un algorithme CRC typique traite tout le contenu du bloc comme un flux de données en série par octet ou par bit sur lequel une division polynomiale continue est effectuée à l'aide d'un générateur polynomial. Le reste de la division représente le contenu comprimé de la mémoire – c'est la «signature» de la mémoire – et est stocké. La signature est recalculée lors de tests ultérieurs et comparée avec celle déjà stockée. Un message de défaillance est produit s'il y a une différence.

Références:

Calculating an error checking character in software. S. Vasa, Computer Design, 5, 1976.

Berechnung von Fehlererkennungswahrscheinlichkeiten bei Signaturregistern. D. Leisengang, Elektronische Rechenanlagen 24, H. 2, S. 55-61, 1982.

A.4.4 Signature d'un mot double (16 bits)

NOTE Cette technique/mesure est mentionnée dans le tableau A.5 de la CEI 61508-2.

But: Détecter toutes les défaillances d'un bit et toutes les défaillances multi-bits dans un mot, ainsi qu'environ 99,998 % de toutes les défaillances de bits possibles.

Description: Cette procédure calcule une signature à l'aide d'un algorithme de contrôle cyclique par redondance (CRC), mais la valeur obtenue a une taille d'au moins deux mots. La signature allongée est stockée, recalculée et comparée comme pour le cas d'un seul mot. Un message de défaillance est produit s'il y a une différence entre les signatures stockées et recalculées.

References:

Error detecting and error correcting codes. R. W. Hamming, The Bell System Technical Journal 29 (2), 147-160, 1950.

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967.

A.4.2 Modified checksum

NOTE This technique/measure is referenced in table A.5 of IEC 61508-2.

Aim: To detect all odd-bit failures, i.e. approximately 50 % of all possible bit failures.

Description: A checksum is created by a suitable algorithm which uses all the words in a block of memory. The checksum may be stored as an additional word in ROM, or an additional word may be added to the memory block to ensure that the checksum algorithm produces a predetermined value. In a later memory test, a checksum is created again using the same algorithm, and the result is compared with the stored or defined value. If a difference is found, a failure message is produced.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.4.3 Signature of one word (8-bit)

NOTE This technique/measure is referenced in table A.5 of IEC 61508-2.

Aim: To detect all one-bit failures and all multi-bit failures within a word, as well as approximately 99,6 % of all possible bit failures.

Description: The contents of a memory block is compressed (using either hardware or software) using a cyclic redundancy check (CRC) algorithm into one memory word. A typical CRC algorithm treats the whole contents of the block as byte-serial or bit-serial data flow, on which a continued polynomial division is carried out using a polynomial generator. The remainder of the division represents the compressed memory contents – it is the "signature" of the memory – and is stored. The signature is computed once again in later tests and compared with one already stored. A failure message is produced if there is a difference.

References:

Calculating an error checking character in software. S. Vasa, Computer Design, 5, 1976.

Berechnung von Fehlererkennungswahrscheinlichkeiten bei Signaturregistern. D. Leisengang, Elektronische Rechenanlagen 24, H. 2, S. 55-61, 1982.

A.4.4 Signature of a double word (16-bit)

NOTE This technique/measure is referenced in table A.5 of IEC 61508-2.

Aim: To detect all one-bit failures and all multi-bit failures within a word, as well as approximately 99,998 % of all possible bit failures.

Description: This procedure calculates a signature using a cyclic redundancy check (CRC) algorithm, but the resulting value is at least two words in size. The extended signature is stored, recalculated and compared as in the single-word case. A failure message is produced if there is a difference between the stored and recalculated signatures.

Référence:

Signaturanalyse in der Datenverarbeitung. D. Leisengang, M. Wagner, Elektronik 32, H. 21, S. 67-72, 1983.

Signaturregister für selbsttestende ICs. B. Könemann, J. Mucha, G. Zwiehoff, Größtintegration/ NTG-Fachtagung Baden-Baden, S. 109-112, April 1977.

A.4.5 Réplication du bloc (par exemple, double ROM avec comparaison par matériel ou logiciel)

NOTE Cette technique/mesure est mentionnée dans le tableau A.5 de la CEI 61508-2.

But: Détecter toutes les défaillances de bit.

Description: L'espace adresse est dupliqué en deux mémoires. La première mémoire est normalement activée. La deuxième mémoire contient les mêmes informations et est accessible en parallèle avec la première. Les sorties sont comparées et un message de défaillance est produit si une différence est détectée. Pour détecter certains types d'erreurs de bit, les données doivent être stockées inversées dans l'une des deux mémoires et inversées à nouveau lors de la lecture.

Références:

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

Computers can now perform vital safety functions safely. Otto Berg von Linde, Railway Gazette International, Vol. 135, n° 11, 1979.

A.5 Gammes de mémoire variable

Objectif général: Détecter les défaillances lors de l'adressage, de l'écriture, du stockage et de la lecture.

A.5.1 Test RAM «échiquier» ou «défilement»

NOTE Cette technique/mesure est mentionnée dans le tableau A.6 de la CEI 61508-2.

But: Détecter les défaillances de bit à prédominance statique.

Description: Une structure de type échiquier de 0 et de 1 est inscrite dans les cellules d'une mémoire binaire. Les cellules sont alors contrôlées par paires pour s'assurer que les contenus sont identiques et corrects. L'adresse de la première cellule d'une paire est variable et l'adresse de la deuxième cellule est formée par l'inversion bit par bit de la première adresse. Dans un premier temps, la plage d'adresses de la mémoire est déroulée vers les adresses supérieures à partir de l'adresse variable et, dans un deuxième temps, elle est déroulée vers les adresses inférieures. Les deux déroulements sont ensuite répétés avec une distribution préalable inversée. Un message de défaillance est produit s'il y a une différence.

Dans le «défilement» d'un test RAM, les cellules d'une mémoire binaire sont initialisées par un train binaire uniforme. Dans un premier déroulement, les cellules sont contrôlées vers le haut: le contenu de chaque cellule est contrôlé et les contenus sont ensuite inversés. La base, qui est créée lors du premier déroulement, est traitée dans un deuxième déroulement vers le bas et de la même façon. Les deux déroulements sont ensuite répétés avec une distribution préalable inversée dans un troisième ou quatrième déroulement. Un message de défaillance est produit s'il y a une différence.

References:

Signaturanalyse in der Datenverarbeitung. D. Leisengang, M. Wagner, Elektronik 32, H. 21, S. 67-72, 1983.

Signaturregister für selbsttestende ICs. B. Könemann, J. Mucha, G. Zwiehoff, Größtintegration/ NTG-Fachtagung Baden-Baden, S. 109-112, April 1977.

A.4.5 Block replication (for example double ROM with hardware or software comparison)

NOTE This technique/measure is referenced in table A.5 of IEC 61508-2.

Aim: To detect all bit failures.

Description: The address space is duplicated in two memories. The first memory is operated in the normal manner. The second memory contains the same information and is accessed in parallel to the first. The outputs are compared and a failure message is produced if a difference is detected. In order to detect certain kinds of bit errors, the data must be stored inversely in one of the two memories and inverted once again when read.

References:

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

Computers can now perform vital safety functions safely. Otto Berg von Linde, Railway Gazette International, Vol. 135, No. 11, 1979.

A.5 Variable memory ranges

Global objective: Detecting failures during addressing, writing, storing and reading.

A.5.1 RAM test "checkerboard" or "march"

NOTE This technique/measure is referenced in table A.6 of IEC 61508-2.

Aim: To detect predominantly static bit failures.

Description: A checker-board type pattern of 0s and 1s is written into the cells of a bit-oriented memory. The cells are then inspected in pairs to ensure that the contents are the same and correct. The address of the first cell of such a pair is variable and the address of the second cell of the pair is formed by inverting bitwise the first address. In the first run, the address range of the memory is run towards higher addresses from the variable address, and in a second run towards lower addresses. Both runs are then repeated with an inverted pre-assignment. A failure message is produced if any difference occurs.

In a RAM test "march" the cells of a bit-oriented memory are initialised by a uniform bit stream. In the first run, the cells are inspected in ascending order: each cell is checked for the correct contents and its contents are inverted. The background, which is created in the first run, is treated in a second run in descending order and in the same manner. Both first runs are repeated with an inverted pre-assignment in a third or fourth run. A failure message is produced if a difference occurs.

Références:

Memory testing. W. G. Fee, LSI Testing (Tutorial at the COMPCON 77 in San Francisco), IEEE Computer Society, W. G. Fee (ed.), 81-88, 1978.

Memory testing. P. Rosenfield, Electronics and Power, H. 1, P. 26-31, 1979.

Halbleiterspeicher-Testfolgen. Th. John, E. Schaefer, Elektronikpraxis, H. 6, 18-26 et H. 7, 10-14, 1980.

A.5.2 Test RAM «walkpath»

NOTE Cette technique/mesure est mentionnée dans le tableau A.6 de la CEI 61508-2.

But: Détecter les défaillances binaires statiques et dynamiques et les interactions entre cellules mémoire.

Description: La plage mémoire à tester est initialisée par un train binaire uniforme. La première cellule est ensuite inversée et la zone mémoire restante est contrôlée pour s'assurer que son contenu est correct. Après cela, la première cellule est inversée à nouveau pour retrouver sa valeur initiale et toute la procédure est répétée pour la cellule suivante. Un deuxième déroulement du «modèle du bit errant» est effectué avec une distribution préalable inverse du contenu. Un message de défaillance est produit s'il y a une différence.

Références:

Memory testing. W. G. Fee, LSI Testing (Tutorial at the COMPCON 77 in San Francisco), IEEE Computer Society, W. G. Fee (ed.), 81-88, 1978.

Techniques for testing the microprocessor family. W. Barraclough, A. Chiang, W. Sohl, Proceedings of the IEEE 64 (6), 943-950, 1976.

A.5.3 Test RAM «galpat» ou «galpat transparent»

NOTE Cette technique/mesure est mentionnée dans le tableau A.6 de la CEI 61508-2.

But: Détecter les défaillances binaires statiques et une grande partie des couplages dynamiques.

Description: Dans le test RAM «galpat», la plage de mémoires choisie est d'abord initialisée uniformément (c'est-à-dire tout en 0 ou tout en 1). La première cellule mémoire à tester est ensuite inversée et toutes les cellules restantes sont contrôlées pour s'assurer que leur contenu est correct. Après chaque accès en lecture à l'une des cellules restantes, la cellule inversée est aussi contrôlée. Cette procédure est répétée pour chaque cellule dans la plage de mémoires choisie. Un deuxième déroulement est effectué avec l'initialisation opposée. Toute différence génère un message de défaillance.

Le test «galpat transparent» est une variante de la procédure ci-dessus: au lieu d'initialiser toutes les cellules dans la plage de mémoires choisie, les contenus existants demeurent inchangés et les signatures sont utilisées pour comparer les contenus des ensembles de cellules. La première cellule à tester dans la plage choisie est sélectionnée et la signature S1 de toutes les cellules restantes dans la plage est calculée et enregistrée. La cellule à tester est ensuite inversée et la signature S2 de toutes les cellules restantes est recalculée. (Après chaque accès en lecture à l'une des cellules restantes, la cellule inversée est aussi contrôlée.) S2 est comparée avec S1 et toute différence génère un message de défaillance. La cellule testée est à nouveau inversée pour rétablir le contenu de départ et la signature S3 de toutes les cellules restantes est recalculée et comparée avec S1. Toute différence produit un message de défaillance. Toutes les cellules mémoire dans la plage choisie sont contrôlées de la même façon.

References:

Memory testing. W. G. Fee, LSI Testing (Tutorial at the COMPCON 77 in San Francisco), IEEE Computer Society, W. G. Fee (ed.), 81-88, 1978.

Memory testing. P. Rosenfield, Electronics and Power, H. 1, P. 26-31, 1979.

Halbleiterspeicher-Testfolgen. Th. John, E. Schaefer, Elektronikpraxis, H. 6, 18-26 and H. 7, 10-14, 1980.

A.5.2 RAM test "walkpath"

NOTE This technique/measure is referenced in table A.6 of IEC 61508-2.

Aim: To detect static and dynamic bit failures, and cross-talk between memory cells.

Description: The memory range to be tested is initialised by a uniform bit stream. The first cell is then inverted and the remaining memory area is inspected to ensure that the background is correct. After this, the first cell is re-inverted to return it to its original value, and the whole procedure is repeated for the next cell. A second run of the "wandering bit model" is carried out with an inverse background pre-assignment. A failure message is produced if a difference occurs.

References:

Memory testing. W. G. Fee, LSI Testing (Tutorial at the COMPCON 77 in San Francisco), IEEE Computer Society, W. G. Fee (ed.), 81-88, 1978.

Techniques for testing the microprocessor family. W. Barraclough, A. Chiang, W. Sohl, Proceedings of the IEEE 64 (6), 943-950, 1976.

A.5.3 RAM test "galpat" or "transparent galpat"

NOTE This technique/measure is referenced in table A.6 of IEC 61508-2.

Aim: To detect static bit failures and a large proportion of dynamic couplings.

Description: In the RAM test "galpat", the chosen range of memory is first initialised uniformly (i.e. all 0s or all 1s). The first memory cell to be tested is then inverted and all the remaining cells are inspected to ensure that their contents are correct. After every read access to one of the remaining cells, the inverted cell is also checked. This procedure is repeated for each cell in the chosen memory range. A second run is carried out with the opposite initialisation. Any difference produces a failure message.

The "transparent galpat" test is a variation on the above procedure: instead of initialising all cells in the chosen memory range, the existing contents are left unchanged and signatures are used to compare the contents of sets of cells. The first cell to be tested in the chosen range is selected, and the signature S1 of all remaining cells in the range is calculated and stored. The cell to be tested is then inverted and the signature S2 of all the remaining cells is recalculated. (After every read access to one of the remaining cells, the inverted cell is also checked.) S2 is compared with S1, and any difference produces a failure message. The cell under test is re-inverted to re-establish the original contents, and the signature S3 of all the remaining cells is recalculated and compared with S1. Any difference produces a failure message. All memory cells in the chosen range are tested in the same manner.

Références:

Entwurf von Selbsttestprogrammen für Mikrocomputer. E. Maehle, Microcomputing. Berichte der Tagung III/79 des German Chapter of the ACM, W. Remmele, H. Schecher, (ed.), Stuttgart, Teubner, 204-216, 1979.

Periodischer Selbsttest einer mikroprocessorgesteuerten Sicherheitsschaltung. U. Stinnesbek, Diplomarbeit am Institut für theoretische Elektrotechnik der RWTH Aachen 1980.

A.5.4 Test RAM «Abraham»

NOTE Cette technique/mesure est mentionnée dans le tableau A.6 de la CEI 61508-2.

But: Détecter toutes les défaillances de collage et de couplage entre les cellules mémoire.

Description: La proportion d'anomalies détectées dépasse celle du test RAM «galpat». Le nombre d'opérations requis pour effectuer le test de toute la mémoire est d'environ 30 *n*, *n* étant le nombre de cellules dans la mémoire. L'utilisation du test peut être rendue transparente pendant le cycle d'exploitation en découpant la mémoire et en testant chaque découpage dans des segments temporels différents.

Référence: Efficient Algorithms for Testing Semiconductor Random-Access Memories. R. Nair, S. M. Thatte, J. A. Abraham, IEEE Trans. Comput. C-27 (6), 572-576, 1978.

A.5.5 Redondance à un bit (par exemple, surveillance de la RAM avec un bit de parité)

NOTE Cette technique/mesure est mentionnée dans le tableau A.6 de la CEI 61508-2.

But: Détecter 50 % de toutes les défaillances binaires possibles dans la plage de mémoire testée.

Description: Chaque mot de la mémoire est allongé d'un bit (le bit de parité) qui complète chaque mot pour avoir un nombre pair ou impair de niveaux logiques '1'. La parité du mot porteur d'information est contrôlée chaque fois qu'il est lu. Si un mauvais nombre d'1 est trouvé, un message de défaillance est produit. Le choix d'une parité paire ou impaire doit être tel que le plus défavorable des mots zéro (que des 0) et un (que des 1) en cas de défaillance ne soit pas un code valide. La parité peut aussi servir à détecter les défaillances d'adressage quand la parité est calculée pour la concaténation du mot porteur d'information et de son adresse.

Référence: Integrierte Digitalbausteine. K. Reiß, H. Liedl, W. Spichall, Berlin, 1970.

A.5.6 Surveillance de la RAM avec un code de Hamming modifié, ou détection des défaillances concernant les données par des codes de détection d'erreurs (EDC)

NOTE Voir également A.4.1 et C.3.2. Cette technique/mesure est mentionnée dans le tableau A.6 de la CEI 61508-2.

But: Détecter toutes les défaillances binaires impaires, toutes les défaillances de deux bits, quelques défaillances de trois bits et multi-bits.

Description: Chaque mot de la mémoire est rallongé par plusieurs bits redondants pour obtenir un code de Hamming modifié avec une distance de Hamming d'au moins quatre. Chaque fois qu'un mot est lu, on peut déterminer si une altération a eu lieu en contrôlant les bits redondants. S'il y a une différence, un message de défaillance est produit. La procédure peut aussi servir à détecter une défaillance d'adressage quand les bits redondants sont calculés pour la concaténation du mot porteur d'information et de son adresse.

References:

Entwurf von Selbsttestprogrammen für Mikrocomputer. E. Maehle, Microcomputing. Berichte der Tagung III/79 des German Chapter of the ACM, W. Remmele, H. Schecher, (ed.), Stuttgart, Teubner, 204-216, 1979.

Periodischer Selbsttest einer mikroprocessorgesteuerten Sicherheitsschaltung. U. Stinnesbek, Diplomarbeit am Institut für theoretische Elektrotechnik der RWTH Aachen 1980.

A.5.4 RAM test "Abraham"

NOTE This technique/measure is referenced in table A.6 of IEC 61508-2.

Aim: To detect all stuck-at and coupling failures between memory cells.

Description: The proportion of faults detected exceeds that of the RAM test "galpat". The number of operations required to perform the entire memory test is about 30 *n*, where *n* is the number of cells in the memory. The test can be made transparent for use during the operating cycle by partitioning the memory and testing each partition in different time segments.

Reference: Efficient Algorithms for Testing Semiconductor Random-Access Memories. R. Nair, S. M. Thatte, J. A. Abraham, IEEE Trans. Comput. C-27 (6), 572-576, 1978.

A.5.5 One-bit redundancy (for example RAM monitoring with a parity bit)

NOTE This technique/measure is referenced in table A.6 of IEC 61508-2.

Aim: To detect 50 % of all possible bit failures in the memory range tested.

Description: Every word of the memory is extended by one bit (the parity bit) which completes each word to an even or odd number of logical 1s. The parity of the data word is checked each time it is read. If the wrong number of 1s is found, a failure message is produced. The choice of even or odd parity should be made such that, whichever of the zero word (nothing but 0s) and the one word (nothing but 1s) is the more unfavourable in the event of a failure, then that word is not a valid code. Parity can also be used to detect addressing failures, when the parity is calculated for the concatenation of the data word and its address.

Reference: Integrierte Digitalbausteine. K. Reiß, H. Liedl, W. Spichall, Berlin, 1970.

A.5.6 RAM monitoring with a modified Hamming code, or detection of data failures with error-detection-correction codes (EDC)

NOTE See also A.4.1 and C.3.2. This technique/measure is referenced in table A.6 of IEC 61508-2.

Aim: To detect all odd-bit failures, all two-bit failures, some three-bit and some multi-bit failures.

Description: Every word of the memory is extended by several redundant bits to produce a modified Hamming code with a Hamming distance of at least 4. Every time a word is read, one can determine whether a corruption has taken place by checking the redundant bits. If a difference is found, a failure message is produced. The procedure can also be used to detect addressing failure, when the redundant bits are calculated for the concatenation of the data word and its address.

Référence:

Error detecting and error correcting codes. R. W. Hamming, The Bell System Technical Journal 29 (2), 147 160, 1950.

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967.

A.5.7 Double RAM avec comparaison matérielle ou logicielle et test de lecture/écriture

NOTE Cette technique/mesure est mentionnée dans le tableau A.6 de la CEI 61508-2.

But: Détecter toutes les défaillances binaires.

Description: L'espace adresse est dupliqué en deux mémoires. La première mémoire est normalement activée. La deuxième mémoire contient les mêmes informations et on y accède en parallèle à la première. Les sorties sont comparées et un message de défaillance est produit si une différence est détectée. Pour détecter certains types d'erreurs binaires, les données doivent être stockées inversées dans l'une des deux mémoires et inversées à nouveau lors de la lecture.

Références:

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

Computers can now perform vital safety functions safely. Otto Berg von Linde, Railway Gazette International, Vol. 135, n° 11, 1979.

A.6 Unités E/S et interfaces (communication externe)

Objectif général: Détecter les défaillances des unités d'entrée et de sortie (numérique, analogique, en série ou parallèle) et prévenir l'envoi de sorties inadmissibles au processus.

A.6.1 Trame de test

NOTE Cette technique/mesure est mentionnée dans les tableaux A.7, A.14 et A.15 de la CEI 61508-2.

But: Détecter les défaillances statiques (défaillances de collage) et les interférences.

Description: Il s'agit d'un test cyclique indépendant du flux de données des unités d'entrée et de sortie. Il utilise une trame de test définie pour comparer les observations avec les valeurs prévues correspondantes. Les informations concernant la trame, la réception de la trame et l'évaluation de la trame de test doivent toutes être indépendantes les unes des autres. Il convient que l'EUC ne soit pas influencé de façon inadmissible par la trame de test.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88383-315-9.

A.6.2 Protection par code

NOTE Cette technique/mesure est mentionnée dans les tableaux A.7, A.16, A.17 et A.19 de la CEI 61508-2.

But: Détecter les défaillances systématiques et les défaillances aléatoires du matériel dans le flux de données d'entrée/de sortie.

References:

Error detecting and error correcting codes. R. W. Hamming, The Bell System Technical Journal 29 (2), 147-160, 1950.

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967.

A.5.7 Double RAM with hardware or software comparison and read/write test

NOTE This technique/measure is referenced in table A.6 of IEC 61508-2.

Aim: To detect all bit failures.

Description: The address space is duplicated in two memories. The first memory is operated in the normal manner. The second memory contains the same information and is accessed in parallel to the first. The outputs are compared and a failure message is produced if a difference is detected. In order to detect certain kinds of bit errors, the data must be stored inversely in one of the two memories and inverted once again when read.

References:

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

Computers can now perform vital safety functions safely. Otto Berg von Linde, Railway Gazette International, Vol. 135, No. 11, 1979.

A.6 I/O-units and interfaces (external communication)

Global objective: To detect failures in input and output units (digital, analogue, serial or parallel) and to prevent the sending of inadmissible outputs to the process.

A.6.1 Test pattern

NOTE This technique/measure is referenced in tables A.7, A.14 and A.15 of IEC 61508-2.

Aim: To detect static failures (stuck-at failures) and cross-talk.

Description: This is a dataflow-independent cyclical test of input and output units. It uses a defined test pattern to compare observations with the corresponding expected values. The test pattern information, the test pattern reception, and test pattern evaluation must all be independent of each other. The EUC should not be inadmissibly influenced by the test pattern.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.6.2 Code protection

NOTE This technique/measure is referenced in tables A.7, A.16, A.17 and A.19 of IEC 61508-2.

Aim: To detect random hardware and systematic failures in the input/output dataflow.

Description: Cette procédure protège les informations d'entrée et de sortie contre les défaillances systématiques et les défaillances aléatoires du matériel. La protection par code permet une détection des défaillances fonction du flux de données dans les unités d'entrée et de sortie basée sur la redondance des informations et/ou la redondance temporelle. Typiquement, l'information redondante se superpose aux données d'entrée et de sortie. Il en résulte alors un moyen de surveillance du fonctionnement correct des circuits d'entrée ou de sortie. De nombreuses techniques sont possibles, par exemple une fréquence porteuse peut être superposée au signal de sortie d'un capteur. L'unité de traitement logique peut alors vérifier la présence de la fréquence porteuse. Un code binaire redondant peut être ajouté sur un canal de sortie afin de permettre la surveillance de la validité du passage d'un signal entre l'unité logique et l'actionneur final.

Référence: Standard input/output tests and monitoring procedures – Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.6.3 Sortie parallèle multi-canaux

NOTE Cette technique/mesure est mentionnée dans le tableau A.7 de la CEI 61508-2.

But: Détecter les défaillances aléatoires du matériel (défaillances dues à un blocage), les défaillances dues à des influences externes, les défaillances de synchronisation, les défaillances d'adressage, les défaillances dues à une dérive et les défaillances transitoires.

Description: Une sortie parallèle à plusieurs canaux dépendant du flux de données, avec des sorties indépendantes, détecte les défaillances aléatoires du matériel. La détection des défaillances est effectuée par des comparateurs externes. S'il y a une défaillance, l'EUC est arrêté directement. Cette mesure n'est efficace que si le flux de données change pendant l'intervalle du test de diagnostic.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.6.4 Sorties surveillées

NOTE Cette technique/mesure est mentionnée dans le tableau A.7 de la CEI 61508-2.

But: Détecter les défaillances individuelles, les défaillances dues à des influences externes, les défaillances de synchronisation, les défaillances d'adressage, les défaillances dues à une dérive (pour les signaux analogiques) et les défaillances transitoires.

Description: Une comparaison dépendante du flux de données des sorties, avec des entrées indépendantes, garantit la conformité avec une plage de tolérances définie (temps, valeur). Une défaillance détectée ne peut pas toujours être mise en relation avec la sortie défectueuse. Cette mesure n'est efficace que si le flux de données change pendant l'intervalle du test de diagnostic.

Références:

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

MSR-Schutzeinrichtungen. Anforderungen und Massnahmen zur gesicherten Funktion. DIN V 19251, February 1995.

Description: This procedure protects the input and output information from both systematic and random hardware failures. Code protection provides dataflow-dependent failure detection of the input and output units, based on information redundancy and/or time redundancy. Typically, redundant information is superimposed on input and/or output data. This then provides a means to monitor the correct operation of the input or output circuits. Many techniques are possible, for example a carrier frequency signal may be superimposed on the output signal of a sensor. The logic unit may then check for the presence of the carrier frequency or redundant code bits may be added to an output channel to allow the monitoring of the validity of a signal passing between the logic unit and final actuator.

Reference: Standard input/output tests and monitoring procedures – Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.6.3 Multi-channel parallel output

NOTE This technique/measure is referenced in table A.7 of IEC 61508-2.

Aim: To detect random hardware failures (stuck-at failures), failures caused by external influences, timing failures, addressing failures, drift failures and transient failures.

Description: This is a dataflow-dependent multi-channel parallel output with independent outputs for the detection of random hardware failures. Failure detection is carried out via external comparators. If a failure occurs, the EUC is switched off directly. This measure is only effective if the dataflow changes during the diagnostic test interval.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.6.4 Monitored outputs

NOTE This technique/measure is referenced in table A.7 of IEC 61508-2.

Aim: To detect individual failures, failures caused by external influences, timing failures, addressing failures, drift failures (for analogue signals) and transient failures.

Description: This is a dataflow-dependent comparison of outputs with independent inputs to ensure compliance with a defined tolerance range (time, value). A detected failure cannot always be related to the defective output. This measure is only effective if the dataflow changes during the diagnostic test interval.

References:

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

MSR-Schutzeinrichtungen. Anforderungen und Massnahmen zur gesicherten Funktion. DIN V 19251, February 1995.

A.6.5 Comparaison/vote majoritaire sur les entrées

NOTE Cette technique/mesure est mentionnée dans les tableaux A.7 et A.14 de la CEI 61508-2.

But: Détecter les défaillances individuelles, les défaillances dues à des influences externes, les défaillances de synchronisation, les défaillances d'adressage, les défaillances dues à une dérive (pour les signaux analogiques) et les défaillances transitoires.

Description: Une comparaison dépendant du flux de données d'entrées indépendantes garantit la conformité avec une plage de tolérances définie (temps, valeur). Il y aura des redondances d'1 parmi 2, 2 parmi 3 ou plus. Cette mesure n'est efficace que si le flux de données change pendant l'intervalle du test de diagnostic.

Références:

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

MSR-Schutzeinrichtungen. Anforderungen und Massnahmen zur gesicherten Funktion. DIN V 19251, February 1995.

A.7 Chemins de données (communication interne)

Objectif général: Détecter les défaillances dues à un défaut dans le transfert d'informations.

A.7.1 Redondance matérielle sur un bit

NOTE Cette technique/mesure est mentionnée dans le tableau A.8 de la CEI 61508-2.

But: Détecter toutes les défaillances binaires impaires, soit 50 % de toutes les défaillances binaires possibles dans le train de données.

Description: Le bus est complété par une ligne (bit) et cette ligne supplémentaire (bit) est utilisée pour détecter les défaillances par un contrôle de parité.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.2 Redondance matérielle sur plusieurs bits

NOTE Cette technique/mesure est mentionnée dans le tableau A.8 de la CEI 61508-2.

But: Détecter les défaillances pendant la communication sur le bus et dans les liaisons de transmission en série.

Description: Le bus est complété par plusieurs lignes (bits) et ces lignes supplémentaires (bits) sont utilisées pour détecter les défaillances avec les techniques de code de Hamming.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.3 Redondance matérielle complète

NOTE Cette technique/mesure est mentionnée dans le tableau A.8 de la CEI 61508-2.

But: Détecter les défaillances pendant la communication en comparant les signaux sur deux bus.

Description: Le bus est doublé et les lignes supplémentaires (bits) sont utilisées pour détecter les défaillances.

A.6.5 Input comparison/voting

NOTE This technique/measure is referenced in tables A.7 and A.14 of IEC 61508-2.

Aim: To detect individual failures, failures caused by external influences, timing failures, addressing failures, drift failures (for analogue signals) and transient failures.

Description: This is a dataflow-dependent comparison of independent inputs to ensure compliance with a defined tolerance range (time, value). There will be 1 out of 2, 2 out of 3 or better redundancy. This measure is only effective if the dataflow changes during the diagnostic test interval.

References:

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

MSR-Schutzeinrichtungen. Anforderungen und Massnahmen zur gesicherten Funktion. DIN V 19251, February 1995.

A.7 Data paths (internal communication)

Global objective: To detect failures caused by a defect in the information transfer.

A.7.1 One-bit hardware redundancy

NOTE This technique/measure is referenced in table A.8 of IEC 61508-2.

Aim: To detect all odd-bit failures, i.e. 50 % of all the possible bit failures in the data stream.

Description: The bus is extended by one line (bit) and this additional line (bit) is used to detect failures by parity checking.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.2 Multi-bit hardware redundancy

NOTE This technique/measure is referenced in table A.8 of IEC 61508-2.

Aim: To detect failures during the communication on the bus and in serial transmission links.

Description: The bus is extended by two or more lines (bits) and these additional lines (bits) are used in order to detect failures by Hamming code techniques.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.3 Complete hardware redundancy

NOTE This technique/measure is referenced in table A.8 of IEC 61508-2.

Aim: To detect failures during the communication by comparing the signals on two buses.

Description: The bus is doubled and the additional lines (bits) are used in order to detect failures.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.4 Inspection utilisant des trames de test

NOTE Cette technique/mesure est mentionnée dans le tableau A.8 de la CEI 61508-2.

But: Détecter les défaillances statiques (défaillances dues à un blocage) et les interférences.

Description: Il s'agit d'un test cyclique, indépendant du flux de données, des chemins de données. Il utilise une trame de test définie pour comparer les observations avec les valeurs prévues correspondantes.

Le contenu de la trame de test, la réception de la trame de test et l'évaluation de la trame de test doivent toutes être indépendantes les unes des autres. Il convient que l'EUC ne soit pas influencé de façon inadmissible par la trame de test.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.5 Redondance de transmission

NOTE Cette technique/mesure est mentionnée dans le tableau A.8 de la CEI 61508-2.

But: Détecter les défaillances transitoires dans la communication sur le bus.

Description: Les informations sont transférées plusieurs fois en séquence. La répétition n'est efficace que contre les défaillances transitoires.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.6 Redondance d'informations

NOTE Cette technique/mesure est mentionnée dans le tableau A.8 de la CEI 61508-2.

But: Détecter les défaillances dans la communication sur le bus.

Description: Les données sont transmises en paquets, accompagnées d'une somme de contrôle calculée pour chacun des paquets. Le récepteur calcule à son tour les sommes de contrôle correspondant aux données reçues et compare les résultats aux sommes de contrôle reçues.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.8 Alimentation

Objectif général: Détecter ou tolérer les défaillances dues à un défaut de l'alimentation.

A.8.1 Protection contre les surtensions avec arrêt de sécurité

NOTE Cette technique/mesure est mentionnée dans le tableau A.9 de la CEI 61508-2.

But: Protéger le système relatif à la sécurité contre les surtensions.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.4 Inspection using test patterns

NOTE This technique/measure is referenced in table A.8 of IEC 61508-2.

Aim: To detect static failures (stuck-at failure) and cross-talk.

Description: This is a dataflow-independent cyclical test of data paths. It uses a defined test pattern to compare observations with the corresponding expected values.

The test pattern information, the test pattern reception, and test pattern evaluation must all be independent of each other. The EUC should not be inadmissibly influenced by the test pattern.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.5 Transmission redundancy

NOTE This technique/measure is referenced in table A.8 of IEC 61508-2.

Aim: To detect transient failures in bus communication.

Description: The information is transferred several times in sequence. The repetition is effective only against transient failures.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.7.6 Information redundancy

NOTE This technique/measure is referenced in table A.8 of IEC 61508-2.

Aim: To detect failures in bus communication.

Description: Data is transmitted in blocks, together with a calculated checksum for each block. The receiver then re-calculates the checksum of the received data and compares the result with the received checksum.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.8 Power supply

Global objective: To detect or tolerate failures caused by a defect in the power supply.

A.8.1 Overvoltage protection with safety shut-off

NOTE This technique/measure is referenced in table A.9 of IEC 61508-2.

Aim: To protect the safety-related system against overvoltage.

Description: Une surtension est détectée assez tôt pour que toutes les sorties puissent être mises en position de sécurité par le sous-programme de mise hors tension ou qu'il y ait un basculement sur une deuxième unité d'alimentation.

Référence: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

A.8.2 Surveillance de la tension (secondaire)

NOTE Cette technique/mesure est mentionnée dans le tableau A.9 de la CEI 61508-2.

But: Surveiller les tensions secondaires et mettre sur la position de sécurité si la tension ne se situe pas dans la fourchette spécifiée.

Description: La tension secondaire est surveillée et une mise hors tension a lieu ou il y a un basculement sur une deuxième unité d'alimentation si la tension ne se situe pas dans la fourchette spécifiée.

A.8.3 Mise hors tension avec arrêt de sécurité

NOTE Cette technique/mesure est mentionnée dans le tableau A.9 de la CEI 61508-2.

But: Couper l'alimentation tout en enregistrant toutes les informations critiques relatives à la sécurité.

Description: La surtension ou la sous-tension est détectée assez tôt pour que l'état interne puisse être sauvegardé dans une mémoire non volatile (si nécessaire), et pour que toutes les sorties puissent être mises en position de sécurité par le sous-programme de mise hors tension, ou pour que toutes les sorties puissent être mises en position de sécurité par le sous-programme de mise hors tension, ou qu'il y ait un basculement sur une deuxième unité d'alimentation.

A.9 Surveillance temporelle et logique de la séquence du programme

NOTE Ce groupe de techniques et mesures est mentionné dans les tableaux A.16, A.17 et A.19 de la CEI 61508-2.

Objectif général: Détecter une séquence de programme défectueuse. Une séquence de programme est défectueuse si les éléments individuels d'un programme (par exemple modules logiciels, sous-programmes ou commandes) sont traités dans une mauvaise séquence ou période de temps ou si l'horloge du processeur présente une anomalie.

A.9.1 Chien de garde avec base de temps séparée sans fenêtre temporelle

NOTE Cette technique/mesure est mentionnée dans les tableaux A.10 et A.12 de la CEI 61508-2.

But: Surveiller le comportement et la vraisemblance de la séquence du programme.

Description: Des éléments de séquencement externes avec une base de temps différente (par exemple des chiens de garde de séquencement) sont périodiquement déclenchés pour surveiller le comportement de l'ordinateur et la vraisemblance de la séquence du programme. Il est important que les points de déclenchement soient correctement placés dans le programme. Le chien de garde n'est pas déclenché à des périodes fixes mais un intervalle maximal est spécifié.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

Description: Overvoltage is detected early enough that all outputs can be switched to a safe condition by the power-down routine or there is a switch-over to a second power unit.

Reference: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

A.8.2 Voltage control (secondary)

NOTE This technique/measure is referenced in table A.9 of IEC 61508-2.

Aim: To monitor the secondary voltages and initiate a safe condition if the voltage is not in its specified range.

Description: The secondary voltage is monitored and a power-down is initiated, or there is a switch-over to a second power unit, if it is not in its specified range.

A.8.3 Power-down with safety shut-off

NOTE This technique/measure is referenced in table A.9 of IEC 61508-2.

Aim: To shut off the power with all safety critical information stored.

Description: Overvoltage or undervoltage is detected early enough so that the internal state can be saved in non-volatile memory (if necessary), and so that all outputs can be set to a safe condition by the power-down routine, or that all outputs can be switched to a safe condition by the power-down routine, or there is a switch-over to a second power unit.

A.9 Temporal and logical program sequence monitoring

NOTE This group of techniques and measures is referenced in tables A.16, A.17 and A.19 of IEC 61508-2.

Global objective: To detect a defective program sequence. A defective program sequence exists if the individual elements of a program (for example software modules, subprograms or commands) are processed in the wrong sequence or period of time, or if the clock of the processor is faulty.

A.9.1 Watch-dog with separate time base without time-window

NOTE This technique/measure is referenced in tables A.10 and A.12 of IEC 61508-2.

Aim: To monitor the behaviour and the plausibility of the program sequence.

Description: External timing elements with a separate time base (for example watch-dog timers) are periodically triggered to monitor the computer's behaviour and the plausibility of the program sequence. It is important that the triggering points are correctly placed in the program. The watch-dog is not triggered at a fixed period, but a maximum interval is specified.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.9.2 Chien de garde avec base de temps séparée et fenêtre temporelle

NOTE Cette technique/mesure est mentionnée dans les tableaux A.10 et A.12 de la CEI 61508-2.

But: Surveiller le comportement et la vraisemblance de la séquence du programme.

Description: Des éléments de séquencement externes avec une base de temps différente (par exemple des chiens de garde de séquencement) sont périodiquement déclenchés pour surveiller le comportement de l'ordinateur et la vraisemblance de la séquence du programme. Il est important que les points de déclenchement soient correctement placés dans le programme. Une limite inférieure et une limite supérieure sont fixées pour le chien de garde de séquencement. Si la séquence de programme met plus ou moins de temps que prévu, une action d'urgence est entreprise.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.9.3 Surveillance logique de la séquence du programme

NOTE Cette technique/mesure est mentionnée dans les tableaux A.10 et A.12 de la CEI 61508-2.

But: Surveiller la bonne séquence des sections individuelles du programme.

Description: La séquence correcte des sections individuelles du programme est surveillée à l'aide d'un logiciel (procédure de comptage, procédure adaptée) ou à l'aide d'éléments de surveillance externes. Il est important que les points de surveillance soient placés correctement dans le programme.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.9.4 Combinaison de surveillance temporelle et logique des séquences du programme

NOTE Cette technique/mesure est mentionnée dans les tableaux A.10 et A.12 de la CEI 61508-2.

But: Surveiller le comportement et la bonne séquence des sections individuelles du programme.

Description: Un moyen temporel (par exemple un chien de garde de séquencement) surveillant la séquence du programme n'est redéclenché que si la séquence des sections du programme est également exécutée correctement.

Référence: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.9.5 Surveillance temporelle avec contrôle en ligne

NOTE Cette technique/mesure est mentionnée dans les tableaux A.10 et A.12 de la CEI 61508-2.

But: Détecter les anomalies de la surveillance temporelle.

Description: La surveillance temporelle est contrôlée lors de la mise en marche et celle-ci n'est possible que si la surveillance temporelle fonctionne correctement. Par exemple, une sonde de température pourrait être testée avec une résistance chauffante lors de la mise en route.

A.9.2 Watch-dog with separate time base and time-window

NOTE This technique/measure is referenced in tables A.10 and A.12 of IEC 61508-2.

Aim: To monitor the behaviour and the plausibility of the program sequence.

Description: External timing elements with a separate time base (for example watch-dog timers) are periodically triggered to monitor the computer's behaviour and the plausibility of the program sequence. It is important that the triggering points are correctly placed in the program. A lower and upper limit is given for the watch-dog timer. If the program sequence takes a longer or shorter time than expected, emergency action is taken.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.9.3 Logical monitoring of program sequence

NOTE This technique/measure is referenced in tables A.10 and A.12 of IEC 61508-2.

Aim: To monitor the correct sequence of the individual program sections.

Description: The correct sequence of the individual program sections is monitored using software (counting procedure, key procedure) or using external monitoring facilities. It is important that the checking points are placed in the program correctly.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.9.4 Combination of temporal and logical monitoring of program sequences

NOTE This technique/measure is referenced in tables A.10 and A.12 of IEC 61508-2.

Aim: To monitor the behaviour and the correct sequence of the individual program sections.

Description: A temporal facility (for example a watch-dog timer) monitoring the program sequence is retriggered only if the sequence of the program sections is also executed correctly.

Reference: Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

A.9.5 Temporal monitoring with on-line check

NOTE This technique/measure is referenced in tables A.10 and A.12 of IEC 61508-2.

Aim: To detect faults in the temporal monitoring.

Description: The temporal monitoring is checked at start-up, and a start is only possible if the temporal monitoring operates correctly. For example, a heat sensor could be checked by a heated resistor at start-up.

A.10 Aération et chauffage

NOTE Ce groupe de techniques et de mesures est mentionné dans les tableaux A.17 et A.19 de la CEI 61508-2.

Objectif général: Maîtriser les défaillances de l'aération ou du chauffage et/ou surveillance de l'aération ou du chauffage s'il s'agit de sécurité.

A.10.1 Capteur de température

NOTE Cette technique/mesure est mentionnée dans le tableau A.11 de la CEI 61508-2.

But: Détecter une température trop élevée ou trop basse avant que le système ne commence à fonctionner hors spécification.

Description: Un capteur de température surveille la température aux points les plus critiques du E/E/PES. Avant que la température ne sorte de la fourchette spécifiée, une action d'urgence est entreprise.

A.10.2 Surveillance des ventilateurs

NOTE Cette technique/mesure est mentionnée dans le tableau A.11 de la CEI 61508-2.

But: Détecter un mauvais fonctionnement des ventilateurs.

Description: Le bon fonctionnement des ventilateurs est surveillé. Si un ventilateur ne fonctionne pas correctement, une action d'entretien (ou en dernier recours, d'urgence) est entreprise.

A.10.3 Actionnement de l'arrêt de sécurité par l'intermédiaire d'un fusible thermique

NOTE Cette technique/mesure est mentionnée dans le tableau A.11 de la CEI 61508-2.

But: Arrêter le système relatif à la sécurité avant que le système ne fonctionne en dehors de sa spécification thermique.

Description: Un fusible thermique est utilisé pour arrêter le système relatif à la sécurité. Pour un PES, l'arrêt est produit par un programme de mise hors tension qui enregistre toutes les informations nécessaires pour une action d'urgence.

A.10.4 Message échelonné des capteurs thermiques et de l'alarme conditionnelle

NOTE Cette technique/mesure est mentionnée dans le tableau A.11 de la CEI 61508-2.

But: Indiquer que le système relatif à la sécurité fonctionne en dehors de sa spécification thermique.

Description: La température est surveillée et une alarme se déclenche si la température est en dehors de la fourchette spécifiée.

A.10.5 Connexion du refroidissement par air forcé et indication d'état

NOTE Cette technique/mesure est mentionnée dans le tableau A.11 de la CEI 61508-2.

But: Prévenir la surchauffe par un refroidissement par air forcé.

Description: La température est surveillée et le refroidissement par air forcé est enclenché si elle est plus élevée qu'une valeur limite spécifiée. L'utilisateur est informé de l'état.

A.10 Ventilation and heating

NOTE This group of techniques and measures is referenced in tables A.17 and A.19 of IEC 61508-2.

Global objective: To control failures in the ventilation or heating, and/or their monitoring, if this is safety-related.

A.10.1 Temperature sensor

NOTE This technique/measure is referenced in table A.11 of IEC 61508-2.

Aim: To detect over- or under-temperature before the system begins to operate outside specification.

Description: A temperature sensor monitors temperature at the most critical points of the E/E/PES. Before the temperature leaves the specified range, emergency action is taken.

A.10.2 Fan control

NOTE This technique/measure is referenced in table A.11 of IEC 61508-2.

Aim: To detect incorrect operation of the fans.

Description: The fans are monitored for correct operation. If a fan is not working properly, maintenance (or ultimately, emergency) action is taken.

A.10.3 Actuation of the safety shut-off via thermal fuse

NOTE This technique/measure is referenced in table A.11 of IEC 61508-2.

Aim: To shut off the safety-related system before the system works outside of its thermal specification.

Description: A thermal fuse is used to shut off the safety-related system. For a PES, the shut-off is introduced by a power-down routine which stores all information necessary for emergency action.

A.10.4 Staggered message from thermo-sensors and conditional alarm

NOTE This technique/measure is referenced in table A.11 of IEC 61508-2.

Aim: To indicate that the safety-related system is working outside its thermal specification.

Description: The temperature is monitored and an alarm is raised if the temperature is outside of a specified range.

A.10.5 Connection of forced-air cooling and status indication

NOTE This technique/measure is referenced in table A.11 of IEC 61508-2.

Aim: To prevent overheating by forced-air cooling.

Description: The temperature is monitored and forced-air cooling is introduced if the temperature is higher than a specified limit. The user is informed of the status.

A.11 Communication et mémoire de masse

Objectif général: Maîtriser les défaillances pendant la communication avec des sources externes et la mémoire de masse.

A.11.1 Séparation entre les lignes d'alimentation et les lignes d'informations

NOTE Cette technique/mesure est mentionnée dans le tableau A.13 de la CEI 61508-2.

But: Minimiser les interférences causées sur les lignes d'informations par des intensités élevées.

Description: Les lignes d'alimentation sont séparées des lignes acheminant les informations. Le champ électrique qui pourrait causer des pointes de tension sur les lignes d'informations diminue avec la distance.

A.11.2 Séparation spatiale des lignes multiples

NOTE Cette technique/mesure est mentionnée dans les tableaux A.13 et A.17 de la CEI 61508-2.

But: Minimiser les interférences induites sur les lignes multiples par les intensités élevées.

Description: Des lignes transportant des signaux dupliqués sont séparées les unes des autres. Le champ électrique qui pourrait causer des pointes de tension sur les lignes multiples diminue avec la distance. Cette mesure réduit aussi les défaillances de cause commune.

A.11.3 Augmentation de l'immunité aux interférences

NOTE Cette technique/mesure est mentionnée dans les tableaux A.13, A.17 et A.19 de la CEI 61508-2.

But: Minimiser les interférences électromagnétiques sur le système relatif à la sécurité.

Description: Des techniques de conception telles que le blindage et le filtrage sont utilisées pour accroître l'immunité du système relatif à la sécurité aux perturbations électromagnétiques qui peuvent être rayonnées ou conduites par les lignes d'alimentation ou de signalisation par rayonnement ou conduction ou encore résulter de décharges électrostatiques.

Références:

CEI 61000-5-2/TR3:1997, *Compatibilité électromagnétique (CEM) – Partie 5: Guides d'installation et d'atténuation – Section 2: Mise à la terre et câblage*.

Noise Reduction Techniques in Electronic Systems. H. W. Ott, John Wiley Interscience, 2nd Edition, 1988.

EMC for Product Designers. Tim Williams, Newnes, 1992, ISBN 0-7506-1264-9.

Grounding and Shielding Techniques in Instrumentation. John Wiley & Sons, New York, 1986.

Principles and Techniques of Electromagnetic Compatibility. C. Christopoulos, CRC Press, 1995.

Gestaltung von Maschinensteuerungen unter Berücksichtigung der elektromagnetischen Verträglichkeit. F. Börner, Sicherheitstechnisches Informations- und Arbeitsblatt 330260, BIA-Handbuch. 20. Lfg. V/93, Erich Schmidt Verlag, Bielefeld.

A.11 Communication and mass-storage

Global objective: To control failures during communication with external sources and mass-storage.

A.11.1 Separation of electrical energy lines from information lines

NOTE This technique/measure is referenced in table A.13 of IEC 61508-2.

Aim: To minimise cross-talk induced by high currents in the information lines.

Description: Electrical energy supply lines are separated from the lines carrying the information. The electrical field which could induce voltage spikes on the information lines decreases with distance.

A.11.2 Spatial separation of multiple lines

NOTE This technique/measure is referenced in tables A.13 and A.17 of IEC 61508-2.

Aim: To minimise cross-talk induced by high currents in multiple lines.

Description: Lines carrying duplicated signals are separated from each other. The electrical field which could induce voltage spikes on the multiple lines decreases with the distance. This measure also reduces common cause failures.

A.11.3 Increase of interference immunity

NOTE This technique/measure is referenced in tables A.13, A.17 and A.19 of IEC 61508-2.

Aim: To minimise electromagnetic interference on the safety-related system.

Description: Design techniques such as shielding and filtering are used to increase the interference immunity of the safety-related system to electromagnetic disturbances which may be radiated or conducted on power or signal lines, or result from electrostatic discharges.

References:

IEC 61000-5-2/TR3:1997, *Electromagnetic compatibility (EMC) – Part 5: Installation and mitigation guidelines – Section 2: Earthing and cabling*.

Noise Reduction Techniques in Electronic Systems. H. W. Ott, John Wiley Interscience, 2nd Edition, 1988.

EMC for Product Designers. Tim Williams, Newnes, 1992, ISBN 0-7506-1264-9.

Grounding and Shielding Techniques in Instrumentation. John Wiley & Sons, New York, 1986.

Principles and Techniques of Electromagnetic Compatibility. C. Christopoulos, CRC Press, 1995.

Gestaltung von Maschinensteuerungen unter Berücksichtigung der elektromagnetischen Verträglichkeit. F. Börner, Sicherheitstechnisches Informations- und Arbeitsblatt 330260, BIA-Handbuch. 20. Lfg. V/93, Erich Schmidt Verlag, Bielefeld.

A.11.4 Transmission de signaux complémentaires

NOTE Cette technique/mesure est mentionnée dans les tableaux A.13 et A.17 de la CEI 61508-2.

But: Détecter les mêmes tensions induites dans les lignes de transmission de signaux multiples.

Description: Toutes les informations sont dupliquées et transmises par des signaux complémentaires (par exemple en logique positive ou négative). Les défaillances de cause commune (dues par exemple à une émission électromagnétique) peuvent être détectées par un comparateur exclusif.

Référence: Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch. 20. Lfg. V/93, Erich Schmidt Verlag, Bielefeld.

A.12 Sondes

Objectif général: Maîtriser les défaillances des capteurs du système relatif à la sécurité.

A.12.1 Capteur de référence

NOTE Cette technique/mesure est mentionnée dans le tableau A.14 de la CEI 61508-2.

But: Détecter le mauvais fonctionnement d'un capteur.

Description: Un capteur de référence indépendant est utilisé pour surveiller le fonctionnement d'un capteur de procédé. Tous les signaux d'entrée sont contrôlés à des intervalles adéquats par le capteur de référence pour détecter les défaillances du capteur de procédé.

Référence: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

A.12.2 Commutateur à action directe

NOTE Cette technique/mesure est mentionnée dans le tableau A.14 de la CEI 61508-2.

But: Ouvrir un contact par une connexion mécanique directe entre la came et le contact de commutation.

Description: Un commutateur à action directe ouvre ses contacts normalement fermés par une connexion mécanique directe entre la came et le contact de commutation. Cela garantit que, chaque fois que la came de commutation est en position de fonctionnement, les contacts de commutation sont ouverts.

Référence: Verriegelung beweglicher Schutzeinrichtungen. F. Kreutzkampf, K. Becker, Sicherheitstechnisches Informations- und Arbeitsblatt 330210, BIA-Handbuch. 1. Lfg. IX/85, Erich Schmidt Verlag, Bielefeld.

A.13 Organes finaux (actionneurs)

Objectif général: Maîtriser les défaillances des organes finaux du système relatif à la sécurité.

A.13.1 Surveillance

NOTE Cette technique/mesure est mentionnée dans le tableau A.15 de la CEI 61508-2.

But: Détecter le mauvais fonctionnement d'un actionneur.

A.11.4 Antivalent signal transmission

NOTE This technique/measure is referenced in tables A.13 and A.17 of IEC 61508-2.

Aim: To detect the same induced voltages in multiple signal transmission lines.

Description: All duplicated information is transmitted with antivalent signals (for example logic 1 and 0). Common cause failures (for example by electromagnetic emission) can be detected by an antivalent comparator.

Reference: Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch. 20. Lfg. V/93, Erich Schmidt Verlag, Bielefeld.

A.12 Sensors

Global objective: To control failures in the sensors of the safety-related system.

A.12.1 Reference sensor

NOTE This technique/measure is referenced in table A.14 of IEC 61508-2.

Aim: To detect the incorrect operation of a sensor.

Description: An independent reference sensor is used to monitor the operation of a process sensor. All input signals are checked at suitable time intervals by the reference sensor to detect failures of the process sensor.

Reference: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

A.12.2 Positive-activated switch

NOTE This technique/measure is referenced in table A.14 of IEC 61508-2.

Aim: To open a contact by a direct mechanical connection between switch cam and contact.

Description: A positive-activated switch opens its normally closed contacts by a direct mechanical connection between switch cam and contact. This ensures that whenever the switch cam is in the operated position, the switch contacts must be open.

Reference: Verriegelung beweglicher Schutzeinrichtungen. F. Kreutzkampf, K. Becker, Sicherheitstechnisches Informations- und Arbeitsblatt 330210, BIA-Handbuch. 1. Lfg. IX/85, Erich Schmidt Verlag, Bielefeld.

A.13 Final elements (actuators)

Global objective: To control failures in the final elements in the safety-related system.

A.13.1 Monitoring

NOTE This technique/measure is referenced in table A.15 of IEC 61508-2.

Aim: To detect the incorrect operation of an actuator.

Description: Le fonctionnement de l'actionneur est surveillé (par exemple par les contacts à action directe d'un relais; voir surveillance des contacts de relais en A.1.2). La redondance induite par cette surveillance peut servir à déclencher une action d'urgence.

Références:

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Verriegelungseinrichtungen. F. Kreutzkampf, W. Hertel, Sicherheitstechnisches Informations- und Arbeitsblatt 330212, BIA-Handbuch. 17. Lfg. X/91, Erich Schmidt Verlag, Bielefeld.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

A.13.2 Surveillance croisée de plusieurs actionneurs

NOTE Cette technique/mesure est mentionnée dans le tableau A.15 de la CEI 61508-2.

But: Détecter les défauts des actionneurs en comparant les résultats.

Description: Chaque actionneur multiple est surveillé par un canal matériel différent. S'il y a un écart, une action d'urgence est entreprise.

A.14 Mesures contre l'environnement physique

NOTE Cette technique/mesure est mentionnée dans le tableau A.17 de la CEI 61508-2.

But: Prévenir les influences de l'environnement physique (eau, poussière, produits corrosifs) à l'origine de défaillances.

Description: L'enveloppe du matériel est conçue pour résister à l'environnement prévisible.

Référence: CEI 60529:1989, *Degrés de protection procurés par les enveloppes (Code IP)*.

Description: The operation of the actuator is monitored (for example by the positively activated contacts of a relay, see monitoring of relay contacts in A.1.2). The redundancy introduced by this monitoring can be used to trigger emergency action.

References:

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Verriegelungseinrichtungen. F. Kreutzkampf, W. Hertel, Sicherheitstechnisches Informations- und Arbeitsblatt 330212, BIA-Handbuch. 17. Lfg. X/91, Erich Schmidt Verlag, Bielefeld.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

A.13.2 Cross-monitoring of multiple actuators

NOTE This technique/measure is referenced in table A.15 of IEC 61508-2.

Aim: To detect faults in actuators by comparing the results.

Description: Each multiple actuator is monitored by a different hardware channel. If a discrepancy occurs, emergency action is taken.

A.14 Measures against the physical environment

NOTE This technique/measure is referenced in table A.17 of IEC 61508-2

Aim: To prevent influences of the physical environment (water, dust, corrosive substances) causing failures.

Description: The enclosure of the equipment is designed to withstand the expected environment.

Reference: IEC 60529:1989, *Degrees of protection provided by enclosures (IP Code)*.

Annexe B
(informative)

Présentation de techniques et mesures pour les E/E/PES: prévention des défaillances systématiques (voir la CEI 61508-2 et la CEI 61508-3)

NOTE Beaucoup de techniques de cette annexe sont applicables au logiciel mais n'ont pas été reproduites dans l'annexe C.

B.1 Mesures et techniques générales

B.1.1 Gestion de projet

NOTE Cette technique/mesure est mentionnée dans les tableaux B.1 à B.6 de la CEI 61508-2.

But: Eviter les défaillances par l'adoption d'un modèle organisationnel ainsi que de règles et de mesures pour le développement et le test des systèmes relatifs à la sécurité.

Description: Les mesures les plus adaptées et les plus importantes sont

– la création d'un modèle organisationnel, en particulier pour l'assurance de la qualité (voir les normes telles que les séries ISO 9000-1 à ISO 9004-1 ou équivalent) qui est exposée dans un manuel sur l'assurance de la qualité; et

– la mise en place de règles et de mesures pour la création et la validation des systèmes relatifs à la sécurité dans les recommandations de projets croisés et de projets spécifiques.

Un certain nombre de principes de base importants sont exposés dans ce qui suit:

– définition d'une organisation de conception:
  – tâches et responsabilités des unités organisationnelles;
  – autorité des services d'assurance de la qualité;
  – indépendance de l'assurance de la qualité (inspection interne) par rapport au développement;
– définition d'un plan séquentiel (modèles d'activité):
  – détermination de toutes les activités pertinentes pendant l'exécution du projet, y compris les inspections internes et leur planification;
  – actualisation du projet;
– définition d'une séquence normalisée pour une inspection interne:
  – planification, exécution et contrôle de l’inspection (théorie de l’inspection);
  – création de mécanismes pour les sous-produits;
  – mise en sécurité des inspections répétitives;
– gestion de configuration:
  – administration et contrôle des versions;
  – détection des effets des modifications;
  – inspection de cohérence après les modifications;
– introduction d'une évaluation quantitative des mesures de l'assurance de qualité:
  – identification des exigences;
  – statistiques des défaillances;
– introduction de méthodes universelles assistées par ordinateur, comprenant les outils et les formations du personnel.

Annex B
(informative)

Overview of techniques and measures for E/E/PES: avoidance of systematic failures (see IEC 61508-2 and IEC 61508-3)

NOTE Many techniques in this annex are applicable to software but have not been duplicated in annex C.

B.1 General measures and techniques

B.1.1 Project management

NOTE This technique/measure is referenced in tables B.1 to B.6 of IEC 61508-2.

Aim: To avoid failures by adoption of an organisational model and rules and measures for development and testing of safety-related systems.

Description: The most important and best measures are

– the creation of an organisational model, especially for quality assurance (see standards such as the series ISO 9000-1 to ISO 9004-1 or similar) which is set down in a quality assurance handbook; and

– the establishment of regulations and measures for the creation and validation of safety-related systems in cross-project and project-specific guidelines.

A number of important basic principles are set down in the following:

– definition of a design organisation:
  – tasks and responsibilities of the organisational units,
  – authority of the quality assurance departments,
  – independence of quality assurance (internal inspection) from development;
– definition of a sequence plan (activity models):
  – determination of all activities which are relevant during execution of the project including internal inspections and their scheduling,
  – project update;
– definition of a standardised sequence for an internal inspection:
  – planning, execution and checking of the inspection (inspection theory),
  – releasing mechanisms for subproducts,
  – the safekeeping of repeat inspections;
– configuration management:
  – administration and checking of versions,
  – detection of the effects of modifications,
  – consistency inspections after modifications;
– introduction of a quantitative assessment of quality assurance measures:
  – requirement acquisition,
  – failure statistics;
– introduction of computer-aided universal methods, tools and training of personnel.

Références:

IEEE: Software Engineering Standards. IEEE/Wiley-Interscience, New York, 1987

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.1.2 Documentation

NOTE 1 Cette technique/mesure est mentionnée dans les tableaux B.1 à B.6 de la CEI 61508-2.

NOTE 2 Voir également l'article 5 et l'annexe A de la CEI 61508-1.

But: Eviter les défaillances et faciliter l'évaluation de la sécurité du système en documentant chaque étape du développement.

Description: L'aptitude à l'exploitation et la sécurité en exploitation ainsi que le soin apporté au développement par toutes les parties intervenant doivent être démontrés lors de l'évaluation. Pour pouvoir montrer le soin apporté au développement et pour garantir la vérification des preuves de la sécurité à tous moments, une importance particulière est donnée à la documentation.

Des mesures courantes importantes sont l'introduction de recommandations et d'une assistance par ordinateur, c'est-à-dire

- recommandations qui
  - spécifient un plan de groupement;
  - demandent une liste de contrôle pour les contenus; et
  - déterminent la forme du document;
- administration de la documentation à l'aide d'une bibliothèque de projet organisée et assistée par ordinateur.

Les mesures individuelles sont:

- la séparation dans la documentation
  - des exigences;
  - du système (documentation pour l'utilisateur); et
  - du développement (y compris l'inspection interne);
- le groupement de la documentation sur le développement conformément au cycle de vie de sécurité;
- la définition de modules de documentation normalisés à partir desquels les documents peuvent être compilés;
- l'identification claire des parties constitutives de la documentation;
- la mise à jour formalisée des révisions;
- le choix de moyens de description clairs et intelligibles:
  - notation formelle pour les déterminations;
  - langage naturel pour les introductions, les justifications et les objectifs;
  - représentations graphiques pour les exemples;
  - définition sémantique des éléments graphiques; et
  - répertoires de mots spécialisés.

References:

IEEE: Software Engineering Standards. IEEE/Wiley-Interscience, New York, 1987.

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.1.2 Documentation

NOTE 1 This technique/measure is referenced in tables B.1 to B.6 of IEC 61508-2.

NOTE 2 See also clause 5 and annex A of IEC 61508-1.

Aim: To avoid failures and facilitate assessment of system safety, by documenting each step during development.

Description: The operational capacity and safety, as well as the care taken in development by all parties involved, has to be demonstrated during assessment. In order to be able to show the development care, and in order to guarantee the verification of the evidence of safety at any time, special importance is given to the documentation.

Important common measures are the introduction of guidelines and computer aid, i.e.

- guidelines, which
  - specify a grouping plan;
  - ask for checklists for the contents; and
  - determine the form of the document;
- administration of the documentation with the help of a computer-aided and organised project library.

Individual measures are:

- separation in the documentation
  - of the requirements,
  - of the system (user-documentation) and
  - of the development (including internal inspection);
- grouping of the development documentation according to the safety lifecycle;
- definition of standardised documentation modules, from which the documents can be compiled;
- clear identification of the constituent parts of the documentation;
- formalised revision update;
- selection of clear and intelligible means of description:
  - formal notation for determinations;
  - natural language for introductions, justifications and representations of intentions;
  - graphical representations for examples;
  - semantic definition of graphical elements; and
  - directories of specialised words.

Références:

CEI 61506:1997, *Mesure et commande dans les processus industriels – Documentation des logiciels d'application.*

EWICS European Workshop on Industrial Computer Systems, TC 7: Safety Related Computers – Software Development and Systems Documentation. Verlag TÜV Rheinland, Köln, 1985.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Entwicklungstechnik sicherheitsverantwortlicher Software in der Eisenbahn-Signaltechnik. U. Feucht, Informatik-Fachberichte 86, Springer Verlag, Berlin, 184-195, 1984.

Richtlinie zur Erstellung und Prüfung sicherheitsrelevanter Software. K. Grimm, G. Heiner, Informatik Fachberichte 86, Springer Verlag, Berlin, 277-288, 1984.

B.1.3 Séparation des systèmes relatifs à la sécurité et des systèmes non relatifs à la sécurité

NOTE Cette technique/mesure est mentionnée dans les tableaux B.1 et B.6 de la CEI 61508-2.

But: Eviter que la partie du système non relative à la sécurité ait une influence néfaste sur la partie relative à la sécurité.

Description: Dans la spécification, il convient de décider si une séparation des systèmes relatifs à la sécurité des systèmes non relatifs à la sécurité est possible. Des spécifications claires doivent être données pour l'interface entre ces deux parties. Une séparation claire réduit l'effort de test des systèmes relatifs à la sécurité.

Référence: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.1.4 Diversité du matériel

NOTE Cette technique/mesure est mentionnée dans les tableaux A.16, A.17 et A.19 de la CEI 61508-2.

But: Détecter les défaillances systématiques pendant le fonctionnement de l'EUC à l'aide de composants diversifiés ayant des taux de défaillance et des types de défaillances différents.

Description: Des types de composants différents sont utilisés pour les différents canaux d'un système relatif à la sécurité. Cela réduit la probabilité des défaillances dues aux causes communes (par exemple surtension, perturbation électromagnétique) et augmente la probabilité de détection de ces défaillances.

L'existence de moyens différents pour effectuer une fonction souhaitée, par exemple des principes physiques différents, offre d'autres façons de résoudre le même problème. Il y a plusieurs types de diversité. La diversité fonctionnelle utilise des approches différentes pour obtenir le même résultat.

Référence: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

References:

IEC 61506:1997, *Industrial-process measurement and control – Documention of application software.*

EWICS European Workshop on Industrial Computer Systems, TC 7: Safety Related Computers – Software Development and Systems Documentation. Verlag TÜV Rheinland, Köln, 1985.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Entwicklungstechnik sicherheitsverantwortlicher Software in der Eisenbahn-Signaltechnik. U. Feucht, Informatik-Fachberichte 86, Springer Verlag, Berlin, 184-195, 1984.

Richtlinie zur Erstellung und Prüfung sicherheitsrelevanter Software. K. Grimm, G. Heiner, Informatik Fachberichte 86, Springer Verlag, Berlin, 277-288, 1984.

B.1.3 Separation of safety-related systems from non-safety-related systems

NOTE This technique/measure is referenced in tables B.1 and B.6 of IEC 61508-2.

Aim: To prevent the non-safety-related part of the system from influencing the safety-related part in undesired ways.

Description: In the specification it should be decided whether a separation of the safety-related systems and non-safety-related systems is possible. Clear specifications should be written for the interfacing of the two parts. A clear separation reduces the effort for testing the safety-related systems.

Reference: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.1.4 Diverse hardware

NOTE This technique/measure is referenced in tables A.16, A.17 and A.19 of IEC 61508-2.

Aim: To detect systematic failures during operation of the EUC, using diverse components with different rates and types of failures.

Description: Different types of components are used for the diverse channels of a safety-related system. This reduces the probability of common cause failures (for example overvoltage, electromagnetic interference), and increases the probability of detecting such failures.

Existence of different means of performing a required function, for example different physical principles, offer other ways of solving the same problem. There are several types of diversity. Functional diversity employs the use of different approaches to achieve the same result.

Reference: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.2 Spécification des exigences relatives aux E/E/PES

Objectif général: Produire une spécification qui soit, autant que possible, complète, sans erreur, sans contradiction et simple à vérifier.

B.2.1 Spécification structurée

NOTE Cette technique/mesure est mentionnée dans les tableaux B.1 et B.6 de la CEI 61508-2.

But: Réduire la complexité en créant une structure hiérarchique des exigences partielles. Eviter les défaillances d'interfaces entre les exigences.

Description: Cette technique structure la spécification fonctionnelle en exigences partielles de façon qu'il y ait des relations visibles les plus simples possible entre ces exigences. Cette analyse est successivement affinée jusqu'à ce que de petites exigences partielles claires puissent être distinguées. Le résultat de la mise au point finale est une structure hiérarchique d'exigences partielles qui offre un cadre pour la spécification des exigences complètes. Cette méthode met l'accent sur les interfaces des exigences partielles et est particulièrement efficace pour éviter les défaillances d'interfaces.

Références:

Structured Analysis and System Specification. T. De Marco, Yourdon Press, Englewood Cliffs, 1979.

ESA PSS 05-02, Guide to the user requirements definition phase, European Space Agency, 1989.

B.2.2 Méthodes formelles

NOTE 1 Voir en C.2.4 les détails sur les méthodes formelles spécifiques.

NOTE 2 Cette technique/mesure est mentionnée dans les tableaux B.1 et B.6 de la CEI 61508-2.

But: Enoncer une spécification de manière non ambiguë et cohérente pour que les erreurs et omissions puissent être détectées.

Description: Les méthodes formelles offrent un moyen de développer une description d'un système à un certain stade de sa spécification ou conception. La description qui en résulte prend une forme mathématique et peut être soumise à une analyse mathématique pour détecter plusieurs types d'incohérence ou d'inexactitude. De plus, la description peut, dans certains cas, être analysée par une machine avec une rigueur similaire à la vérification de la syntaxe d'un programme source par un compilateur ou animée pour afficher différents aspects du comportement du système décrit. L'animation peut renforcer la confiance en ce que le système satisfait aux exigences réelles aussi bien qu'aux exigences spécifiées formellement parce qu'elle améliore la reconnaissance par l'homme du comportement spécifié.

Une méthode formelle offre généralement une notation (c'est généralement une forme de calculs mathématiques discrets qui est utilisée), une technique pour élaborer une description dans cette notation et diverses formes d'analyse pour contrôler une description de propriétés d'exactitude différentes.

Partant d'une spécification mathématiquement formelle, la conception peut être transformée par une série de mises au point étape par étape en une conception de circuit logique.

Références:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

HOL: A Machine Orientated Formulation of Higher Order Logic. M. Gordon, University of Cambridge Technical Report Number 68, 1985.

B.2 E/E/PES safety requirements specification

Global objective: To produce a specification which is, as far as possible, complete, free from mistakes, free from contradiction, and simple to verify.

B.2.1 Structured specification

NOTE This technique/measure is referenced in tables B.1 and B.6 of IEC 61508-2.

Aim: To reduce complexity by creating a hierarchical structure of partial requirements. To avoid interface failures between the requirements.

Description: This technique structures the functional specification into partial requirements such that the simplest possible, visible relations exist between the latter. This analysis is successively refined until small clear partial requirements can be distinguished. The result of the final refinement is a hierarchical structure of partial requirements which provide a framework for the specification of the complete requirements. This method emphasises the interfaces of the partial requirements and is particularly effective for avoiding interface failures.

References:

Structured Analysis and System Specification. T. De Marco, Yourdon Press, Englewood Cliffs, 1979.

ESA PSS 05-02, Guide to the user requirements definition phase, European Space Agency, 1989.

B.2.2 Formal methods

NOTE 1 See C.2.4 for details of specific formal methods.

NOTE 2 This technique/measure is referenced in tables B.1 and B.6 of IEC 61508-2.

Aim: To express a specification unambiguously and consistently, so that mistakes and omissions can be detected.

Description: Formal methods provide a means of developing a description of a system at some stage in its specification or design. The resulting description takes a mathematical form and can be subjected to mathematical analysis to detect various classes of inconsistency or incorrectness. Moreover, the description can in some cases be analysed by a machine with a rigour similar to the syntax checking of a source program by a compiler, or animated to display various aspects of the behaviour of the system described. Animation can give extra confidence that the system meets the real requirement as well as the formally specified requirement, because it improves human recognition of the specified behaviour.

A formal method will generally offer a notation (generally some form of discrete mathematics being used), a technique for deriving a description in that notation, and various forms of analysis for checking a description for different correctness properties.

Starting from a mathematically formal specification, the design can be transformed by a series of step-wise refinements to a logic circuit design.

References:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

HOL: A Machine Orientated Formulation of Higher Order Logic. M. Gordon, University of Cambridge Technical Report Number 68, 1985.

B.2.3 Méthodes semi-formelles

But: Enoncer des parties d'une spécification de manière non ambiguë et cohérente pour que les erreurs et omissions puissent être détectées.

NOTE Cette technique/mesure est mentionnée dans les tableaux B.1, B.2 et B.6 de la CEI 61508-2 et dans les tableaux A.1, A.2 et A.4 de la CEI 61508-3.

B.2.3.1 Généralités

But: Prouver que la conception est conforme à sa spécification.

Description: Les méthodes semi-formelles offrent un moyen de développer une description d'un système à un stade de son développement, c'est-à-dire la spécification, la conception ou le codage. La description peut, dans certains cas, être analysée par une machine ou animée pour afficher différents aspects du comportement du système. L'animation peut renforcer la confiance en ce que le système satisfait aux exigences réelles aussi bien qu'aux exigences spécifiées.

Deux méthodes semi-formelles sont décrites dans les paragraphes suivants.

B.2.3.2 Automates finis/diagrammes de changement d'états

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.7 de la CEI 61508-3.

But: Modéliser, spécifier ou mettre en place la structure de commande d'un système.

Description: Beaucoup de systèmes peuvent être décrits en termes d'états, d'entrées et d'actions. Ainsi, dans l'état E1, à réception d'une entrée I, un système peut exécuter l'action A et passer à l'état E2. En décrivant les actions d'un système pour chaque entrée dans chaque état, il est possible de décrire entièrement un système. Le modèle de système résultant est appelé automate fini. Il est souvent représenté sous forme de «diagramme de changement d'états» montrant comment le système se déplace d'un état à un autre ou sous forme de matrice dont les dimensions sont l'état et l'entrée. Les cellules de la matrice contiennent l'action et le nouvel état qui est le résultat de la réception de l'entrée dans l'état donné.

Lorsqu'un système est compliqué ou a une structure naturelle, cela est reflété par un automate fini à couches.

Une spécification ou conception exprimée en automate fini peut subir des tests

– de complétude (il faut que le système ait une action et un nouvel état pour chaque entrée dans chaque état);
– de cohérence (un seul changement d'état est décrit pour chaque paire état/entrée); et
– d'accessibilité (s'il est possible ou non de passer d'un état à l'autre par n'importe quelle séquence d'entrées).

Ces propriétés sont importantes pour les systèmes critiques. Les outils pour faciliter ces contrôles sont facilement développés. Il y a aussi des algorithmes qui permettent la génération automatique des cas d'essai pour vérifier la réalisation d'un automate fini ou pour animer un modèle d'automate fini.

Référence: Introduction to the theory of Finite State Machines. A. Gill, McGraw-Hill, 1962.

B.2.3 Semi-formal methods

Aim: To express parts of a specification unambiguously and consistently, so that some mistakes and omissions can be detected.

NOTE This technique/measure is referenced in tables B.1, B.2 and B.6 of IEC 61508-2 and in tables A.1, A.2 and A.4 of IEC 61508-3.

B.2.3.1 General

Aim: To prove that the design meets its specification.

Description: Semi-formal methods provide a means of developing a description of a system at some stage in its development, i.e. specification, design or coding. The description can in some cases be analysed by machine or animated to display various aspects of the system behaviour. Animation can give extra confidence that the system meets the real requirement as well as the specified requirement.

Two semi-formal methods are described in the following subclauses.

B.2.3.2 Finite state machines/state transition diagrams

NOTE This technique/measure is referenced in tables B.5 and B.7 of IEC 61508-3.

Aim: To model, specify or implement the control structure of a system.

Description: Many systems can be described in terms of their states, their inputs, and their actions. Thus when in state S1, on receiving input I a system might carry out action A and move to state S2. By describing a system's actions for every input in every state we can describe a system completely. The resulting model of the system is called a finite state machine. It is often drawn as a so-called state transition diagram showing how the system moves from one state to another, or as a matrix in which the dimensions are state and input, and the matrix cells contain the action and new state resulting from receiving the input when in the given state.

Where a system is complicated or has a natural structure this can be reflected in a layered finite state machine.

A specification or design expressed as a finite state machine can be checked for

- completeness (the system must have an action and new state for every input in every state);
- consistency (only one state change is described for each state/input pair); and
- reachability (whether or not it is possible to get from one state to another by any sequence of inputs).

These are important properties for critical systems. Tools to support these checks are easily developed. Algorithms also exist that allow the automatic generation of test cases for verifying a finite state machine implementation or for animating a finite state machine model.

Reference: Introduction to the theory of Finite State Machines. A. Gill, McGraw-Hill, 1962.

B.2.3.3 Réseaux de Pétri temporels

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.7 de la CEI 61508-3.

But: Modéliser les aspects pertinents du comportement du système, évaluer et éventuellement améliorer les exigences d'exploitation et de sécurité par l'analyse et la reconception.

Description: Les réseaux de Pétri appartiennent à une catégorie de modèles théoriques graphiques qui conviennent pour représenter les informations et le flux de commande dans les systèmes qui ont des traitements simultanés et ont un comportement asynchrone.

Un réseau de Pétri est un réseau d'endroits et de transitions. Les endroits peuvent être «marqués» ou non. Une transition est «permise» quand tous les endroits d'entrée au réseau sont marqués. Quand elle est permise, elle peut (mais n'est pas obligée de) «se déclencher». Si elle se déclenche, les endroits d'entrée à la transition deviennent non marqués et, à la place, chaque endroit de sortie de la transition est marqué.

Les dangers potentiels peuvent être représentés comme des états particuliers (marquages) dans le modèle. Le modèle du réseau de Pétri peut être élargi pour permettre des fonctions temporelles du système. Bien que les réseaux de Pétri «classiques» se concentrent sur les aspects de flux de commande, plusieurs extensions ont été proposées pour incorporer le flux de données dans le modèle.

Références:

Petri nets: Properties, analysis and applications. T. Murata, Proc. IEEE 77 (4), 541-580, April 1989.

Safety analysis using Petri nets. N. G. Leveson, J. L. Stolzy, Proc. 15th Ann. Int. Symp on Fault-Tolerant Computing, 358-363, IEEE, 1985.

Using Petri nets for safety analysis of unmanned Metro system. M. El Koursi, P. Ozello, Proc. SAFECOMP 92, 135-139, Pergamon Press, 1992.

Net theory and applications. W. Brauer (ed.), Lecture Remarques in Computer Science, Vol. 84, Springer Verlag, 1980.

Petri net theory and modelling of systems. J. L. Peterson, Prentice Hall, 1981.

A tool for requirements specification and analysis of real time software based on timed Petri nets. S. Bologna, F. Pisacane, C. Ghezzi, D. Mandrioli, Proc. SAFECOMP 88, 9-11 November 1988. Fulda, Fed. Rep. of Germany, 1988.

B.2.4 Outils de spécification assistée par ordinateur

NOTE Cette technique/mesure est mentionnée dans les tableaux B.1 et B.6 de la CEI 61508-2 et dans les tableaux A.1 et A.2 de la CEI 61508-3.

B.2.4.1 Généralités

But: Utiliser des techniques de spécification formelles pour faciliter la détection automatique de la complétude et de la cohérence.

Description: Cette technique donne une spécification sous forme de base de données qui peut être inspectée automatiquement pour évaluer la cohérence et la complétude. L'outil de spécification peut animer divers aspects du système spécifié pour l'utilisateur. En général, cette technique facilite non seulement la création de la spécification mais aussi la conception et d'autres phases du cycle de vie du projet. Les outils de spécification peuvent être classés selon les paragraphes suivants.

B.2.3.3 Time Petri nets

NOTE This technique/measure is referenced in tables B.5 and B.7 of IEC 61508-3.

Aim: To model relevant aspects of the system behaviour and to assess and possibly improve safety and operational requirements through analysis and re-design.

Description: Petri nets belong to a class of graph theoretic models which are suitable for representing information and control flow in systems that exhibit concurrency and have asynchronous behaviour.

A Petri net is a network of places and transitions. The places may be "marked" or "unmarked". A transition is "enabled" when all the input places to it are marked. When enabled, it is permitted (but not obliged) to "fire". If it fires, the input places to the transition become unmarked, and each output place from the transition is marked instead.

The potential hazards can be represented as particular states (markings) in the model. The Petri net model can be extended to allow for timing features of the system. Although "classical" Petri nets concentrate on control flow aspects, several extensions have been proposed to incorporate data flow into the model.

References:

Petri nets: Properties, analysis and applications. T. Murata, Proc. IEEE 77 (4), 541-580, April 1989.

Safety analysis using Petri nets. N. G. Leveson, J. L. Stolzy, Proc. 15th Ann. Int. Symp on Fault-Tolerant Computing, 358-363, IEEE, 1985.

Using Petri nets for safety analysis of unmanned Metro system. M. El Koursi, P. Ozello, Proc. SAFECOMP '92, 135-139, Pergamon Press, 1992.

Net theory and applications. W. Brauer (ed.), Lecture Notes in Computer Science, Vol. 84, Springer Verlag, 1980.

Petri net theory and modelling of systems. J. L. Peterson, Prentice Hall, 1981.

A tool for requirements specification and analysis of real time software based on timed Petri nets. S. Bologna, F. Pisacane, C. Ghezzi, D. Mandrioli, Proc. SAFECOMP 88, 9-11 November 1988. Fulda, Fed. Rep. of Germany, 1988.

B.2.4 Computer-aided specification tools

NOTE This technique/measure is referenced in tables B.1 and B.6 of IEC 61508-2 and in tables A.1 and A.2 of IEC 61508-3.

B.2.4.1 General

Aim: To use formal specification techniques to facilitate automatic detection of ambiguity and completeness.

Description: The technique produces a specification in the form of a database that can be automatically inspected to assess consistency and completeness. The specification tool can animate various aspects of the specified system to the user. In general, the technique supports not only the creation of the specification but also of the design and other phases of the project lifecycle. Specification tools can be classified according to the following subclauses.

B.2.4.2 Outils orientés vers aucune méthode spécifique

But: Aider l'utilisateur à écrire une spécification valable en fournissant des raccourcis et des liens entre les parties en relation.

Description: L'outil de spécification effectue une partie du travail habituel de l'utilisateur et facilite la gestion du projet. Il n'applique aucune méthodologie de spécification particulière. L'indépendance relative par rapport à la méthode offre aux utilisateurs une grande liberté mais leur apporte peu de soutien spécialisé nécessaire lors de la création des spécifications. Cela rend la familiarisation avec le système plus difficile.

Référence: Integrierte Rechnerunterstützung für Entwicklung, Projektmanagement und Produktverwaltung mit EPOS. R. L.auber, P. Lempp, Elektron. Rechenanlagen 27, Heft 2, 68-74, 1985.

B.2.4.3 Procédure orientée vers le modèle avec une analyse hiérarchique

But: Aider l'utilisateur à écrire une spécification valable en assurant la cohérence entre la description des actions et des données à différents niveaux d'abstraction.

Description: Cette méthode donne une représentation fonctionnelle du système désiré (analyse structurée) à des niveaux d'abstraction différents (degré de précision). L'analyse à des niveaux différents influence les actions et les données. L'évaluation de la cohérence et de la complétude est possible entre les niveaux hiérarchiques ainsi qu'entre deux unités fonctionnelles (modules) sur le même niveau.

Référence: Structured Analysis for Requirement Definition. D. T. Ross, K. E. Schomann jr, IEEE Trans. on SE, January 1977.

B.2.4.4 Modèles d'entité

But: Aider l'utilisateur à écrire une spécification valable en se concentrant sur les entités du système et aux relations entre ces entités.

Description: Le système désiré est décrit comme un ensemble d'objets et leurs relations. L'outil permet de déterminer quelles relations peuvent être interprétées par le système. En général, les relations permettent une description de la structure hiérarchique des objets, du flux de données, des relations entre les données et des données qui sont soumises à certains procédés de fabrication. La procédure classique est élargie aux applications de commande de procédés. Les capacités d'inspection et l'assistance à l'utilisateur dépendent de la variété des relations illustrées. D'autre part, un grand nombre de possibilités de représentation rend l'application de cette technique compliquée.

Références:

PSL/PSA Computer-aided Technique for Structured Documentation and Analysis of Information Processing. D. Teichroew, E. A. Hershey, IEEE Trans on SE, Jan 1977.

Computer Aided Software Development. D. Teichroew, E. A. Hershey, Y. Lamamoto, Beitrag in: Verfahren und Hilfsmittel fur Spezifikation und Entwurf von Prozeßautomatisierungs-systemen. Hommel (ed.), Bericht KfK-PDV 154, Kernforschungszentrum Karlsruhe, 1978.

PCSL und ESPRESO – zwei Ansätze zur Formalisierung der Prozessrechner Software-spezifikation. J. Ludewig, GI-Fachtagung Prozessrechner 1981, Informatik-Fachberichte Bd. 39, Springer Verlag, Berlin, 1981.

B.2.4.2 Tools oriented towards no specific method

Aim: To help the user write a good specification by providing prompts and links between related parts.

Description: The specification tool takes over some routine work from the user and supports the project management. It does not enforce any particular specification methodology. The relative independence with regard to method allows users a great deal of freedom but also gives them little of the specialised support necessary when creating specifications. This makes familiarisation with the system more difficult.

Reference: Integrierte Rechnerunterstützung für Entwicklung, Projektmanagement und Produktverwaltung mit EPOS. R. Lauber, P. Lempp, Elektron. Rechenanlagen 27, Heft 2, 68-74, 1985.

B.2.4.3 Model orientated procedure with hierarchical analysis

Aim: To help the user write a good specification by ensuring consistency between descriptions of actions and data at various levels of abstraction.

Description: This method gives a functional representation of the desired system (structured analysis) at various levels of abstraction (degree of precision). The analysis at various levels acts on both actions and data. Assessment of ambiguity and completeness is possible between hierarchical levels as well as between two functional units (modules) on the same level.

Reference: Structured Analysis for Requirement Definition. D. T. Ross, K. E. Schomann jr, IEEE Trans. on SE, Januay 1977.

B.2.4.4 Entity models

Aim: To help the user write a good specification by focusing on entities within the system and relationships between them.

Description: The desired system is described as a collection of objects and their relationships. The tool enables one to determine which relationships can be interpreted by the system. In general, the relationships permit a description of the hierarchical structure of the objects, the data flow, the relationships between the data, and which data are subject to certain manufacturing processes. The classical procedure has been extended for process control applications. Inspection capabilities and support for the user depend on the variety of relationships illustrated. On the other hand, a large number of representation possibilities makes the application of this technique complex.

References:

PSL/PSA Computer-aided Technique for Structured Documentation and Analysis of Information Processing. D. Teichroew, E. A. Hershey, IEEE Trans on SE, Jan 1977.

Computer Aided Software Development. D. Teichroew, E. A. Hershey, Y. Lamamoto, Beitrag in: Verfahren und Hilfsmittel fur Spezifikation und Entwurf von Prozeßautomatisierungs-systemen. Hommel (ed.), Bericht KfK-PDV 154, Kernforschungszentrum Karlsruhe, 1978.

PCSL und ESPRESO – zwei Ansätze zur Formalisierung der Prozessrechner Software-spezifikation. J. Ludewig, GI-Fachtagung Prozessrechner 1981, Informatik-Fachberichte Bd. 39, Springer Verlag, Berlin, 1981.

B.2.4.5 Interrogation et réponse

But: Aider l'utilisateur à écrire une spécification valable en identifiant les relations entre les rapports entre les réponses aux différents stimuli.

Description: Les relations entre les objets du système sont spécifiées dans une notation qui rend compte des stimuli et des réponses. Un langage simple et facilement élargi est utilisé; il contient des éléments de langage qui représentent des objets, des relations, des caractéristiques et des structures.

Références:

A Requirements Engineering Methodology for Real-time Processing Requirements. M. W. Alford, IEEE Trans on SE, January 1977.

The Specification System X-SPEX – Introduction and Experiences. G. Dahll, J. Lahti, Proc. SAFECOMP '83, 111-118.

B.2.5 Listes de contrôle

NOTE Cette technique/mesure est mentionnée dans les tableaux B.1, B.2 et B.6 de la CEI 61508-2 et dans les tableaux A.10 et B.8 de la CEI 61508-3.

But: Attirer l'attention et gérer l'évaluation critique de tous les aspects importants du système par une phase du cycle de vie de sécurité assurant une couverture complète sans établir les exigences exactes.

Description: Ensemble de questions auxquelles la personne effectuant la liste de contrôle doit répondre. Beaucoup de questions sont d'ordre général et la personne chargée de l'évaluation doit les interpréter de la façon qui semble la plus appropriée au système particulier évalué. Les listes de contrôle peuvent être utilisées pour toutes les phases du cycle de vie de sécurité global des logiciels et des E/E/PES et sont particulièrement utiles comme outil pour l'évaluation de la sécurité fonctionnelle.

Pour s'adapter à une grande diversité des systèmes à valider, la plupart des listes de contrôle comportent des questions applicables à plusieurs types de systèmes. Par conséquent, il peut très bien y avoir des questions dans la liste de contrôle utilisée qui sont inadaptées au système considéré et qu'il convient donc d'ignorer. Il se peut aussi qu'il faille, pour un système particulier, ajouter à la liste de contrôle standard des questions spécifiques relatives à ce système.

Dans tous les cas, il convient qu'il soit clair que l'utilisation des listes de contrôle dépende essentiellement de la compétence et du jugement de l'ingénieur choisissant et appliquant la liste de contrôle. Par conséquent, il convient que les décisions prises par l'ingénieur et concernant la ou les listes de contrôle choisies, ainsi que toutes les questions supplémentaires ou superflues, soient entièrement documentées et justifiées. L'objectif est de s'assurer que l'application des listes de contrôle peut être examinée et que les mêmes résultats seront obtenus, sauf si des critères différents sont utilisés.

Lorsqu'on prépare la liste de contrôle, il faut être aussi concis que possible. Quand une longue justification est nécessaire, il convient que cela soit fait par référence à de la documentation supplémentaire. Il convient d'utiliser des critères tels que «réussite, échec, non concluant», ou des types de réponses restrictives similaires, pour documenter les résultats de chaque question. Cette précision simplifie beaucoup la procédure pour obtenir une conclusion générale des résultats de l'évaluation de la liste de contrôle.

B.2.4.5 Incentive and answer

Aim: To help the user write a good specification by identifying stimulus-response relationships.

Description: The relationships between the objects of the system are specified in a notation of "incentives" and "answers". A simple and easily expanded language is used which contains language elements which represent objects, relationships, characteristics and structures.

References:

A Requirements Engineering Methodology for Real-time Processing Requirements. M. W. Alford, IEEE Trans on SE, January 1977.

The Specification System X-SPEX – Introduction and Experiences. G. Dahll, J. Lahti, Proc. SAFECOMP '83, 111-118.

B.2.5 Checklists

NOTE This technique/measure is referenced in tables B.1, B.2 and B.6 of IEC 61508-2 and in tables A.10 and B.8 of IEC 61508-3.

Aim: To draw attention to, and manage critical appraisal of, all important aspects of the system by safety lifecycle phase, ensuring comprehensive coverage without laying down exact requirements.

Description: A set of questions to be answered by the person performing the checklist. Many of the questions are of a general nature and the assessor must interpret them as seems most appropriate to the particular system being assessed. Checklists can be used for all phases of the overall, E/E/PES and software safety lifecycles and are particularly useful as a tool to aid the functional safety assessment.

To accommodate wide variations in systems being validated, most checklists contain questions which are applicable to many types of system. As a result, there may well be questions in the checklist being used which are not relevant to the system being dealt with and which should be ignored. Equally there may be a need, for a particular system, to supplement the standard checklist with questions specifically directed at the system being dealt with.

In any case it should be clear that the use of checklists depends critically on the expertise and judgement of the engineer selecting and applying the checklist. As a result, the decisions taken by the engineer, with regard to the checklist(s) selected, and any additional or superfluous questions, should be fully documented and justified. The objective is to ensure that the application of the checklists can be reviewed and that the same results will be achieved unless different criteria are used.

The object in completing a checklist is to be as concise as possible. When extensive justification is necessary this should be done by reference to additional documentation. Pass, fail and inconclusive, or some similar restricted set of responses should be used to document the results for each question. This conciseness greatly simplifies the procedure of reaching an overall conclusion as to the results of the checklist assessment.

Références:

CEI 61346 (toutes les parties), *Systèmes industriels, installations et appareils, et produits industriels – Principes de structuration et désignations de référence*.

CEI 60880:1986, *Logiciel pour les calculateurs utilisés dans les systèmes de sûreté des centrales nucléaires*.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Programmable Electronic Systems (PES) in Safety Related Application. Health and Safety Executive, UK, 1987.

Dependability of Critical Computer Systems 2, F. J. Redmill, Elsevier Applied Science, 1989, ISBN 1-85166-381-9.

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

B.2.6 Inspection de la spécification

NOTE Cette technique/mesure est mentionnée dans les tableaux B.1 et B.6 de la CEI 61508-2.

But: Eviter l'incomplétude et la contradiction dans la spécification.

Description: L'inspection est une technique générale dans laquelle diverses qualités d'un document de spécification sont évaluées par une équipe indépendante. L'équipe d'inspection remet les questions au créateur qui doit y répondre de façon satisfaisante. Il convient que l'évaluation soit (si possible) effectuée par une équipe qui n'était pas impliquée dans la création de la spécification. Le degré d'indépendance requis est déterminé par les niveaux d'intégrité de sécurité exigés du système. Il convient que les inspecteurs indépendants puissent reconstruire la fonction opérationnelle du système de façon indiscutable sans se référer à d'autres spécifications. Il faut qu'ils vérifient aussi que tous les aspects techniques et de sécurité pertinents des mesures organisationnelles et opérationnelles sont couverts. Cette procédure s'est avérée très efficace en pratique.

Références:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

CEI 61160:1992, *Revue de conception formalisée*.

B.3 Conception et développement des E/E/PES

Objectif général: Produire une conception stable du système relatif à la sécurité, conforme à la spécification.

B.3.1 Respect des lignes directrices et des normes

NOTE Cette technique/mesure est mentionnée dans le tableau B.2 de la CEI 61508-2.

But: Respecter les normes du secteur d'application (non précisées dans cette norme).

Description: Il convient que les lignes directrices soient suivies lors de la conception du système relatif à la sécurité. Il convient que ces lignes directrices donnent d'abord des systèmes relatifs à la sécurité pratiquement sans défaillances et ensuite facilitent la validation de sécurité qui s'ensuit. Elles peuvent être universelles, spécifiques à un projet ou spécifiques à une seule phase.

References:

IEC 61346 (all parts), *Industrial systems, installations and equipment and industrial products – Structuring principles and reference designation*.

IEC 60880:1986, *Software for computers in the safety systems of nuclear power stations*.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Programmable Electronic Systems (PES) in Safety Related Application. Health and Safety Executive, UK, 1987.

Dependability of Critical Computer Systems 2, F. J. Redmill, Elsevier Applied Science, 1989, ISBN 1-85166-381-9.

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

B.2.6 Inspection of the specification

NOTE This technique/measure is referenced in tables B.1 and B.6 of IEC 61508-2.

Aim: To avoid incompleteness and contradiction in the specification.

Description: Inspection is a general technique in which various qualities of a specification document are assessed by an independent team. The inspection team puts questions to the creator, who must answer them satisfactorily. The examination should (if possible) be carried out by a team that was not involved in the creation of the specification. The required degree of independence is determined by the safety integrity levels demanded of the system. The independent inspectors should be able to reconstruct the operational function of the system in an indisputable manner without referring to any further specifications. They must also check that all relevant safety and technical aspects in the operational and organisational measures are covered. This procedure has proved itself to be very effective in practice.

References:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

IEC 61160:1992, *Formal design review*.

B.3 E/E/PES design and development

Global objective: To produce a stable design of the safety-related system in conformance with the specification.

B.3.1 Observance of guidelines and standards

NOTE This technique/measure is referenced in table B.2 of IEC 61508-2.

Aim: To observe application sector standards (not specified in this standard).

Description: Guidelines should be complied with during the design of the safety-related system. These guidelines should firstly lead to safety-related systems which are practically free from failures, and secondly facilitate the subsequent safety validation. They can be universally valid, specific to a project, or specific only to a single phase.

Références:

EWICS European Workshop on Industrial Computer Systems, TC 7: Safety Related Computers – Software Development and Systems Documentation. Verlag TÜV Rheinland, Köln, 1985.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Deutsche Bundesbahn: Richtlinien-Entwürfe 42500 to 42550 für das Handbuch "Grundsätze zur technischen Zulassung in der Signal- und Nachrichtentechnik". Bundesbahn-Zentralamt München, August 1987.

Richtlinie zur Erstellung und Prüfung sicherheitsrelevanter Software. K. Grimm, G. Heiner, Informatik Fachberichte 86, Springer Verlag, Berlin, 277-288, 1984.

B.3.2 Conception structurée

NOTE Cette technique/mesure est mentionnée dans les tableaux B.2 et B.6 de la CEI 61508-2.

But: Réduire la complexité en créant une structure hiérarchique d'exigences partielles. Eviter les défaillances d'interface entre les exigences. Simplifier la vérification.

Description: Lors de la conception du matériel, il convient d'utiliser des critères ou des méthodes spécifiques. Par exemple, ce qui suit peut être exigé:

– une conception de circuit hiérarchiquement structurée;
– une utilisation de composants de circuit fabriqués et testés.

De même, lors de la conception du logiciel, l'utilisation d'organigrammes structurés permet de créer une structure non ambiguë de modules logiciels. Cette structure présente la manière suivant laquelle les différents modules sont en relation, les données précises qui sont échangées entre les modules, et les commandes exactes qui existent entre les modules.

Références:

Structured Development for Real-Time Systems (3 Volumes). P. T. Yourdon, P. T. Yourdon Press, 1985.

Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

Essential Systems Analysis. St. M. McMenamin, F. Palmer, Yourdon Inc, 1984.

The Use of Structured Methods in the Development of Large Software-Based Avionic Systems. D. J. Hatley, Proceedings DASC, Baltimore, 1984.

An Overview of JSD, J. R. Cameron, IEEE Trans SE-12 No. 2, February 1986.

System Development. M. Jackson, Prentice-Hall, 1983.

MASCOT 3 User Guide. MASCOT Users Forum, RSRE, Malvern, England, 1987.

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward, S. J. Mellor, Yourdon Press, 1985.

Structured Analysis for Requirements Definition, D. T. Ross, K. E. Schoman Jr, IEEE Trans Software Eng, Vol. SE-3, 6-15, 1977.

Structured Analysis (SA): A language for communicating ideas. D. T. Ross, IEEE Trans. Software Eng, Vol. SE-3 (1), 16-34.

References:

EWICS European Workshop on Industrial Computer Systems, TC 7: Safety Related Computers – Software Development and Systems Documentation. Verlag TÜV Rheinland, Köln, 1985.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Deutsche Bundesbahn: Richtlinien-Entwürfe 42500 to 42550 für das Handbuch "Grundsätze zur technischen Zulassung in der Signal- und Nachrichtentechnik". Bundesbahn-Zentralamt München, August 1987.

Richtlinie zur Erstellung und Prüfung sicherheitsrelevanter Software. K. Grimm, G. Heiner, Informatik Fachberichte 86, Springer Verlag, Berlin, 277-288, 1984.

B.3.2 Structured design

NOTE This technique/measure is referenced in tables B.2 and B.6 of IEC 61508-2.

Aim: To reduce complexity by creating a hierarchical structure of partial requirements. To avoid interface failures between the requirements. To simplify verification.

Description: When designing the hardware, specific criteria or methods should be used. For example, the following might be required:

– a hierarchically structured circuit design;

– use of manufactured and tested circuit parts.

Similarly, when designing the software, the use of structure charts enables an unambiguous structure of the software modules to be created. This structure shows how the modules relate to each other, the precise data which passes between modules, and the precise controls that exist between modules.

References:

Structured Development for Real-Time Systems (3 Volumes). P. T. Yourdon, P. T. Yourdon Press, 1985.

Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

Essential Systems Analysis. St. M. McMenamin, F. Palmer, Yourdon Inc, 1984.

The Use of Structured Methods in the Development of Large Software-Based Avionic Systems. D. J. Hatley, Proceedings DASC, Baltimore, 1984.

An Overview of JSD, J. R. Cameron, IEEE Trans SE-12 No. 2, February 1986.

System Development. M. Jackson, Prentice-Hall, 1983.

MASCOT 3 User Guide. MASCOT Users Forum, RSRE, Malvern, England, 1987.

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward, S. J. Mellor, Yourdon Press, 1985.

Structured Analysis for Requirements Definition, D. T. Ross, K. E. Schoman Jr, IEEE Trans. Software Eng, Vol. SE-3, 6-15, 1977.

Structured Analysis (SA): A language for communicating ideas. D. T. Ross, IEEE Trans. Software Eng, Vol. SE-3 (1), 16-34.

Applications and Extensions of SADT. D. T. Ross, Computer, 25-34, April 1985.

Structured Analysis and Design Technique – Application on Safety Systems. W. Heins, Risk Assessment and Control Courseware, Module B1, chapter 11, Delft University of Technology, Safety Science Group, PO Box 5050, 2600 GB Delft, Netherlands, 1989.

CEI 61346 (toutes les parties), *Systèmes industriels, installations et appareils, et produits industriels – Principes de structuration et désignations de référence*.

B.3.3 Utilisation de composants ayant fait leurs preuves

NOTE Cette technique/mesure est mentionnée dans les tableaux B.2 et B.6 de la CEI 61508-2.

But: Réduire le risque de nombreuses anomalies nouvelles et non détectées en utilisant des composants ayant des caractéristiques spécifiques.

Description: Le choix de composants éprouvés est effectué par le fabricant, pour ce qui est de la sécurité, en fonction de la fiabilité des composants (par exemple l'utilisation d'unités physiques testées en exploitation pour satisfaire aux exigences élevées de sécurité ou le stockage de programmes relatifs à la sécurité dans des mémoires sûres uniquement). La sécurité des mémoires peut se rapporter à un accès non autorisé à des influences environnementales (compatibilité électromagnétique, radiation, etc.) ainsi qu'à la réaction des composants en cas de défaillance.

Références:

Reliability Testing for Industrial use. W. T. Greenwood, Computer 10 (7), 26-30, 1977.

Independent Test Labs: Caveat Emptor. E. Rubinstein, IEEE Spectrum, 14 (6), 44-50, 1977.

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

CEI 61163-1:1995, *Déverminage sous contraintes – Partie 1: Entités réparables fabriquées en lots*.

Zuverlässigkeit elektronischer Komponenten. T. Bajenescu, VDE-Verlag, Berlin, 1985.

B.3.4 Modularisation

NOTE Cette technique/mesure est mentionnée dans les tableaux B.2 et B.6 de la CEI 61508-2.

But: Réduire la complexité et éviter les défaillances liées à l'interfaçage de sous-systèmes.

Description: Chaque sous-système, à tous les niveaux de la conception, est clairement défini et est de taille limitée (quelques fonctions uniquement). Les interfaces entre les sous-systèmes sont maintenues aussi simples que possible et les interactions (c'est-à-dire les données communes, l'échange d'informations) sont réduites au minimum. La complexité des sous-systèmes individuels est aussi limitée.

Références:

EWICS European Workshop on Industrial Computer Systems, TC 7: Safety Related Computers – Software Development and Systems Documentation. TÜV Rheinland, Köln, 1985.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Software Reliability – Principles and Practices. G. J. Myers, Wiley-Interscience, New York, 1976.

Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

Applications and Extensions of SADT. D. T. Ross, Computer, 25-34, April 1985.

Structured Analysis and Design Technique – Application on Safety Systems. W. Heins, Risk Assessment and Control Courseware, Module B1, chapter 11, Delft University of Technology, Safety Science Group, PO Box 5050, 2600 GB Delft, Netherlands, 1989.

IEC 61346 (all parts), *Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations*.

B.3.3 Use of well-tried components

NOTE This technique/measure is referenced in tables B.2 and B.6 of IEC 61508-2.

Aim: To reduce the risk of numerous first time and undetected faults by the use of components with specific characteristics.

Description: The selection of well-tried components is carried out by the manufacturer, with regard to safety according to the reliability of the components (for example the use of operationally tested physical units to meet high safety requirements, or the storing of safety-related programs in safe memories only). The safety of memories can refer to unauthorised access as well as environmental influences (electromagnetic compatibility, radiation, etc) and the response of the components in the event of a failure occurring.

References:

Reliability Testing for Industrial use. W. T. Greenwood, Computer 10 (7), 26-30, 1977.

Independent Test Labs: Caveat Emptor. E. Rubinstein, IEEE Spectrum, 14 (6), 44-50, 1977.

Microcomputers in safety technique – an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

IEC 61163-1:1995, *Reliability stress screening – Part 1: Repairable items manufactured in lots*.

Zuverlässigkeit elektronischer Komponenten. T. Bajenescu, VDE-Verlag, Berlin, 1985.

B.3.4 Modularisation

NOTE This technique/measure is referenced in tables B.2 and B.6 of IEC 61508-2.

Aim: To reduce complexity and avoid failures, related to interfacing between subsystems.

Description: Every subsystem, at all levels of the design, is clearly defined and is of restricted size (only a few functions). The interfaces between subsystems are kept as simple as possible and the cross-section (i.e. shared data, exchange of information) is minimised. The complexity of individual subsystems is also restricted.

References:

EWICS European Workshop on Industrial Computer Systems, TC 7: Safety Related Computers – Software Development and Systems Documentation. TÜV Rheinland, Köln, 1985.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Software Reliability – Principles and Practices. G. J. Myers, Wiley-Interscience, New York, 1976.

Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

B.3.5 Outils de conception assistée par ordinateur

NOTE Cette technique/mesure est mentionnée dans les tableaux B.2 et B.6 de la CEI 61508-2 et dans le tableau A.4 de la CEI 61508-3.

But: Effectuer la procédure de conception plus systématiquement. Inclure des éléments de construction automatiques appropriés qui sont déjà disponibles et éprouvés.

Description: Il convient que les outils de conception assistée par ordinateur (CAO) soient utilisés pour la conception du matériel et des logiciels lorsqu'ils existent et qu'ils soient justifiés par la complexité du système. Il convient que la précision de ces outils soit prouvée par des tests spécifiques, par un rapport détaillé sur une utilisation satisfaisante ou par une vérification indépendante de leurs résultats pour le système relatif à la sécurité particulier en cours de conception.

Références:

Verification – The Practical Problems. J. T. Webb and D. J. Mannering, SARSS 87, November 1987, Altrincham, England, Elsevier Applied Science, 1987, ISBN 1-85166-167-0.

An Experience in Design and Validation of Software for a Reactor Protection System. S. Bologna, E. de Agostino et al, IFAC Workshop, SAFECOMP 1979, Stuttgart, 16-18 May 1979, Pergamon Press, 1979.

B.3.6 Simulation

NOTE Cette technique/mesure est mentionnée dans les tableaux B.2, B.5 et B.6 de la CEI 61508-2.

But: Effectuer une inspection complète et systématique d'un circuit électrique/électronique, des caractéristiques fonctionnelles et du dimensionnement correct des composants.

Description: La fonction du circuit du système relatif à la sécurité est simulée sur ordinateur par un modèle comportemental de logiciel. Les composants individuels du circuit ont chacun leur propre comportement simulé et la réponse du circuit dans lequel ils sont assemblés est étudiée en observant les spécifications aux limites de chaque composant.

Références:

Proc. Working Conference on Prototyping. Namur, October 1983, Budde et al, Springer Verlag, 1984.

Using an executable specification language for an information system. S. Urban et al, IEEE Trans Software Engineering, Vol. SE-11 n° 7, July 1985.

Verification and validation of Real-time Software. W. J. Quirk (ed.), Springer Verlag, Berlin, 1985.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.3.7 Inspection (revues et analyses)

NOTE Cette technique/mesure est mentionnée dans les tableaux B.2 et B.6 de la CEI 61508-2.

But: Révéler les écarts entre la spécification et la réalisation.

Description: Les fonctions spécifiées du système relatif à la sécurité sont étudiées et évaluées pour s'assurer que le système relatif à la sécurité est conforme aux exigences de la spécification. Tout point de doute concernant la réalisation et l'utilisation du produit est documenté afin qu'il puisse être résolu. Contrairement à un sondage, l'auteur est passif et l'inspecteur est actif pendant la procédure d'inspection.

B.3.5 Computer-aided design tools

NOTE This technique/measure is referenced in tables B.2 and B.6 of IEC 61508-2 and in table A.4 of IEC 61508-3.

Aim: To carry out the design procedure more systematically. To include appropriate automatic construction elements which are already available and tested.

Description: Computer-aided design tools (CAD) should be used during the design of both hardware and software when available and justified by the complexity of the system. The correctness of such tools should be demonstrated by specific testing, by an extensive history of satisfactory use, or by independent verification of their output for the particular safety-related system that is being designed.

References:

Verification – The Practical Problems. J. T. Webb and D. J. Mannering, SARSS 87, November 1987, Altrincham, England, Elsevier Applied Science, 1987, ISBN 1-85166-167-0.

An Experience in Design and Validation of Software for a Reactor Protection System. S. Bologna, E. de Agostino et al, IFAC Workshop, SAFECOMP 1979, Stuttgart, 16-18 May 1979, Pergamon Press, 1979.

B.3.6 Simulation

NOTE This technique/measure is referenced in tables B.2, B.5 and B.6 of IEC 61508-2.

Aim: To carry out a systematic and complete inspection of an electrical/electronic circuit, of both the functional performance and the correct dimensioning of the components.

Description: The function of the safety-related system circuit is simulated on a computer via a software behavioural model. Individual components of the circuit each have their own simulated behaviour, and the response of the circuit in which they are connected is examined by looking at the marginal data of each component.

References:

Proc. Working Conference on Prototyping. Namur, October 1983, Budde et al, Springer Verlag, 1984.

Using an executable specification language for an information system. S. Urban et al, IEEE Trans Software Engineering, Vol. SE-11 No. 7, July 1985.

Verification and validation of Real-time Software. W. J. Quirk (ed.), Springer Verlag, Berlin, 1985.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.3.7 Inspection (reviews and analysis)

NOTE This technique/measure is referenced in tables B.2 and B.6 of IEC 61508-2.

Aim: To reveal discrepancies between the specification and implementation.

Description: Specified functions of the safety-related system are examined and evaluated to ensure that the safety-related system conforms to the requirements given in the specification. Any points of doubt concerning the implementation and use of the product are documented so they may be resolved. In contrast to a walk-through, the author is passive and the inspector is active during the inspection procedure.

Références:

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

ANSI/IEE Std. 1028:1997, IEEE Standard for software reviews and audits.

B.3.8 Sondage

NOTE Cette technique/mesure est mentionnée dans le tableau B.6 de la CEI 61508-2.

But: Révéler les écarts entre la spécification et la réalisation.

Description: Les fonctions spécifiées du projet du système relatif à la sécurité sont étudiées et évaluées pour s'assurer que le système relatif à la sécurité satisfait aux exigences exposées dans la spécification. Les doutes et points faibles éventuels concernant la réalisation et l'utilisation du produit sont documentés pour qu'ils puissent être résolus. Contrairement à une inspection, l'auteur est actif et l'inspecteur est passif lors du sondage.

Références:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

Methodisches Testen von Programmen. G. J. Myers, Oldenbourg Verlag, München, Wien, 1987.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

ANSI/IEE Std. 1028:1997, IEEE Standard for software reviews and audits.

B.4 Procédures d'exploitation et de maintenance des E/E/PES

Objectif général: Développer des procédures qui aident à éviter les défaillances au cours de l'exploitation et de la maintenance du système relatif à la sécurité.

B.4.1 Instructions d'exploitation et de maintenance

NOTE Cette technique/mesure est mentionnée dans le tableau B.4 de la CEI 61508-2.

But: Eviter des erreurs pendant l'exploitation et la maintenance du système relatif à la sécurité.

Description: Les instructions pour l'utilisateur contiennent les informations essentielles sur la façon d'utiliser et de maintenir le système relatif à la sécurité. Dans des cas particuliers, il y aura aussi dans ces instructions des exemples sur la façon d'installer le système relatif à la sécurité en général. Il faut que toutes les instructions soient facilement compréhensibles. Il convient que des schémas soient utilisés pour décrire les interdépendances et les procédures complexes.

Référence: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

References:

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

ANSI/IEE Std. 1028:1997, IEEE Standard for software reviews and audits.

B.3.8 Walk-through

NOTE This technique/measure is referenced in table B.6 of IEC 61508-2.

Aim: To reveal discrepancies between the specification and implementation.

Description: Specified functions of the safety-related system draft are examined and evaluated to ensure that the safety-related system complies with the requirements given in the specification. Doubts and potential weak points concerning the realisation and use of the product are documented so that they may be resolved. In contrast to an inspection, the author is active and the inspector is passive during the walk-through.

References:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

Methodisches Testen von Programmen. G. J. Myers, Oldenbourg Verlag, München, Wien, 1987.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

ANSI/IEE Std. 1028:1997, IEEE Standard for software reviews and audits.

B.4 E/E/PES operation and maintenance procedures

Global objective: To develop procedures which help to avoid failures during the operation and maintenance of the safety-related system.

B.4.1 Operation and maintenance instructions

NOTE This technique/measure is referenced in table B.4 of IEC 61508-2.

Aim: To avoid mistakes during operation and maintenance of the safety-related system.

Description: User instructions contain essential information on how to use and how to maintain the safety-related system. In special cases, these instructions will also include examples on how to install the safety-related system in general. All instructions must be easily understood. Figures and schematics should be used to describe complex procedures and dependencies.

Reference: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.4.2 Convivialité en termes d'utilisation

NOTE Cette technique/mesure est mentionnée dans le tableau B.4 de la CEI 61508-2.

But: Réduire la complexité pendant l'exploitation du système relatif à la sécurité.

Description: La bonne exploitation du système relatif à la sécurité peut dépendre, dans une certaine mesure, de l'homme. Prenant en compte la conception du système qui convient et l'agencement du lieu de travail, la personne qui développe le système relatif à la sécurité doit s'assurer que

– le besoin d'intervention de l'homme est réduit au strict minimum;
– l'intervention nécessaire est aussi simple que possible;
– le risque de dommage potentiel résultant d'erreurs de l'utilisateur est réduit;
– les moyens d'intervention et les indications sont conçus en fonction d'exigences ergonomiques;
– les moyens mis à la disposition de l'opérateur sont simples, bien repérés et d'utilisation intuitive;
– l'opérateur n'est pas surchargé, même dans les situations extrêmes;
– la formation sur les procédures et les moyens d'intervention est adaptée au niveau de connaissance et de motivation de l'utilisateur qui reçoit la formation.

B.4.3 Convivialité en termes de maintenance

NOTE Cette technique/mesure est mentionnée dans le tableau B.4 de la CEI 61508-2.

But: Simplifier les procédures de maintenance du système relatif à la sécurité et concevoir les moyens nécessaires pour un diagnostic et une réparation efficaces.

Description: La maintenance préventive et la réparation sont souvent effectuées dans des conditions difficiles et sous la pression des délais. Par conséquent, la personne qui développe le système relatif à la sécurité doit s'assurer que

– les mesures de maintenance relative à la sécurité sont nécessaires aussi rarement que possible; l'idéal serait qu'elles ne soient pas du tout nécessaires;
– des outils de diagnostic suffisants, sensibles et faciles à manipuler sont inclus pour les réparations inévitables; il convient que les outils comprennent toutes les interfaces nécessaires;
– si des outils de diagnostic séparés doivent être développés ou obtenus, il convient alors qu'ils soient disponibles à temps.

B.4.4 Possibilités d'exploitation limitées

NOTE Cette technique/mesure est mentionnée dans les tableaux B.4 et B.6 de la CEI 61508-2.

But: Réduire les possibilités d'exploitation pour l'utilisateur normal.

Description: Cette approche réduit les possibilités d'exploitation en

– limitant l'exploitation dans des modes d'exploitation spéciaux, par exemple par des interrupteurs à clé;
– limitant le nombre d'éléments en exploitation;
– limitant le nombre de modes d'exploitation généralement possibles.

Référence: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.4.2 User friendliness

NOTE This technique/measure is referenced in table B.4 of IEC 61508-2.

Aim: To reduce complexity during operation of the safety-related system.

Description: The correct operation of the safety-related system may depend to some degree on human operation. By considering the relevant system design and the design of the workplace, the safety-related system developer must ensure that

– the need for human intervention is restricted to an absolute minimum;

– the necessary intervention is as simple as possible;

– the potential for harm from operator error is minimised;

– the intervention facilities and indication facilities are designed according to ergonomic requirements;

– the operator facilities are simple, well labelled and intuitive to use;

– the operator is not overstrained, even in extreme situations;

– training on intervention procedures and facilities is adapted to the level of knowledge and motivation of the trainee user.

B.4.3 Maintenance friendliness

NOTE This technique/measure is referenced in table B.4 of IEC 61508-2.

Aim: To simplify maintenance procedures of the safety-related system and to design the necessary means for effective diagnosis and repair.

Description: Preventive maintenance and repair is often carried out under difficult circumstances and under pressure from deadlines. Therefore, the safety-related system developer should ensure that

– safety-related maintenance measures are necessary as seldom as possible or even, ideally, not necessary at all;

– sufficient, sensible and easy-to-handle diagnosis tools are included for those repairs that are unavoidable – tools should include all necessary interfaces;

– if separate diagnosis tools have to be developed or obtained, then these should be available on time.

B.4.4 Limited operation possibilities

NOTE This technique/measure is referenced in tables B.4 and B.6 of IEC 61508-2.

Aim: To reduce the operation possibilities for the normal user.

Description: This approach reduces the operation possibilities by

– limiting the operation within special operating modes, for example by key switches;

– limiting the number of operating elements;

– limiting the number of generally possible operating modes.

Reference: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.4.5 Exploitation uniquement par des opérateurs qualifiés

NOTE Cette technique/mesure est mentionnée dans les tableaux B.4 et B.6 de la CEI 61508-2.

But: Eviter les défaillances en exploitation dues à une mauvaise utilisation.

Description: L'opérateur du système relatif à la sécurité est formé à un niveau correspondant au niveau d'intégrité de sécurité et de complexité du système relatif à la sécurité. La formation comprend l'apprentissage des bases du procédé de production et la connaissance de la relation entre le système relatif à la sécurité et l'EUC.

Référence: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.4.6 Protection contre les erreurs humaines

NOTE Cette technique/mesure est mentionnée dans le tableau B.6 de la CEI 61508-2.

But: Protéger le système contre tous les types d'erreurs de l'opérateur.

Description: Les entrées erronées (valeur, temps, etc.) sont détectées par des contrôles de vraisemblance ou par surveillance de l'EUC. Pour intégrer ces moyens dans la conception, il est nécessaire d'établir très tôt quelles entrées sont possibles et quelles entrées sont autorisées.

B.4.7 (Non utilisé)

B.4.8 Protection contre les modifications

NOTE Cette technique/mesure est mentionnée dans le tableau A.18 de la CEI 61508-2.

But: Protéger le système relatif à la sécurité contre les modifications du matériel par des moyens techniques.

Description: Les modifications ou manipulations sont automatiquement détectées, par exemple par des contrôles de vraisemblance des signaux des capteurs, une détection par des procédés techniques et par des tests de démarrage automatiques. Si une modification est détectée, alors une action d'urgence est entreprise.

B.4.9 Accusé de réception des entrées

NOTE Cette technique/mesure est mentionnée dans les tableaux A.18 et A.19 de la CEI 61508-2.

But: Une erreur pendant l'exploitation est détectée par l'opérateur lui-même avant d'activer l'EUC.

Description: Une entrée dans l'EUC par le système relatif à la sécurité est renvoyée à l'opérateur avant d'être envoyée à l'EUC de façon que l'opérateur puisse détecter et corriger une erreur. Tout comme les actions personnelles non provoquées et anormales, il convient que la conception du système prenne en compte les limites de vitesse supérieure et inférieure et le sens de la réaction humaine. Cela éviterait, par exemple, que l'opérateur appuie sur les touches trop vite et que le système prenne deux frappes pour une seule ou qu'une même touche soit frappée deux fois parce que le système (affichage) aura été trop lent à réagir à la première frappe. Il convient que la frappe de la même touche ne soit pas valide plus d'une fois à la suite pour la saisie de données critiques; il ne faut pas que le fait d'appuyer sur les touches «entrée» ou «oui» un nombre de fois illimité conduise à une action dangereuse du système.

Il convient que des procédures de temporisation soient prévues avec des questions à choix multiples (oui/non, etc.) pour les fois où l'opérateur hésite et laisse le système en attente.

La possibilité de réinitialiser un PES relatif à la sécurité rend le système vulnérable, à moins que le logiciel/matériel ne soient conçus en ayant à l'esprit de telles possibilités.

Référence: DIN V VDE 0801: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Beuth-Verlag, Berlin, 1990.

B.4.5 Operation only by skilled operators

NOTE This technique/measure is referenced in tables B.4 and B.6 of IEC 61508-2.

Aim: To avoid operating failures caused by misuse.

Description: The safety-related system operator is trained to a level which is appropriate to the complexity and safety integrity level of the safety-related system. Training includes studying the background of the production process and knowing the relationship between the safety-related system and the EUC.

Reference: Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.4.6 Protection against operator mistakes

NOTE This technique/measure is referenced in table B.6 of IEC 61508-2.

Aim: To protect the system against all classes of operator mistakes.

Description: Wrong inputs (value, time, etc) are detected via plausibility checks or monitoring of the EUC. To integrate these facilities into the design it is necessary to state at a very early stage which inputs are possible and which are permissible.

B.4.7 (Not used)

B.4.8 Modification protection

NOTE This technique/measure is referenced in table A.18 of IEC 61508-2.

Aim: To protect the safety-related system against hardware modifications by technical means.

Description: Modifications or manipulations are detected automatically, for example by plausibility checks for the sensor signals, detection by the technical process and by automatic start-up tests. If a modification is detected, then emergency action is taken.

B.4.9 Input acknowledgement

NOTE This technique/measure is referenced in tables A.18 and A.19 of IEC 61508-2.

Aim: A mistake during operation is detected by the operator himself before activating the EUC.

Description: An input to the EUC via the safety-related system is echoed to the operator before being sent to the EUC so that the operator has the possibility to detect and correct a mistake. As well as abnormal, unprovoked personnel action, the system design should consider top/bottom speed limits and direction of human reaction. This would avoid, for example, the operator pressing keys faster than expected, causing the system to read a double keystroke as a single one, or a key to be pressed twice because the system (display) was too slow to react to the first instance. The same key stroke should not be valid more than once in succession for critical data entry; pressing the "enter" or "yes" key unlimited times must not lead to an unsafe action of the system.

Time-out procedures should be included with multiple choice questions (yes/no, etc.) to cater for when the operator may not make up his mind and leave the system waiting.

Ability to reboot a safety-related PES makes the system vulnerable unless both software/hardware are designed with such occasions in mind.

Reference: DIN V VDE 0801: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Beuth-Verlag, Berlin, 1990.

B.5 Intégration des E/E/PES

Objectif général: Eviter les défaillances pendant la phase d'intégration et révéler toutes les défaillances qui ont lieu pendant cette phase et pendant la phase précédente.

B.5.1 Test fonctionnel

NOTE Cette technique/mesure est mentionnée dans les tableaux B.3 et B.5 de la CEI 61508-2 et dans les tableaux A.5, A.6 et A.7 de la CEI 61508-3.

But: Révéler les défaillances pendant les phases de spécification et de conception. Eviter les défaillances lors de la réalisation et de l'intégration du logiciel et du matériel.

Description: Pendant les tests fonctionnels, des revues sont effectuées pour voir si les caractéristiques spécifiées du système ont été respectées. Le système reçoit les paramètres qui caractérisent correctement le fonctionnement normalement attendu. Les sorties sont observées et leur réaction est comparée avec celle indiquée dans la spécification. Les écarts par rapport à la spécification et les indications d'une spécification incomplète sont documentés.

Le test fonctionnel des composants électroniques conçus pour une architecture à plusieurs canaux implique habituellement que les composants fabriqués soient testés avec des composants similaires préalablement validés. En plus de cela, il est recommandé de tester les composants fabriqués conjointement avec d'autres composants similaires du même lot afin d'identifier les anomalies de mode commun qui autrement seraient restées masquées.

Il faut aussi que la puissance de travail du système soit suffisante, voir C.5.20.

Références:

Functional Program Testing and Analysis. W. E. Howden, McGraw-Hill, 1987.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

B.5.2 Test «boîte noire»

NOTE Cette technique/mesure est mentionnée dans les tableaux B.3, B.5 et B.6 de la CEI 61508-2 et dans les tableaux A.5, A.6 et A.7 de la CEI 61508-3.

But: Contrôler le comportement dynamique dans des conditions réelles de fonctionnement. Révéler les défaillances pour que la spécification fonctionnelle soit satisfaite et évaluer son utilité et sa robustesse.

Description: Les fonctions d'un système ou d'un programme sont exécutées dans un environnement spécifié avec des données de test spécifiées déduites systématiquement de la spécification conformément aux critères établis. Cela met en évidence le comportement du système et permet une comparaison avec la spécification. Aucune connaissance de la structure interne du système n'est utilisée pour diriger le test. Le but est de déterminer si l'unité fonctionnelle effectue correctement toutes les fonctions exigées par la spécification. La technique consistant à former des classes d'équivalence est un exemple des critères pour définir des données de test «boîte noire». L'ensemble des données d'entrée est subdivisé en des fourchettes de valeurs d'entrée (classes d'équivalence) à l'aide de la spécification. Les cas de test sont donc formés à partir

– des données dans les fourchettes permises;
– des données en dehors des fourchettes permises;
– des données aux limites des fourchettes;
– des valeurs extrêmes;
– des combinaisons des classes ci-dessus.

B.5 E/E/PES integration

Global objective: To avoid failures during the integration phase and to reveal any failures that are made during this and previous phases.

B.5.1 Functional testing

NOTE This technique/measure is referenced in tables B.3 and B.5 of IEC 61508-2 and in tables A.5, A.6 and A.7 of IEC 61508-3.

Aim: To reveal failures during the specification and design phases. To avoid failures during implementation and the integration of software and hardware.

Description: During the functional tests, reviews are carried out to see whether the specified characteristics of the system have been achieved. The system is given input data which adequately characterises the normally expected operation. The outputs are observed and their response is compared with that given by the specification. Deviations from the specification and indications of an incomplete specification are documented.

Functional testing of electronic components designed for a multi-channel architecture usually involves the manufactured components being tested with pre-validated partner components. In addition to this, it is recommended that the manufactured components be tested in combination with other partner components of the same batch, in order to reveal common mode faults which would otherwise have remained masked.

Also, the working capacity of the system has to be sufficient, see guidance in C.5.20.

References:

Functional Program Testing and Analysis. W. E. Howden, McGraw-Hill, 1987.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

B.5.2 Black-box testing

NOTE This technique/measure is referenced in tables B.3, B.5 and B.6 of IEC 61508-2 and in tables A.5, A.6 and A.7 of IEC 61508-3.

Aim: To check the dynamic behaviour under real functional conditions. To reveal failures to meet functional specification, and to assess utility and robustness.

Description: The functions of a system or program are executed in a specified environment with specified test data which is derived systematically from the specification according to established criteria. This exposes the behaviour of the system and permits a comparison with the specification. No knowledge of the internal structure of the system is used to guide the testing. The aim is to determine whether the functional unit carries out correctly all the functions required by the specification. The technique of forming equivalence classes is an example of the criteria for blackbox test data. The input data space is subdivided into specific input value ranges (equivalence classes) with the aid of the specification. Test cases are then formed from the

- data from permissible ranges;
- data from inadmissible ranges;
- data from the range limits;
- extreme values;
- and combinations of the above classes.

D'autres critères peuvent être efficaces pour choisir les cas de test dans les différentes activités de test (test du module, test d'intégration et test du système). Par exemple, pour le test du système dans le cadre d'une validation, on se fie au critère «conditions d'exploitation extrêmes».

Références:

Functional Testing and Analysis. W. E. Howden, McGraw-Hill Book Company, New York, 1987.

Software Testing and Validation Techniques. E. Miller, W. E. Howden, IEEE Computer Society, New York, 1978.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Methodik systematischen Testens von Software. K. Grimm, 30 (4), 1988.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.5.3 Test statistique

NOTE Cette technique/mesure est mentionnée dans les tableaux B.3, B.5 et B.6 de la CEI 61508-2.

But: Vérifier le comportement dynamique du système relatif à la sécurité et évaluer son utilité et sa robustesse.

Description: Cette approche teste un système ou un programme avec des données d'entrée choisies en fonction de la distribution statistique attendue des données d'entrée réelles, c'est-à-dire le profil opérationnel.

Références:

Software Testing via Environmental Simulation (CONTESSE Report). Available until December 1998 from: Ray Browne, CIID, DTI, 151 Buckingham Palace Road, London, SW1W 9SS, UK, 1994.

Aspects of Development and Verification of Reliable Process Computer Software. W. Ehrenberger, IFAC-IFIP Conference Proceedings, 35-48, 1980.

Verification and validation of Real-time Software. W. J. Quirk (ed.), Springer Verlag, Berlin, 1985.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

B.5.4 Retour d'expérience

NOTE 1 Voir aussi en C.2.10 une mesure similaire et à l'annexe D une approche statistique, dans le contexte du logiciel.

NOTE 2 Cette technique/mesure est mentionnée dans les tableaux B.3 et B.5 de la CEI 61508-2.

But: Utiliser le retour d'expérience de différentes applications comme l'une des mesures permettant d'éviter les anomalies, soit pendant l'intégration E/E/PES, et/ou pendant la validation de sécurité E/E/PES.

Other criteria can be effective in order to select test cases in the various test activities (module test, integration test and system test). For example, the criterion "extreme operational conditions" is relied upon for the system test within the framework of a validation.

References:

Functional Testing and Analysis. W. E. Howden, McGraw-Hill Book Company, New York, 1987.

Software Testing and Validation Techniques. E. Miller, W. E. Howden, IEEE Computer Society, New York, 1978.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Methodik systematischen Testens von Software. K. Grimm, 30 (4), 1988.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.5.3 Statistical testing

NOTE This technique/measure is referenced in tables B.3, B.5 and B.6 of IEC 61508-2.

Aim: To check the dynamic behaviour of the safety-related system and to assess its utility and robustness.

Description: This approach tests a system or program with input data selected according to the expected statistical distribution of the real operating inputs – the operational profile.

References:

Software Testing via Environmental Simulation (CONTESSE Report). Available until December 1998 from: Ray Browne, CIID, DTI, 151 Buckingham Palace Road, London, SW1W 9SS, UK, 1994.

Aspects of Development and Verification of Reliable Process Computer Software. W. Ehrenberger, IFAC-IFIP Conference Proceedings, 35-48, 1980.

Verification and validation of Real-time Software. W. J. Quirk (ed.), Springer Verlag, Berlin, 1985.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

B.5.4 Field experience

NOTE 1 See also C.2.10 for a similar measure and annex D for a statistical approach, both in the context of software.

NOTE 2 This technique/measure is referenced in tables B.3 and B.5 of IEC 61508-2.

Aim: To use field experience from different applications as one of the measures to avoid faults either during E/E/PES integration and/or during E/E/PES safety validation.

Description: Utilisation de composants ou de sous-systèmes dont l'utilisation montre qu'ils n'ont pas d'anomalies, ou alors seulement des anomalies sans importance, lors d'un fonctionnement pratiquement sans remplacement pendant une période suffisante et dans beaucoup d'applications différentes. La personne qui développe le système doit faire attention aux fonctions qui ont en fait été validées par le retour d'expérience, en particulier pour les composants complexes avec une multitude de fonctions possibles (par exemple le système d'exploitation, les circuits intégrés). Par exemple, considérons les sous-programmes d'autotest pour la détection d'anomalies: s'il n'y a pas de panne du matériel pendant l'exploitation, ces sous-programmes ne peuvent pas être considérés comme validés par l'expérience dans la mesure où ils n'ont jamais effectué leur fonction de détection d'anomalies.

Pour que la mention «validé par le retour d'expérience» puisse être appliquée, les conditions suivantes doivent être remplies:

– spécification inchangée;
– 10 systèmes dans des applications différentes;
– 10^5 heures en exploitation et au moins un an d'historique de service.

Le retour d'expérience est démontré par la documentation du fournisseur et/ou de la société d'exploitation. Il faut que cette documentation contienne au moins

– la désignation exacte du système et de son composant, y compris la vérification de la version du matériel;
– les utilisateurs et le temps d'application;
– le nombre d'heures de fonctionnement;
– les procédures pour le choix des systèmes et applications obtenus pour l'argumentation;
– les procédures pour la détection et l'enregistrement des anomalies ainsi que pour leur correction.

Références:

DIN V VDE 0801 A1: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Änderung 1 zu DIN V VDE 0801/01.90. Beuth-Verlag, Berlin, 1994.

Guidelines for safe automation of chemical processes. CCPS, AIChE, New York, 1993.

B.6 Validation de la sécurité des E/E/PES

Objectif général: Prouver que le système E/E/PE relatif à la sécurité est conforme à la spécification relative aux exigences de sécurité des E/E/PES.

B.6.1 Tests fonctionnels dans des conditions environnementales

NOTE Cette technique/mesure est mentionnée dans le tableau B.5 de la CEI 61508-2.

But: Evaluer si le système relatif à la sécurité est protégé contre les influences environnementales habituelles.

Description: Le système est soumis à diverses conditions environnementales (conformément aux normes des séries CEI 60068 ou CEI 61000 par exemple) pendant lesquelles la fiabilité des fonctions de sécurité (ainsi que leur compatibilité avec les normes mentionnées) sont évaluées.

Description: Use of components or subsystems, which have been shown by experience to have no, or only unimportant, faults when used, essentially unchanged, over a sufficient period of time in numerous different applications. Particularly for complex components with a multitude of possible functions (for example operating system, integrated circuits), the developer shall pay attention to which functions have actually been tested by the field experience. For example, consider self-test routines for fault detection: if no break-down of the hardware occurs within the operating period, the routines cannot be said to have been tested, since they have never performed their fault detection function.

For field experience to apply, the following requirements must have been fulfilled:

– unchanged specification;

– 10 systems in different applications;

– 10^5 operating hours and at least one year of service history.

The field experience is demonstrated through documentation of the vendor and/or operating company. This documentation must contain at least

– the exact designation of the system and its component, including version control for hardware;

– the users and time of application;

– the operating hours;

– the procedures for the selection of the systems and applications procured to the proof;

– the procedures for fault detection and fault registration as well as fault removal.

References:

DIN V VDE 0801 A1: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Änderung 1 zu DIN V VDE 0801/01.90. Beuth-Verlag, Berlin, 1994.

Guidelines for safe automation of chemical processes. CCPS, AIChE, New York, 1993.

B.6 E/E/PES safety validation

Global objective: To prove that the E/E/PE safety-related system conforms to the E/E/PES safety requirements specification.

B.6.1 Functional testing under environmental conditions

NOTE This technique/measure is referenced in table B.5 of IEC 61508-2.

Aim: To assess whether the safety-related system is protected against typical environmental influences.

Description: The system is put under various environmental conditions (for example according to the standards in the IEC 60068 series or the IEC 61000 series), during which the safety functions are assessed for their reliability (and compatibility with the standards mentioned).

Références:

CEI 61000-4-1:1992, *Compatibilité électromagnétique (CEM) – Partie 4: Techniques d'essai et de mesure – Section 1: Vue d'ensemble sur les essais d'immunité*.

CEI 60068-1:1988, *Essais d'environnement – Première partie: Généralités et guide*.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

B.6.2 Essai d'immunité aux interférences et aux ondes de choc

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.6 de la CEI 61508-2.

But: Evaluer la capacité du système relatif à la sécurité à supporter les ondes de choc.

Description: Le système est chargé avec un programme d'application typique et toutes les lignes périphériques (toutes les interfaces numériques, analogiques et série, ainsi que les connexions et l'alimentation du bus, etc.) sont soumises à des signaux parasites normalisés. Afin d'obtenir un état quantitatif, il est recommandé d'approcher prudemment la limite de l'onde de choc. La catégorie de parasites choisie n'est pas respectée si une défaillance apparaît.

Référence: Guide for surge withstand capability (SWC) test. ANSI C.37.90-1974.

CEI 61000-4-5:1995, *Compatibilité électromagnétique (CEM) – Partie 4: Techniques d'essai et de mesure – Section 5: Essai d'immunité aux ondes de choc*.

B.6.3 (Non utilisé)

B.6.4 Analyse statique

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.6 de la CEI 61508-2 et dans le tableau A.9 de la CEI 61508-3.

But: Eviter les anomalies systématiques qui peuvent causer des pannes du système testé, soit au début soit après des années d'exploitation.

Description: Cette approche systématique et éventuellement assistée par ordinateur inspecte les caractéristiques statiques spécifiques du système prototype pour assurer la complétude, la cohérence et l'absence d'ambiguïté par rapport à l'exigence en question (par exemple recommandations de construction, spécifications du système et fiche technique de l'appareil). Une analyse statique est reproductible. Elle est appliquée à un prototype qui a atteint un stade bien défini de réalisation. Des exemples d'analyse statique pour le matériel et les logiciels sont

– l'analyse de cohérence du flux de données (telle que le test consistant en la vérification qu'un objet de donnée est interprété partout avec la même valeur);
– l'analyse du flux de commandes (telle que la détermination du chemin d'accès ou de code inaccessible);
– l'analyse de l'interface (telle que la recherche du transfert de variables entre différents modules logiciels);
– l'analyse du flux de données pour détecter les séquences suspectes de création, désignation et suppression de variables;
– le test du respect des recommandations spécifiques (par exemple distances dans l'air et lignes de fuite, distance d'assemblage, disposition de l'unité physique, unités physiques mécaniquement sensibles, utilisation exclusive des unités physiques qui ont été mises en place).

References:

IEC 61000-4-1:1992, *Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 1: Overview of immunity tests*.

IEC 60068-1:1988, *Environmental testing – Part 1: General and guidance*.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

B.6.2 Interference surge immunity testing

NOTE This technique/measure is referenced in tables B.5 and B.6 of IEC 61508-2.

Aim: To check the capacity of the safety-related system to handle peak surges.

Description: The system is loaded with a typical application program, and all the peripheral lines (all digital, analogue and serial interfaces as well as the bus connections and power supply, etc.) are subjected to standard noise signals. In order to obtain a quantitative statement, it is sensible to approach the surge limit carefully. The chosen class of noise is not complied with if the function fails.

References:

Guide for surge withstand capability (SWC) test. ANSI C.37.90-1974.

IEC 61000-4-5:1995, *Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 5: Surge immunity testing*.

B.6.3 (Not used)

B.6.4 Static analysis

NOTE This technique/measure is referenced in tables B.5 and B.6 of IEC 61508-2 and in table A.9 of IEC 61508-3.

Aim: To avoid systematic faults that can lead to breakdowns in the system under test, either early or after many years of operation.

Description: This systematic and possibly computer-aided approach inspects specific static characteristics of the prototype system to ensure completeness, consistency, lack of ambiguity regarding the requirement in question (for example construction guidelines, system specifications, and an appliance data sheet). A static analysis is reproducible. It is applied to a prototype which has reached a well-defined stage of completion. Some examples of static analysis, for hardware and software, are

– consistency analysis of the data flow (such as testing if a data object is interpreted everywhere as the same value);

– control flow analysis (such as path determination, determination of non-accessible code);

– interface analysis (such as investigation of variable transfer between various software modules);

– dataflow analysis to detect suspicious sequences of creating, referencing and deleting variables;

– testing adherence to specific guidelines (for example creepage distances and clearances, assembly distance, physical unit arrangement, mechanically sensitive physical units, exclusive use of the physical units which were introduced).

Référence:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.6.5 Analyse dynamique

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.6 de la CEI 61508-2 et dans les tableaux A.5 et A.9 de la CEI 61508-3.

But: Détecter les défaillances de la spécification en inspectant le comportement dynamique d'un prototype à un stade avancé de sa réalisation.

Description: L'analyse dynamique d'un système relatif à la sécurité est effectuée en soumettant un prototype presque opérationnel du système relatif à la sécurité à des données d'entrée typiques de l'environnement prévu en exploitation. L'analyse est satisfaisante si le comportement observé du système relatif à la sécurité est conforme au comportement requis. Il faut corriger toute défaillance du système relatif à la sécurité et analyser à nouveau la nouvelle version opérationnelle.

Références:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.6.6 Analyse des défaillances

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.6 de la CEI 61508-2.

B.6.6.1 Analyse des modes de défaillance et de leurs effets

But: Analyser une conception du système en étudiant toutes les sources possibles de défaillances des composants de ce système et en déterminant les effets de ces défaillances sur le comportement et la sécurité du système.

Description: L'analyse est généralement faite lors d'une réunion d'ingénieurs. Les composants d'un système sont analysés l'un après l'autre pour établir l'ensemble des modes de défaillance du composant, leurs causes et leurs effets, ainsi que des procédures et recommandations pour la détection. Si les recommandations sont prises en charge, elles sont documentées comme des actions de correction qui ont été prises.

Références:

CEI 60812:1985, *Techniques d'analyse de la fiabilité des systèmes – Procédure d'analyse des modes de défaillance et de leurs effets (AMDE)*.

System Reliability Engineering Methodology: A Discussion of the State of the Art. J. B. Fussel, J. S. Arend, Nuclear Safety 20 (5), 1979.

Reliability Technology. A. E. Green, A. J. Bourne, Wiley-Interscience, 1972.

Fault Tree Handbook. W. E. Vesely et al, NUREG-0942, Division of System Safety Office of Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission, Washington, DC 20555, 1981.

References:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.6.5 Dynamic analysis

NOTE This technique/measure is referenced in tables B.5 and B.6 of IEC 61508-2 and in tables A.5 and A.9 of IEC 61508-3.

Aim: To detect specification failures by inspecting the dynamic behaviour of a prototype at an advanced state of completion.

Description: The dynamic analysis of a safety-related system is carried out by subjecting a near-operational prototype of the safety-related system to input data which is typical of the intended operating environment. The analysis is satisfactory if the observed behaviour of the safety-related system conforms to the required behaviour. Any failure of the safety-related system must be corrected and the new operational version must then be reanalysed.

References:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

VDI-Gemeinschaftsausschuß Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.6.6 Failure analysis

NOTE This technique/measure is referenced in tables B.5 and B.6 of IEC 61508-2.

B.6.6.1 Failure modes and effects analysis

Aim: To analyse a system design, by examining all possible sources of failure of a system's components and determining the effects of these failures on the behaviour and safety of the system.

Description: The analysis usually takes place through a meeting of engineers. Each component of a system is analysed in turn to give a set of failure modes for the component, their causes and effects, detection procedures and recommendations. If the recommendations are acted upon, they are documented as remedial action taken.

References:

IEC 60812:1985, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA).*

System Reliability Engineering Methodology: A Discussion of the State of the Art. J. B. Fussel, J. S. Arend, Nuclear Safety 20 (5), 1979.

Reliability Technology. A. E. Green, A. J. Bourne, Wiley-Interscience, 1972.

Fault Tree Handbook. W. E. Vesely et al, NUREG-0942, Division of System Safety Office of Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission, Washington, DC 20555, 1981.

B.6.6.2 Diagramme cause-conséquence

NOTE Cette technique/mesure est mentionnée dans les tableaux A.10, B.3 et B.4 de la CEI 61508-3.

But: Modéliser, sous forme graphique, la séquence des événements qui peut se développer dans un système comme conséquence d'une association d'événements initiaux.

Description: Cette technique peut être considérée comme une association d'analyses par arbre de panne et par arbre d'événement. Partant d'un événement critique, un diagramme cause-conséquence est tracé en amont et en aval. Vers l'amont, il équivaut à un arbre de panne, l'événement critique étant l'événement amont donné. Vers l'aval, les conséquences possibles d'un événement sont définies. Le graphique peut comporter des symboles graphiques décrivant les conditions de propagation le long des différentes branches partant du sommet. Des temporisations peuvent aussi être incluses. Ces conditions peuvent aussi être décrites avec des arbres de panne. Les lignes de propagation peuvent être associées à des symboles logiques pour rendre le diagramme plus compact. Un ensemble de symboles standard est défini pour une utilisation dans les diagrammes cause-conséquence. Les diagrammes peuvent être utilisés pour calculer la probabilité de certaines conséquences critiques.

Référence: The Cause Consequence Diagram Method as a Basis for Quantitative Accident Analysis. B. S. Nielsen, Riso-M-1374, 1971.

B.6.6.3 Analyse par arbre d'événement

NOTE Cette technique/mesure est mentionnée dans le tableau B.4 de la CEI 61508-3.

But: Modéliser, sous forme graphique, la séquence des événements qui peut se développer dans un système après un événement initiateur et indiquer ainsi comment des conséquences graves peuvent se produire.

Description: Au sommet du diagramme apparaissent les conditions de la séquence qui sont pertinentes pour la progression des événements suivant l'événement initiateur. Partant sous l'événement initiateur, qui est la cible de l'analyse, une ligne est tracée vers la première condition de la séquence. Là, le diagramme se divise en deux branches, «oui» et «non», décrivant ainsi comment les événements futurs dépendent de la condition. Pour chacune de ces branches, on continue vers la condition suivante de la même façon. Toutes les conditions, toutefois, ne conviennent pas à toutes les branches. On continue jusqu'à la fin de la séquence et chaque branche de l'arbre ainsi construit représente une conséquence possible. L'arbre des événements peut servir à calculer la probabilité des différentes conséquences sur la base de la probabilité et du nombre de conditions dans la séquence.

Référence: Event Trees and their Treatment on PC Computers. N. Limnious and J. P. Jeannette, Reliability Engineering, Vol. 18, n° 3, 1987.

B.6.6.4 Analyse des modes de défaillance, de leurs effets et de leur criticité

NOTE Cette technique/mesure est mentionnée dans les tableaux A.10 et B.4 de la CEI 61508-3.

But: Etablir un ordre de criticité des composants qui pourraient être à l'origine d'une blessure, d'un préjudice ou d'une dégradation du système par le biais de défaillances uniques, afin de déterminer quels composants peuvent avoir besoin d'une attention particulière et de mesures de surveillance pendant la conception ou l'exploitation.

Description: La criticité peut être classée de plusieurs façons. La méthode la plus laborieuse est décrite par la Society for Automotive Engineers (SAE) dans la norme ARP 926. Dans cette procédure, le degré de criticité d'un composant est indiqué par le nombre de défaillances d'un type particulier attendu au cours d'un cycle d'un million d'utilisations se produisant dans un mode critique. Le degré de criticité est une fonction de neuf paramètres, dont la plupart doivent être mesurés. Une méthode très simple pour déterminer la criticité est de multiplier la probabilité de la défaillance du composant par celle du préjudice qui pourrait être causé; cette méthode est similaire à l'évaluation simple du facteur de risque.

B.6.6.2 Cause consequence diagrams

NOTE This technique/measure is referenced in tables A.10, B.3 and B.4 of IEC 61508-3.

Aim: To model, in a diagrammatic form, the sequence of events that can develop in a system as a consequence of combinations of basic events.

Description: The technique can be regarded as a combination of fault tree and event tree analysis. Starting from a critical event, a cause consequence graph is traced backwards and forwards. In the backward direction it is equivalent to a fault tree with the critical event as the given top event. In the forward direction the possible consequences arising from an event are determined. The graph can contain vertex symbols which describe the conditions for propagation along different branches from the vertex. Time delays can also be included. These conditions can also be described with fault trees. The lines of propagation can be combined with logical symbols, to make the diagram more compact. A set of standard symbols is defined for use in cause consequence diagrams. The diagrams can be used to compute the probability of occurrence of certain critical consequences.

Reference: The Cause Consequence Diagram Method as a Basis for Quantitative Accident Analysis. B. S. Nielsen, Riso-M-1374, 1971.

B.6.6.3 Event tree analysis

NOTE This technique/measure is referenced in table B.4 of IEC 61508-3.

Aim: To model, in a diagrammatic form, the sequence of events that can develop in a system after an initiating event, and thereby indicate how serious consequences can occur.

Description: On the top of the diagram is written the sequence conditions that are relevant in the progression of events that follow the initiating event. Starting under the initiating event, which is the target of the analysis, a line is drawn to the first condition in the sequence. There the diagram branches off into "yes" and "no" branches, describing how future events depend on the condition. For each of these branches, one continues to the next condition in a similar way. Not all conditions are, however, relevant for all branches. One continues to the end of the sequence, and each branch of the tree constructed in this way represents a possible consequence. The event tree can be used to compute the probability of the various consequences, based on the probability and number of conditions in the sequence.

Reference: Event Trees and their Treatment on PC Computers. N. Limnious and J. P. Jeannette, Reliability Engineering, Vol. 18, No. 3, 1987.

B.6.6.4 Failure modes, effects and criticality analysis

NOTE This technique/measure is referenced in tables A.10 and B.4 of IEC 61508-3.

Aim: To rank the criticality of components which could result in injury, damage or system degradation through single-point failures, in order to determine which components might need special attention and necessary control measures during design or operation.

Description: Criticality can be ranked in many ways. The most laborious method is described by the Society for Automotive Engineers (SAE) in ARP 926. In this procedure, the criticality number for any component is indicated by the number of failures of a specific type expected during each million operations occurring in a critical mode. The criticality number is a function of nine parameters, most of which have to be measured. A very simple method for criticality determination is to multiply the probability of component failure by the damage that could be generated; this method is similar to simple risk factor assessment.

Références:

Design Analysis Procedure for Failure Modes, Effects and Criticality Analysis (FMECA). Aerospace Recommended Practice (ARP) 926, Society of Automotive Engineers (SAE), USA, 15 September 1967.

CEI 60812:1985, *Techniques d'analyse de la fiabilité des systèmes – Procédure d'analyse des modes de défaillance et de leurs effets (AMDE)*.

B.6.6.5 Analyse par arbre de panne

NOTE Cette technique/mesure est mentionnée dans les tableau B.4 de la CEI 61508-3.

But: Faciliter l'analyse des événements ou des associations d'événements qui auront une conséquence grave ou dangereuse.

Description: Partant d'un événement qui serait la cause immédiate d'une conséquence grave ou dangereuse (l'événement amont), l'analyse est effectuée le long d'un chemin arborescent. Les associations de causes sont décrites avec des opérateurs logiques (et, ou, etc.). Les causes intermédiaires sont analysées de la même façon et ainsi de suite jusqu'aux événements initiaux où s'arrête l'analyse.

La méthode est graphique et un ensemble de symboles normalisés est utilisé pour tracer l'arbre de panne. La technique est principalement destinée à l'analyse des systèmes matériels, mais on a aussi essayé d'appliquer cette approche à l'analyse des défaillances de logiciels.

Références:

CEI 61025:1990, *Analyse par arbre de panne (AAP)*.

System Reliability Engineering Methodology: A Discussion of the State of the Art. J. B. Fussel and J. S. Arend, Nuclear Safety 20 (5), 1979.

Fault Tree Handbook. W. E. Vesely et al, NUREG-0492, Division of System Safety Office of Nuclear Reactor Regulation, US Nuclear Regulatory Commission Washington, DC 20555, 1981.

Reliability Technology. A. E. Greene and A. J. Bourne, Wiley-Interscience, 1972.

B.6.7 Analyse des cas les plus défavorables

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.6 de la CEI 61508-2.

But: Eviter les défaillances systématiques provenant d'associations défavorables des conditions de l'environnement et des tolérances des composants.

Description: L'aptitude à l'exploitation du système et le dimensionnement du composant sont étudiés ou calculés sur une base théorique. Les conditions de l'environnement sont modifiées et on leur attribue leurs valeurs limites les plus élevées possibles. Les réactions essentielles du système sont inspectées et comparées avec la spécification.

B.6.8 Test fonctionnel étendu

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.6 de la CEI 61508-2.

But: Révéler les défaillances au cours des phases de spécification, de conception et de développement. Contrôler le comportement du système relatif à la sécurité en cas d'entrées rares ou non spécifiées.

References:

Design Analysis Procedure for Failure Modes, Effects and Criticality Analysis (FMECA). Aerospace Recommended Practice (ARP) 926, Society of Automotive Engineers (SAE), USA, 15 September 1967.

IEC 60812:1985, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*.

B.6.6.5 Fault tree analysis

NOTE This technique/measure is referenced in table B.4 of IEC 61508-3.

Aim: To aid in the analysis of events, or combinations of events, that will lead to a hazard or serious consequence.

Description: Starting at an event which would be the immediate cause of a hazard or serious consequence (the "top event"), analysis is carried out along a tree path. Combinations of causes are described with logical operators (and, or, etc). Intermediate causes are analysed in the same way, and so on, back to basic events where analysis stops.

The method is graphical, and a set of standardised symbols are used to draw the fault tree. The technique is mainly intended for the analysis of hardware systems, but there have also been attempts to apply this approach to software failure analysis.

References:

IEC 61025:1990, *Fault tree analysis (FTA)*.

System Reliability Engineering Methodology: A Discussion of the State of the Art. J. B. Fussel and J. S. Arend, Nuclear Safety 20 (5), 1979.

Fault Tree Handbook. W. E. Vesely et al, NUREG-0492, Division of System Safety Office of Nuclear Reactor Regulation, US Nuclear Regulatory Commission Washington, DC 20555, 1981.

Reliability Technology. A. E. Greene and A. J. Bourne, Wiley-Interscience, 1972.

B.6.7 Worst-case analysis

NOTE This technique/measure is referenced in tables B.5 and B.6 of IEC 61508-2.

Aim: To avoid systematic failures which arise from unfavourable combinations of the environmental conditions and the component tolerances.

Description: The operational capacity of the system and the component dimensioning is examined or calculated on a theoretical basis. The environmental conditions are changed to their highest permissible marginal values. The most essential responses of the system are inspected and compared with the specification.

B.6.8 Expanded functional testing

NOTE This technique/measure is referenced in tables B.5 and B.6 of IEC 61508-2.

Aim: To reveal failures during the specification and design and development phases. To check the behaviour of the safety-related system in the event of rare or unspecified inputs.

Description: Le test fonctionnel étendu étudie le comportement fonctionnel du système relatif à la sécurité en réponse aux conditions d'entrée qui ne sont supposées se produire que rarement (par exemple défaillance grave) ou qui ne sont pas dans la spécification du système relatif à la sécurité (par exemple mauvaise utilisation). Pour les conditions rares, le comportement observé du système relatif à la sécurité est comparé à la spécification. Lorsque la réaction du système relatif à la sécurité n'est pas spécifiée, il faut contrôler que la sécurité de l'usine est préservée par la réaction observée.

Références:

Functional Program Testing and Analysis. W. E. Howden, McGraw-Hill, 1987.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

B.6.9 Test du cas le plus défavorable

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.6 de la CEI 61508-2.

But: Tester les cas spécifiés lors de l'analyse des cas les plus défavorables.

Description: L'aptitude à l'exploitation du système et le dimensionnement des composants sont testés dans les conditions des cas les plus défavorables. Les conditions environnementales sont modifiées et on leur attribue leurs valeurs limites les plus élevées possibles. Les réactions essentielles du système sont inspectées et comparées avec la spécification.

B.6.10 Test d'insertion d'anomalie

NOTE Cette technique/mesure est mentionnée dans les tableaux B.5 et B.6 de la CEI 61508-2.

But: Introduire ou simuler des erreurs dans le matériel du système et documenter sa réaction.

Description: Il s'agit d'une méthode qualitative d'évaluation de la fiabilité. Il est préférable d'utiliser des schémas de câblages, des schémas électriques et des schémas de blocs fonctionnels détaillés pour décrire l'emplacement et le type de défaillance et la façon dont elle est introduite; par exemple: l'alimentation de différents modules peut être coupée; l'alimentation, le bus ou les lignes d'adresses peuvent être court-circuités ou en circuit ouvert; les composants ou leurs ports peuvent mis en circuit ouvert ou en court-circuit; les relais peuvent ne pas se fermer ou ne pas s'ouvrir ou le faire au mauvais moment, etc. Les défaillances du système qui en résultent sont classées, comme dans les tableaux I et II de la norme CEI 60812, par exemple. En principe, des anomalies uniques, en régime établi, sont introduites. Toutefois, si une anomalie n'est pas identifiée par les tests de diagnostic intégrés ou n'est pas évidente, elle peut être laissée dans le système et l'effet d'une deuxième anomalie peut être étudié. Le nombre d'anomalies peut facilement atteindre des centaines.

Le travail est effectué par une équipe multidisciplinaire et il convient que le fournisseur du système soit présent et consulté. La moyenne des temps de bon fonctionnement pour les anomalies qui ont de graves conséquences doit être calculée ou estimée. Si le temps calculé est court, il convient d'apporter des modifications.

Références:

Integrity Testing of Process Control Systems. R. J. Lasher, Control Engineering 36 (11), 152-164, October 1989.

CEI 61069-5:1994, *Mesure et commande dans les processus industriels – Appréciation des propriétés d'un système en vue de son évaluation – Partie 5: Evaluation de la sûreté de fonctionnement d'un système*.

CEI 60812:1985, *Techniques d'analyse de la fiabilité des systèmes – Procédure d'analyse des modes de défaillance et de leurs effets (AMDE)*.

Description: Expanded functional testing reviews the functional behaviour of the safety-related system in response to input conditions which are expected to occur only rarely (for example major failure), or which are outside the specification of the safety-related system (for example incorrect operation). For rare conditions, the observed behaviour of the safety-related system is compared with the specification. Where the response of the safety-related system is not specified, one should check that the plant safety is preserved by the observed response.

References:

Functional Program Testing and Analysis. W. E. Howden, McGraw-Hill, 1987.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

B.6.9 Worst-case testing

NOTE This technique/measure is referenced in tables B.5 and B.6 of IEC 61508-2.

Aim: To test the cases specified during worst-case analysis.

Description: The operational capacity of the system and the component dimensioning is tested under worst-case conditions. The environmental conditions are changed to their highest permissible marginal values. The most essential responses of the system are inspected and compared with the specification.

B.6.10 Fault insertion testing

NOTE This technique/measure is referenced in tables B.5 and B.6 of IEC 61508-2.

Aim: To introduce or simulate faults in the system hardware and document the response.

Description: This is a qualitative method of assessing dependability. Preferably, detailed functional block, circuit and wiring diagrams are used in order to describe the location and type of fault and how it is introduced; for example: power can be cut from various modules; power, bus or address lines can be open/short-circuited; components or their ports can be opened or shorted; relays can fail to close or open, or do it at the wrong time, etc. Resulting system failures are classified, as in tables I and II of IEC 60812, for example. In principle, single steady-state faults are introduced. However, in case a fault is not revealed by the built-in diagnostic tests or otherwise does not become evident, it can be left in the system and the effect of a second fault considered. The number of faults can easily increase to hundreds.

The work is done by a multidisciplinary team and the vendor of the system should be present and consulted. The mean operating time between failure for faults that have grave consequences should be calculated or estimated. If the calculated time is low, modifications should be made.

References:

Integrity Testing of Process Control Systems. R. J. Lasher, Control Engineering 36 (11), 152-164, October 1989.

IEC 61069-5:1994, *Industrial-process measurement and control – Evaluation of system properties for the purpose of system assessment – Part 5: Assessment of system dependability*.

IEC 60812:1985, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*.

Annexe C
(informative)

Présentation de techniques et mesures pour l'obtention de l'intégrité de sécurité logicielle (voir la CEI 61508-3)

C.1 Généralités

La présentation des techniques contenue dans la présente annexe ne doit pas être considérée comme complète ou exhaustive.

Documents généraux de référence:

System – Safety Society of America System Safety Analysis Handbook. System Safety Society, New Mexico Chapter. PO Box 95424, Albuquerque NM, USA.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

Encyclopaedia of Software Engineering. Ed. J. Marciniak. John Wiley & Sons, 1994, ISBN 0-471-54004-8.

Software Engineer's Reference Book. Ed. J. McDermid. Butterworth-Heinemann, 1991, ISBN 0-7506-1040-9.

C.2 Prescriptions et conception détaillée

NOTE Des techniques et mesures concernées sont également décrites en B.2.

C.2.1 Méthodes structurées

NOTE Cette technique/mesure est référencée dans les tableaux A.2 et A.4 de la CEI 61508-3.

C.2.1.1 Généralités

But: Le but principal des méthodes structurées est de promouvoir la qualité de développement du logiciel en portant l'attention sur les premières parties du cycle de vie. Pour ce faire, les méthodes font appel à des procédures et notations (assistées par ordinateur) à la fois précises et intuitives, pour déterminer et documenter des prescriptions et des caractéristiques d'implémentation dans un ordre logique et de manière structurée.

Description: Il existe toute une gamme de méthodes structurées. Certaines sont prévues pour les fonctions traditionnelles de traitement des données et des transactions alors que d'autres (MASCOT, JSD, Yourdon temps réel) sont plus orientées commande de procédé et application temps réel (qui tendent à être plus critiques pour la sécurité).

Les méthodes structurées sont essentiellement des «outils de réflexion» permettant l'approche et le partage systématiques d'un problème ou d'un système. Elles comportent les caractéristiques principales suivantes:

– ordre de réflexion logique, partage d'un problème important en parties plus facilement gérables;
– analyse et documentation du système total, comprenant l'environnement aussi bien que le système requis;
– décomposition des données et de la fonction du système requis;
– listes de contrôle, c'est-à-dire listes des éléments nécessitant une analyse;
– faible servitude intellectuelle – méthodes simples, intuitives et pragmatiques.

Annex C
(informative)

Overview of techniques and measures for achieving software safety integrity (see IEC 61508-3)

C.1 General

The overview of techniques contained in this annex should not be regarded as either complete or exhaustive.

Some general references are:

System – Safety Society of America System Safety Analysis Handbook. System Safety Society, New Mexico Chapter. PO Box 95424, Albuquerque NM, USA.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

Encyclopaedia of Software Engineering. Ed. J. Marciniak. John Wiley & Sons, 1994, ISBN 0-471-54004-8.

Software Engineer's Reference Book. Ed. J. McDermid. Butterworth-Heinemann, 1991, ISBN 0-7506-1040-9.

C.2 Requirements and detailed design

NOTE Relevant techniques and measures are found in B.2.

C.2.1 Structured methods

NOTE This technique/measure is referenced in tables A.2 and A.4 of IEC 61508-3.

C.2.1.1 General

Aim: The main aim of structured methods is to promote the quality of software development by focusing attention on the early parts of the lifecycle. The methods aim to achieve this through both precise and intuitive procedures and notations (assisted by computers), to determine and document requirements and implementation features in a logical order and a structured manner.

Description: A range of structured methods exist. Some are designed for traditional data-processing and transaction processing functions, while others (MASCOT, JSD, real-time Yourdon) are more oriented to process control and real-time applications (which tend to be more safety critical).

Structured methods are essentially "thought tools" for systematically perceiving and partitioning a problem or system. Their main features are the following:

– a logical order of thought, breaking a large problem into manageable stages;

– analysis and documentation of the total system, including the environment as well as the required system;

– decomposition of data and function in the required system;

– checklists, i.e. lists of the sort of things that need analysis;

– low intellectual overhead – simple, intuitive, pragmatic.

Les notations supports pour l'analyse et la documentation de problèmes et d'entités du système (par exemple les flux de données et de traitements) tendent à être précises, mais les notations exprimant les fonctions de traitement effectuées par ces entités tendent à être plus informelles. Cependant, certaines méthodes utilisent en partie des notations mathématiques formelles (par exemple, la méthode JSD utilise des expressions régulières et les méthodes Yourdon, SOM et SDL utilisent des machines à états finis). Une précision accrue non seulement réduit l'importance des erreurs de compréhension mais permet le traitement automatique.

Un autre avantage des notations structurées est leur visibilité qui permet la vérification intuitive d'une spécification ou d'une conception par l'utilisateur en fonction de ses connaissances approfondies mais non formalisées.

La présente bibliographie décrit cinq méthodes structurées de façon plus précise: CORE, JSD, MASCOT, Yourdon temps réel et SADT.

Références:

Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward and S. J. Mellor, Yourdon Press, 1985.

Essential Systems Analysis. St. M. McMenamin, F. Palmer, Yourdon Inc, New York, 1984.

The Use of Structured Methods in the Development of Large Software-Based Avionic Systems. D. J. Hatley. Proc. DASC, Baltimore, 1984.

C.2.1.2 CORE – Controlled Requirements Expression

But: S'assurer que toutes les prescriptions sont déterminées et exprimées.

Description: Cette méthode a pour but de combler l'écart entre le client/utilisateur final et l'analyste. Elle n'est pas mathématiquement rigoureuse mais facilite le processus de communication (la méthode CORE permet l'expression des prescriptions plutôt que la spécification). L'approche est structurée et l'expression passe par différents niveaux d'affinement. La méthode CORE permet une vue élargie du problème, en tenant compte de la connaissance de l'environnement dans lequel le système sera utilisé ainsi que des points de vue respectifs des différents types d'utilisateur. Cette méthode comporte les lignes directrices et la tactique à utiliser pour reconnaître les écarts par rapport à la «conception générale». Les écarts peuvent être corrigés ou identifiés explicitement et documentés. Ainsi, les spécifications peuvent ne pas être complètes mais les problèmes non résolus et les zones à haut risque sont détectés et doivent être pris en compte dans la suite de la conception.

Référence: Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

C.2.1.3 JSD – Jackson System Development

But: Méthode de développement couvrant le développement des systèmes logiciels à partir des prescriptions jusqu'au programme, en portant une attention particulière aux systèmes temps réel.

Description: La méthode JSD est une procédure de développement en plusieurs étapes dans laquelle le développeur modélise le comportement du monde réel devant servir de base aux fonctions du système, détermine les fonctions requises et les introduit dans le modèle, puis transforme la spécification résultante en une spécification réalisable dans l'environnement cible. Elle couvre donc les phases traditionnelles de spécification et conception et de développement tout en différant des méthodes traditionnelles par le fait qu'elle n'est pas de conception descendante.

The supporting notations for analysing and documenting problems and system entities (for example processes and data flows) tend to be precise, but notations for expressing the processing functions performed by these entities tend to be more informal. However, some methods do make partial use of (mathematically) formal notations (for example, JSD makes use of regular expressions; Yourdon, SOM and SDL utilise finite state machines). Increased precision not only reduces the scope for misunderstanding, it provides scope for automatic processing.

Another benefit of structured notations is their visibility, enabling a specification or design to be checked intuitively by a user, against his powerful but unstated knowledge.

This overview describes five structured methods in more detail: Controlled Requirements Expression, Jackson System Development, MASCOT, real-time Yourdon, and Structured Analysis and Design Technique (SADT).

References:

Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward and S. J. Mellor, Yourdon Press, 1985.

Essential Systems Analysis. St. M. McMenamin, F. Palmer, Yourdon Inc, New York, 1984.

The Use of Structured Methods in the Development of Large Software-Based Avionic Systems. D. J. Hatley. Proc. DASC, Baltimore, 1984.

C.2.1.2 CORE – Controlled Requirements Expression

Aim: To ensure that all the requirements are determined and expressed.

Description: This approach is intended to bridge the gap between the customer/end user and the analyst. It is not mathematically rigorous but aids communication – CORE is designed for requirements expression rather than specification. The approach is structured, and the expression goes through various levels of refinement. The CORE method encourages a wider view of the problem, bringing in a knowledge of the environment in which the system will be used and the differing viewpoints of the various types of user. CORE includes guidelines and tactics for recognising departures from the "grand design". Departures can be corrected or explicitly identified and documented. Thus specifications may not be complete, but unresolved problems and high-risk areas are detected and have to be considered in the subsequent design.

Reference: Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

C.2.1.3 JSD – Jackson System Development

Aim: A development method covering the development of software systems from requirements through to code, with special emphasis on real-time systems.

Description: JSD is a staged development procedure in which the developer models the real world behaviour upon which the system functions are to be based, determines the required functions and inserts them into the model, and transforms the resulting specification into one that is realisable in the target environment. It therefore covers the traditional phases of specification and design and development but takes a somewhat different view from the traditional methods in not being top-down.

De plus, cette méthode porte une grande attention à l'étape initiale de découverte des entités dans l'environnement réel du système en cours de réalisation ainsi qu'à la modélisation des entités et à ce qui peut leur arriver. Une fois cette analyse du «monde réel» effectuée et le modèle créé, les fonctions requises du système sont analysées afin de déterminer comment elles peuvent être intégrées dans le modèle réel. Les descriptions structurées de tous les processus du modèle sont ajoutées au modèle de système résultant et le tout est alors transformé en programmes susceptibles de fonctionner dans l'environnement logiciel et matériel prévu.

Références:

An Overview of JSD. J. R. Cameron. IEEE Transactions on Software Engineering, SE-12, n° 2, February 1986.

System Development. M. Jackson, Prentice-Hall, 1983.

C.2.1.4 MASCOT – Modular Approach to Software Construction, Operation and Test

But: Conception et mise en œuvre de systèmes temps réel.

Description: La méthode MASCOT est une méthode de conception supportée par un système de programmation. C'est une méthode systématique permettant d'exprimer la structure des systèmes temps réel indépendamment du matériel prévu et du langage d'implémentation. Elle impose une étude de conception disciplinée permettant d'obtenir une structure fortement modulaire tout en assurant une étroite correspondance entre les éléments fonctionnels liés à la conception et les éléments de construction utilisés pour l'intégration du système. Un système est conçu en termes de réseau de processus concurrents qui communiquent à travers des canaux. Ces canaux peuvent être constituées soit de pôles de données fixes soit de files d'attente (pipelines de données). Le contrôle d'accès à ces voies est décrit indépendamment du processus en termes de mécanismes d'accès qui servent également à appliquer les règles d'ordonnancement des processus. Des versions récentes de la méthode MASCOT ont été conçues en tenant compte d'une implémentation à l'aide du langage ADA.

La méthode MASCOT supporte une stratégie de réception basée sur l'essai et la vérification des modules logiciels simples ou d'ensembles plus importants de modules logiciels liés du point de vue fonctionnel. La réalisation d'une implémentation MASCOT est prévue à partir d'un noyau MASCOT (jeu de primitives d'ordonnancement sous-jacent à l'implémentation et supportant les mécanismes d'accès).

Référence: MASCOT 3 User Guide. MASCOT Users Forum. RSRE, Malvern, England, 1987.

C.2.1.5 Yourdon temps réel

But: Spécification et conception de systèmes temps réel.

Description: Le schéma de développement servant de base à cette technique suppose une évolution en trois étapes du système en cours de développement. La première étape comprend la construction d'un «modèle essentiel» décrivant le comportement requis par le système. La deuxième comprend la construction d'un modèle d'implémentation décrivant les structures et les mécanismes qui, lorsqu'ils sont implémentés, concrétisent le comportement requis. La troisième étape comprend la réalisation à proprement parler du système en matériel et logiciel. Les trois étapes correspondent approximativement aux phases traditionnelles de spécification et conception et de développement mais portent une attention plus grande au fait qu'à chaque étape, le développeur est engagé dans une activité de modélisation.

Le modèle essentiel est constitué de deux parties:

– le modèle lié à l'environnement, contenant une description de la frontière entre le système et son environnement ainsi qu'une description des événements externes auxquels il faut que le système réponde; et

Moreover, it places great emphasis on the early stage of discovering the entities in the real world that are the concern of the system being built and on modelling them and what can happen to them. Once this analysis of the "real world" has been done and a model created, the system's required functions are analysed to determine how they can fit into this real-world model. The resulting system model is augmented with structured descriptions of all the processes in the model and the whole is then transformed into programs that will operate in the target software and hardware environment.

References:

An Overview of JSD. J. R. Cameron. IEEE Transactions on Software Engineering, SE-12, No. 2, February 1986.

System Development. M. Jackson, Prentice-Hall, 1983.

C.2.1.4 MASCOT – Modular Approach to Software Construction, Operation and Test

Aim: The design and implementation of real-time systems.

Description: MASCOT is a design method supported by a programming system. It is a systematic method of expressing the structure of real-time systems in a way that is independent of the target hardware or implementation language. It imposes a disciplined approach to design that yields a highly modular structure, ensuring a close correspondence between the functional elements in the design and the construction elements appearing in system integration. A system is designed in terms of a network of concurrent processes that communicate through channels. Channels can be either pools of fixed data or queues (pipelines of data). Control of access to channels is described independently of the processes in terms of access mechanisms that also enforce scheduling rules on the processes. Recent versions of MASCOT have been designed with ADA implementation in mind.

MASCOT supports an acceptance strategy based on the test and verification of single software modules and larger collections of functionally related software modules. A MASCOT implementation is intended to be built upon a MASCOT kernel – a set of scheduling primitives that underlie the implementation and support the access mechanisms.

Reference: MASCOT 3 User Guide. MASCOT Users Forum. RSRE, Malvern, England, 1987.

C.2.1.5 Real-time Yourdon

Aim: The specification and design of real-time systems.

Description: The development scheme underlying this technique assumes a three-stage evolution of a system being developed. The first stage involves the building of an "essential model", one that describes the behaviour required by the system. The second involves the building of an implementation model which describes the structures and mechanisms that, when implemented, embody the required behaviour. The third stage involves the actual building of the system in hardware and software. The three stages correspond roughly to the traditional specification and design and development phases but lay greater emphasis on the fact that at each stage the developer is engaged in a modelling activity.

The essential model is in two parts:

– the environmental model, containing a description of the boundary between the system and its environment and a description of the external events to which the system must respond; and

– le modèle de comportement, contenant les schémas décrivant la transformation effectuée par le système en réponse aux événements ainsi qu'une description des données qu'il faut que le système gère pour répondre.

Le modèle d'implémentation se divise également en sous-modèles, couvrant l'allocation des processus individuels aux processeurs et la décomposition des processus en modules logiciels.

La technique d'acquisition de ces modèles comporte un certain nombre d'autres techniques bien connues: diagrammes de flux de données, graphes de transformation, langage structuré, diagrammes de changement d'état et réseaux de Pétri. De plus, cette méthode comporte des techniques pour simuler la conception du système proposé soit sur papier soit mécaniquement à partir des modèles réalisés.

Références:

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward and S. J. Mellor. Yourdon Press, 1985.

Strategies for Real-time System Specification. D. Hatley, E. Pirbhai, Dorset Publishing House, 1988.

C.2.1.6 SADT – Structured Analysis and Design Technique

But: Modéliser et analyser, sous forme de diagrammes utilisant des flux d'informations, les processus de prise de décision et les tâches de gestion associées à un système complexe.

Description: Dans la méthode SADT, le concept de diagramme appelé actigramme joue un rôle central. Ce diagramme regroupe les activités dans des «boîtes actions». Chaque boîte action possède un nom unique et est reliée aux autres boîtes actions par des liaisons factorielles (représentées par des flèches) possédant également un nom unique. Chaque pavé fonctionnel peut être hiérarchiquement décomposé en pavés fonctionnels et liaisons auxiliaires. Il existe quatre types de facteurs: entrées, commandes, mécanismes et sorties:

– **entrée:** indiquée par une flèche entrante située sur le côté gauche d'un pavé fonctionnel. Les entrées peuvent représenter des éléments matériels ou immatériels et sont susceptibles de concerner une ou plusieurs activités d'un même pavé fonctionnel;
– **commande**: une instruction, une procédure, un critère de choix, etc. La commande guide l'exécution d'une activité et est représentée par une flèche entrante située à la partie supérieure d'un pavé fonctionnel;
– **mécanisme:** ressources telles que personnel, unité administrative ou équipements nécessaires au fonctionnement de l'activité;
– **sortie:** tout élément résultant d'une activité et représenté par une flèche sortante située sur le côté droit d'un pavé fonctionnel.

Lorsque plusieurs activités sont étroitement liées les unes aux autres par un grand nombre de liaisons factorielles, il peut être préférable de considérer ces activités comme un groupe indivisible contenu dans une seule boîte action, sans plus de détails en ce qui concerne le contenu de la boîte. Le principe directeur concernant le regroupement des activités en boîtes actions est que les boîtes résultantes ne doivent être reliées deux à deux que par un petit nombre de facteurs.

La hiérarchisation des actigrammes se poursuit jusqu'à ce que les boîtes actions ne nécessitent plus aucun détail supplémentaire. Ce stade est atteint lorsque les activités à l'intérieur des boîtes sont inséparables or lorsque tout détail supplémentaire relatif aux boîtes sort du cadre consacré à l'analyse du système.

– the behavioural model, which contains schemes describing the transformation carried out by the system in response to events and a description of the data the system must hold in order to respond.

The implementation model also divides into submodels, covering the allocation of individual processes to processors and the decomposition of the processes into software modules.

To capture these models, the technique combines a number of other well-known techniques: data-flow diagrams, transformation graphs, structured English, state transition diagrams and Petri nets. Additionally, the method contains techniques for simulating a proposed system design either on paper or mechanically from the models that are drawn up.

References:

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward and S. J. Mellor. Yourdon Press, 1985.

Strategies for Real-time System Specification. D. Hatley, E. Pirbhai, Dorset Publishing House, 1988.

C.2.1.6 SADT – Structured Analysis and Design Technique

Aim: To model and analyse, in a diagrammatic form using information flows, the decision-making processes and the management tasks associated with a complex system.

Description: In SADT, the concept of an activity-factor diagram plays a central role. An A/F diagram consists of activities grouped in so-called "action boxes". Each action box has a unique name, and is linked to other action boxes by factor relations (drawn as arrows) which are also given unique names. Each action box can be hierarchically decomposed into subsidiary action boxes and relations. There are four types of factors: inputs, controls, mechanisms and outputs:

– **input:** indicated by an arrow that enters an action box at the left-hand side. Inputs can represent material or immaterial things and they are suitable for manipulation by one or more activities in an action box;

– **control:** typically an instruction, procedure, choice criterion or so on. A control guides the execution of an activity and is shown by an arrow entering the top side of an action box;

– **mechanism:** a resource such as personnel, organisational unit or equipment, needed for an activity to perform its task;

– **output:** anything that an activity produces, pictured by an arrow leaving an action box at the right-hand side.

When activities are strongly related to each other by many factor relations, it is perhaps better to consider these activities as an indivisible group, contained in one action box, with no further detailing of its content. The guiding principle for grouping of activities into action boxes is that the resulting boxes are coupled pairwise by only a few factors.

The model hierarchy of A/F diagrams is pursued until a further detailing of the action boxes is meaningless. This stage is reached when the activities within the boxes are inseparable or when further detailing of the action boxes falls outside the scope of the system analysis.

Références:

Structured Analysis for Requirements Definition. D. T. Ross, K. E. Schoman Jr. IEEE Transactions on Software Engineering, Vol. SE-3, 1, 6-15, 1977.

Structured Analysis (SA): A Language for Communicating Ideas. D. T. Ross. IEEE Transactions on Software Engineering, Vol. SE-3, 1, 16-34, 1977.

Applications and Extensions of SADT. D. T. Ross. Computer, 25-34, April 1985.

Structured Analysis and Design Technique – Application on Safety Systems. W. Heins. Risk Assessment and Control Courseware, Module B1, Chapter 11, Delft University of Technology, Safety Science Group, PO Box 5050, 2600 GB Delft, Netherlands, 1989.

C.2.2 Diagrammes de flux de données

NOTE Cette technique/mesure est référencée dans les tableaux B.5 et B.7 de la CEI 61508-3.

But: Décrire le flux de données d'un programme sous forme de diagramme.

Description: Les diagrammes de flux de données montrent comment les données d'entrée sont transformées en données de sortie, chaque étape de l'organigramme montrant une transformation distincte.

Les diagrammes de flux de données comportent trois éléments:

– flèches annotées: elles représentent le flux de données entrant et sortant associé à chaque centre de transformation, avec annotations documentant ce qu'est la donnée;
– bulles annotées: elles représentent les centres de transformation, avec annotations documentant la transformation;
– opérateurs (ET, OU exclusif): ces opérateurs servent à relier les flèches annotées.

Chaque bulle d'un diagramme de flux de données peut être considérée comme une boîte noire autonome qui transforme ses entrées en sorties dès que les premières sont disponibles. Un des principaux avantages des bulles réside dans le fait qu'elles montrent les transformations sans faire aucune supposition en ce qui concerne la manière dont ces transformations sont implémentées. Un diagramme de flux de données pur ne comporte pas d'éléments de commande ou de séquencement, ces éléments étant fournis par des extensions temps réel aux notations, comme dans le cas de la méthode Yourdon temps réel (voir C.2.1.5).

La meilleure méthode pour la préparation des diagrammes de flux de données consiste à considérer les entrées du système puis à traiter les sorties correspondantes. Il faut que chaque bulle représente une transformation distincte (il convient que la sortie de la bulle soit, d'un certain point de vue, différente de son entrée). Il n'existe pas de règles pour déterminer la structure générale du diagramme et la construction d'un diagramme de flux de données constitue l'un des aspects créatifs de la conception d'un système. Comme toutes les autres conceptions, il s'agit d'une procédure itérative dont les premières ébauches sont affinées en vue d'obtenir le diagramme final.

Références:

Software Engineering. I. Sommerville, Addison-Wesley, 3rd Edition, 1989.

ISO 5807:1985, *Traitement de l'information – Symboles de documentation et conventions applicables aux données, aux organigrammes de programmation et d'analyse, aux schémas des réseaux de programmes et des ressources de système*.

ISO/CEI 8631:1989, *Technologies de l'information – Structures de programmes et normes pour leur représentation*.

References:

Structured Analysis for Requirements Definition. D. T. Ross, K. E. Schoman Jr. IEEE Transactions on Software Engineering, Vol. SE-3, 1, 6-15, 1977.

Structured Analysis (SA): A Language for Communicating Ideas. D. T. Ross. IEEE Transactions on Software Engineering, Vol. SE-3, 1, 16-34, 1977.

Applications and Extensions of SADT. D. T. Ross. Computer, 25-34, April 1985.

Structured Analysis and Design Technique – Application on Safety Systems. W. Heins. Risk Assessment and Control Courseware, Module B1, Chapter 11, Delft University of Technology, Safety Science Group, PO Box 5050, 2600 GB Delft, Netherlands, 1989.

C.2.2 Data flow diagrams

NOTE This technique/measure is referenced in tables B.5 and B.7 of IEC 61508-3.

Aim: To describe the data flow through a program in a diagrammatic form.

Description: Data flow diagrams document how data input is transformed to output, with each stage in the diagram representing a distinct transformation.

Data flow diagrams are made up of three components:

– annotated arrows – represent data flow in and out of the transformation centres, with the annotations documenting what the data is;

– annotated bubbles – represent transformation centres, with the annotation documenting the transformation;

– operators (and, xor) – these operators are used to link the annotated arrows.

Each bubble in a data flow diagram can be considered as a stand-alone black box which, as soon as its inputs are available, transforms them to its outputs. One of the principal advantages is that they show transformations without making any assumptions about how these transformations are implemented. A pure data flow diagram does not include control information or sequencing information, but this is catered for by real-time extensions to the notation, as in real-time Yourdon (see C.2.1.5).

The preparation of data flow diagrams is best approached by considering system inputs and working towards system outputs. Each bubble must represent a distinct transformation – its output should, in some way, be different from its input. There are no rules for determining the overall structure of the diagram and constructing a data flow diagram is one of the creative aspects of system design. Like all design, it is an iterative procedure with early attempts refined in stages to produce the final diagram.

References:

Software Engineering. I. Sommerville, Addison-Wesley, 3rd Edition, 1989.

ISO 5807:1985, *Information processing – Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts*.

ISO/IEC 8631:1989, *Information technology – Program constructs and conventions for their preparation*.

C.2.3 Diagrammes de structures

NOTE Cette technique/mesure est référencée dans le tableau B.5 de la CEI 61508-3.

But: Montrer la structure d'un programme sous forme de diagramme.

Description: Les diagrammes de structures constituent une notation complémentaire aux diagrammes de flux de données. Ils décrivent le système programmé et la hiérarchie des différentes parties sous forme graphique, comme un arbre. Ils montrent comment les éléments d'un diagramme de flux de données peuvent être mis en œuvre en tant que hiérarchie d'unités de programme.

Un diagramme de structure montre les relations entre les modules de programme sans toutefois inclure les informations concernant l'ordre d'activation de ces unités. Ces diagrammes sont réalisés à l'aide des quatre symboles suivants:

– rectangle annoté du nom du module;
– ligne reliant les rectangles en créant une structure;
– flèche cerclée (cercle vide), annotée du nom des données échangées entre les éléments du diagramme de structure (normalement, la flèche cerclée est parallèle à la ligne reliant les rectangles du diagramme);
– flèche cerclée (cercle plein), annotée du nom du signal de commande qui passe d'un module à un autre dans le diagramme, la flèche étant là aussi parallèle à la ligne reliant les deux modules.

Il est possible d'obtenir un certain nombre de diagrammes de structures différents à partir d'un diagramme de flux de données qui n'est pas d'une simplicité triviale.

Les diagrammes de flux de données représentent la relation entre les informations et les fonctions du système. Les diagrammes de structure représentent la manière dont les éléments du système sont implémentés. Les deux techniques, bien que différentes, donnent une représentation valable du système.

Références:

Software Engineering. I. Sommerville, Addison-Wesley, 3rd Edition, 1989.

Structured Design. L. L. Constantine and E. Yourdon, Englewood Cliffs, New Jersey, Prentice Hall, 1979.

Reliable Software Through Composite Design. G. J. Myers, New York, Van Nostrand, 1975.

C.2.4 Méthodes formelles

NOTE Cette technique/mesure est référencée dans les tableaux A.1, A.2, A.4 et B.5 de la CEI 61508-3.

C.2.4.1 Généralités

But: Développement du logiciel basé d'une certaine manière sur les mathématiques. Ces méthodes comprennent des techniques de conception et de codage formels.

Description: Les méthodes formelles permettent de développer la description d'un système à n'importe quel stade de sa spécification de développement, de sa conception ou de son implémentation. La description résultante est constituée d'une notation stricte qui peut être soumise à une analyse mathématique en vue de détecter différentes classes d'incohérence ou d'inexactitude. De plus, la description peut, dans certains cas, être analysée par une machine avec une rigueur similaire à la vérification de la syntaxe d'un programme source par un compilateur, ou animée afin de visualiser différents aspects du comportement du système décrit. L'animation peut fournir une assurance supplémentaire en ce qui concerne la conformité du système à la prescription réelle ainsi qu'à la prescription formellement spécifiée, étant donné qu'elle améliore l'identification par l'opérateur du comportement spécifié.

C.2.3 Structure diagrams

NOTE This technique/measure is referenced in table B.5 of IEC 61508-3.

Aim: To show the structure of a program diagrammatically.

Description: Structure diagrams are a notation which complements data flow diagrams. They describe the programming system and a hierarchy of parts and display this graphically, as a tree. They document how elements of a data flow diagram can be implemented as a hierarchy of program units.

A structure chart shows relationships between program modules without including any information about the order of activation of these units. They are drawn using the following four symbols:

– a rectangle annotated with the name of the module;

– a line connecting these rectangles creating structure;

– a circled arrow (circle empty), annotated with the name of data passed to and from elements in the structure chart (normally, the circled arrow is drawn parallel to the line connecting the rectangles in the chart);

– a circled arrow (circle filled), annotated with the name of the control signal passing from one module to another in the structure chart, again the arrow is drawn parallel to the line connecting the two modules.

From any non-trivial data flow diagram, it is possible to derive a number of different structure charts.

Data flow diagrams depict the relationship between information and functions in the system. Structure charts depict the way elements of the system are implemented. Both techniques present valid, though different, views of the system.

References:

Software Engineering. I. Sommerville, Addison-Wesley, 3rd Edition, 1989.

Structured Design. L. L. Constantine and E. Yourdon, Englewood Cliffs, New Jersey, Prentice Hall, 1979.

Reliable Software Through Composite Design. G. J. Myers, New York, Van Nostrand, 1975.

C.2.4 Formal methods

NOTE This technique/measure is referenced in tables A.1, A.2, A.4 and B.5 of IEC 61508-3.

C.2.4.1 General

Aim: The development of software in a way that is based on mathematics. This includes formal design and formal coding techniques.

Description: Formal methods provide a means of developing a description of a system at some stage in its specification, design or implementation. The resulting description is in a strict notation that can be subjected to mathematical analysis to detect various classes of inconsistency or incorrectness. Moreover, the description can in some cases be analysed by machine with a rigour similar to the syntax checking of a source program by a compiler, or animated to display various aspects of the behaviour of the system described. Animation can give extra confidence that the system meets the real requirement as well as the formally specified requirement, because it improves human recognition of the specified behaviour.

Une méthode formelle offre généralement une notation (généralement grâce à l'utilisation d'une forme de représentation mathématique discrète), une technique permettant d'obtenir une description dans le cadre de cette notation et différentes formes d'analyse permettant de vérifier la conformité des différents aspects de la description.

NOTE La description ci-dessus figure également en B.2.2.

Plusieurs méthodes formelles sont décrites dans les paragraphes suivants de la présente norme: CCS, CSP, HOL, LOTOS, OBJ, logique temporelle, VDM et Z. A noter que les autres techniques, telles que les machines à états finis (voir B.2.3.2) et les réseaux de Pétri (voir B.2.3.3), peuvent être considérées comme des méthodes formelles selon le degré de conformité de ces techniques, telles qu'elles sont utilisées, à une base mathématique rigoureuse.

Références:

The Practice of Formal Methods in Safety-Critical Systems. S. Liu, V. Stavridou, B. Dutertre, J. Systems Software 28, 77-87, Elsevier, 1995.

Formal Methods: Use and Relevance for the Development of Safety-Critical Systems. L. M. Barroca, J. A. McDermid, The Computer Journal 35 (6), 579-599, 1992.

How to Produce Correct Software – An Introduction to Formal Specification and Program Development by Transformations. E. A. Boiten et al, The Computer Journal 35 (6), 547-554, 1992.

C.2.4.2 CCS – Calculus of Communicating Systems

But: CCS est un moyen de décrire et comparer le comportement de systèmes appartenant à des processus de communication concurrents.

Description: La méthode CCS est une technique de calcul mathématique applicable au comportement des systèmes. La conception des systèmes est modélisée comme un réseau de processus indépendants fonctionnant séquentiellement ou en parallèle. Les processus communiquent au moyen de ports (similaires aux voies utilisées pour la méthode CSP), la communication n'étant établie que lorsque les deux processus sont prêts. Le non-déterminisme peut être modélisé. A partir d'une description abstraite de haut niveau du système complet (connue sous le nom d'analyse), il est possible d'affiner le système par paliers afin d'obtenir une composition de plusieurs processus de communication dont le comportement total corresponde à celui requis pour le système entier. Il est également possible d'utiliser une conception ascendante en combinant les processus et en déduisant les propriétés du système résultant en utilisant les règles d'inférences liées aux règles de composition.

Références:

Communication and Concurrency. R. Milner, Prentice-Hall, 1989.

The Specification of Complex Systems. B. Cohen, W. T. Harwood and M. I. Jackson, Addison Wesley, 1986.

C.2.4.3 CSP – Communicating Sequential Processes

But: CSP est une technique de spécification de systèmes logiciels concurrents, c'est-à-dire de systèmes de processus de communication fonctionnant concurremment.

Description: La méthode CSP fournit un langage pour la spécification de systèmes de processus, ainsi que la preuve permettant de vérifier que la mise en œuvre des processus est conforme à leurs spécifications (connue sous le nom d'analyse, constituée d'une séquence événements autorisée).

A formal method will generally offer a notation (generally some form of discrete mathematics being used), a technique for deriving a description in that notation, and various forms of analysis for checking a description for different correctness properties.

NOTE The above description may also be found in B.2.2.

Several formal methods are described in the following subsections of this overview – CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z. Note that other techniques, such as finite state machines (see B.2.3.2) and Petri nets (see B.2.3.3), may be considered as formal methods, depending on how strictly the techniques, as used, conform to a rigorous mathematical basis.

References:

The Practice of Formal Methods in Safety-Critical Systems. S. Liu, V. Stavridou, B. Dutertre, J. Systems Software 28, 77-87, Elsevier, 1995.

Formal Methods: Use and Relevance for the Development of Safety-Critical Systems. L. M. Barroca, J. A. McDermid, The Computer Journal 35 (6), 579-599, 1992.

How to Produce Correct Software – An Introduction to Formal Specification and Program Development by Transformations. E. A. Boiten et al, The Computer Journal 35 (6), 547-554, 1992.

C.2.4.2 CCS – Calculus of Communicating Systems

Aim: CCS is a means of describing and reasoning about the behaviour of systems of concurrent, communicating processes.

Description: CCS is a mathematical calculus concerned with the behaviour of systems. The system design is modelled as a network of independent processes operating sequentially or in parallel. Processes can communicate via ports (similar to CSP's channels), the communication only taking place when both processes are ready. Non-determinism can be modelled. Starting from a high-level abstract description of the entire system (known as a trace), it is possible to carry out a step-wise refinement of the system into a composition of communicating processes whose total behaviour is that required of the whole system. Equally, it is possible to work in a bottom-up fashion, combining processes and deducing the properties of the resulting system using inference rules related to the composition rules.

References:

Communication and Concurrency. R. Milner, Prentice-Hall, 1989.

The Specification of Complex Systems. B. Cohen, W. T. Harwood and M. I. Jackson, Addison Wesley, 1986.

C.2.4.3 CSP – Communicating Sequential Processes

Aim: CSP is a technique for the specification of concurrent software systems, i.e. systems of communicating processes operating concurrently.

Description: CSP provides a language for the specification of systems of processes and proof for verifying that the implementation of processes satisfies their specifications (described as a trace – a permissible sequence of events).

Un système est modélisé comme un réseau de processus indépendants combinés séquentiellement ou en parallèle. Chaque processus est décrit en fonction de tous ses comportements possibles. Les processus peuvent communiquer (données de synchronisation ou d'échange) au moyen de voies, la communication n'étant établie que lorsque les deux processus sont prêts. Le rythme relatif des événements peut être modélisé.

La théorie concernant la méthode CSP a été directement incorporée dans l'architecture du transputer INMOS, et le langage OCCAM permet à un système spécifié grâce à la méthode CSP d'être directement mis en œuvre dans un réseau de transputers.

Référence: Communicating Sequential Processes. C. A. R. Hoare, Prentice-Hall, 1985.

C.2.4.4 HOL – Higher Order Logic

But: Ce langage formel est prévu pour servir de base à la spécification et la vérification du matériel.

Description: La méthode HOL utilise une notation logique particulière et son système support, ces deux éléments ayant été développés par le laboratoire informatique de l'Université de Cambridge. La notation logique est en grande partie dérivée de la théorie simple des types de Church et le système support est basé sur le système LCF (logique des fonctions calculables).

Références:

HOL: A Machine Orientated Formulation of Higher Order Logic. M. Gordon, University of Cambridge Technical Report, n° 68, 1985.

Specification and Verification Using Higher-Order Logic: A Case Study, F. K. Hanna and N. Daeche, in: Formal Aspects of VLSI Design: Proceedings of the 1985 Edinburgh Workshop on VLSI, pp.179-213, G. Milne and P. A. Subrahmanyam (Eds.), North Holland, 1986.

Application of formal methods to the VIPER microprocessor. W. J. Cullyer, C. H. Pygott, Proc. IEEE 134, 133-141, 1987.

C.2.4.5 LOTOS

But: LOTOS est un moyen pour décrire et comparer le comportement de systèmes appartenant à des processus de communication concurrents.

Description: La méthode LOTOS (language for temporal ordering specification) est basée sur la méthode CCS avec certains éléments supplémentaires des algèbres associées CSP et CIRCAL (calcul circuit). Elle surmonte la faiblesse de la méthode CCS en ce qui concerne la prise en compte des structures de données et des expressions de valeur en la combinant avec une deuxième composante basée sur le langage de type données abstraites ACT ONE. La composante de définition du processus de la méthode LOTOS pourrait, toutefois, être utilisée avec d'autres formalismes pour la description des types de données abstraits.

Référence: ISO 8807:1989, *Systèmes de traitement de l'information – Interconnexion de systèmes ouverts (OSI) – LOTOS – Technique de description formelle basée sur l'organisation temporelle de comportement observationnel.* (Publiée actuellement en anglais seulement.)

C.2.4.6 OBJ

But: Fournir une spécification de système précise avec retour utilisateur et validation du système avant implémentation.

A system is modelled as a network of independent processes, composed sequentially or in parallel. Each process is described in terms of all of its possible behaviours. Processes can communicate (synchronise or exchange data) via channels, the communication only taking place when both processes are ready. The relative timing of events can be modelled.

The theory behind CSP was directly incorporated into the architecture of the INMOS transputer, and the OCCAM language allows a CSP-specified system to be directly implemented on a network of transputers.

Reference: Communicating Sequential Processes. C. A. R. Hoare, Prentice-Hall, 1985.

C.2.4.4 HOL – Higher Order Logic

Aim: This is a formal language intended as a basis for hardware specification and verification.

Description: HOL refers to a particular logic notation and its machine support system, both of which were developed at the University of Cambridge computer laboratory. The logic notation is mostly taken from Church's simple theory of types and the machine support system is based upon the LCF (logic of computable functions) system.

References:

HOL: A Machine Orientated Formulation of Higher Order Logic. M. Gordon, University of Cambridge Technical Report, No. 68, 1985.

Specification and Verification Using Higher-Order Logic: A Case Study, F. K. Hanna and N. Daeche, in: Formal Aspects of VLSI Design: Proceedings of the 1985 Edinburgh Workshop on VLSI, pp.179-213, G. Milne and P. A. Subrahmanyam (Eds.), North Holland, 1986.

Application of formal methods to the VIPER microprocessor. W. J. Cullyer, C. H. Pygott, Proc. IEEE 134, 133-141, 1987.

C.2.4.5 LOTOS

Aim: LOTOS is a means for describing and reasoning about the behaviour of systems of concurrent, communicating processes.

Description: LOTOS (language for temporal ordering specification) is based on CCS with additional features from the related algebras CSP and CIRCAL (circuit calculus). It overcomes the weakness of CCS in the handling of data structures and value expressions by combining it with a second component based on the abstract data type language ACT ONE. The process description component of LOTOS could, however, be used with other formalisms for the description of abstract data types.

Reference: ISO 8807:1989, *Information processing systems – Open Systems Inter-connection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*.

C.2.4.6 OBJ

Aim: To provide a precise system specification with user feed-back and system validation prior to implementation.

Description: OBJ est un langage de spécification algébrique. Les utilisateurs spécifient les prescriptions en termes d'équations algébriques. Les aspects liés au comportement ou à la réalisation du système sont spécifiés en termes d'opérations agissant sur des types de données abstraits (ADT). Un type de donnée abstrait est similaire à un paquetage ADA où le comportement opérationnel est visible alors que les détails d'implémentation sont «cachés».

Une spécification OBJ, ainsi que l'implémentation ultérieure par paliers, est soumise aux mêmes techniques de démonstration formelles que les autres études formelles. De plus, étant donné que les aspects de la spécification OBJ liés à la réalisation sont exécutables par une machine, il convient d'effectuer la validation du système directement à partir de la spécification. L'exécution consiste essentiellement à évaluer une fonction par substitution d'équations (réécriture) jusqu'à ce qu'une valeur spécifique de sortie soit obtenue. Cette exécutabilité permet à l'utilisateur final du système envisagé d'avoir une «vue» de celui-ci au moment de la spécification du système sans être nécessairement au courant des techniques de spécification formelles associées.

Comme les autres techniques ADT, OBJ est uniquement applicable aux systèmes séquentiels ou aux aspects séquentiels des systèmes concurrents. OBJ a été utilisé pour la spécification d'applications industrielles aussi bien restreintes qu'importantes.

Références:

An Introduction to OBJ: A language for Writing and Testing Specifications. J. A. Goguen and J. Tardo, Specification of Reliable Software, IEEE Press 1979, reprinted in Software Specification Techniques, N. Gehani, A. McGrettrick (eds), Addison-Wesley, 1985.

Algebraic Specification for Practical Software Production. C. Rattray, Cogan Press, 1987.

An Algebraic Approach to the Standardisation and Certification of Graphics Software. R. Gnatz, Computer Graphics Forum 2 (2/3), 1983.

C.2.4.7 Logique temporelle

But: Expression directe des prescriptions concernant la sécurité et l'exploitation et démonstration formelle du maintien de ces caractéristiques au cours des stades de développement ultérieurs.

Description: La logique des prédicats de premier ordre standard ne comporte aucune notion de temps. La logique temporelle augmente les possibilités de la logique de premier ordre en ajoutant des opérateurs modaux (par exemple «désormais» ou «éventuellement»). Ces opérateurs peuvent être utilisés pour qualifier des assertions concernant le système. Par exemple, il peut être prescrit que des propriétés de sécurité maintiennent l'opérateur «désormais» alors qu'il peut être prescrit que d'autres états du système puissent être atteints «éventuellement» à partir d'autres états initiateurs. Les formules temporelles sont interprétées pendant les séquences d'états (comportements). La notion d'état dépend du niveau de description choisi. Cela peut s'appliquer au système entier, à un composant du système ou au programme informatique.

Les contraintes et les intervalles de temps quantifiés ne sont pas traités de manière explicite dans la logique temporelle. La datation absolue doit être traitée par la création d'états temporels supplémentaires dans le cadre de la description des états.

Références:

Temporal Logic of Programs. F. Kroger. EATCS Monographs on Computer Science, Vol. 8, Springer Verlag, 1987.

Design for Safety using Temporal Logic. J. Gorski. SAFECOMP 86, Sarlat, France, Pergamon Press, October 1986.

Description: OBJ is an algebraic specification language. Users specify requirements in terms of algebraic equations. The behavioural, or constructive, aspects of the system are specified in terms of operations acting on abstract data types (ADT). An ADT is like an ADA package where the operator behaviour is visible whilst the implementation details are "hidden".

An OBJ specification, and subsequent step-wise implementation, is amenable to the same formal proof techniques as other formal approaches. Moreover, since the constructive aspects of the OBJ specification are machine-executable, it is straightforward to achieve system validation from the specification itself. Execution is essentially the evaluation of a function by equation substitution (rewriting) which continues until specific output value is obtained. This executability allows end-users of the envisaged system to gain a "view" of the eventual system at the system specification stage without the need to be familiar with the underlying formal specification techniques.

As with all other ADT techniques, OBJ is only applicable to sequential systems, or to sequential aspects of concurrent systems. OBJ has been used for the specification of both small- and large-scale industrial applications.

References:

An Introduction to OBJ: A language for Writing and Testing Specifications. J. A. Goguen and J. Tardo, Specification of Reliable Software, IEEE Press 1979, reprinted in Software Specification Techniques, N. Gehani, A. McGrettrick (eds), Addison-Wesley, 1985.

Algebraic Specification for Practical Software Production. C. Rattray, Cogan Press, 1987.

An Algebraic Approach to the Standardisation and Certification of Graphics Software. R. Gnatz, Computer Graphics Forum 2 (2/3), 1983.

C.2.4.7 Temporal logic

Aim: Direct expression of safety and operational requirements and formal demonstration that these properties are preserved in the subsequent development steps.

Description: Standard first-order predicate logic contains no concept of time. Temporal logic extends first-order logic by adding modal operators (for example "henceforth" and "eventually"). These operators can be used to qualify assertions about the system. For example, safety properties might be required to hold "henceforth", whilst other desired system states might be required to be attained "eventually" from some other initiating state. Temporal formulas are interpreted on sequences of states (behaviours). What constitutes a "state" depends on the chosen level of description. It can refer to the whole system, a system component or the computer program.

Quantified time intervals and constraints are not handled explicitly in temporal logic. Absolute timing has to be handled by creating additional time states as part of the state description.

References:

Temporal Logic of Programs. F. Kroger. EATCS Monographs on Computer Science, Vol. 8, Springer Verlag, 1987.

Design for Safety using Temporal Logic. J. Gorski. SAFECOMP 86, Sarlat, France, Pergamon Press, October 1986.

The Temporal Logic of Programs. A Pnueli, 18th Annual Symposium on Foundations of Computer Science, IEEE, 1977.

Verifying Concurrent Processes Using Temporal Logic, Hailpern, T. Brent, Springer Verlag, 1981.

C.2.4.8 VDM, VDM++ – Vienna Development Method

But: Spécification et implémentation systématiques de programmes séquentiels (VDM) et de programmes temps réel (VDM++) concurrents.

Description: VDM est une technique de spécification fondée sur les mathématiques et une technique d'affinement des implémentations de manière à permettre un contrôle de conformité par rapport à la spécification.

La technique de spécification est basée sur des modèles, en ce sens que l'état du système est modélisé en termes de structures suivant la théorie des ensembles, servant à décrire les invariants (prédicats), et les opérations associées à cet état sont modélisées en spécifiant leurs conditions initiales et leurs conditions finales en termes d'états du système. Les opérations peuvent être contrôlées afin de préserver les invariants du système.

L'implémentation de la spécification est effectuée par la réification de l'état du système en termes de structures de données dans le langage d'exécution et par le perfectionnement des opérations en termes de programmes dans le langage cible. Les phases de réification et de perfectionnement entraînent des obligations de contrôle pour établir qu'elles sont correctes. C'est le concepteur qui choisit de rendre obligatoire ou non l'exécution de ces contrôles.

La méthode VDM est principalement utilisée durant la phase de spécification mais elle peut être également utilisée durant les phases de conception et d'implémentation aboutissant au code source. Elle est uniquement applicable à des programmes séquentiels ou aux processus séquentiels de systèmes concurrents.

L'extension temps réel alternative et orientée objet de VDM, VDM++, est un langage de spécification formelle basé sur le langage ISO VDM-SL, et sur le langage orienté objet Smalltalk.

VDM++ fournit une large gamme de constructions de telle sorte qu'un utilisateur peut spécifier formellement et de façon orientée objet des systèmes temps réel concurrents. Dans VDM++, une spécification formelle complète consiste en une variété de spécifications de classes, et optionnellement d'un espace de travail.

Les possibilités temps réel de VDM++ sont:

– les expressions temporelles, destinées à noter l'instant effectif et l'instant lié à la méthode d'invocation dans un corps de méthodes donné;
– les post-expressions datées, ajoutées à une méthode pour spécifier les limites supérieures ou inférieures de l'instant d'exécution pour une réalisation correcte du programme;
– les variables temporelles continues ont été introduites. Moyennant des hypothèses sur les résultats il est possible de spécifier des relations (par exemple, des équations différentielles) entre ces fonctions du temps. Cette caractéristique a démontré une grande utilité dans la spécification d'exigences relatives à des systèmes qui fonctionnent dans un environnement temporel continu. Des étapes de perfectionnement conduisent à des solutions logicielles discrètes pour ce type de systèmes.

The Temporal Logic of Programs. A. Pnueli, 18th Annual Symposium on Foundations of Computer Science, IEEE, 1977.

Verifying Concurrent Processes Using Temporal Logic, Hailpern, T. Brent, Springer Verlag, 1981.

C.2.4.8 VDM, VDM++ – Vienna Development Method

Aim: The systematic specification and implementation of sequential (VDM) and concurrent real-time (VDM++) programs.

Description: VDM is a mathematically based specification technique and a technique for refining implementations in a way that allows proof of their correctness with respect to the specification.

The specification technique is model-based in that the system state is modelled in terms of set-theoretic structures on which are described invariants (predicates), and operations on that state are modelled by specifying their pre- and post-conditions in terms of the system state. Operations can be proved to preserve the system invariants.

The implementation of the specification is done by the reification of the system state in terms of data structures in the target language and by refinement of the operations in terms of the program in the target language. Reification and refinement steps give rise to proof obligations that establish their correctness. Whether or not these obligations are carried out is determined by the designer.

VDM is principally used in the specification stage but can be used in the design and implementation stages leading to source code. It can only be applied to sequentially structured programs or the sequential processes in concurrent systems.

The object-oriented and concurrent real-time extension of VDM, VDM++, is a formal specification language based on the ISO language VDM-SL and on the object-oriented language Smalltalk.

VDM++ provides a wide range of constructs such that a user can formally specify concurrent real-time systems in an object-oriented fashion. In VDM++ a complete formal specification consists of a collection of class specifications and optionally a workspace.

Real-time provisions of VDM++ are:

– temporal expressions are provided to denote both the current moment and the method invocation moment within a method body;

– a timed post expression can be added to a method to specify the upper (or lower) bounds of the execution time for correct implementations;

– time continuous variables have been introduced. With assumption and effect clauses one can specify relations (for example differential equations) between these functions of time. This feature has proven to be very useful in the specification of requirements of systems which operate in a time continuous environment. Refinement steps lead to discrete software solutions for these kinds of systems.

Références:

ISO/IEC 13817-1:1997, *Technologies de l'information – Langages de programmation, leurs environnements et interfaces logiciel système – Méthode de développement de Vienne – Langage de spécification – Partie 1: Langage de base*.

Conformity Clause for VDM-SL, G. I. Parkin and B. A. Wichmann, Lecture Notes in Computer Science 670, FME'93 Industrial-Strength Formal Methods, First International Symposium of Formal Methods in Europe. Editors: J. C. P. Woodcock and P. G. Larsen. Springer Verlag, 501-520.

Major VDM+ – Language characteristics: http://www.ifad.dk/products/vdmlangchar.html

Systematic Software Development using VDM. C. B. Jones. Prentice-Hall, 2nd Edition, 1990.

Software Development – A Rigorous Approach. C. B. Jones, Prentice-Hall, 1980.

Formal Specification and Software Development. D. Bjorner and C. B. Jones, Prentice-Hall, 1982.

The Specification of Complex Systems. B. Cohen, W. T. Harwood and M. I. Jackson. Addison Wesley, 1986.

C.2.4.9 Z

But: Z est une notation de langage de spécification pour systèmes séquentiels et une technique de conception qui permet au développeur de passer d'une spécification Z à des algorithmes exécutables d'une manière qui permet de prouver leur conformité par rapport à la spécification.

Z est principalement utilisé durant l'étape de spécification, mais une méthode a été imaginée pour passer de la spécification à la conception et à l'implémentation. Cette méthode est la plus adaptée au développement de systèmes séquentiels orientés données.

Description: Comme VDM, la technique de spécification est basée sur des modèles en ce sens que l'état du système est modélisé en termes de structures dans la théorie des ensembles servant à définir les invariants (prédicats), et les opérations associées à cet état sont modélisées en spécifiant leurs préconditions et postconditions en termes d'états du système. Les opérations peuvent être prouvées du point de vue de la préservation des invariants du système, démontrant ainsi leur cohérence. La partie formelle d'une spécification est divisée en schémas permettant la structuration des spécifications par affinage.

Normalement, une spécification Z est un mélange de méthode Z formelle et de texte explicatif informel en langage naturel (un texte formel est parfois trop concis pour permettre une lecture facile et nécessite souvent une explication, alors que le langage naturel informel peut facilement devenir vague et imprécis).

Contrairement à VDM, Z est une notation plutôt qu'une méthode complète. Toutefois, une méthode associée (appelée méthode B) a été développée et peut être utilisée conjointement avec la méthode Z. La méthode B est basée sur le principe d'affinage par paliers.

Références:

The Z Notation – A Reference Manual. J. M. Spivey. Prentice-Hall, 1992.

Specification Case Studies. Edited by I. Hayes, Prentice-Hall, 1987.

The B Method. J. R. Abrial et al, VDM '91 Formal Software Development Methods, (S. Prehen and W. J. Toetenel, Eds), Springer Verlag, 398-405, 1991.

Specification of the UNIX Filestore. C. Morgan and B. Sufrin. IEEE Transactions on Software Engineering, SE-10, 2, March 1984.

References:

ISO/IEC 13817-1:1997, *Information technology – Programming languages, their environments and system software interfaces – Vienna Development Method – Specification Language – Part 1: Base language*.

Conformity Clause for VDM-SL, G. I. Parkin and B. A. Wichmann, Lecture Notes in Computer Science 670, FME'93 Industrial-Strength Formal Methods, First International Symposium of Formal Methods in Europe. Editors: J. C. P. Woodcock and P. G. Larsen. Springer Verlag, 501-520.

Major VDM+ – Language characteristics: http://www.ifad.dk/products/vdmlangchar.htmI

Systematic Software Development using VDM. C. B. Jones. Prentice-Hall. 2nd Edition, 1990.

Software Development – A Rigorous Approach. C. B. Jones. Prentice-Hall, 1980.

Formal Specification and Software Development. D. Bjorner and C. B. Jones, Prentice-Hall, 1982.

The Specification of Complex Systems. B. Cohen, W. T. Harwood and M. I. Jackson. Addison Wesley, 1986.

C.2.4.9 Z

Aim: Z is a specification language notation for sequential systems and a design technique that allows the developer to proceed from a Z specification to executable algorithms in a way that allows proof of their correctness with respect to the specification.

Z is principally used in the specification stage but a method has been devised to go from specification into a design and an implementation. It is best suited to the development of data-oriented, sequential systems.

Description: Like VDM, the specification technique is model-based in that the system state is modelled in terms of set-theoretic structures on which are described invariants (predicates), and operations on that state are modelled by specifying their pre- and post-conditions in terms of the system state. Operations can be proved to preserve the system invariants thereby demonstrating their consistency. The formal part of a specification is divided into schemas which allow the structuring of specifications through refinement.

Typically, a Z specification is a mixture of formal Z and informal explanatory text in natural language. (Formal text on its own can be too terse for easy reading and often its purpose needs to be explained, while the informal natural language can easily become vague and imprecise.)

Unlike VDM, Z is a notation rather than a complete method. However, an associated method (called B) has been developed which can be used in conjunction with Z. The B method is based on the principle of step-wise refinement.

References:

The Z Notation – A Reference Manual. J. M. Spivey. Prentice-Hall, 1992.

Specification Case Studies. Edited by I. Hayes, Prentice-Hall, 1987.

The B Method. J. R. Abrial et al, VDM '91 Formal Software Development Methods, (S. Prehen and W. J. Toetenel, eds), Springer Verlag, 398-405, 1991.

Specification of the UNIX Filestore. C. Morgan and B. Sufrin. IEEE Transactions on Software Engineering, SE-10, 2, March 1984.

C.2.5 Programmation défensive

NOTE Cette technique/mesure est référencée dans le tableau A.4 de la CEI 61508-3.

But: Produire des programmes capables de détecter les flux de commandes ou de données anormaux ou les valeurs de données anormales en cours d'exécution et de réagir à ces anomalies de manière prédéterminée et acceptable.

Description: Un grand nombre de techniques peuvent être utilisées pendant la programmation en vue de vérifier l'absence d'anomalies relatives aux commandes ou aux données. Ces techniques peuvent être appliquées systématiquement pendant toute la programmation d'un système en vue de réduire les probabilités d'erreur concernant le traitement des données.

Il y a deux domaines de techniques défensives qui se superposent. Un logiciel intrinsèquement dépourvu d'erreurs est prévu pour prendre en compte ses propres défauts de conception. Ces défauts peuvent être dus à des erreurs de conception ou de codage ou à des prescriptions erronées. Les techniques défensives comprennent, entre autres:

– un contrôle des limites applicable aux variables;
– un contrôle de vraisemblance des valeurs, dans la mesure du possible;
– un contrôle du type, du dimensionnement et des limites des paramètres de procédure en début de procédure.

Ces trois premières recommandations permettent de s'assurer que les nombres manipulés par le programme sont plausibles, à la fois en termes de fonction du programme et de signification physique des variables.

Il convient que les paramètres de consultation seule et de lecture-écriture soient séparés et leur accès vérifié. Il convient que les fonctions traitent tous les paramètres comme des paramètres à consultation seule. Il convient que les constantes littérales ne soient pas accessibles en écriture. Cela permet de détecter tout écrasement accidentel ou mauvaise utilisation des variables.

Un logiciel tolérant aux anomalies est conçu pour «prévoir» les défaillances dans son propre environnement ou utiliser les conditions s'écartant des conditions nominales ou prévues, et réagir d'une manière prédéterminée. Les techniques comprennent les contrôles suivants.

– Contrôle de vraisemblance des variables d'entrée et des variables intermédiaires avec signification physique;
– Contrôle de l'effet des variables de sortie, de préférence par observation directe des changements d'état du système associé;
– Contrôle de la configuration du logiciel, y compris sur l'existence et l'accessibilité du matériel prévu et sur la complétude du logiciel. Cela est particulièrement important pour maintenir l'intégrité après les procédures de maintenance.

Certaines techniques de programmation défensive, telles que le contrôle des séquences des flux de commandes, prennent également en compte les défaillances extérieures.

Références:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989, ISBN 1-85166-381-9.

Software Engineering Aspects of Real-time Programming Concepts. E. Schoitsch, Computer Physics Communications 41, North Holland, Amsterdam, 1986.

C.2.5 Defensive programming

NOTE This technique/measure is referenced in table A.4 of IEC 61508-3.

Aim: To produce programs which detect anomalous control flow, data flow or data values during their execution and react to these in a predetermined and acceptable manner.

Description: Many techniques can be used during programming to check for control or data anomalies. These can be applied systematically throughout the programming of a system to decrease the likelihood of erroneous data processing.

There are two overlapping areas of defensive techniques. Intrinsic error-safe software is designed to accommodate its own design shortcomings. These shortcomings may be due to mistakes in design or coding, or to erroneous requirements. The following lists some of the defensive techniques:

– variables should be range checked;

– where possible, values should be checked for plausibility;

– parameters to procedures should be type, dimension and range checked at procedure entry.

These first three recommendations help to ensure that the numbers manipulated by the program are reasonable, both in terms of the program function and physical significance of the variables.

Read-only and read-write parameters should be separated and their access checked. Functions should treat all parameters as read-only. Literal constants should not be write-accessible. This helps detect accidental overwriting or mistaken use of variables.

Fault tolerant software is designed to "expect" failures in its own environment or use outside nominal or expected conditions, and behave in a predefined manner. Techniques include the following.

– Input variables and intermediate variables with physical significance should be checked for plausibility.

– The effect of output variables should be checked, preferably by direct observation of associated system state changes.

– The software should check its configuration, including both the existence and accessibility of expected hardware and also that the software itself is complete – this is particularly important for maintaining integrity after maintenance procedures.

Some of the defensive programming techniques, such as control flow sequence checking, also cope with external failures.

References:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989, ISBN 1-85166-381-9.

Software Engineering Aspects of Real-time Programming Concepts. E. Schoitsch, Computer Physics Communications 41, North Holland, Amsterdam, 1986.

C.2.6 Règles de conception et de codage

NOTE Cette technique/mesure est référencée dans le tableau A.4 de la CEI 61508-3.

C.2.6.1 Généralités

But: Faciliter l'aptitude à la vérification, encourager une étude objective en équipe et mettre en œuvre une méthode de conception standard.

Description: Les règles à observer sont approuvées par les participants au début du projet. Ces règles comprennent les méthodes de conception et développement à observer (par exemple, les méthodes JSP, MASCOT, les réseaux de Pétri, etc.) et les règles de codage associées (voir C.2.6.2).

Ces règles sont établies pour faciliter le développement, la vérification, l'évaluation et la maintenance. Par conséquent, elles doivent tenir compte des outils disponibles, en particulier analyseurs et outils de rétro-ingénierie.

Références:

CEI 60880:1986, *Logiciel pour les calculateurs utilisés dans les systèmes de sûreté des centrales nucléaires*.

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Verein Deutscher Ingenieure. Software-Zuverlassigkeit – Grundlagen, Konstruktive Mass-nahmen, Nachweisverfahren. VDI-Verlag, 1993, ISBN 3-18-401185-2.

C.2.6.2 Règles de codage

NOTE Cette technique/mesure est référencée dans le tableau B.1 de la CEI 61508-3.

But: Faciliter l'aptitude au contrôle du code produit.

Description: Les règles détaillées à observer sont complètement approuvées avant codage. Ces règles comprennent typiquement

– des détails de modularisation comme, par exemple, la forme des interfaces et la taille des modules logiciels;
– l'utilisation de l'encapsulation, de l'héritage (limité en profondeur) et du polymorphisme, en cas de langages orientés objet;
– l'utilisation limitée ou l'évitement de certaines structures de langage comme, par exemple, «goto», «equivalence», objets dynamiques, données dynamiques, structures de données dynamiques, récursion, pointeurs, sorties, etc.;
– des restrictions relatives aux interruptions autorisées pendant l'exécution du code des opérations critiques relatives à la sécurité;
– la présentation du code (listing);
– l'interdiction de sauts inconditionnels (par exemple «goto» dans les programmes élaborés dans un langage de haut niveau.

Ces règles sont établies pour faciliter les tests, la vérification, l'évaluation et la maintenance des modules logiciels. Par conséquent, il convient qu'elles tiennent compte des outils disponibles, en particulier les analyseurs.

NOTE Pour plus d'informations sur ce sujet, voir C.2.6.3 à C.2.6.7.

C.2.6 Design and coding standards

NOTE This technique/measure is referenced in table A.4 of IEC 61508-3.

C.2.6.1 General

Aim: To facilitate verifiability, to encourage a team-centred, objective approach and to enforce a standard design method.

Description: The rules to be adhered to are agreed at the outset of the project between the participants. These rules comprise the design and development methods to be followed (for example JSP, MASCOT, Petri nets, etc.) and the related coding standards (see C.2.6.2).

These rules are made to allow for ease of development, verification, assessment and maintenance. Therefore they should take into account available tools, in particular analysers and reverse engineering tools.

References:

IEC 60880:1986, *Software for computers in the safety systems of nuclear power stations*.

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988. ISBN 1-85166-203-0.

Verein Deutscher Ingenieure. Software-Zuverlassigkeit – Grundlagen, Konstruktive Mass-nahmen, Nachweisverfahren. VDI-Verlag, 1993, ISBN 3-18-401185-2.

C.2.6.2 Coding standards

NOTE This technique/measure is referenced in table B.1 of IEC 61508-3.

Aim: To facilitate verifiability of the produced code.

Description: The detailed rules to be adhered to are fully agreed before coding. These rules comprise typically

- details of modularisation, for example interface shapes, software module sizes;
- use of encapsulation, inheritance (restricted in depth) and polymorphism, in the case of object oriented languages;
- limited use or avoidance of certain language constructs, for example "goto", "equivalence", dynamic objects, dynamic data, dynamic data structures, recursion, pointers, exits, etc;
- restrictions on interrupts enabled during the execution of safety-critical code;
- layout of the code (listing);
- no unconditional jumps (for example "goto") in programs in higher level languages.

These rules are made to allow for ease of software module testing, verification, assessment and maintenance. Therefore they should take into account available tools, in particular analysers.

NOTE For more information on this topic, see C.2.6.3 to C.2.6.7

C.2.6.3 Pas de variables dynamiques ni d'objets dynamiques

NOTE Cette technique/mesure est référencée dans le tableau B.1 de la CEI 61508-3.

But: Exclure

– tout recouvrement de mémoire non désiré ou non détecté.
– tout goulot d'étranglement des ressources pendant la durée d'exécution (liée à la sécurité).

Description: Dans le cas de cette mesure, les variables dynamiques et les objets dynamiques sont celles et ceux qui font l'objet d'une attribution de mémoire et dont l'adresse absolue est déterminée au moment de l'exécution. La valeur de l'attribution de mémoire et de l'adresse correspondante dépend de l'état du système au moment de l'attribution, ce qui signifie l'impossibilité d'un contrôle par le compilateur ou tout autre outil hors ligne.

Etant donné que le nombre de variables et d'objets dynamiques, ainsi que l'espace mémoire disponible pour l'allocation de nouveaux objets ou variables dynamiques, dépend de l'état du système au moment de l'allocation, il est possible que des défaillances surviennent au moment de l'allocation ou de l'utilisation des variables ou des objets. Par exemple, si l'espace mémoire disponible à l'emplacement alloué par le système est insuffisant, l'écrasement accidentel du contenu mémoire d'une autre variable peut se produire. Ces incidents sont évités lorsque aucun objet ou variable dynamique n'est utilisé.

C.2.6.4 Contrôle en ligne pendant la création de variables dynamiques ou d'objets dynamiques

NOTE 1 Cette technique/mesure est référencée dans le tableau B.1 de la CEI 61508-3.

But: Contrôler que la mémoire qui doit être allouée à des variables et à des objets dynamiques est libre avant cette allocation, en s'assurant que l'allocation de variables et d'objets dynamiques pendant la durée d'exécution n'a aucune incidence sur les variables, les données ou le code existants.

Description: Dans le cas de cette mesure, les variables dynamiques sont celles qui font l'objet d'une attribution de mémoire et dont l'adresse absolue est déterminée au moment de l'exécution (dans ce sens, les variables sont également les attributs des instances d'objets).

Grâce à des moyens matériels ou logiciels, la mémoire est contrôlée en vue de s'assurer qu'elle est libre avant l'allocation d'une variable ou d'un objet dynamique (par exemple pour éviter tout débordement de pile). Si l'allocation n'est pas autorisée (comme, par exemple, lorsque la mémoire à l'adresse déterminée est insuffisante), il faut faire les actions appropriées. Après l'utilisation d'une variable ou d'un objet dynamique (comme, par exemple, après la sortie d'un sous-programme), il faut libérer toute la mémoire allouée à cette variable.

NOTE 2 Une solution alternative est de démontrer statistiquement que la mémoire convient dans tous les cas.

C.2.6.5 Utilisation limitée des interruptions

NOTE Cette technique/mesure est référencée dans le tableau B.1 de la CEI 61508-3.

But: Assurer que le logiciel est vérifiable et testable.

C.2.6.3 No dynamic variables or dynamic objects

NOTE This technique/measure is referenced in table B.1 of IEC 61508-3.

Aim: To exclude

– unwanted or undetected overlay of memory;

– bottlenecks of resources during (safety-related) run-time.

Description: In the case of this measure, dynamic variables and dynamic objects are those variables and objects that have their memory allocated and absolute addresses determined at run-time. The value of allocated memory and its addresses depend on the state of the system at the moment of allocation, which means that it cannot be checked by the compiler or any other off-line tool.

Because the number of dynamic variables and objects, and the existing free memory space for allocating new dynamic variables or objects, depends on the state of the system at the moment of allocation it is possible for faults to occur when allocating or using the variables or objects. For example, when the amount of free memory at the location allocated by the system is insufficient, the memory contents of another variable can be inadvertently overwritten. If dynamic variables or objects are not used, these faults are avoided.

C.2.6.4 On-line checking during creation of dynamic variables or dynamic objects

NOTE 1 This technique/measure is referenced in table B.1 of IEC 61508-3.

Aim: To check that the memory to be allocated to dynamic variables and objects is free before allocation takes place, ensuring that the allocation of dynamic variables and objects during run-time does not impact existing variables, data or code.

Description: In the case of this measure, dynamic variables are those variables that have their memory allocated and absolute addresses determined at run-time (variables in this sense are also the attributes of object instances).

By means of hardware or software, the memory is checked to ensure it is free before a dynamic variable or object is allocated to it (for example, to avoid stack overflow). If allocation is not allowed (for example if the memory at the determined address is not sufficient), appropriate action must be taken. After a dynamic variable or object has been used (for example, after exiting a subroutine) the whole memory which was allocated to it must be freed.

NOTE 2 An alternative is to demonstrate statically that memory will be adequate in all cases.

C.2.6.5 Limited use of interrupts

NOTE This technique/measure is referenced in table B.1 of IEC 61508-3.

Aim: To keep software verifiable and testable.

Description: Il convient que l'utilisation des interruptions soit limitée. Les interruptions peuvent être utilisées si elles simplifient le système. Il convient que le traitement logiciel des interruptions soit inhibé durant les opérations critiques (comme, par exemple, les opérations à durée critique ou les changements critiques pour les données) des fonctions exécutées. Lorsque des interruptions sont utilisées, il convient que les opérations non interruptibles aient un temps de traitement maximal défini, de manière à pouvoir calculer le temps maximal durant lequel une interruption est inhibée. Il convient que l'utilisation et le masquage des interruptions soient soigneusement documentés.

C.2.6.6 Utilisation limitée des pointeurs

NOTE Cette technique/mesure est référencée dans le tableau B.1 de la CEI 61508-3.

But: Eviter les problèmes résultant de l'accession à des données sans contrôle préalable des limites et du type de pointeur. Permettre le test et la vérification des modules du logiciel. Limiter les conséquences des défaillances.

Description: Les opérations arithmétiques relatives à un pointeur du logiciel d'application peuvent être utilisées au niveau du code source uniquement si le type de données et les limites de valeur du pointeur (afin de s'assurer que la référence du pointeur est située dans l'espace d'adresse correct) sont contrôlés avant l'accès. Il convient que la communication entre les différentes tâches du logiciel d'application ne soit pas réalisée par référence directe entre les tâches. Il convient que les échanges de données soient effectués au moyen du système d'exploitation.

C.2.6.7 Utilisation limitée de la récursion

NOTE Cette technique/mesure est référencée dans le tableau B.1 de la CEI 61508-3.

But: Eviter l'utilisation non vérifiable et non testable des appels de sous-programme.

Description: En cas d'utilisation de la récursion, il faut qu'il y ait un critère clair permettant de prédire la profondeur de récursion.

C.2.7 Programmation structurée

NOTE Cette technique/mesure est référencée dans le tableau A.4 de la CEI 61508-3.

But: Concevoir et implémenter le programme de manière à ce qu'il soit pratique de l'analyser sans exécution. Le programme peut contenir uniquement un comportement absolu minimal statistiquement non testable.

Description: Les principes suivants doivent être appliqués en vue de minimiser la complexité structurelle:

– diviser le programme en modules logiciel de taille plus petite et appropriée en s'assurant que ces modules sont découplés autant qu'il est possible et que toutes les interactions sont explicites;

– composer le flux de commandes des modules logiciels à l'aide de constructions structurées: séquences, itérations et sélection;

– faire en sorte que le nombre de chemins à travers un module logiciel soit aussi faible que possible et que la relation entre les paramètres d'entrée et de sortie soit aussi simple que possible;

– éviter les branchements compliqués et, en particulier, les branchements inconditionnels (goto) dans les langages de haut niveau;

– lorsque cela est possible, lier les contraintes de bouclage et les branchements aux paramètres d'entrée;

– éviter les décisions de branchement et de bouclage reposant sur l'utilisation de calculs complexes.

Description: The use of interrupts should be restricted. Interrupts may be used if they simplify the system. Software handling of interrupts should be inhibited during critical parts (for example time critical, critical to data changes) of the executed functions. If interrupts are used, then parts not interruptible should have a specified maximum computation time, so that the maximum time for which an interrupt is inhibited can be calculated. Interrupt usage and masking should be thoroughly documented.

C.2.6.6 Limited use of pointers

NOTE This technique/measure is referenced in table B.1 of IEC 61508-3.

Aim: To avoid the problems caused by accessing data without first checking range and type of the pointer. To support modular testing and verification of software. To limit the consequence of failures.

Description: In the application software, pointer arithmetic may be used at source code level only if pointer data type and value range (to ensure that the pointer reference is within the correct address space) are checked before access. Inter-task communication of the application software should not be done by direct reference between the tasks. Data exchange should be done via the operating system.

C.2.6.7 Limited use of recursion

NOTE This technique/measure is referenced in table B.1 of IEC 61508-3.

Aim: To avoid unverifiable and untestable use of subroutine calls.

Description: If recursion is used, there must be a clear criterion which makes predictable the depth of recursion.

C.2.7 Structured programming

NOTE This technique/measure is referenced in table A.4 of IEC 61508-3.

Aim: To design and implement the program in a way that it is practical to analyse without it being executed. The program may contain only an absolute minimum of statistically untestable behaviour.

Description: The following principles should be applied to minimise structural complexity:

– divide the program into appropriately small software modules, ensuring they are decoupled as far as possible and all interactions are explicit;

– compose the software module control flow using structured constructs, that is sequences, iterations and selection;

– keep the number of possible paths through a software module small, and the relation between the input and output parameters as simple as possible;

– avoid complicated branching and, in particular, avoid unconditional jumps (goto) in higher level languages;

– where possible, relate loop constraints and branching to input parameters;

– avoid using complex calculations as the basis of branching and loop decisions.

Il convient que les fonctions du langage de programmation qui encouragent l'approche ci-dessus soient utilisées de préférence aux autres fonctions qui sont (prétendues) plus efficaces, sauf lorsque l'efficacité est une priorité absolue (comme, par exemple, dans le cas de certains systèmes critiques relatifs à la sécurité).

Références:

Notes on structured programming. E. W. Dijkstra, Structured Programming, Academic Press, London, 1972, ISBN 0-12-200550-3.

A Discipline of Programming. E. W. Dijkstra. Englewood Cliffs NJ, Prentice-Hall, 1976.

A Software Tool for Top-down Programming. D. C. Ince. Software – Practice and Experience, Vol. 13, n° 8, August 1983.

Verification – The Practical Problems. J. T. Webb and D. J. Mannering, SARSS 87, Nov. 1987, Altrincham, England, Elsevier Applied Science, 1987, ISBN 1-85166-167-0.

An Experience in Design and Validation of Software for a Reactor Protection System. S. Bologna, E. de Agostino et al, IFAC Workshop, SAFECOMP, 1979, Stuttgart, 16-18 May 1979, Pergamon Press, 1979.

C.2.8 Masquage/encapsulation des informations

NOTE Cette technique/mesure est référencée dans le tableau B.9 de la CEI 61508-3.

But: Prévenir tout accès involontaire aux données ou procédures et ainsi maintenir une bonne structure du programme.

Description: Les données qui sont globalement accessibles à tous les éléments logiciels peuvent être modifiées accidentellement ou de manière erronée par l'un de ces éléments. Toute modification de ces structures de données peut nécessiter l'examen détaillé du code et des modifications importantes.

Le masquage des informations est une approche générale pour minimiser ces difficultés. Les structures des données essentielles sont «masquées» et ne peuvent être manipulées qu'à l'aide d'un jeu déterminé de procédures d'accès. Cela permet de modifier les structures internes et d'utiliser des procédures supplémentaires sans affecter le comportement du reste du logiciel. Par exemple, un répertoire donné peut comporter les procédures d'accès «insertion», «annulation» et «recherche». Les procédures d'accès et les structures des données internes peuvent être réécrites à l'aide de ces procédures (par exemple pour utiliser une méthode de consultation différente ou pour stocker les noms sur un disque dur) sans affecter le comportement logique du reste du logiciel.

A ce propos, il convient d'utiliser la notion de type de donnée abstrait. Si une aide directe n'est pas fournie, il peut être nécessaire de vérifier que l'abstraction n'a pas été rompue par inadvertance.

Références:

Software Engineering: Planning for Change. D. A. Lamb. Prentice-Hall, 1988.

On the Design and Development of Program Families. D. L. Parnas. IEEE Trans SE-2, March 1976.

Features of the programming language which encourage the above approach should be used in preference to other features which are (allegedly) more efficient, except where efficiency takes absolute priority (for example some safety critical systems).

References:

Notes on structured programming. E. W. Dijkstra, Structured Programming, Academic Press, London, 1972, ISBN 0-12-200550-3.

A Discipline of Programming. E. W. Dijkstra. Englewood Cliffs NJ, Prentice-Hall, 1976.

A Software Tool for Top-down Programming. D. C. Ince. Software – Practice and Experience, Vol. 13, No. 8, August 1983.

Verification – The Practical Problems. J. T. Webb and D. J. Mannering, SARSS 87, Nov. 1987, Altrincham, England, Elsevier Applied Science, 1987, ISBN 1-85166-167-0.

An Experience in Design and Validation of Software for a Reactor Protection System. S. Bologna, E. de Agostino et al, IFAC Workshop, SAFECOMP, 1979, Stuttgart, 16-18 May 1979, Pergamon Press, 1979.

C.2.8 Information hiding/encapsulation

NOTE This technique/measure is referenced in table B.9 of IEC 61508-3.

Aim: To prevent unintended access to data or procedures and thereby support a good program structure.

Description: Data that is globally accessible to all software components can be accidentally or incorrectly modified by any of these components. Any changes to these data structures may require detailed examination of the code and extensive modifications.

Information hiding is a general approach for minimising these difficulties. The key data structures are "hidden" and can only be manipulated through a defined set of access procedures. This allows the internal structures to be modified or further procedures to be added without affecting the functional behaviour of the remaining software. For example, a name directory might have access procedures "insert", "delete" and "find". The access procedures and internal data structures could be re-written (for example to use a different look-up method or to store the names on a hard disk) without affecting the logical behaviour of the remaining software using these procedures.

In this connection, the concept of abstract data types should be used. If direct support is not provided, then it may be necessary to check that the abstraction has not been inadvertently broken.

References:

Software Engineering: Planning for Change. D. A. Lamb. Prentice-Hall, 1988.

On the Design and Development of Program Families. D. L. Parnas. IEEE Trans SE-2, March 1976.

C.2.9 Approche modulaire

NOTE Cette technique/mesure est référencée dans les tableaux A.4 et B.9 de la CEI 61508-3.

But: Décomposition d'un système logiciel en petites parties compréhensibles de manière à limiter la complexité du système.

Description: Une approche modulaire ou modularisation comporte plusieurs règles concernant les phases de conception, de codage et de maintenance d'un projet de logiciel. Ces règles varient selon la méthode de conception employée. La plupart des méthodes comportent les règles suivantes:

– il convient qu'un module logiciel ait une seule tâche ou fonction bien définie à remplir;
– il convient que les connexions entre modules logiciels soient limitées et strictement définies; la cohérence doit être forte au niveau de chaque module;
– il convient que les ensembles de sous-programmes soient réalisés de manière à obtenir plusieurs niveaux de modules logiciels;
– il convient que la taille des sous-programmes soit limitée à une certaine valeur spécifiée comme, par exemple, deux à quatre fois la taille de l'écran;
– il convient que les sous-programmes aient une seule entrée et une seule sortie;
– il convient que les modules logiciels communiquent avec d'autres modules logiciels à travers leurs interfaces (lorsque des variables globales ou communes sont utilisées, il convient qu'elles soient bien structurées, que leur accès soit contrôlé et que leur utilisation soit chaque fois justifiée);
– il convient que toutes les interfaces des modules logiciels soient parfaitement documentées;
– il convient qu'une interface de modules logiciels contienne le nombre minimal de paramètres nécessaires à sa fonction.

Référence: Structured Design – Fundamentals of a Discipline of Computer Program and Systems Design. E. Yourdon, L. L. Constantine, Prentice-Hall, 1979, ISBN 0-13-854471-9.

C.2.10 Utilisation de modules logiciels et composants éprouvés/vérifiés

NOTE 1 Cette technique/mesure est référencée dans la table A.4 de la CEI 61508-3.

NOTE 2 Voir à l'annexe D quelques aspects mathématiques sous-jacents à l'estimation numérique suivante. Voir également en B.5.4 une approche de mesure et statistique similaire.

But: Eviter la nécessité pour les modules logiciels et les conceptions de composants matériels de devoir être amplement revalidés ou reconçus pour chaque application nouvelle. Profiter des conceptions qui n'ont pas été vérifiées formellement ou rigoureusement mais qui bénéficient d'une expérience opérationnelle considérable.

Description: Cette mesure vérifie que les modules logiciels et composants sont suffisamment exempts d'erreurs de conception systématiques et/ou de défaillances opérationnelles. L'utilisation de modules logiciels et de composants éprouvés (c'est-à-dire ceux qui ont fait leurs preuves en utilisation) ne sera suffisante que dans de rares cas en tant que seule mesure permettant d'assurer l'intégrité nécessaire relative à la sécurité. Pour les composants complexes comportant un grand nombre de fonctions possibles (comme, par exemple, un système d'exploitation), il est essentiel de déterminer quelles fonctions ont réellement fait leurs preuves en utilisation. Par exemple, lorsqu'un programme de test automatique est utilisé en vue de détecter les défaillances matérielles, on ne pourra considérer que ce programme a fait ses preuves que si aucune défaillance matérielle n'est intervenue pendant la période de fonctionnement.

C.2.9 Modular approach

NOTE This technique/measure is referenced in tables A.4 and B.9 of IEC 61508-3.

Aim: Decomposition of a software system into small comprehensible parts in order to limit the complexity of the system.

Description: A modular approach or modularisation contains several rules for the design, coding and maintenance phases of a software project. These rules vary according to the design method employed during design. Most methods contain the following rules:

- a software module should have a single well-defined task or function to fulfil;
- connections between software modules should be limited and strictly defined, coherence in one software module shall be strong;
- collections of subprograms should be built providing several levels of software modules;
- subprogram sizes should be restricted to some specified value, typically two to four screen sizes;
- subprograms should have a single entry and a single exit only;
- software modules should communicate with other software modules via their interfaces – where global or common variables are used they should be well structured, access should be controlled and their use should be justified in each instance;
- all software module interfaces should be fully documented;
- any software module's interface should contain only those parameters necessary for its function.

Reference: Structured Design – Fundamentals of a Discipline of Computer Program and Systems Design. E. Yourdon, L. L. Constantine, Prentice-Hall, 1979, ISBN 0-13-854471-9.

C.2.10 Use of trusted/verified software modules and components

NOTE 1 This technique/measure is referenced in table A.4 of IEC 61508-3.

NOTE 2 See annex D for some mathematical aspects supporting the following numerical estimates. See also B.5.4 for a similar measure and statistical approach.

Aim: To avoid the need for software modules and hardware component designs to be extensively revalidated or redesigned for each new application. To take advantage of designs which have not been formally or rigorously verified, but for which considerable operational history is available.

Description: This measure verifies that the software modules and components are sufficiently free from systematic design faults and/or operational failures. Only in rare cases will the employment of trusted software modules and components (i.e. those which are proven in use) be sufficient as the sole measure to ensure that the necessary safety integrity is achieved. For complex components with many possible functions (for example an operating system), it is essential to establish which functions are actually sufficiently proven in use. For example, where a self-test routine is provided to detect hardware faults, if no hardware failure occurs within the operating period, one cannot consider the self-test routine for fault detection as being proven by use.

Un composant ou module logiciel est suffisamment éprouvé si l'intégrité relative à la sécurité a été contrôlée ou si les critères suivants sont remplis:

– spécification inchangée;
– systèmes utilisés dans des applications différentes;
– au moins un an d'historique en service;
– temps de fonctionnement dépendant du niveau d'intégrité relative à la sécurité ou du nombre approprié de sollicitations; démonstration d'un taux de défaillance non relative à la sécurité inférieur à
  – 10^{-2} par sollicitation (an) avec un niveau de confiance de 95 % nécessite 300 exploitations opérationnelles (ans),
  – 10^{-5} par sollicitation (an) avec un niveau de confiance de 99,9 % nécessite 690 000 exploitations opérationnelles (ans);
– il faut que toute l'expérience en exploitation se rapporte à un profil de sollicitation connu en ce qui concerne les fonctions du module logiciel, afin de s'assurer qu'une expérience en exploitation grandissante conduit véritablement à une connaissance accrue du comportement du module logiciel par rapport au profil de sollicitation;
– pas de défaillances relatives à la sécurité.

NOTE 3 Une défaillance qui, dans un certain contexte, peut ne pas être critique par rapport à la sécurité, peut, dans un autre contexte, être critique par rapport à la sécurité, et inversement.

Les éléments suivants doivent être documentés en vue de vérifier qu'un composant ou module logiciel remplit les critères ci-dessus:

– identification exacte de chaque système et de ses composants, y compris les numéros de version (à la fois pour les composants logiciels et matériels);
– identification des utilisateurs et durée d'utilisation;
– temps d'exécution;
– procédure de sélection de l'utilisateur: système utilisé et cas d'application;
– procédures de détection et de prise en compte des défaillances et d'élimination des anomalies.

Références:

DIN V VDE 0801 A1: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Änderung 1 zu DIN V VDE 0801/01.90. Beuth-Verlag, Berlin, 1994.

Guidelines for safe automation of chemical processes. CCPS, AIChE, New York, 1993.

C.3 Conception d'architecture

C.3.1 Détection d'anomalie et diagnostic

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Détecter les anomalies d'un système susceptible d'occasionner une défaillance en vue d'établir les bases des contre-mesures à utiliser pour minimiser les conséquences des défaillances.

Description: La détection d'anomalie est l'activité permettant de vérifier la présence d'états erronés à l'intérieur d'un système (états causés par une anomalie du système ou sous-système à contrôler). L'objectif principal de la détection d'anomalie est de neutraliser les effets de résultats faux. Un système qui agit conjointement avec des composants parallèles et qui «rend la main» lorsqu'il détecte que ses propres résultats sont incorrects est appelé «dispositif d'auto-contrôle».

A component or software module can be sufficiently trusted if it is already verified to the required safety integrity level, or if it fulfils the following criteria:

- unchanged specification;
- systems in different applications;
- at least one year of service history;
- operating time according to the safety integrity level or suitable number of demands; demonstration of a non-safety-related failure rate of less than
  - 10^{-2} per demand (year) with a confidence of 95 % requires 300 operational runs (years),
  - 10^{-5} per demand (year) with a confidence of 99,9 % requires 690 000 operational runs (years);
- all of the operating experience must relate to a known demand profile of the functions of the software module, to ensure that increased operating experience genuinely leads to an increased knowledge of the behaviour of the software module relative to that demand profile;
- no safety-related failures.

NOTE 3 A failure which may not be safety critical in one context can be safety critical in another, and vice versa.

To enable verification that a component or software module fulfils the criteria, the following must be documented:

- exact identification of each system and its components, including version numbers (for both software and hardware);
- identification of users, and time of application;
- operating time;
- procedure for the selection of the user-applied systems and application cases;
- procedures for detecting and registering failures, and for removing faults.

References:

DIN V VDE 0801 A1: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Änderung 1 zu DIN V VDE 0801/01.90. Beuth-Verlag, Berlin, 1994.

Guidelines for safe automation of chemical processes. CCPS, AIChE, New York, 1993

C.3 Architecture design

C.3.1 Fault detection and diagnosis

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To detect faults in a system, which might lead to a failure, thus providing the basis for counter-measures in order to minimise the consequences of failures.

Description: Fault detection is the activity of checking a system for erroneous states (caused by a fault within the (sub)system to be checked). The primary goal of fault detection is to inhibit the effect of wrong results. A system which acts in combination with parallel components, relinquishing control when it detects its own results are incorrect, is called self-checking.

La détection d'anomalie est fondée sur les principes de redondance (principalement pour la détection des anomalies matérielles – voir CEI 61508-2 annexe A) et de diversité (anomalies logicielles). Un système de vote est nécessaire pour décider de la conformité des résultats. Les méthodes spéciales applicables sont: la programmation par assertion, la programmation N-versions et l'utilisation d'un dispositif externe de sécurité ainsi que l'introduction d'éléments matériels: capteurs additionnels, boucles de contrôle, codes détecteurs d'erreur, etc.

La détection d'anomalie peut être obtenue par des contrôles effectués sur les valeurs ou les temps à différents niveaux, en particulier au niveau physique (température, tension, etc.), logique (codes détecteurs d'erreur), fonctionnel (assertions) ou externe (contrôles de vraisemblance). Les résultats de ces contrôles peuvent être mémorisés et associés aux données concernées en vue de faciliter la recherche des défaillances.

Les systèmes complexes sont constitués de sous-systèmes. L'efficacité de la détection d'anomalie, du diagnostic et de la compensation des anomalies dépend de la complexité des interactions entre les sous-systèmes, laquelle complexité a un effet sur la propagation des anomalies.

Il convient que le diagnostic d'anomalie soit utilisé au niveau du plus petit sous-système, puisque les petits sous-systèmes permettent un diagnostic d'anomalie plus détaillé (détection des états erronés).

Des systèmes d'information intégrés au niveau de l'entreprise peuvent communiquer l'état des systèmes relatifs à la sécurité de manière périodique aux autres systèmes de supervision (y compris les informations de test de diagnostic). Toute détection d'anomalie peut être mise en évidence et utilisée pour déclencher une action corrective avant que la situation devienne dangereuse. Enfin, en cas d'incident, la documentation de telles anomalies peut faciliter l'analyse ultérieure.

Référence: Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

C.3.2 Codes de détection et correction d'erreurs

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Détecter et corriger les erreurs concernant les informations sensibles.

Description: Pour une information de *n* bits, un bloc codé de *k* bits est généré afin de permettre la détection et la correction de *r* erreurs. Deux exemples types sont les codes de Hamming et les codes polynomiaux.

A noter que, dans les systèmes relatifs à la sécurité, il sera normalement nécessaire d'éliminer les anomalies plutôt que de tenter de les corriger, étant donné que seule une portion prédéterminée d'erreurs peut être corrigée de manière satisfaisante.

Références:

The Technology of Error Correcting Codes. E. R. Berlekamp, Proc. IEEE 68 (5), 1980.

A Short Course on Error Correcting Codes. N. J. A. Sloane, Springer Verlag, Wien, 1975.

C.3.3 Programmation par assertion des défaillances

NOTE Cette technique/mesure est référencée dans le tableau A.18 de la CEI 61508-2 et dans le tableau A.2 de la CEI 61508-3.

But: Détecter les anomalies de conception logicielle résiduelles pendant l'exécution d'un programme en vue d'éviter les défaillances du système critiques pour la sécurité et continuer l'exécution afin d'obtenir un haut niveau de fiabilité.

Fault detection is based on the principles of redundancy (mainly to detect hardware faults – see IEC 61508-2 annex A) and diversity (software faults). Some sort of voting is needed to decide on the correctness of results. Special methods applicable are: assertion programming, N-version programming and the safety bag technique; and for hardware: introducing additional sensors, control loops, error checking codes, etc.

Fault detection may be achieved by checks in the value domain or in the time domain on different levels, especially physical (temperature, voltage etc), logical (error detecting codes), functional (assertions) or external (plausibility checks). The results of these checks may be stored and associated with the data affected to allow failure tracking.

Complex systems are composed of subsystems. The efficiency of fault detection, diagnosis and fault compensation depends on the complexity of the interactions among the subsystems, which influences the propagation of faults.

Fault diagnosis should be applied at the smallest subsystem level, since smaller subsystems allow a more detailed diagnosis of faults (detection of erroneous states).

Integrated enterprise-wide information systems can routinely communicate the status of safety systems, including diagnostic testing information, to other supervisory systems. If an anomaly is detected, it can be highlighted and used to trigger corrective action before a hazardous situation develops. Lastly, if an incident does occur, documentation of such anomalies can aid the subsequent investigation.

Reference: Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

C.3.2 Error detecting and correcting codes

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To detect and correct errors in sensitive information.

Description: For an information of *n* bits, a coded block of *k* bits is generated which enables *r* errors to be detected and corrected. Two example types are Hamming codes and polynomial codes.

It should be noted that in safety-related systems it will normally be necessary to discard faulty data rather than try to correct it, since only a predetermined fraction of errors may be corrected properly.

References:

The Technology of Error Correcting Codes. E. R. Berlekamp, Proc. IEEE 68 (5), 1980.

A Short Course on Error Correcting Codes. N. J. A. Sloane, Springer Verlag, Wien, 1975.

C.3.3 Failure assertion programming

NOTE This technique/measure is referenced in table A.18 of IEC 61508-2, and table A.2 of IEC 61508-3.

Aim: To detect residual software design faults during execution of a program, in order to prevent safety critical failures of the system and to continue operation for high reliability.

Description: L'idée directrice de la méthode de programmation par assertion est de contrôler une précondition (la validité des conditions initiales est vérifiée avant l'exécution d'une séquence d'instructions) et une postcondition (les résultats sont vérifiés après l'exécution d'une séquence d'instructions). Si la précondition ou postcondition n'est pas remplie, le traitement signale l'erreur.

Par exemple,

```
assertion de < précondition>;
    action 1;
       :
       :
    action x;
   assertion de < postcondition>;
```

Références:

A Discipline of Programming. E. W. Dijkstra, Prentice-Hall, 1976.

The Science of Programming. D. Gries, Springer Verlag, 1981.

Software Development – A Rigorous Approach. C. B. Jones, Prentice-Hall, 1980.

C.3.4 Dispositif externe de sécurité

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Assurer la protection du logiciel contre les anomalies de spécification et d'implémentation résiduelles susceptibles d'affecter la sécurité.

Description: Un dispositif externe de sécurité est un système de surveillance qui est mis en œuvre à partir d'un ordinateur indépendant selon une spécification différente. Ce dispositif a pour seul objectif de vérifier que l'ordinateur principal exécute des opérations sûres mais pas nécessairement correctes. Il surveille l'ordinateur principal de manière permanente et empêche tout état du système contraire à la sécurité. De plus, en cas de détection d'un état d'ordinateur potentiellement dangereux, il faut que le système soit ramené à l'état sûr soit par le dispositif externe de sécurité soit par l'ordinateur principal.

Il convient que le matériel et le logiciel de sécurité externes soient classés et qualifiés suivant le SIL approprié.

Référence: Using AI Techniques to Improve Software Safety. Proc. IFAC SAFECOMP 88, Sarlat, France, Pergamon Press, October 1986.

C.3.5 Diversité logicielle (programmation diversifiée)

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Détecter et masquer les anomalies résiduelles de conception et d'implémentation logicielle pendant l'exécution d'un programme en vue d'éviter les défaillances du système critiques pour la sécurité et de continuer l'exécution afin d'obtenir un haut niveau de fiabilité.

Description: En programmation diversifiée, une spécification de programme donnée est conçue et mise en œuvre *N* fois de différentes manières. Les mêmes valeurs d'entrée sont attribuées aux *N* versions et les résultats produits par les *N* versions sont comparés. Si le résultat est considéré comme valide, il est transmis aux sorties de l'ordinateur.

Description: The assertion programming method follows the idea of checking a pre-condition (before a sequence of statements is executed, the initial conditions are checked for validity) and a post-condition (results are checked after the execution of a sequence of statements). If either the pre-condition or the post-condition is not fulfilled, the processing reports the error.

For example,

```
assert < pre-condition>;
    action 1;
       :
       :
    action x;
   assert < post-condition>;
```

References:

A Discipline of Programming. E. W. Dijkstra, Prentice-Hall, 1976.

The Science of Programming. D. Gries, Springer Verlag, 1981.

Software Development – A Rigorous Approach. C. B. Jones, Prentice-Hall, 1980.

C.3.4 Safety bag

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To protect against residual specification and implementation faults in software which adversely affect safety.

Description: A safety bag is an external monitor, implemented on an independent computer to a different specification. This safety bag is solely concerned with ensuring that the main computer performs safe, not necessarily correct, actions. The safety bag continuously monitors the main computer. The safety bag prevents the system from entering an unsafe state. In addition, if it detects that the main computer is entering a potentially hazardous state, the system has to be brought back to a safe state either by the safety bag or the main computer.

Hardware and software of the safety bag should be classified and qualified according to the appropriate SIL.

Reference: Using AI Techniques to Improve Software Safety. Proc. IFAC SAFECOMP 88, Sarlat, France, Pergamon Press, October 1986.

C.3.5 Software diversity (diverse programming)

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: Detect and mask residual software design and implementation faults during execution of a program, in order to prevent safety critical failures of the system, and to continue operation for high reliability.

Description: In diverse programming a given program specification is designed and implemented *N* times in different ways. The same input values are given to the *N* versions, and the results produced by the *N* versions are compared. If the result is considered to be valid, the result is transmitted to the computer outputs.

Les *N* versions peuvent être exécutées en parallèle sur des ordinateurs séparés, avec l'alternative de pouvoir exécuter toutes les versions sur le même ordinateur et de soumettre les résultats à un vote interne. Différentes stratégies de vote peuvent être utilisées pour les *N* versions selon les prescriptions de l'application, comme suit.

– Si le système a un état sûr, il est alors possible de demander la conformité complète (toutes les versions sont conformes); dans le cas contraire, une valeur de sortie entraînant la mise à l'état sûr du système est utilisée. Pour les systèmes à déclenchement simple, le vote peut être systématiquement orienté dans le sens de la sécurité. Dans ce cas, la réponse sûre serait de déclencher lorsqu'un déclenchement est demandé par une version quelconque. Cette approche ne nécessite normalement que deux versions (*N*=2).
– Pour les systèmes sans état sûr, des stratégies de vote majoritaire peuvent être utilisées. En cas de non-conformité collective, des approches probabilistes peuvent être utilisées en vue de maximiser les chances de sélectionner la valeur correcte comme, par exemple, en prenant la valeur moyenne, en figeant temporairement les sorties jusqu'au retour en conformité, etc.

Cette technique n'élimine pas les anomalies résiduelles de conception logicielle, ni les erreurs dans l'interprétation de la spécification, mais elle fournit une mesure de détection et de masquage avant que la sécurité ne soit affectée.

Références:

Dependable Computing: From Concepts to Design Diversity. A. Avizienis and J. C. Laprie, Proc. IEEE 74 (5), May 1986.

A Theoretical Basis for the Analysis of Multi-version Software subject to Co-incident Failures. D. E. Eckhardt and L. D. Lee, IEEE Trans SE-11 (12), 1985.

Computers can now perform vital safety functions safely. Otto Berg Von Linde, Railway Gazette International, Vol. 135, n° 11, 1979.

C.3.6 Bloc de récupération

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Accroître les chances du programme de remplir finalement sa fonction.

Description: Plusieurs segments de programme sont écrits, souvent indépendamment les uns des autres, et chaque segment est prévu pour remplir la même fonction. Le programme final est réalisé à partir de ces segments. Le premier segment, appelé segment primaire, est exécuté en premier. Cette exécution est suivie d'un test d'acceptation applicable au résultat calculé. Si le test est satisfaisant, le résultat est accepté et envoyé aux parties suivantes du système. Si le test n'est pas satisfaisant, tous les effets de bord du premier segment sont remis à l'état initial et le second segment, appelé segment auxiliaire, est exécuté. Cette exécution est également suivie d'un test d'acceptation avec les mêmes conclusions que dans le premier cas. D'autres segments auxiliaires (deuxième, troisième, etc.) peuvent être utilisés si nécessaire.

Références:

System Structure for Software Fault Tolerance. B. Randall. IEEE Trans Software Engineering, Vol. SE-1, n° 2, 1975.

Fault Tolerance – Principles and Practice. T. Anderson, P. A. Lee, Prentice-Hall, 1981.

The *N* versions can run in parallel on separate computers, alternatively all versions can be run on the same computer and the results subjected to an internal vote. Different voting strategies can be used on the *N* versions, depending on the application requirements, as follows.

– If the system has a safe state, then it is feasible to demand complete agreement (all *N* agree) otherwise an output value is used that will cause the system to reach the safe state. For simple trip systems the vote can be biased in the safe direction. In this case the safe action would be to trip if either version demanded a trip. This approach typically uses only two versions (*N*=2).

– For systems with no safe state, majority voting strategies can be employed. For cases where there is no collective agreement, probabilistic approaches can be used in order to maximise the chance of selecting the correct value, for example, taking the middle value, temporary freezing of outputs until agreement returns, etc.

This technique does not eliminate residual software design faults, nor does it avoid errors in the interpretation of the specification, but it provides a measure to detect and mask before they can affect safety.

References:

Dependable Computing: From Concepts to Design Diversity. A. Avizienis and J. C. Laprie, Proc. IEEE 74 (5), May 1986.

A Theoretical Basis for the Analysis of Multi-version Software subject to Co-incident Failures. D. E. Eckhardt and L. D. Lee, IEEE Trans SE-11 (12), 1985.

Computers can now perform vital safety functions safely. Otto Berg von Linde, Railway Gazette International, Vol. 135, No. 11, 1979.

C.3.6 Recovery block

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To increase the likelihood of the program eventually performing its intended function.

Description: Several different program sections are written, often independently, each of which is intended to perform the same desired function. The final program is constructed from these sections. The first section, called the primary, is executed first. This is followed by an acceptance test of the result it calculates. If the test is passed then the result is accepted and passed on to subsequent parts of the system. If it fails, any side-effects of the first are reset and the second section, called the first alternative, is executed. This too is followed by an acceptance test and is treated as in the first case. A second, third or even more alternatives can be provided if desired.

References:

System Structure for Software Fault Tolerance. B. Randall. IEEE Trans Software Engineering, Vol. SE-1, No. 2, 1975.

Fault Tolerance – Principles and Practice. T. Anderson, P. A. Lee, Prentice-Hall, 1981.

C.3.7 Récupération arrière

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Permettre un fonctionnement correct en présence d'une ou plusieurs anomalies.

Description: Lorsqu'une anomalie a été détectée, le système est ramené à un état interne précédent dont la cohérence a déjà été démontrée. Cette méthode nécessite la sauvegarde fréquente de l'état interne au niveau de points de contrôle bien définis. Cela peut être fait globalement (pour la base de données complète) ou par incréments (changements intervenant uniquement entre les points de contrôle). Le système doit alors compenser les changements qui se sont produits pendant ce temps en utilisant la journalisation (vérification à rebours des actions), la compensation (tous les effets des changements sont annulés) ou l'interaction externe (manuelle).

Référence: Software Fault Tolerance (Trends in Software, n° 3), M. R. Lyu (ed.), John Wiley & Sons, April 1995, ISBN 0471950688.

C.3.8 Récupération avant

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Permettre un fonctionnement correct en présence d'une ou plusieurs anomalies.

Description: Lorsqu'une anomalie a été détectée, l'état présent du système est manipulé afin d'obtenir un état qui sera cohérent quelques instants plus tard. Ce concept est particulièrement adapté aux systèmes temps réel ayant une base de données réduite et une vitesse de changement d'état interne rapide. On suppose qu'au moins une partie de l'état du système peut être imposée à l'environnement et que seulement une partie des états du système est influencée (forcée) par l'environnement.

Référence: Software Fault Tolerance (Trends in Software, n° 3), M. R. Lyu (ed.), John Wiley & Sons, April 1995, ISBN 0471950688.

C.3.9 Mécanismes de récupération d’anomalie par relance

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Tenter la récupération fonctionnelle à partir d'une condition d’anomalie détectée, à l'aide de mécanismes de relance.

Description: Suite à une détection d’anomalie ou d'erreur, des tentatives sont effectuées en vue de rétablir la situation en exécutant de nouveau le même code. La récupération par relance peut être aussi complète que de relancer le système ou d’exécuter une procédure de redémarrage ou peut se faire par une petite tâche de replanification et de redémarrage après dépassement du temps imparti par le logiciel ou après une action de supervision d'une tâche. Les techniques de relance sont communément utilisées pour la récupération des anomalies ou d'erreurs de communication et les conditions de relance peuvent être signalées à partir d'une erreur de protocole de communication (checksum, etc.) ou à partir d'un dépassement du temps de réponse d'accusé de réception de communication.

Référence: Reliable Computer Systems: Design and Evaluation, D. P. Siewiorek and R. S. Schwartz, A. K. Peters Ltd., 1998, ISBN 156881092X.

C.3.7 Backward recovery

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To provide correct functional operation in the presence of one or more faults.

Description: If a fault has been detected, the system is reset to an earlier internal state, the consistency of which has been proven before. This method implies saving of the internal state frequently at so-called well-defined checkpoints. This may be done globally (for the complete database) or incrementally (changes only between checkpoints). Then the system has to compensate for the changes which have taken place in the meantime by using journalling (audit trail of actions), compensation (all effects of these changes are nullified) or external (manual) interaction.

Reference: Software Fault Tolerance (Trends in Software, No. 3), M. R. Lyu (ed.), John Wiley & Sons, April 1995, ISBN 0471950688.

C.3.8 Forward recovery

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To provide correct functional operation in the presence of one or more faults.

Description: If a fault has been detected, the current state of the system is manipulated to obtain a state, which will be consistent some time later. This concept is especially suited for real-time systems with a small database and fast rate of change of internal state. It is assumed that at least part of the system state may be imposed onto the environment, and only part of the system states are influenced (forced) by the environment.

Reference: Software Fault Tolerance (Trends in Software, No. 3), M. R. Lyu (ed.), John Wiley & Sons, April 1995, ISBN 0471950688.

C.3.9 Re-try fault recovery mechanisms

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To attempt functional recovery from a detected fault condition by re-try mechanisms.

Description: In the event of a detected fault or error condition, attempts are made to recover the situation by re-executing the same code. Recovery by re-try can be as complete as a reboot and a re-start procedure or a small re-scheduling and re-starting task, after a software time-out or a task monitoring action. Re-try techniques are commonly used in communication fault or error recovery, and re-try conditions could be flagged from a communication protocol error (checksum, etc.) or from a communication acknowledgement response time-out.

Reference: Reliable Computer Systems: Design and Evaluation, D. P. Siewiorek and R. S. Schwartz, A. K. Peters Ltd., 1998, ISBN 156881092X.

C.3.10 Mémorisation de cas d'exécution

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Forcer le logiciel à avoir une défaillance en toute sécurité lorsqu'il essaie de suivre un chemin qui n'est pas autorisé.

Description: Tous les détails significatifs de chaque exécution de programme sont documentés. En fonctionnement normal, chaque exécution de programme est comparée avec les détails documentés précédemment. Si une différence existe, une mesure de sécurité est prise.

La documentation d'exécution peut être la séquence des chemins de décision-à-décision individuels (chemins DD) ou la séquence des accès individuels à des tableaux, enregistrements ou volumes, ou les deux.

Différentes méthodes de mémorisation des chemins d'exécution sont possibles. Les méthodes d'adressage calculé peuvent être utilisées pour mettre en correspondance la séquence d'exécution sur un nombre important unique ou une séquence de nombres. En fonctionnement normal, la valeur du trajet d'exécution doit être contrôlée par rapport aux cas mémorisés avant toute opération de sortie.

Etant donné que les combinaisons possibles de chemins de décision-à-décision au cours d'un programme sont très importantes, il peut s'avérer impossible de traiter les programmes comme un tout. Dans ce cas, la technique peut être appliquée au niveau des modules logiciels.

Référence: Fail-safe Software – Some Principles and a Case Study. W. Ehrenberger. Proc. SARSS 1987, Altrincham, Manchester, UK, Elsevier Applied Science, 1987.

C.3.11 Dégradation «élégante»

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Maintenir les fonctions les plus critiques du système disponibles en dépit des défaillances, par abandon des fonctions les moins critiques.

Description: Cette technique donne la priorité aux différentes fonctions qui doivent être exécutées par le système. La conception garantit qu'il existe suffisamment de ressources pour exécuter toutes les fonctions du système, celles-ci étant traitées dans l'ordre des priorités. Par exemple, les fonctions de consignation des erreurs et événements peuvent être moins prioritaires que les fonctions de commande du système, ce qui permet de continuer à commander le système en cas de défaillance de la partie matérielle chargée de la consignation des erreurs. De plus, en cas de défaillance de la partie matérielle chargée de la commande du système, sans défaillance de celle chargée de la consignation des erreurs, c'est cette dernière partie qui assume la fonction de commande.

Cette technique s'applique principalement au matériel mais également au système entier. Il faut la prendre en compte dès le début de la phase de conception.

Références:

Space Shuttle Software. C. T. Sheridan, Datamation, Vol. 24, July 1978.

The Evolution of Fault-Tolerant Computing. Vol. 1 of Dependable Computing and Fault-Tolerant Systems, Edited by A. Avizienis, H. Kopetz and J. C. Laprie, Springer Verlag, 1987, ISBN 3-211-81941-X.

Fault Tolerance, Principle and Practices. T. Anderson and P. A. Lee, Vol. 3 of Dependable Computing and Fault-Tolerant Systems, Springer Verlag, 1987, ISBN 3-211-82077-9.

C.3.10 Memorising executed cases

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To force the software to fail safely if it attempts to execute a path which is not allowed.

Description: All relevant details of each program execution is documented. During normal operation each program execution is compared with the previously documented details. If it differs, a safety action is taken.

The execution documentation can contain the sequence of the individual decision-to-decision paths (DD paths) or the sequence of the individual accesses to arrays, records or volumes, or both.

Different methods of storing execution paths are possible. Hash-coding methods can be used to map the execution sequence onto a single large number or sequence of numbers. During normal operation the execution path value must be checked against the stored cases before any output operation occurs.

Since the possible combinations of decision-to-decision paths during one program is very large, it may not be feasible to treat programs as a whole. In this case, the technique can be applied at software module level.

Reference: Fail-safe Software – Some Principles and a Case Study. W. Ehrenberger. Proc. SARSS 1987, Altrincham, Manchester, UK, Elsevier Applied Science, 1987.

C.3.11 Graceful degradation

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To maintain the more critical system functions available, despite failures, by dropping the less critical functions.

Description: This technique gives priorities to the various functions to be carried out by the system. The design ensures that if there is insufficient resources to carry out all the system functions, the higher priority functions are carried out in preference to the lower ones. For example, error and event logging functions may be lower priority than system control functions, in which case system control would continue if the hardware associated with error logging were to fail. Further, should the system control hardware fail, but not the error logging hardware, then the error logging hardware would take over the control function.

This is predominantly applied to hardware but is applicable to the total system. It must be taken into account from the topmost design phase.

References:

Space Shuttle Software. C. T. Sheridan, Datamation, Vol. 24, July 1978.

The Evolution of Fault-Tolerant Computing. Vol. 1 of Dependable Computing and Fault-Tolerant Systems, Edited by A. Avizienis, H. Kopetz and J. C. Laprie, Springer Verlag, 1987, ISBN 3-211-81941-X.

Fault Tolerance, Principle and Practices. T. Anderson and P. A. Lee, Vol. 3 of Dependable Computing and Fault-Tolerant Systems, Springer Verlag, 1987, ISBN 3-211-82077-9.

C.3.12 Correction d'anomalie en utilisant les techniques d'intelligence artificielle

NOTE 1 Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Etre capable de réagir aux dangers éventuels de manière très souple en utilisant une combinaison de méthodes et de modèles de processus et une certaine sorte d'analyse de sécurité et de fiabilité en ligne.

Description: La prévision des anomalies (tendances obtenues par calcul), la correction des anomalies et les actions de maintenance et de surveillance peuvent être réalisées de manière très efficace sur des voies diversifiées du système en utilisant des systèmes basés sur l'intelligence artificielle (IA) puisque les règles peuvent être directement déduites des spécifications et vérifiées par comparaison avec ces spécifications. Certaines anomalies communes qui sont introduites dans les spécifications en pensant déjà implicitement à certaines règles de conception et d'implémentation peuvent être évitées de manière efficace grâce à cette approche, en particulier lorsque l'on applique une combinaison de modèles et de méthodes d'une manière fonctionnelle et descriptive.

Les méthodes sont sélectionnées de telle manière que les anomalies puissent être corrigées et les effets des défaillances minimisés en vue d'obtenir l'intégrité de sécurité désirée.

NOTE 2 Voir en C.3.2 un avertissement sur la correction des données erronées, et la CEI 61508-3, tableau A.2, point 5, pour des recommandations négatives relatives à cette technique.

Références:

Automatic Programming Techniques Applied to Software Development: An approach based on exception handling. M. Bidoit et al, Proc. 1st Int. Conf. on Applications of Artificial Intelligence to Engineering Problems, Southampton, 165-177, 1986.

Artificial Intelligence and the Design of Expert Systems. G. F. Luger and W. A. Stubblefield, Benjamin/Cummings, 1989.

C.3.13 Reconfiguration dynamique

NOTE Cette technique/mesure est référencée dans le tableau A.2 de la CEI 61508-3.

But: Conserver la fonctionnalité du système en dépit d'une anomalie interne.

Description: L'architecture logique du système doit être telle qu'elle puisse être mise en correspondance sur un sous-ensemble des ressources disponibles du système. Cette architecture doit être capable de détecter la défaillance d'une ressource physique et de mettre à nouveau en correspondance l'architecture logique sur les ressources limitées encore en fonctionnement. Bien que le concept soit plus traditionnellement limité à la récupération à partir d'unités matérielles défaillantes, il s'applique également aux unités logicielles défaillantes lorsque la «redondance en exécution» est suffisante pour permettre une relance logicielle, ou lorsqu'il y a suffisamment de données redondantes pour qu'une défaillance individuelle et isolée soit considérée comme peu importante.

Il faut prendre en compte cette technique au tout début de la phase de conception du système.

Références:

Critical Issues in the Design of Reconfigurable Control Computer, H. Schmid, J. Lam, R. Naro and K. Weir, FTCS 14 June 1984, IEEE, 1984.

Assigning Processes to Processors: A Fault-tolerant Approach. G. Kar and C. N. Nikolaou, Watson Research Centre, Yorktown, June 1984.

C.3.12 Artificial intelligence fault correction

NOTE 1 This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To be able to react to possible hazards in a very flexible way by introducing a combination of methods and process models and some kind of on-line safety and reliability analysis.

Description: Fault forecasting (calculating trends), fault correction, maintenance and supervisory actions may be supported by artificial intelligence (AI) based systems in a very efficient way in diverse channels of a system, since the rules might be derived directly from the specifications and checked against these. Certain common faults which are introduced into specifications, by implicitly already having some design and implementation rules in mind, may be avoided effectively by this approach, especially when applying a combination of models and methods in a functional or descriptive manner.

The methods are selected in such a way that faults may be corrected and the effects of failures be minimised, in order to meet the desired safety integrity.

NOTE 2 See C.3.2 for warning about correcting faulty data, and item 5, table A.2 of IEC 61508-3 for negative recommendations concerning this technique.

References:

Automatic Programming Techniques Applied to Software Development: An approach based on exception handling. M. Bidoit et al, Proc. 1st Int. Conf. on Applications of Artificial Intelligence to Engineering Problems, Southampton, 165-177, 1986.

Artificial Intelligence and the Design of Expert Systems. G. F. Luger and W. A. Stubblefield, Benjamin/Cummings, 1989.

C.3.13 Dynamic reconfiguration

NOTE This technique/measure is referenced in table A.2 of IEC 61508-3.

Aim: To maintain system functionality despite an internal fault.

Description: The logical architecture of the system has to be such that it can be mapped onto a subset of the available resources of the system. The architecture needs to be capable of detecting a failure in a physical resource and then remapping the logical architecture back onto the restricted resources left functioning. Although the concept is more traditionally restricted to recovery from failed hardware units, it is also applicable to failed software units if there is sufficient "run-time redundancy" to allow a software re-try or if there is sufficient redundant data to make the individual and isolated failure be of little importance.

This technique must be considered at the first system design stage.

References:

Critical Issues in the Design of Reconfigurable Control Computer, H. Schmid, J. Lam, R. Naro and K. Weir, FTCS 14 June 1984, IEEE, 1984.

Assigning Processes to Processors: A Fault-tolerant Approach. G. Kar and C. N. Nikolaou, Watson Research Centre, Yorktown, June 1984.

C.4 Outils de développement et langages de programmation

C.4.1 Langages de programmation fortement typés

NOTE Cette technique/mesure est référencée dans le tableau A.3 de la CEI 61508-3.

But: Réduire la probabilité d'anomalies en utilisant un langage qui permette un contrôle de haut niveau par le compilateur.

Description: Lorsqu'un langage de programmation fortement typé est compilé, plusieurs contrôles sont effectués en ce qui concerne le mode d'utilisation des types de variable comme, par exemple, dans les procédures d'appel et l'accès aux données externes. La compilation échouera et un message d'erreur sera émis pour chaque utilisation non conforme aux règles prédéfinies.

De tels langages servent normalement à définir des types de données définis par l'utilisateur à partir de types de données liés à un langage de base (comme les nombres entiers ou réels). Ces types peuvent alors être utilisés exactement de la même façon que les types de base. Des contrôles stricts sont imposés afin de s'assurer que le type correct est utilisé. Ces contrôles sont imposés au programme entier, même si celui-ci a été réalisé à partir d'unités compilées séparément. Les contrôles permettent également de vérifier que le nombre et le type d'arguments des procédures sont bien adaptés, même lorsqu'ils sont référencés à partir de modules logiciels compilés séparément.

Les langages fortement typés supportent en général d'autres aspects de bonne pratique de génie logiciel tels que des structures de contrôle facilement analysables (par exemple si.. alors.. sinon, faire.. pendant, etc.) qui conduisent à des programmes bien structurés.

Des exemples types de langages fortement typés sont les langages Pascal, ADA et Modula 2.

Références:

In Search of Effective Diversity: a Six Language Study of Fault-Tolerant Flight Control Software. A. Avizienis, M. R. Lyu and W. Schutz. 18th Symposium on Fault-Tolerant Computing, Tokyo, Japan, 27-30 June 1988, IEEE Computer Society Press, 1988, ISBN 0-8186-0867-6.

ISO/CEI 8652:1995, *Technologies de l'information – Langages de programmation – Ada*. (Publiée actuellement en anglais seulement.)

ISO/CEI 10514-1:1996, *Technologies de l'information – Langages de programmation – Partie 1: Modula-2, langage de base*.

ISO 7185:1990, T*echnologies de l'information – Langages de programmation – Pascal*. (Publiée actuellement en anglais seulement.)

C.4.2 Sous-ensembles de langages

NOTE Cette technique/mesure est référencée dans le tableau A.3 de la CEI 61508-3.

But: Réduire la probabilité d'introduction d'anomalies de programmation et augmenter la probabilité de détection des anomalies restantes.

Description: Le langage est examiné afin de déterminer les structures de programmation qui sont soit sujettes aux erreurs soit difficiles à analyser, par exemple en utilisant des méthodes d'analyse statique. Un sous-ensemble du langage est alors défini en vue d'exclure ces structures.

C.4 Development tools and programming languages

C.4.1 Strongly typed programming languages

NOTE This technique/measure is referenced in table A.3 of IEC 61508-3.

Aim: Reduce the probability of faults by using a language which permits a high level of checking by the compiler.

Description: When a strongly typed programming language is compiled, many checks are made on how variable types are used, for example in procedure calls and external data access. Compilation will fail and an error message be produced for any usage that does not conform to predefined rules.

Such languages usually allow user-defined data types to be defined from the basic language data types (such as integer, real). These types can then be used in exactly the same way as the basic type. Strict checks are imposed to ensure the correct type is used. These checks are imposed over the whole program, even if this is built from separately compiled units. The checks also ensure that the number and the type of procedure arguments match even when referenced from separately compiled software modules.

Strongly typed languages usually support other aspects of good software engineering practice such as easily analysable control structures (for example if.. then.. else, do.. while, etc.) which lead to well-structured programs.

Typical examples of strongly typed languages are Pascal, Ada and Modula 2.

References:

In Search of Effective Diversity: a Six Language Study of Fault-Tolerant Flight Control Software. A. Avizienis, M. R. Lyu and W. Schutz. 18th Symposium on Fault-Tolerant Computing, Tokyo, Japan, 27-30 June 1988, IEEE Computer Society Press, 1988, ISBN 0-8186-0867-6.

ISO/IEC 8652:1995, *Information technology – Programming languages – Ada*.

ISO/IEC 10514-1:1996, *Information technology – Programming languages – Part 1: Modula-2, Base Language*.

ISO 7185:1990, *Information technology – Programming languages – Pascal*.

C.4.2 Language subsets

NOTE This technique/measure is referenced in table A.3 of IEC 61508-3.

Aim: To reduce the probability of introducing programming faults and increase the probability of detecting any remaining faults.

Description: The language is examined to determine programming constructs which are either error-prone or difficult to analyse, for example, using static analysis methods. A language subset is then defined which excludes these constructs.

Références:

Requirements for programming languages in safety and security software standard. B. A. Wichmann. Computer Standards and Interfaces. Vol. 14, pp 433-441, 1992.

Safer C: Developing Software for High-integrity and Safety-critical Systems. L. Hatton, McGraw-Hill, 1994, ISBN 0-07-707640-0.

C.4.3 Outils certifiés et traducteurs certifiés

NOTE Cette technique/mesure est référencée dans le tableau A.3 de la CEI 61508-3.

But: Des outils sont nécessaires pour assister les développeurs au cours des différentes phases de développement d'un logiciel. Dans la mesure du possible, il convient que les outils soient certifiés de manière à assurer un certain niveau de fiabilité en ce qui concerne la conformité des sorties.

Description: La certification d'un outillage sera généralement réalisée par un organisme indépendant, souvent national, à l'aide de critères déterminés de manière indépendante, typiquement dans des normes nationales ou internationales. L'idéal serait que les outils utilisés pendant les phases de développement (spécification, conception, codage, essai et validation) et ceux utilisés pour la gestion de configuration soient soumis à la certification.

Actuellement, seuls les compilateurs (traducteurs) sont soumis régulièrement aux procédures de certification; ces procédures sont établies par les organismes de certification nationaux et permettent de tester les compilateurs (traducteurs) par rapport aux normes internationales comme celles relatives aux langages Ada et Pascal.

Il est important de noter que les outils certifiés et les traducteurs certifiés sont en général certifiés uniquement par rapport à leur propre langage ou normes de processus. Il ne sont en général en aucune façon certifiés en ce qui concerne la sécurité.

Références:

Pascal Validation Suite. UK Distributor: BSI Quality Assurance, PO Box 375, Milton Keynes, MK14 6LL.

Ada Validation Suite. UK Distributor: National Computing Centre (NCC), Oxford Road, Manchester, England.

C.4.4 Outils et traducteurs: confiance accrue résultant de l'utilisation

NOTE Cette technique/mesure est référencée dans le tableau A.3 de la CEI 61508-3.

But: Eviter des difficultés dues aux défaillances du traducteur susceptibles de se produire pendant le développement, la vérification et la maintenance d'un progiciel.

Description: Utilisation d'un traducteur pour lequel il n'existe pas de preuve de fonctionnement incorrect au cours de plusieurs projets antérieurs. Il convient d'éviter l'utilisation de traducteurs sans expérience d'exploitation ou comportant des anomalies graves connues à moins qu'il n'existe quelque autre attestation de fonctionnement correct (voir par exemple C.4.4.1).

Au cas où le traducteur aurait montré de petites déficiences, les structures de langage correspondantes seront notées afin d'être soigneusement évitées au cours d'un projet relatif à la sécurité.

References:

Requirements for programming languages in safety and security software standard. B. A. Wichmann. Computer Standards and Interfaces. Vol. 14, pp 433-441, 1992.

Safer C: Developing Software for High-integrity and Safety-critical Systems. L. Hatton, McGraw-Hill, 1994, ISBN 0-07-707640-0.

C.4.3 Certified tools and certified translators

NOTE This technique/measure is referenced in table A.3 of IEC 61508-3.

Aim: Tools are necessary to help developers in the different phases of software development. Wherever possible, tools should be certified so that some level of confidence can be assumed regarding the correctness of the outputs.

Description: The certification of a tool will generally be carried out by an independent, often national, body, against independently set criteria, typically national or international standards. Ideally, the tools used in all development phases (specification, design, coding, testing and validation) and those used in configuration management, should be subject to certification.

To date, only compilers (translators) are regularly subject to certification procedures; these are laid down by national certification bodies and they exercise compilers (translators) against international standards such as those for Ada and Pascal.

It is important to note that certified tools and certified translators are usually certified only against their respective language or process standards. They are usually not certified in any way with respect to safety.

References:

Pascal Validation Suite. UK Distributor: BSI Quality Assurance, PO Box 375, Milton Keynes, MK14 6LL.

Ada Validation Suite. UK Distributor: National Computing Centre (NCC), Oxford Road, Manchester, England.

C.4.4 Tools and translators: increased confidence from use

NOTE This technique/measure is referenced in table A.3 of IEC 61508-3.

Aim: To avoid any difficulties due to translator failures which can arise during development, verification and maintenance of a software package.

Description: A translator is used, where there has been no evidence of improper performance over many prior projects. Translators without operating experience or with any serious known faults should be avoided unless there is some other assurance of correct performance (for example, see C.4.4.1).

If the translator has shown small deficiencies, the related language constructs are noted down and carefully avoided during a safety related project.

Une autre procédure consiste à limiter l'utilisation du langage aux éléments communément utilisés.

La présente recommandation est basée sur l'expérience acquise au cours de plusieurs projets. Il a été démontré que les traducteurs immatures constituent un sérieux handicap pour le développement des logiciels. Ils rendent généralement impossible le développement des logiciels relatifs à la sécurité.

Il faut également savoir qu'il n'existe actuellement aucune méthode permettant de démontrer la conformité de toutes les parties d'un outil ou d'un traducteur.

C.4.4.1 Comparaison du programme source et du code exécutable

But: Vérifier que les outils utilisés pour produire une image PROM n'ont introduit aucune erreur dans l'image PROM.

Description: L'image PROM est soumise à un processus de rétro-ingénierie pour obtenir les modules «objets». Ces modules «objets» sont soumis à un processus de rétro-ingénierie pour obtenir des fichiers en assembleur. En utilisant des techniques appropriées les fichiers en langage assembleur ainsi régénérés sont comparés avec le code source réel utilisés à l'origine pour définir la PROM.

L'avantage majeur de cette technique est que les outils (compilateur, éditeur de liens, etc.) utilisés pour produire l'image PROM ne nécessitent pas d'être validés pour tous les programmes. Cette technique permet de vérifier que le fichier source utilisé pour le système relatif à la sécurité considéré est correctement transformé.

Références:

Demonstrating Equivalence of Source Code and PROM Contents. D. J. Pavey and L. A. Winsborrow. The Computer Journal Vol. 36, n° 7, 1993.

Formal demonstration of equivalence of source code and PROM contents: an industrial example. D. J. Pavey and L. A. Winsborrow. Mathematics of Dependable Systems, Ed. C. Mitchell and V. Stavridou, Clarendon Press, 1995, ISBN 0-198534-91-4.

Retrospective Formal Verification of Reactor Protection System Software. D. J. Pavey, L. A. Winsborrow, A. R. Lawrence. Proceedings of the Second Safety Through Quality Conference, 1995, ISBN 1-897851-06-5.

Assuring Correctness in a Safety Critical Software Application. L. A. Winsborrow and D. J. Pavey. High Integrity Systems, Vol. 1, n° 5, pp 453-459, 1996.

C.4.5 Bibliothèque des modules logiciels et composants éprouvés/vérifiés

NOTE Cette technique/mesure est référencée dans le tableau A.3 de la CEI 61508-3.

But: Eviter la nécessité pour les modules logiciels et les conceptions de composants matériels de devoir être amplement revalidés ou reconçus pour chaque application nouvelle. Egalement promouvoir les conceptions qui n'ont pas été validées formellement ou rigoureusement mais qui bénéficient d'un historique d'exploitation considérable.

Description: Les PES (systèmes électroniques programmables) correctement conçus et structurés sont constitués d'un certain nombre de modules et composants matériels et logiciels clairement distincts et dont les interactions sont clairement spécifiées.

Another version to this way of working is to restrict the usage of the language to only its commonly used features.

This recommendation is based on the experience from many projects. It has been shown that immature translators are a serious handicap to any software development. They make a safety-related software development generally infeasible.

It is also known, presently, that no method exists to prove the correctness for all tool or translator parts.

C.4.4.1 Comparison of source program and executable code

Aim: To check that the tools used to produce a PROM image have not introduced any errors into the PROM image.

Description: The PROM image is reverse-engineered to obtain the constituent "object" modules. These "object" modules are reverse-engineered into assembly language files. Using suitable techniques the reverse generated assembly language files are compared with the actual source files originally used to produce the PROM.

The major advantage of the technique is that the tools (compilers, linkers etc.) used to produce the PROM image do not have to be validated for all programs. The technique verifies that source file used for the particular safety-related system are correctly transformed.

References:

Demonstrating Equivalence of Source Code and PROM Contents. D. J. Pavey and L. A. Winsborrow. The Computer Journal Vol. 36, No. 7, 1993.

Formal demonstration of equivalence of source code and PROM contents: an industrial example. D. J. Pavey and L. A. Winsborrow. Mathematics of Dependable Systems, Ed. C. Mitchell and V. Stavridou, Clarendon Press, 1995, ISBN 0-198534-91-4.

Retrospective Formal Verification of Reactor Protection System Software. D. J. Pavey, L. A. Winsborrow, A. R. Lawrence. Proceedings of the Second Safety Through Quality Conference, 1995, ISBN 1-897851-06-5.

Assuring Correctness in a Safety Critical Software Application. L. A. Winsborrow and D. J. Pavey. High Integrity Systems, Vol. 1, No. 5, pp 453-459, 1996.

C.4.5 Library of trusted/verified software modules and components

NOTE This technique/measure is referenced in table A.3 of IEC 61508-3.

Aim: To avoid the need for software modules and hardware component designs to be extensively revalidated or redesigned for each new application. Also to promote designs which have not been formally or rigorously validated but for which considerable operational history is available.

Description: Well-designed and structured PESs are made up of a number of hardware and software components and modules which are clearly distinct and which interact with each other in clearly specified ways.

Différents PES conçus pour des applications différentes contiendront un certain nombre de modules logiciels ou de composants qui sont identiques ou très semblables. La constitution d'une bibliothèque comprenant de tels modules d'application générale permet de partager la plupart des ressources nécessaires à la validation des conceptions entre plusieurs applications.

De plus, l'utilisation de ces modules logiciels pour de multiples applications fournit la preuve empirique d'une exploitation satisfaisante. Cette preuve empirique accroît légitimement la confiance que les utilisateurs sont susceptibles d'avoir dans les modules logiciels.

C.2.10 décrit une des approches selon laquelle un module logiciel peut être considéré comme digne de confiance.

Références:

Software Reuse and Reverse Engineering in Practice. P. A. V. Hall (ed.), Chapman & Hall, 1992, ISBN 0-412-39980-6.

DIN V VDE 0801 A1: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Änderung 1 zu DIN V VDE 0801/01.90. Beuth-Verlag, Berlin, 1994.

C.4.6 Langages de programmation adéquats

NOTE Cette technique/mesure est référencée dans le tableau A.3 de la CEI 61508-3.

But: Prendre en compte les prescriptions de la présente Norme internationale autant que possible, en particulier en ce qui concerne la programmation défensive, le typage fort, la programmation structurée et éventuellement les assertions. Il convient que le langage de programmation choisi aboutisse à un code facilement vérifiable avec un minimum d'effort et faciliter le développement, la vérification et la maintenance du programme.

Description: Il convient que le langage soit complètement et clairement défini. Il convient que le langage soit orienté vers l'utilisateur ou le problème plutôt que vers le processeur/la plateforme de la machine. Les langages communément utilisés et leurs sous-ensembles sont préférables aux langages spécialisés.

En plus des caractéristiques déjà mentionnées, le langage doit permettre

– une structure par blocs;
– le contrôle du temps de traduction; et
– le contrôle du type d'exécution et des limites de tableaux.

Il convient que le langage favorise

– l'utilisation de modules logiciels gérables et de petite taille;
– la restriction d'accès aux données à des modules logiciels spécifiques;
– la définition de sous-ensembles de variables; et
– tout autre type de construction de limitation des erreurs.

Si le fonctionnement sûr du système dépend de contraintes temps réel, il convient alors que le langage permette également le traitement des exceptions/interruptions.

Il est souhaitable que le langage soit supporté par un traducteur approprié, des bibliothèques appropriées des modules logiciels préexistants, un programme de mise au point et des outils concernant à la fois le contrôle et le développement des versions.

Actuellement, au moment de l'établissement de la présente norme, on ne sait pas encore s'il est préférable d'utiliser des langages orientés objet ou d'autre langages conventionnels.

Different PESs designed for differing applications will contain a number of software modules or components which are the same or very similar. Building up a library of such generally applicable software modules allows much of the resource necessary for validating the designs to be shared by more than one application.

Furthermore, the use of such software modules in multiple applications provides empirical evidence of successful operational use. This empirical evidence justifiably enhances the trust which users are likely to have in the software modules.

C.2.10 describes one approach by which a software module may be classified as trusted.

References:

Software Reuse and Reverse Engineering in Practice. P. A. V. Hall (ed.), Chapman & Hall, 1992, ISBN 0-412-39980-6.

DIN V VDE 0801 A1: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Änderung 1 zu DIN V VDE 0801/01.90. Beuth-Verlag, Berlin, 1994.

C.4.6 Suitable programming languages

NOTE This technique/measure is referenced in table A.3 of IEC 61508-3.

Aim: To support the requirements of this International Standard as much as possible, in particular defensive programming, strong typing, structured programming and possibly assertions. The programming language chosen should lead to an easily verifiable code with a minimum of effort and facilitate program development, verification and maintenance.

Description: The language should be fully and unambiguously defined. The language should be user- or problem-orientated rather than processor/platform machine-orientated. Widely used languages or their subsets are preferred to special purpose languages.

In addition to the already referenced features the language should provide for

– block structure;
– translation time checking; and
– run-time type and array bound checking.

The language should encourage

– the use of small and manageable software modules;
– restriction of access to data in specific software modules;
– definition of variable subranges; and
– any other type of error-limiting constructs.

If safe operation of the system is dependent upon real-time constraints, then the language should also provide for exception/interrupt handling.

It is desirable that the language is supported by a suitable translator, appropriate libraries of pre-existing software modules, a debugger and tools for both version control and development.

Currently, at the time of developing this standard, it is not clear whether object-oriented languages are to be preferred to other conventional ones.

Les éléments suivants rendent la vérification difficile et il convient donc de les éviter:

– les branchement inconditionnels, à l'exception des appels de sous-programmes;
– la récursion;
– les pointeurs, segments de mémoire ou tout type d'objets ou de variables dynamiques;
– la gestion des interruptions au niveau du code source;
– les entrées ou sorties multiples de boucles, blocs ou sous-programmes;
– la déclaration ou initialisation implicite de variables;
– les enregistrements de variantes et les équivalences; et
– les paramètres de procédure.

Les langages de bas niveau, en particulier les langages assembleurs, présentent des problèmes liés au fait qu'ils sont par nature, orientés vers le processeur/la plateforme de la machine.

L'une des propriétés souhaitables d'un langage est que sa conception et son utilisation donnent des programmes dont l'exécution est prévisible. Pour un langage de programmation défini de façon appropriée, il existe un sous-ensemble de ce langage qui assure que l'exécution d'un programme est prévisible. Ce sous-ensemble ne peut pas, en général, être déterminé statiquement, bien que de nombreuses contraintes statiques puissent aider à assurer une exécution prévisible. Cela nécessiterait, typiquement, que les indices de matrices n'aient aucune limite, qu'aucun dépassement numérique n'apparaisse, etc.

Le tableau C.1 donne les recommandations applicables aux langages de programmation spécifiques.

Références:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

CEI 60880:1986, *Logiciel pour les calculateurs utilisés dans les systèmes de sûreté des centrales nucléaires*.

CEI 61131-3:1993, *Automates programmables – Partie 3: Langages de programmation.*

ISO/CEI 1539-1:1997, *Technologies de l'information – Langages de programmation – Fortran – Partie 1: Langage de base.*

ISO/CEI 7185:1990, *Technologies de l'information – Langages de programmation – Pascal.* (Publiée actuellement en anglais seulement.)

ISO/CEI 8652:1995, *Technologies de l'information – Langages de programmation – Ada*.

ISO/CEI 9899:1990, *Langages de programmation – C*.

ISO/CEI/TR 10206:1991, *Technologies de l'information – Langages de programmation – Pascal étendu.*

ISO/CEI 10514-1:1996, *Technologies de l'information – Langages de programmation – Partie 1: Modula-2, langage de base*.

ISO/CEI 10514-3:1998, *Technologies de l'information – Langages de programmation – Partie 3: Modula-2 orienté objet.*

ISO/CEI 14882:1998, *Langages de programmation – C++*.

ISO/IEC/TR 15942, *Guidance for the use of the Ada programming language in high integrity systems*.[1)]

_____

1) A publier.

Features which make verification difficult and therefore should be avoided are

- unconditional jumps excluding subroutine calls;
- recursion;
- pointers, heaps or any type of dynamic variables or objects;
- interrupt handling at source code level;
- multiple entries or exits of loops, blocks or subprograms;
- implicit variable initialisation or declaration;
- variant records and equivalence; and
- procedural parameters.

Low-level languages, in particular assembly languages, present problems due to their processor/platform machine-orientated nature.

A desirable language property is that its design and use should result in programs whose execution is predictable. Given a suitably defined programming language, there is a subset which ensures that program execution is predictable. This subset cannot (in general) be statically determined, although many static constraints may assist in ensuring predictable execution. This would typically require a demonstration that array indices are within bounds, and that numeric overflow cannot arise, etc.

Table C.1 gives recommendations for specific programming languages.

References:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

IEC 60880:1986, *Software for computers in the safety systems of nuclear power stations*.

IEC 61131-3:1993, *Programmable controllers – Part 3: Programming languages*.

ISO/IEC 1539-1:1997, *Information technology – Programming languages – Fortran – Part 1: Base language*.

ISO/IEC 7185:1990, *Information technology – Programming languages – Pascal*.

ISO/IEC 8652:1995, *Information technology – Programming languages – Ada*.

ISO/IEC 9899:1990, *Programming languages – C*.

ISO/IEC/TR 10206:1991, *Information technology – Programming languages – Extended Pascal*.

ISO/IEC 10514-1:1996, *Information technology – Programming languages – Part 1: Modula-2, Base Language*.

ISO/IEC 10514-3:1998, *Information technology – Programming languages – Part 3: Object Oriented Modula-2.*

ISO/IEC 14882:1998, *Programming languages – C++*.

ISO/IEC/TR 15942, *Guidance for the use of the Ada programming language in high integrity systems*.[1)]

[1)] To be published.

Tableau C.1 – Recommandations applicables aux langages de programmation spécifiques

	Langage de programmation	SIL1	SIL2	SIL3	SIL4
1	ADA	HR	HR	R	R
2	Sous-ensemble ADA	HR	HR	HR	HR
3	MODULA-2	HR	HR	R	R
4	Sous-ensemble MODULA-2	HR	HR	HR	HR
5	PASCAL	HR	HR	R	R
6	Sous-ensemble PASCAL	HR	HR	HR	HR
7	FORTRAN 77	R	R	R	R
8	Sous-ensemble FORTRAN 77	HR	HR	HR	HR
9	C	R	–	NR	NR
10	Sous-ensemble C avec norme de codage et utilisation d'outils d'analyse statique	HR	HR	HR	HR
11	PL/M	R	–	NR	NR
12	Sous-ensemble PL/M avec norme de codage	HR	R	R	R
13	Assembleur	R	R	–	–
14	Sous-ensemble assembleur avec norme de codage	R	R	R	R
15	Diagrammes à contacts	R	R	R	R
16	Diagramme à contacts avec sous-ensemble de langage défini	HR	HR	HR	HR
17	Diagrammes de blocs fonctionnels	R	R	R	R
18	Diagrammes de blocs fonctionnels avec sous-ensemble de langage défini	HR	HR	HR	HR
19	Texte structuré	R	R	R	R
20	Texte structuré avec sous-ensemble de langage défini	HR	HR	HR	HR
21	Diagramme fonctionnel séquentiel	R	R	R	R
22	Diagramme fonctionnel séquentiel avec sous-ensemble de langage défini	HR	HR	HR	HR
23	Liste d'instructions	R	–	NR	NR
24	Liste d'instructions avec sous-ensemble de langage défini	HR	R	R	R

NOTE 1 Les recommandations R, HR et – sont explicitées dans l'annexe A de la CEI 61508-3.

NOTE 2 Le logiciel système comprend le système d'exploitation, les fonctions spécifiques intégrées dans les drivers et les modules logiciels faisant partie du système. Le logiciel est normalement fourni par le vendeur du système relatif à la sécurité. Le sous-ensemble de langage doit être soigneusement choisi afin d'éviter les structures complexes pouvant conduire à des anomalies d'implémentation. Il convient que les contrôles soient réalisés en vue de vérifier l'utilisation correcte du sous-ensemble de langage.

NOTE 3 Le logiciel d'application est un logiciel qui a été développé pour une application spécifique relative à la sécurité. Dans beaucoup de cas, ce logiciel est développé par l'utilisateur final ou par le fournisseur d'application. Lorsque les recommandations sont applicables à plusieurs langages de programmation, le développeur choisira celui qui est le plus utilisé dans l'industrie ou dans l'unité de production concernée. Il convient que le sous-ensemble de langage soit soigneusement choisi afin d'éviter les structures complexes pouvant conduire à des anomalies d'implémentation. Il convient que des contrôles soient réalisés en vue de vérifier l'utilisation correcte du sous-ensemble de langage.

NOTE 4 Tout langage spécifique ne figurant pas dans le présent tableau, n'a pas lieu d'être considéré comme un langage à exclure. Il convient toutefois qu'il soit conforme à la présente Norme internationale.

NOTE 5 Pour les entrées 15 à 24, voir la CEI 61131-3.

Table C.1 – Recommendations for specific programming languages

	Programming language	SIL1	SIL2	SIL3	SIL4
1	ADA	HR	HR	R	R
2	ADA with subset	HR	HR	HR	HR
3	MODULA-2	HR	HR	R	R
4	MODULA-2 with subset	HR	HR	HR	HR
5	PASCAL	HR	HR	R	R
6	PASCAL with subset	HR	HR	HR	HR
7	FORTRAN 77	R	R	R	R
8	FORTRAN 77 with subset	HR	HR	HR	HR
9	C	R	–	NR	NR
10	C with subset and coding standard, and use of static analysis tools	HR	HR	HR	HR
11	PL/M	R	–	NR	NR
12	PL/M with subset and coding standard	HR	R	R	R
13	Assembler	R	R	–	–
14	Assembler with subset and coding standard	R	R	R	R
15	Ladder diagrams	R	R	R	R
16	Ladder diagram with defined subset of language	HR	HR	HR	HR
17	Functional block diagram	R	R	R	R
18	Function block diagram with defined subset of language	HR	HR	HR	HR
19	Structured text	R	R	R	R
20	Structured text with defined subset of language	HR	HR	HR	HR
21	Sequential function chart	R	R	R	R
22	Sequential function chart with defined subset of language	HR	HR	HR	HR
23	Instruction list	R	–	NR	NR
24	Instruction list with defined subset of language	HR	R	R	R

NOTE 1 The recommendations R, HR and – are explained in annex A of IEC 61508-3.

NOTE 2 System software includes the operating system, drivers, embedded functions and software modules provided as part of the system. The software is typically provided by the safety system vendor. The language subset should be carefully selected to avoid complex structures which may result in implementation faults. Checks should be performed to check for proper use of the language subset.

NOTE 3 The application software is the software developed for a specific safety application. In many cases this software is developed by the end user or by an application oriented contractor. Where a number of programming languages have the same recommendation, the developer should select one which is commonly used by personnel in the industry or facility. The language subset should be carefully selected to avoid complex structures which may result in implementation faults. Checks should be performed to check for proper use of the language subset.

NOTE 4 If a specific language is not listed in the table, it must not be assumed that it is excluded. It should conform to this International Standard.

NOTE 5 For entries 15-24, see IEC 61131-3.

C.5 Vérification et modification

C.5.1 Test probabiliste

NOTE Cette technique/mesure est référencée dans les tableaux A.5, A.7 et A.9 de la CEI 61508-3.

But: Obtenir une indication quantitative concernant les propriétés de confiance du logiciel examiné.

Description: Cette indication quantitative peut prendre en compte les niveaux de fiabilité et de signification associés et peut correspondre à

– une probabilité de défaillance par sollicitation;
– une probabilité de défaillance pendant une certaine période de temps; et
– une probabilité de confinement des erreurs.

A partir de ces indications, d'autres paramètres peuvent être obtenus, tels que:

– la probabilité d'exécution sans défaillances;
– la probabilité de survie;
– la disponibilité;
– le MTBF ou le taux de défaillance; et
– la probabilité d'exécution sûre.

Les considérations probabilistes sont basées soit sur un test probabiliste soit sur l'expérience en exploitation. Normalement, le nombre de cas de test ou de cas observés en exploitation est très important. Typiquement, le test du mode de fonctionnement à la demande nécessite un temps beaucoup plus court que pour le mode de fonctionnement en continu.

Les outils de test automatisés sont normalement utilisés pour fournir les données de test et superviser les sorties de test. Des tests importants sont réalisés sur les grands ordinateurs centraux avec les équipements périphériques de simulation de processus appropriés. Les données de test sont sélectionnées d'un point de vue à la fois systématique et aléatoire. Par exemple, le test global utilise un profil de données de test alors que la sélection aléatoire de données peut être utilisée pour la réalisation de tests individuels détaillés.

L'utilisation de bancs de test logiciels, l'exécution et la supervision de test individuels sont déterminées en fonction des objectifs détaillés des tests, comme décrit ci-dessus. D'autres conditions importantes résultent des conditions mathématiques préalables qu'il faut remplir pour que l'évaluation atteigne l'objectif prévu par le test.

Les chiffres probabilistes concernant le comportement de tout objet testé peuvent être également obtenus d'après l'expérience en exploitation. Sous réserve que les mêmes conditions soient remplies, les mêmes méthodes mathématiques que celles utilisées pour l'évaluation des résultats de test peuvent être appliquées.

En pratique, il est très difficile de démontrer des niveaux de fiabilité extrêmement élevés à l'aide de ces techniques.

Références:

Software Testing via Environmental Simulation (CONTESSE Report). Available until December 1998 from: Ray Browne, CIID, DTI, 151 Buckingham Palace Road, London, SW1W 9SS, UK, 1994.

C.5 Verification and modification

C.5.1 Probabilistic testing

NOTE This technique/measure is referenced in tables A.5, A.7 and A.9 of IEC 61508-3.

Aim: To gain a quantitative figure about the reliability properties of the investigated software.

Description: This quantitative figure may take into account the related levels of confidence and significance and can give

– a failure probability per demand;

– a failure probability during a certain period of time; and

– a probability of error containment.

From these figures other parameters may be derived such as:

– probability of failure free execution;

– probability of survival;

– availability;

– MTBF or failure rate; and

– probability of safe execution.

Probabilistic considerations are either based on a probabilistic test or on operating experience. Usually, the number of test cases or observed operating cases is very large. Typically, the testing of the demand mode of operation involves considerably less elapsed time than the continuous mode of operation.

Automated testing tools are normally employed to provide test data and supervise test outputs. Large tests are run on large host computers with the appropriate process simulation periphery. Test data is selected both according to systematic and random hardware viewpoints. The overall test control, for example, guarantees a test data profile, while random selection can govern individual test cases in detail.

Individual test harnesses, test executions and test supervisions are determined by the detailed test aims as described above. Other important conditions are given by the mathematical prerequisites that must be fulfilled if the test evaluation is to meet its intended test aim.

Probabilistic figures about the behaviour of any test object may also be derived from operating experience. Provided the same conditions are met, the same mathematics can be applied as for the evaluation of test results.

In practice, it is very difficult to demonstrate ultra-high levels of reliability using these techniques.

References:

Software Testing via Environmental Simulation (CONTESSE Report). Available until December 1998 from: Ray Browne, CIID, DTI, 151 Buckingham Palace Road, London, SW1W 9SS, UK, 1994.

Validation of ultra high dependability for software based systems. B. Littlewood and L Strigini. Comm. ACM 36 (11), 69-80, 1993.

Handbook of Software Reliability Engineering. M. R. Lyu (ed.). IEEE Computer Society Press, McGraw-Hill, 1995, ISBN 0-07-039400-8.

C.5.2 Enregistrement et analyse de données

NOTE Cette technique/mesure est référencée dans les tableaux A.5 et A.8 de la CEI 61508-3.

But: Documenter toutes les données, décisions et justifications associées du projet de logiciel en vue de faciliter la vérification, la validation, l'évaluation et la maintenance.

Description: Une documentation détaillée est maintenue pendant un projet, pouvant inclure

– les tests réalisés sur chaque module logiciel;
– les décisions et leurs justifications;
– les problèmes et leurs solutions.

Cette documentation peut être analysée au cours et à la fin du projet en vue d'obtenir une grande variété d'informations. En particulier, l'enregistrement des données est très important pour la maintenance des systèmes informatiques étant donné que la raison de certaines décisions prises pendant le projet de développement n'est pas toujours connue des ingénieurs de maintenance.

Référence: Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989, ISBN 1-85166-381-9.

C.5.3 Test d'interface

NOTE Cette technique/mesure est référencée dans le tableau A.5 de la CEI 61508-3.

But: Détecter les erreurs dans les interfaces de sous-programmes.

Description: Plusieurs niveaux de détail ou de complétude des tests sont possibles. Les niveaux les plus importants sont les tests mettant en jeu

– toutes les variables d'interface à leurs valeurs extrêmes
– toutes les variables d'interface considérées individuellement à leurs valeurs extrêmes avec les autres variables d'interface à leur valeur normale;
– toutes les valeurs du domaine de chaque variable d'interface avec les autres variables d'interface à leur valeur normale;
– toutes les valeurs de toutes les variables combinées (ceci peut seulement être réalisé sur les petites interfaces);
– les conditions de test spécifiées concernant chaque appel de chaque sous-programme.

Ces tests sont particulièrement importants lorsque les interfaces ne contiennent pas d'assertions permettant de détecter les valeurs de paramètres incorrectes. Ils sont également importants après l'élaboration de nouvelles configurations de sous-programmes préexistants.

C.5.4 Analyse des valeurs aux limites

NOTE Cette technique/mesure est référencée dans les tableaux B.2, B.3 and B.8 de la CEI 61508-3.

But: Détecter les erreurs logicielles aux frontières ou limites des paramètres.

Validation of ultra high dependability for software based systems. B. Littlewood and L Strigini. Comm. ACM 36 (11), 69-80, 1993.

Handbook of Software Reliability Engineering. M. R. Lyu (ed.). IEEE Computer Society Press, McGraw-Hill, 1995, ISBN 0-07-039400-8.

C.5.2 Data recording and analysis

NOTE This technique/measure is referenced in tables A.5 and A.8 of IEC 61508-3.

Aim: To document all data, decisions and rationale in the software project to allow for easier verification, validation, assessment and maintenance.

Description: Detailed documentation is maintained during a project, which could include

– testing performed on each software module;

– decisions and their rationale;

– problems and their solutions.

During and at the conclusion of the project this documentation can be analysed to establish a wide variety of information. In particular, data recording is very important for the maintenance of computer systems as the rationale for certain decisions made during the development project is not always known by the maintenance engineers.

Reference: Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989, ISBN 1-85166-381-9.

C.5.3 Interface testing

NOTE This technique/measure is referenced in table A.5 of IEC 61508-3.

Aim: To detect errors in the interfaces of subprograms.

Description: Several levels of detail or completeness of testing are feasible. The most important levels are tests for

– all interface variables at their extreme values;

– all interface variables individually at their extreme values with other interface variables at normal values;

– all values of the domain of each interface variable with other interface variables at normal values;

– all values of all variables in combination (this will only be feasible for small interfaces);

– the specified test conditions relevant to each call of each subroutine.

These tests are particularly important if the interfaces do not contain assertions that detect incorrect parameter values. They are also important after new configurations of pre-existing subprograms have been generated.

C.5.4 Boundary value analysis

NOTE This technique/measure is referenced in tables B.2, B.3 and B.8 of IEC 61508-3.

Aim: To detect software errors occurring at parameter limits or boundaries.

Description: Le domaine d'entrée du programme est divisé en un certain nombre de classes d'entrée d'après la relation d'équivalence (voir C.5.7). Il convient que les tests couvrent les frontières et les extrêmes des classes. Ils vérifient que les frontières du domaine d'entrée de la spécification coïncident avec celles du programme. L'utilisation de la valeur zéro dans le cas d'une traduction directe aussi bien qu'indirecte est souvent sujette à des erreurs et nécessite une attention spéciale:

– diviseur par zéro;
– caractères ASCII blancs;
– élément de liste ou pile vide;
– matrice pleine;
– entrée de table à zéro.

Normalement, les limites pour les entrées correspondent directement avec celles de la plage de sortie. Il convient que les cas de test soient écrits de manière à forcer la sortie à atteindre ses valeurs limites. Il faut également considérer s'il est possible ou non de spécifier un cas de test pour lequel la sortie serait amenée à dépasser les valeurs limites de la spécification.

Si la sortie est une séquence de données comme, par exemple, une table imprimée, il convient de porter une attention particulière aux premier et dernier éléments ainsi qu'aux listes contenant zéro, un et deux éléments.

Références:

CEI 61704, *Lignes directrices pour le choix de méthodes de test en vue de l'évaluation de la fiabilité des logiciels*[1)].

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

C.5.5 Estimation des erreurs

NOTE Cette technique/mesure est référencée dans les tableaux B.2 et B.8 de la CEI 61508-3.

But: Eliminer les erreurs de programmation communes.

Description: L'expérience acquise durant les tests et l'intuition alliée à la connaissance et à la curiosité du système testé peut entraîner la prise en compte de quelques cas de tests hors catégorie n'appartenant pas au jeu de cas de test prévu.

Les valeurs spéciales ou les combinaisons de valeurs peuvent entraîner des erreurs. Certains cas de test intéressants peuvent être extraits des listes de contrôle. La robustesse du système peut être également mise en question. Exemple: Est-il possible d'appuyer trop rapidement ou trop souvent sur les boutons de la face avant ? Que se passe-t-il lorsque l'on appuie simultanément sur deux boutons ?

Référence: The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

C.5.6 Implantation d'erreurs

NOTE Cette technique/mesure est référencée dans le tableau B.2 de la CEI 61508-3.

But: S'assurer qu'un jeu de cas de test est adéquat.

---

1) A publier.

Description: The input domain of the program is divided into a number of input classes according to the equivalence relation (see C.5.7). The tests should cover the boundaries and extremes of the classes. The tests check that the boundaries in the input domain of the specification coincide with those in the program. The use of the value zero, in a direct as well as in an indirect translation, is often error-prone and demands special attention:

– zero divisor;
– blank ASCII characters;
– empty stack or list element;
– full matrix;
– zero table entry.

Normally the boundaries for input have a direct correspondence to the boundaries for the output range. Test cases should be written to force the output to its limited values. Consider also if it is possible to specify a test case which causes the output to exceed the specification boundary values.

If the output is a sequence of data, for example a printed table, special attention should be paid to the first and the last elements and to lists containing none, one and two elements.

References:

IEC 61704, *Guide to the selection of software test methods for reliability assessment*[1)].

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

C.5.5 Error guessing

NOTE This technique/measure is referenced in tables B.2 and B.8 of IEC 61508-3.

Aim: To remove common programming mistakes.

Description: Testing experience and intuition combined with knowledge and curiosity about the system under test may add some uncategorised test cases to the designed test case set.

Special values or combinations of values may be error-prone. Some interesting test cases may be derived from inspection checklists. It may also be considered whether the system is robust enough. For example: can the buttons be pushed on the front-panel too fast or too often? What happens if two buttons are pushed simultaneously?

Reference: The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

C.5.6 Error seeding

NOTE This technique/measure is referenced in table B.2 of IEC 61508-3.

Aim: To ascertain whether a set of test cases is adequate.

---

1) To be published.

Description: Certains types d'erreurs connues sont implantés (semés) dans le programme, puis le programme est exécuté avec les cas de test en condition de test. Si seulement quelques-unes des erreurs implantées sont découvertes, le jeu de cas de test n'est pas adéquat. Le rapport entre les erreurs implantées découvertes et le nombre total d'erreurs implantées correspond à une estimation du rapport entre les erreurs réelles découvertes et le nombre total d'erreurs, ce qui permet d'estimer le nombre d'erreurs restantes et, par conséquent, l'effort de test restant.

$$\frac{\textit{Erreurs implantées découvertes}}{\textit{Nombre total d'erreurs implantées}} = \frac{\textit{Erreurs réelles découvertes}}{\textit{Nombre total d'erreurs réelles}}$$

La détection de toutes les erreurs implantées peut signifier que le jeu de cas de test est adéquat ou que les erreurs implantées étaient trop faciles à découvrir. Les limites de la méthode sont celles qui permettent d'obtenir tout résultat utilisable, les types d'erreurs ainsi que la position des implantations devant refléter la distribution statistique des erreurs réelles.

Référence: Software Fault Injection, J. M. Voas and G. McGraw, Wiley 1998.

C.5.7 Classes d'équivalence et test des partitions d'entrée

NOTE Cette technique/mesure est référencée dans les tableaux B.2 et B.3 de la CEI 61508-3.

But: Tester le logiciel de manière adéquate à l'aide d'un minimum de données de test. Les données de test sont obtenues en sélectionnant les partitions du domaine d'entrée requises pour tester le logiciel.

Description: La stratégie de test est basée sur la relation d'équivalence des entrées, ce qui permet de déterminer une partition du domaine d'entrée.

Les cas de test sont sélectionnés dans le but de couvrir toutes les partitions précédemment spécifiées. Un cas de test au moins est pris en compte pour chaque classe d'équivalence.

Les deux possibilités de base pour la partition des entrées sont les suivantes:

– classes d'équivalence déduites de la spécification; l'interprétation de la spécification peut être orientée entrée (par exemple les valeurs sélectionnées sont traitées de la même manière) ou orientée sortie (par exemple le jeu de valeurs conduisant au même résultat fonctionnel);
– les classes d'équivalence peuvent être déduites de la structure interne du programme. Dans ce cas, les résultats des classes d'équivalence sont déterminés à partir de l'analyse statique du programme (par exemple le jeu de valeurs conduisant au même trajet en cours d'exécution).

Références:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

C.5.8 Tests basés sur la structure

NOTE Cette technique/mesure est référencée dans le tableau B.2 de la CEI 61508-3.

But: Appliquer des tests servant à tester certains sous-ensembles de la structure du programme.

Description: Some known types of mistake are inserted (seeded) into the program, and the program is executed with the test cases under test conditions. If only some of the seeded errors are found, the test case set is not adequate. The ratio of found seeded errors to the total number of seeded errors is an estimate of the ratio of found real errors to total number errors. This gives a possibility of estimating the number of remaining errors and thereby the remaining test effort.

$$\frac{Found\ \ seeded\ \ errors}{Total\ \ number\ \ of\ \ seeded\ \ errors} = \frac{Found\ \ real\ \ errors}{Total\ \ number\ \ of\ \ real\ \ errors}$$

The detection of all the seeded errors may indicate either that the test case set is adequate, or that the seeded errors were too easy to find. The limitations of the method are that in order to obtain any usable results, the types of mistake as well as the seeding positions must reflect the statistical distribution of real errors.

Reference: Software Fault Injection, J. M. Voas and G. McGraw, Wiley 1998.

C.5.7 Equivalence classes and input partition testing

NOTE This technique/measure is referenced in tables B.2 and B.3 of IEC 61508-3.

Aim: To test the software adequately using a minimum of test data. The test data is obtained by selecting the partitions of the input domain required to exercise the software.

Description: This testing strategy is based on the equivalence relation of the inputs, which determines a partition of the input domain.

Test cases are selected with the aim of covering all the partitions previously specified. At least one test case is taken from each equivalence class.

There are two basic possibilities for input partitioning which are

– equivalence classes derived from the specification – the interpretation of the specification may be either input orientated, for example the values selected are treated in the same way, or output orientated, for example the set of values lead to the same functional result;

– equivalence classes derived from the internal structure of the program – the equivalence class results are determined from static analysis of the program, for example the set of values leading to the same path being executed.

References:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

C.5.8 Structure-based testing

NOTE This technique/measure is referenced in table B.2 of IEC 61508-3.

Aim: To apply tests which exercise certain subsets of the program structure.

Description: Fondé sur l'analyse du programme, un jeu de données d'entrée est choisi de manière à tester un fort pourcentage (pourcentage cible souvent prédéfini) du code de programme. Les mesures de couverture du code varieront comme suit en fonction du niveau de rigueur requis.

– **Instructions:** ce test est le moins rigoureux étant donné qu'il est possible d'exécuter toutes les instructions de codage sans tester tous les branchements d'une instruction conditionnelle.

– **Branchements:** il convient de vérifier tous les côtés de chaque branchement, ce qui peut être impossible pour certains types de codes défensifs.

– **Conditions composées:** chaque condition d'un branchement conditionnel composé (c'est-à-dire avec liaison ET/OU) est testée. Voir MCDC (Modified condition decision coverage, réf. DO178B).

– **Séquence de code linéaire avec branchement:** une séquence de code linéaire avec branchement est une séquence linéaire d'instructions de codage, y compris les instructions conditionnelles, terminée par un branchement. Beaucoup de sous-chemins potentiels seront indisponibles à cause des contraintes imposées aux données d'entrée par l'exécution du code précédent.

– **Flux de données:** les chemins d'exécution sont sélectionnés sur la base de l'utilisation des données comme, par exemple, un chemin servant à la fois à l'écriture et à la lecture d'une même variable.

– **Graphe d'appel:** un programme est composé de sous-programmes qui peuvent être appelés à partir d'autres sous-programmes. Le graphe d'appel représente l'arbre des appels de sous-programmes à l'intérieur du programme. Les tests sont prévus pour couvrir tous les appels de l'arbre.

– **Chemin de base:** l'un des chemins d'un ensemble fini de chemins qui vont du début à la fin, de telle sorte que tous les arcs y soient inclus. (Les combinaisons de chemins qui se superposent dans cet ensemble de base peuvent être des chemins quelconques constituant la partie correspondante du programme.) Les tests de tous les chemins de base ont fait leurs preuves dans leur efficacité à détecter les erreurs.

Références:

Reliability of the Path Analysis Testing Strategy. W. Howden. IEEE Trans Software Engineering, Vol. SE-3, 1976.

Software considerations in airborne systems and equip certification, DO178B, RTCA, December 1992.

Structure testing, McCabe; NBS Special Publication 500-99, 1982.

A software reliability study, Walsh [USA] National Computer Conference, 1979.

C.5.9 Analyse du flux de commandes

NOTE Cette technique/mesure est référencée dans le tableau B.8 de la CEI 61508-3.

But: Détecter les structures de programme de faible qualité et potentiellement incorrectes.

Description: L'analyse du flux de commandes est une technique de test statique pour identifier des zones suspectes du code qui ne suivent pas des pratiques de bonne programmation. Le programme est analysé et on produit un graphe orienté permettant de vérifier de manière plus approfondie la présence des éléments suivants:

– code inaccessible résultant, par exemple, des branchements inconditionnels partant des blocs de code inaccessibles;

Description: Based on analysis of the program, a set of input data is chosen so that a large (and often prespecified target) percentage of the program code is exercised. Measures of code coverage will vary as follows, depending upon the level of rigour required.

- **Statements**: this is the least rigorous test since it is possible to execute all code statements without exercising both branches of a conditional statement.
- **Branches**: both sides of every branch should be checked. This may be impractical for some types of defensive code.
- **Compound conditions**: every condition in a compound conditional branch (i.e. linked by AND/OR) is exercised. See MCDC (modified condition decision coverage, ref. DO178B).
- **LCSAJ**: a linear code sequence and jump is any linear sequence of code statements, including conditional statements, terminated by a jump. Many potential subpaths will be infeasible due to constraints on the input data imposed by the execution of earlier code.
- **Data flow**: the execution paths are selected on the basis of data usage; for example, a path where the same variable is both written and read.
- **Call graph**: a program is composed of subroutines which may be invoked from other subroutines. The call graph is the tree of subroutine invocations in the program. Tests are designed to cover all invocations in the tree.
- **Basis path**: one of a minimal set of finite paths from start to finish, such that all arcs are included. (Overlapping combinations of paths in this basis set can form any path through that part of the program.) Tests of all basis path has been shown to be efficient for locating errors.

References:

Reliability of the Path Analysis Testing Strategy. W. Howden. IEEE Trans Software Engineering, Vol. SE-3, 1976.

Software considerations in airborne systems and equip certification, DO178B, RTCA, December 1992.

Structure testing, McCabe; NBS Special Publication 500-99, 1982.

A software reliability study, Walsh [USA] National Computer Conference, 1979.

C.5.9 Control flow analysis

NOTE This technique/measure is referenced in table B.8 of IEC 61508-3.

Aim: To detect poor and potentially incorrect program structures.

Description: Control flow analysis is a static testing technique for finding suspect areas of code that do not follow good programming practice. The program is analysed producing a directed graph which can be further analysed for

- inaccessible code, for instance unconditional jumps which leaves blocks of code unreachable;

– code noué. Un code bien structuré ayant un graphe de commande qui peut être réduit à un simple sommet par réductions successives. A l'inverse, un code faiblement structuré peut n'être réduit qu'à un nœud composé de plusieurs sommets.

Référence: Information Flow and Data Flow of While Programs. J. F. Bergeretti and B. A. Carre, ACM Trans. on Prog. Lang. and Syst., 1985.

C.5.10 Analyse du flux de données

NOTE Cette technique/mesure est référencée dans le tableau B.8 de la CEI 61508-3.

But: Détecter les structures de programme faibles et potentiellement incorrectes.

Description: L'analyse du flux de données est une technique de test statique permettant de combiner les informations obtenues à partir de l'analyse du flux de commandes avec celles concernant la lecture et l'écriture des variables dans différentes parties du code. L'analyse permet de vérifier la présence des éléments suivants:

– variables pouvant être lues avant qu'une valeur ne leur soit affectée (ce qui peut être évité en affectant toujours une valeur au moment de la déclaration de toute nouvelle variable);

– variables pouvant être écrites plusieurs fois avant d'être lues (ce qui peut indiquer une omission de code);

– variables pouvant être écrites mais jamais lues (ce qui peut indiquer un code redondant).

Un anomalie de flux de données ne correspond pas toujours directement à une anomalie de programme mais le nombre d'anomalies susceptibles d'être contenues dans le code sera d'autant moins important que celles du flux de données seront évitées.

Référence: Information Flow and Data Flow of While Programs. J. F. Bergeretti and B. A. Carre, ACM Trans. on Prog. Lang. and Syst., 1985.

C.5.11 Analyse de circuit parasite

NOTE Cette technique/mesure est référencée dans le tableau B.8 de la CEI 61508-3.

But: Détecter tout flux logique ou trajet imprévu à l'intérieur d'un système et qui, dans certaines conditions, donne naissance à une fonction non désirée ou interdit une fonction désirée.

Description: Un trajet de circuit parasite peut avoir une cause matérielle ou logicielle, être occasionné par une intervention de l'opérateur ou résulter d'une combinaison de ces trois facteurs. Les circuits parasites ne sont pas le résultat de défaillances matérielles mais constituent des conditions latentes accidentellement conçues dans le système ou codées dans les programmes logiciels et pouvant entraîner un mauvais fonctionnement dans certaines conditions.

Les catégories de circuits parasites sont les suivantes

– trajets parasites occasionnant un flux de courant, d'énergie ou de données de séquence logique le long d'un trajet imprévu ou dans une direction non désirée;

– synchronisation parasite au cours de laquelle les événements se produisent selon une séquence imprévue ou incompatible;

– indications parasites entraînant un affichage ambigu ou erroné des conditions d'exploitation du système, ce qui peut provoquer une intervention intempestive de l'opérateur;

– labels parasites désignant les fonctions du système de façon incorrecte ou imprécise comme, par exemple, les entrées du système, les commandes, les affichages, les bus, etc., ce qui peut tromper l'opérateur et l'amener à appliquer un stimulus incorrect au système.

– knotted code. Well-structured code has a control graph which is reducible by successive graph reductions to a single node. In contrast, poorly structured code can only be reduced to a knot composed of several nodes.

References:

Information Flow and Data Flow of While Programs. J. F. Bergeretti and B. A. Carre, ACM Trans. on Prog. Lang. and Syst., 1985.

C.5.10 Data flow analysis

NOTE This technique/measure is referenced in table B.8 of IEC 61508-3.

Aim: To detect poor and potentially incorrect program structures.

Description: Data flow analysis is a static testing technique that combines the information obtained from the control flow analysis with information about which variables are read or written in different portions of code. The analysis can check for

– variables that may be read before they are assigned a value – this can be avoided by always assigning a value when declaring a new variable;

– variables that are written more than once without being read – this could indicate omitted code;

– variables that are written but never read – this could indicate redundant code.

A data flow anomaly will not always directly correspond to a program fault, but if anomalies are avoided the code is less likely to contain faults.

References:

Information Flow and Data Flow of While Programs. J. F. Bergeretti and B. A. Carre, ACM Trans. on Prog. Lang. and Syst., 1985.

C.5.11 Sneak circuit analysis

NOTE This technique/measure is referenced in table B.8 of IEC 61508-3.

Aim: To detect an unexpected path or logic flow within a system which, under certain conditions, initiates an undesired function or inhibits a desired function.

Description: A sneak circuit path may consist of hardware, software, operator actions, or combinations of these elements. Sneak circuits are not the result of hardware failure but are latent conditions inadvertently designed into the system or coded into the software programs, which can cause it to malfunction under certain conditions.

Categories of sneak circuits are

– sneak paths which cause current, energy, or logical sequence to flow along an unexpected path or in an unintended direction;

– sneak timing in which events occur in an unexpected or conflicting sequence;

– sneak indications which cause an ambiguous or false display of system operating conditions, and thus may result in an undesired action by the operator;

– sneak labels which incorrectly or imprecisely label system functions, for example, system inputs, controls, displays, buses, etc, and thus may mislead an operator into applying an incorrect stimulus to the system.

L'analyse des circuits parasites repose sur la reconnaissance des structures topologiques de base comme structure matérielle ou logicielle (par exemple, six structures de base ont été proposées pour le logiciel). L'analyse est réalisée à l'aide d'une liste de questions concernant l'utilisation des éléments topologiques de base et les relations entre ces éléments.

Références:

Sneak Analysis and Software Sneak Analysis. S. G. Godoy and G. J. Engels. J. Aircraft Vol. 15, n° 8, 1978.

Sneak Circuit Analysis. J. P. Rankin, Nuclear Safety, Vol. 14, n° 5, 1973.

C.5.12 Exécution symbolique

NOTE Cette technique/mesure est référencée dans le tableau B.8 de la CEI 61508-3.

But: Montrer la concordance entre le code source et la spécification.

Description: Les variables du programme sont évaluées après avoir remplacé le côté gauche par le côté droit pour toutes les affectations. Les branchements conditionnels et les boucles sont traduites en expressions booléennes. Le résultat final consiste en une expression symbolique pour chaque variable de programme. Cela peut être vérifié par rapport à l'expression prévue.

Références:

Formal Program Verification using Symbolic Execution. R. B. Dannenberg and G. W. Ernst. IEEE Transactions on Software Engineering, Vol. SE-8, n° 1, 1982.

Symbolic Execution and Software Testing. J. C. King, Communications of the ACM, Vol. 19, n° 7, 1976.

C.5.13 Preuve formelle

NOTE Cette technique/mesure est référencée dans le tableau A.9 de la CEI 61508-3.

But: Prouver la conformité d'un programme ou d’une spécification sans l'exécuter, à l'aide de règles et de modèles théoriques et mathématiques.

Description: Un certain nombre d'assertions sont énoncées en divers points du programme et sont utilisées comme préconditions et postconditions associées à différents chemins dans le programme. La preuve consiste à démontrer que le programme passe des préconditions aux postconditions selon un ensemble de règles logiques et que le programme s'arrête.

Plusieurs méthodes formelles sont décrites dans la présente norme comme, par exemple les méthodes CCS, CSP, HOL, LOTOS, OBJ, logique temporelle, VDM et Z (voir C.2.4 pour la description de ces méthodes).

Une technique alternative à la Preuve Formelle est l’Argument Rigoureux. Une présentation de la preuve formelle donnant les principales étapes mais pas tous les détails mathématiques est préparée. Il s’agit d’une vérification technique plus faible qui établit qu’une preuve serait possible si elle était recherchée.

Référence:

Software Development – A Rigorous Approach. C. B. Jones. Prentice-Hall, 1980.

Systematic Software Development using VDM. C. B. Jones. Prentice-Hall, 2nd Edition, 1990.

Sneak circuit analysis relies on the recognition of basic topological patterns with the hardware or software structure (for example, six basic patterns have been proposed for software). Analysis takes place with the aid of a checklist of questions about the use and relationships between the basic topological components.

References:

Sneak Analysis and Software Sneak Analysis. S. G. Godoy and G. J. Engels. J. Aircraft Vol. 15, No. 8, 1978.

Sneak Circuit Analysis. J. P. Rankin, Nuclear Safety, Vol. 14, No. 5, 1973.

C.5.12 Symbolic execution

NOTE This technique/measure is referenced in table B.8 of IEC 61508-3.

Aim: To show the agreement between the source code and the specification.

Description: The program variables are evaluated after substituting the left-hand side by the right- hand side in all assignments. Conditional branches and loops are translated into Boolean expressions. The final result is a symbolic expression for each program variable. This can be checked against the expected expression.

References:

Formal Program Verification using Symbolic Execution. R. B. Dannenberg and G. W. Ernst. IEEE Transactions on Software Engineering, Vol. SE-8, No. 1, 1982.

Symbolic Execution and Software Testing. J. C. King, Communications of the ACM, Vol. 19, No. 7, 1976.

C.5.13 Formal proof

NOTE This technique/measure is referenced in table A.9 of IEC 61508-3.

Aim: To prove the correctness of a program or specification without executing it, using theoretical and mathematical models and rules.

Description: A number of assertions are stated at various locations in the program, and they are used as pre- and post-conditions to various paths in the program. The proof consists of showing that the program transfers the pre-conditions into the post-conditions according to a set of logical rules, and that the program terminates.

Several formal methods are described in this overview, for instance, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z (see C.2.4 for descriptions of these methods).

An alternative technique to Formal Proof is Rigorous Argument. An outline of the formal proof is prepared in which the main steps are presented but not all the mathematical detail is included. This is a weaker verification technique that establishes that a proof would be possible if it were to be attempted.

References:

Software Development – A Rigorous Approach. C. B. Jones. Prentice-Hall, 1980.

Systematic Software Development using VDM. C. B. Jones. Prentice-Hall, 2nd Edition, 1990.

C.5.14 Métriques de complexité

NOTE Cette technique/mesure est référencée dans les tableaux A.9 et A.10 de la CEI 61508-3.

But: Prédire les attributs des programmes à partir des caractéristiques du logiciel proprement dit ou de son développement ou d'après l'historique des tests.

Description: Ces modèles évaluent certaines caractéristiques du logiciel et les relient à l'attribut désiré comme la fiabilité ou la complexité. Des outils logiciels sont nécessaires à l'évaluation de la plupart des mesures. Une partie des métriques applicables est indiquée ci-dessous:

– complexité théorique du graphe: cette mesure peut être appliquée au début du cycle de vie en vue d'évaluer les compromis techniques et est basée sur la complexité du graphe de commande du programme représenté pas son nombre cyclomatique;
– nombre de manières d'activer un certain module logiciel (accessibilité) (plus un module logiciel est accessible, plus il a de chances de pouvoir être mis au point);
– science des métriques de type Halstead: cette mesure calcule la longueur du programme en comptant le nombre d'opérateurs et d'opérandes. Elle fournit une mesure de complexité et de volume qui sert de base de comparaison lors de l'évaluation des ressources de développement futures;
– nombre d'entrées et de sorties pour chaque module logiciel: la minimisation du nombre de points d'entrée/sortie est un facteur clé dans l'utilisation des techniques de conception et de programmation structurées.

Références:

Software Metrics: A Rigorous and Practical Approach. N. E. Fenton, International Thomson Computer Press, 1996, ISBN 1-85032-275-9, 2nd Edition.

A Complexity Measure. T. J. McCabe. IEEE Trans on Software Engineering, Vol. SE-2, n° 4, December 1976.

Models and Measurements for Quality Assessments of Software. S. N. Mohanty. ACM Computing Surveys, Vol. 11, n° 3, September 1979.

Elements of Software Science. M. H. Halstead. Elsevier, North Holland, New York, 1977.

C.5.15 Inspection selon Fagan

NOTE Cette technique/mesure est référencée dans le tableau B.8 de la CEI 61508-3.

But: Révéler les erreurs et les anomalies durant toutes les phases du développement d'un programme.

Description: Audit «formel» portant sur les documents d'assurance qualité et destiné à détecter les erreurs et les anomalies. La procédure d'inspection comprend cinq étapes: planification, préparation, contrôle, reprise et suivi. Chacune de ces étapes comporte son propre objectif. Il faut que le développement complet du système (spécification, conception, codage et test) soit inspecté.

Référence: Design and Code Inspections to Reduce Errors in Program Development. M. E. Fagan, IBM Systems Journal, n° 3, 1976.

C.5.14 Complexity metrics

NOTE This technique/measure is referenced in tables A.9 and A.10 of IEC 61508-3.

Aim: To predict the attributes of programs from properties of the software itself or from its development or test history.

Description: These models evaluate some structural properties of the software and relate this to a desired attribute such as reliability or complexity. Software tools are required to evaluate most of the measures. Some of the metrics which can be applied are given below:

– graph theoretic complexity – this measure can be applied early in the lifecycle to assess trade-offs, and is based on the complexity of the program control graph, represented by its cyclomatic number;

– number of ways to activate a certain software module (accessibility) – the more a software module can be accessed, the more likely it is to be debugged;

– Halstead type metrics science – this measure computes the program length by counting the number of operators and operands; it provides a measure of complexity and size that forms a baseline for comparison when estimating future development resources;

– number of entries and exits per software module – minimising the number of entry/exit points is a key feature of structured design and programming techniques.

References:

Software Metrics: A Rigorous and Practical Approach. N. E. Fenton, International Thomson Computer Press, 1996, ISBN 1-85032-275-9, 2nd Edition.

A Complexity Measure. T. J. McCabe. IEEE Trans on Software Engineering, Vol. SE-2, No. 4, December 1976.

Models and Measurements for Quality Assessments of Software. S. N. Mohanty. ACM Computing Surveys, Vol. 11, No. 3, September 1979.

Elements of Software Science. M. H. Halstead. Elsevier, North Holland, New York, 1977.

C.5.15 Fagan inspections

NOTE This technique/measure is referenced in table B.8 of IEC 61508-3.

Aim: To reveal mistakes and faults in all phases of the program development.

Description: A "formal" audit on quality assurance documents aimed at finding mistakes and faults. The inspection procedure consists of five stages: planning, preparation, inspection, rework and follow-up. Each of these stages has its own separate objective. The complete system development (specification, design, coding and testing) must be inspected.

Reference: Design and Code Inspections to Reduce Errors in Program Development. M. E. Fagan, IBM Systems Journal, No. 3, 1976.

C.5.16 Lectures croisées/revues de conception

NOTE Cette technique/mesure est référencée dans le tableau B.8 de la CEI 61508-3.

But: Détecter les anomalies dans des produits du processus de développement aussitôt et aussi économiquement que possible.

Description: La CEI a publié un guide sur les revues de conception formelles qui comprend une description générale de ces revues, leurs objectifs, des détails sur les différents types de revue et la composition d'une équipe de revue de conception ainsi que les obligations et responsabilités des membres de cette équipe. Ce document de la CEI fournit également les lignes directrices générales relatives à la planification et à la conduite des revues de conception formalisées ainsi que des détails spécifiques concernant le rôle des spécialistes indépendants au sein d'une équipe de revue de conception. Par exemple, les fonctions des spécialistes couvrent, entre autres, la fiabilité, le soutien logistique de maintenance et la disponibilité.

La CEI recommande «qu'une revue de conception soit conduite pour tous les nouveaux produits, processus et applications, ainsi que pour les révisions de produits existants et les processus de fabrication qui affecteraient les fonctions, les performances, la sécurité, la sûreté de fonctionnement, la compatibilité électromagnétique, les incidences sur les coûts ou autres caractéristiques concernant le produit ou le processus, les utilisateurs, les observateurs ou la population en général».

Une lecture croisée de code est réalisée par une équipe de lecture croisée qui sélectionne un jeu réduit de cas de tests sur papier représentatifs des jeux d'entrées et de sorties correspondantes prévues du programme. Les données de test sont alors tracées en suivant la logique du programme.

Références:

CEI 61160:1992, *Revue de conception formalisée*.
Amendement 1 (1994).

Software Inspection. T. Gilb, D. Graham, Addison-Wesley, 1993, ISBN 0-201-63181-4.

C.5.17 Prototypage/animation

NOTE Cette technique/mesure est référencée dans les tableaux B.3 et B.5 de la CEI 61508-3.

But: Vérifier la faisabilité de la mise en œuvre du système par rapport aux contraintes imposées. Communiquer au client l'interprétation du spécificateur concernant le système afin de mettre en évidence les erreurs de compréhension.

Description: Un sous-ensemble de fonctions du système, des contraintes et des prescriptions de caractéristiques de fonctionnement est sélectionné. Un prototype est construit à l'aide d'outils de haut niveau. A ce stade, il n'est pas nécessaire de prendre en compte les contraintes telles que l'ordinateur cible, le langage d'implémentation, la taille du programme, la maintenabilité, la fiabilité et la disponibilité. Le prototype est évalué par rapport aux critères du client et les prescriptions applicables au système peuvent être modifiées au vu de cette évaluation.

Références:

The emergence of rapid prototyping as a real-time software development tool. J. E. Cooling, T. S. Hughes, Proc. 2nd Int. Conf. on Software Engineering for Real-time Systems, Cirencester, UK, IEE, 1989.

C.5.16 Walk-throughs/design reviews

NOTE This technique/measure is referenced in table B.8 of IEC 61508-3.

Aim: To detect faults in some product of the development as soon and as economically as possible.

Description: IEC has published a guide on formal design reviews, which includes a general description of formal design reviews, their objectives, details of the various design review types, the composition of a design review team and their associated duties and responsibilities. The IEC document also provides general guidelines for planning and conducting formal design reviews, as well as specific details concerning the role of independent specialists within a design review team. Examples of specialist functions include, amongst others, reliability, maintenance support and availability.

The IEC recommend that a "formal design review should be conducted for all new products/processes, new applications, and revisions to existing products and manufacturing processes which affect the function, performance, safety, reliability, ability to inspect maintainability, availability, ability to cost, and other characteristics affecting the end product/process, users or bystanders".

A code walk-through consists of a walk-through team selecting a small set of paper test cases, representative sets of inputs and corresponding expected outputs for the program. The test data is then manually traced through the logic of the program.

References:

IEC 61160:1992, *Formal design review*.
Amendment 1 (1994).

Software Inspection. T. Gilb, D. Graham, Addison-Wesley, 1993, ISBN 0-201-63181-4.

C.5.17 Prototyping/animation

NOTE This technique/measure is referenced in tables B.3 and B.5 of IEC 61508-3.

Aim: To check the feasibility of implementing the system against the given constraints. To communicate the specifier’s interpretation of the system to the customer, in order to locate misunderstandings.

Description: A subset of system functions, constraints, and performance requirements are selected. A prototype is built using high-level tools. At this stage, constraints such as the target computer, implementation language, program size, maintainability, reliability and availability need not be considered. The prototype is evaluated against the customer’s criteria and the system requirements may be modified in the light of this evaluation.

References:

The emergence of rapid prototyping as a real-time software development tool. J. E. Cooling, T. S. Hughes, Proc. 2nd Int. Conf. on Software Engineering for Real-time Systems, Cirencester, UK, IEE, 1989.

Software evolution through rapid prototyping. Luqi, IEEE Computer 22 (5), 13-27, May 1989.

Approaches to Prototyping. R. Budde et al, Springer Verlag, 1984, ISBN 3-540-13490-5.

Proc. Working Conference on Prototyping. Namur, October 1983, Budde et al, Springer Verlag, 1984.

Using an executable specification language for an information system. S. Urban et al. IEEE Trans Software Engineering, Vol. SE-11 n° 7, July 1985.

C.5.18 Simulation du procédé

NOTE Cette technique/mesure est référencée dans le tableau B.3 de la CEI 61508-3.

But: Tester la fonction d'un système logiciel et son interface avec le monde extérieur, sans lui permettre de modifier, d'aucune sorte, le monde réel.

Description: Création d'un système permettant uniquement la réalisation de tests et simulant le comportement de l'équipement commandé (EUC).

La simulation ne peut être que de type logiciel ou une combinaison d'éléments matériels et logiciels. Il faut

- qu'elle fournisse des entrées équivalentes à celles utilisées après l'installation réelle de l'équipement commandé;
- qu'elle réponde aux sorties en provenance du logiciel testé d'une manière représentant fidèlement l'unité commandée;
- qu'elle prévoie des entrées opérateur pour toutes les perturbations auxquelles le système testé doit être confronté.

Lors du test du logiciel, la simulation peut être celle du matériel cible avec ses entrées et ses sorties.

Référence: Software Testing via Environmental Simulation (CONTESSE Report). Available until December 1998 from: Ray Browne, CIID, DTI, 151 Buckingham Palace Road, London, SW1W 9SS, UK, 1994.

C.5.19 Prescriptions relatives au fonctionnement

NOTE Cette technique/mesure est référencée dans le tableau B.6 de la CEI 61508-3.

But: Etablir les prescriptions relatives au fonctionnement d'un système logiciel qui soient démontrables.

Description: Une analyse des spécifications concernant le système et les prescriptions applicables au logiciel est effectuée afin de spécifier toutes les prescriptions relatives au fonctionnement générales, spécifiques, explicites et implicites.

Chaque prescription relative au fonctionnement est examinée à tour de rôle afin de déterminer

- le critère de réussite à obtenir;
- si une mesure par rapport au critère de réussite peut être obtenue;
- la précision potentielle de ces mesures;
- les stades du projet auxquels les mesures peuvent être estimées; et
- les stades du projet auxquels les mesures peuvent être effectuées.

Software evolution through rapid prototyping. Luqi, IEEE Computer 22 (5), 13-27, May 1989.

Approaches to Prototyping. R. Budde et al, Springer Verlag, 1984, ISBN 3-540-13490-5.

Proc. Working Conference on Prototyping. Namur, October 1983, Budde et al, Springer Verlag, 1984.

Using an executable specification language for an information system. S. Urban et al. IEEE Trans Software Engineering, Vol. SE-11 No. 7, July 1985.

C.5.18 Process simulation

NOTE This technique/measure is referenced in table B.3 of IEC 61508-3.

Aim: To test the function of a software system, together with its interface to the outside world, without allowing it to modify the real world in any way.

Description: The creation of a system, for testing purposes only, which mimics the behaviour of the equipment under control (EUC).

The simulation may be software only or a combination of software and hardware. It must

- provide inputs, equivalent to the inputs which will exist when the EUC is actually installed;
- respond to outputs from the software being tested in a way which faithfully represents the controlled plant;
- have provision for operator inputs to provide any perturbations with which the system under test is required to cope.

When software is being tested the simulation may be a simulation of the target hardware with its inputs and outputs.

Reference: Software Testing via Environmental Simulation (CONTESSE Report). Available until December 1998 from: Ray Browne, CIID, DTI, 151 Buckingham Palace Road, London, SW1W 9SS, UK, 1994.

C.5.19 Performance requirements

NOTE This technique/measure is referenced in table B.6 of IEC 61508-3.

Aim: To establish demonstrable performance requirements of a software system.

Description: An analysis is performed of both the system and the software requirements specifications to specify all general and specific, explicit and implicit performance requirements.

Each performance requirement is examined in turn to determine

- the success criteria to be obtained;
- whether a measure against the success criteria can be obtained;
- the potential accuracy of such measurements;
- the project stages at which the measurements can be estimated; and
- the project stages at which the measurements can be made.

Le caractère pratique de chaque prescription relative au fonctionnement est alors analysé en vue d'obtenir une liste des prescriptions relatives au fonctionnement, des critères de réussite et des mesures potentielles. Les objectifs principaux sont les suivants:

– associer chaque prescription relative au fonctionnement avec au moins une mesure;
– dans la mesure du possible, sélectionner des mesures précises et efficaces susceptibles d'être utilisées aussitôt que possible au cours du développement;
– spécifier les prescriptions relatives au fonctionnement et critères de réussite essentiels et facultatifs; et
– dans la mesure du possible, il convient de profiter de la possibilité d'utiliser une seule mesure pour plusieurs prescriptions relatives au fonctionnement.

C.5.20 Modélisation du fonctionnement

NOTE Cette technique/mesure est référencée dans les tableaux B.2 et B.5 de la CEI 61508-3.

But: S'assurer que la capacité de travail du système est suffisante pour satisfaire aux prescriptions spécifiées.

Description: La spécification des prescriptions comprend les prescriptions de capacité de traitement et de réponse applicables aux fonctions spécifiques qui peuvent être combinées aux contraintes d'utilisation des ressources totales du système. La conception de système proposée est comparée aux prescriptions énoncées en

– produisant un modèle des processus du système et de leurs interactions;
– déterminant l'utilisation des ressources par chaque processus comme, par exemple, le temps d'exécution, la bande passante des communications, les mémoires, etc.;
– déterminant la distribution des demandes envoyées au système dans des conditions moyennes puis dans les cas les plus défavorables;
– calculant la capacité de traitement et les temps de réponse dans des conditions moyennes puis dans les cas les plus défavorables pour chaque fonction individuelle du système.

Pour les systèmes simples, une solution analytique peut être suffisante, alors que pour les systèmes plus complexes, une certaine forme de simulation serait plus appropriée en vue d'obtenir des résultats précis.

Avant une modélisation détaillée, un contrôle plus simple portant sur le «budget des ressources» peut être effectué en additionnant les ressources requises pour tous les processus. Si les prescriptions dépassent la capacité du système prévue, la conception est impossible. Même si la conception satisfait à ce contrôle, la modélisation du fonctionnement peut faire apparaître que des délais et temps de réponse trop longs sont obtenus à cause de l'insuffisance des ressources. Pour éviter cette situation, les ingénieurs conçoivent souvent des systèmes n'utilisant qu'une partie des ressources totales (par exemple 50 %) afin de réduire les risques liés à l'insuffisance des ressources.

Référence: The Design of Real-time Systems: From Specification to Implementation and Verification. H. Kopetz et al, Software Engineering Journal 72-82, 1991.

C.5.21 Tests d'avalanche/de stress

NOTE Cette technique/mesure est référencée dans le tableau B.6 de la CEI 61508-3.

But: Appliquer à l'objet testé une charge de travail exceptionnellement élevée en vue de démontrer que l'objet testé supportera facilement les charges de travail normales.

The practicability of each performance requirement is then analysed in order to obtain a list of performance requirements, success criteria and potential measurements. The main objectives are:

– each performance requirement is associated with at least one measurement;
– where possible, accurate and efficient measurements are selected which can be used as early in the development as possible;
– essential and optional performance requirements and success criteria are specified; and
– where possible, advantage should be taken of the possibility of using a single measurement for more than one performance requirement.

C.5.20 Performance modelling

NOTE This technique/measure is referenced in tables B.2 and B.5 of IEC 61508-3.

Aim: To ensure that the working capacity of the system is sufficient to meet the specified requirements.

Description: The requirements specification includes throughput and response requirements for specific functions, perhaps combined with constraints on the use of total system resources. The proposed system design is compared against the stated requirements by

– producing a model of the system processes, and their interactions;
– determining the use of resources by each process, for example, processor time, communications bandwidth, storage devices, etc;
– determining the distribution of demands placed upon the system under average and worst-case conditions;
– computing the mean and worst-case throughput and response times for the individual system functions.

For simple systems an analytic solution may be sufficient, while for more complex systems some form of simulation may be more appropriate to obtain accurate results.

Before detailed modelling, a simpler "resource budget" check can be used which sums the resources requirements of all the processes. If the requirements exceed designed system capacity, the design is infeasible. Even if the design passes this check, performance modelling may show that excessive delays and response times occur due to resource starvation. To avoid this situation, engineers often design systems to use some fraction (for example 50 %) of the total resources so that the probability of resource starvation is reduced.

Reference: The Design of Real-time Systems: From Specification to Implementation and Verification. H. Kopetz et al, Software Engineering Journal 72-82, 1991.

C.5.21 Avalanche/stress testing

NOTE This technique/measure is referenced in table B.6 of IEC 61508-3.

Aim: To burden the test object with an exceptionally high workload in order to show that the test object would stand normal workloads easily.

Description: Un grand nombre de conditions de test peuvent être appliquées pour les tests d'avalanche/de stress. Certaines de ces conditions sont les suivantes:

– en cas de fonctionnement en mode scrutation, un nombre beaucoup plus important de changements par unité de temps se produisent à l'entrée de l'objet testé que dans des conditions normales;

– en cas de fonctionnement à la demande, le nombre de demandes envoyées à l'objet testé par unité de temps dépasse celui prévu pour des conditions normales;

– si la taille d'une base de données joue un rôle important, cette taille est alors supérieure à celle prévue pour des conditions normales;

– les dispositifs influents sont respectivement réglés de manière à fonctionner à leur vitesse maximale ou minimale;

– pour les cas extrêmes, tous les facteurs influents sont, dans la mesure du possible, simultanément placés dans des conditions limites.

Dans ces conditions de test, le comportement dans le temps de l'objet testé peut être évalué. L'effet des changements de charge peut être observé. La dimension correcte des tampons internes, variables dynamiques, piles, etc. peut être vérifiée.

C.5.22 Temps de réponse et contraintes mémoire

NOTE Cette technique/mesure est référencée dans le tableau B.6 de la CEI 61508-3.

But: S'assurer que le système satisfera aux prescriptions temporelles et à celles concernant la mémoire.

Description: La spécification des prescriptions applicables au système et au logiciel comprend les prescriptions de mémoire et de réponse applicables aux fonctions spécifiques qui peuvent être combinées aux contraintes d'utilisation des ressources totales du système.

Une analyse est effectuée pour déterminer les demandes de répartition dans des conditions moyennes et dans les cas les plus défavorables. Cette analyse nécessite une estimation de l'utilisation des ressources et le temps écoulé pour chaque fonction du système. Ces estimations peuvent être obtenues de plusieurs manières comme, par exemple, par comparaison avec un système existant ou par prototypage et évaluation des performances des systèmes à durée critique.

C.5.23 Analyse d'impact

NOTE Cette technique/mesure est référencée dans le tableau A.8 de la CEI 61508-3.

But: Déterminer les effets d'une modification ou d'une amélioration apportée à un système logiciel sur les autres modules logiciels de ce système ainsi que sur les autres systèmes.

Description: Avant d'apporter une modification ou une amélioration au logiciel, il convient qu'une analyse soit effectuée pour déterminer l'impact de la modification ou de l'amélioration sur le logiciel afin de déterminer quels systèmes logiciels et modules logiciels sont concernés.

Après l'analyse, une décision doit être prise en ce qui concerne la revérification du système logiciel. Cette décision dépend du nombre de modules logiciels concernés, de la criticité des modules logiciels concernés et de la nature de la modification. Des décisions possibles sont les suivantes:

– revérification du module logiciel modifié seulement;

– revérification de tous les modules logiciels concernés identifiés; ou

– revérification du système complet.

Référence: Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989. ISBN 1-85166-381-9.

Description: There are a variety of test conditions which can be applied for avalanche/stress testing. Some of these test conditions are:

– if working in a polling mode then the test object gets much more input changes per time unit as under normal conditions;

– if working on demands then the number of demands per time unit to the test object is increased beyond normal conditions;

– if the size of a database plays an important role then it is increased beyond normal conditions;

– influential devices are tuned to their maximum speed or lowest speed respectively;

– for the extreme cases, all influential factors, as far as is possible, are put to the boundary conditions at the same time.

Under these test conditions the time behaviour of the test object can be evaluated. The influence of load changes can be observed. The correct dimension of internal buffers or dynamic variables, stacks, etc. can be checked.

C.5.22 Response timing and memory constraints

NOTE This technique/measure is referenced in table B.6 of IEC 61508-3.

Aim: To ensure that the system will meet its temporal and memory requirements.

Description: The requirements specification for the system and the software includes memory and response requirements for specific functions, perhaps combined with constraints on the use of total system resources.

An analysis is performed to determine the distribution demands under average and worst-case conditions. This analysis requires estimates of the resource usage and elapsed time of each system function. These estimates can be obtained in several ways, for example comparison with an existing system or the prototyping and benchmarking of time critical systems.

C.5.23 Impact analysis

NOTE This technique/measure is referenced in table A.8 of IEC 61508-3.

Aim: To determine the effect that a change or an enhancement to a software system will have to other software modules in that software system as well as to other systems.

Description: Prior to a modification or enhancement being performed on the software, an analysis should be undertaken to determine the impact of the modification or enhancement on the software, and to also determine which software systems and software modules are affected.

After the analysis has been completed a decision is required concerning the reverification of the software system. This depends on the number of software modules affected, the criticality of the affected software modules and the nature of the change. Possible decisions are:

– only the changed software module is reverified;

– all affected software modules are reverified; or

– the complete system is reverified.

Reference: Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989. ISBN 1-85166-381-9.

C.5.24 Gestion de configuration logicielle

NOTE Cette technique/mesure est référencée dans le tableau A.8 de la CEI 61508-3.

But: Le but de la gestion de configuration logicielle est d'assurer la cohérence des groupes de fournitures de développement au fur et à mesure de la modification de ces fournitures. La gestion de configuration s'applique en général au développement du matériel et du logiciel.

Description: La gestion de configuration logicielle est une technique qui est utilisée tout au long du développement. Elle consiste essentiellement à documenter la production de chaque version de chaque fourniture «significative» et chaque relation entre les différentes versions des différentes fournitures. La documentation résultante permet au développeur de déterminer les effets de la modification d'une fourniture (spécialement d'un de ses composants) sur les autres fournitures. En particulier, les systèmes ou sous-systèmes peuvent être reconstruits de manière sûre à partir d'ensembles cohérents des versions de composants.

Références:

Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs. MIL-STD-483.

Software Configuration Management. J. K. Buckle. Macmillan Press, 1982.

Software Configuration Management. W. A. Babich. Addison-Wesley, 1986.

Configuration Management Requirements for Defence Equipment. UK Ministry of Defence Standard 05-57 Issue 3, July 1993.

C.6 Evaluation de la sécurité fonctionnelle

NOTE Les techniques et mesures concernées sont également décrites en B.6.

C.6.1 Tables de décision (tables de vérité)

NOTE Cette technique/mesure est référencée dans les tableaux A.10 et B.7 de la CEI 61508-3.

But: Fournir une spécification et une analyse claires et cohérentes des combinaisons logiques complexes et de leurs relations.

Description: Cette méthode utilise des tableaux à deux dimensions pour décrire de manière concise les relations entre les variables de programme booléennes.

La concision et la nature tabulaire de la méthode rend celle-ci appropriée en tant que moyen d'analyse des combinaisons logiques complexes exprimées sous forme de code.

La méthode est potentiellement exécutable lorsqu'elle est utilisée en tant que spécification.

C.6.2 Etude de danger et d'opérabilité (HAZOP)

But: Déterminer les dangers mettant en cause la sécurité d'un système proposé ou existant, les causes possibles, les conséquences et les actions recommandées pour minimiser leurs chances d'apparition.

C.5.24 Software configuration management

NOTE This technique/measure is referenced in table A.8 of IEC 61508-3.

Aim: Software configuration management aims to ensure the consistency of groups of development deliverables as those deliverables change. Configuration management in general applies to both hardware and software development.

Description: Software configuration management is a technique used throughout development. In essence, it requires documenting the production of every version of every significant deliverable and of every relationship between different versions of the different deliverables. The resulting documentation allows the developer to determine the effect on other deliverables of a change to one deliverable (especially one of its components). In particular, systems or subsystems can be reliably re-built from consistent sets of component versions.

References:

Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs. MIL-STD-483.

Software Configuration Management. J. K. Buckle. Macmillan Press, 1982.

Software Configuration Management. W. A. Babich. Addison-Wesley, 1986.

Configuration Management Requirements for Defence Equipment. UK Ministry of Defence Standard 05-57 Issue 3, July 1993.

C.6 Functional safety assessment

NOTE Relevant techniques and measures may also be found in B.6

C.6.1 Decision tables (truth tables)

NOTE This technique/measure is referenced in tables A.10 and B.7 of IEC 61508-3.

Aim: To provide a clear and coherent specification and analysis of complex logical combinations and relationships.

Description: This method uses two dimensional tables to concisely describe logical relationships between Boolean program variables.

The conciseness and tabular nature of the method makes it appropriate as a means of analysing complex logical combinations expressed in code.

The method is potentially executable if used as a specification.

C.6.2 Hazard and Operability Study (HAZOP)

Aim: To determine safety hazards in a proposed or existing system, their possible causes and consequences, and recommend action to minimise the chance of their occurrence.

Description: Une équipe d'ingénieurs dont l'expérience couvre l'ensemble du système concerné effectue un examen structuré d'une conception, au cours d'une série de réunions programmées. Ces ingénieurs examinent les aspects fonctionnels de la conception ainsi que la manière dont le système sera exploité en pratique (y compris les interventions humaines et la maintenance). Le chef d'équipe encourage la créativité des membres de l'équipe en ce qui concerne la dénonciation des dangers potentiels et conduit la procédure en présentant chaque partie du système en fonction de plusieurs mots clés: «aucun», «plus de», «moins de», «partie de», «plus que» (ou «aussi bien que») et «autre que». Chaque type de condition ou mode de défaillance est considéré en fonction de sa possibilité, de la manière dont il peut se produire, des conséquences possibles (y-a-t-il un danger?), de la manière dont il peut être évité et de la dépense à engager ou non pour l'éviter.

A un stade ultérieur, il est souvent nécessaire d'effectuer une analyse de danger plus poussée (souvent appelée «évaluation probabiliste ou quantitative de risque») en vue d'étudier les dangers majeurs plus en détail.

Les études de dangers peuvent être réalisées à plusieurs stades du développement d'un projet mais sont plus efficaces lorsqu'elles sont réalisées suffisamment tôt afin de pouvoir être prises en compte pour les décisions majeures concernant la conception et l'opérabilité. Ces études sont utiles si un calendrier fixe de réunions est prévu dans le cadre du projet, si chaque réunion est prévue pour une durée d'au moins une demi-journée et si un maximum de quatre réunions par semaine est prévu en vue de permettre le suivi de la documentation d'accompagnement. La documentation de ces réunions constituera une partie substantielle du dossier concernant les dangers et la sécurité du système.

La technique HAZOP s'est développée dans l'industrie manufacturière et elle est difficile à appliquer sans modification aux éléments logiciels des PES (systèmes électroniques programmables). Différentes méthodes dérivées applicables aux techniques HAZOP pour les PES (ou «CHAZOP»: techniques HAZOP pour systèmes informatiques) ont été proposées. Ces méthodes utilisent de nouveaux mots clés et/ou suggèrent des solutions pour la couverture systématique de l'architecture du système et du logiciel.

Références:

Draft Interim Defence Standard 00-58/1: "A Guide to HAZOP Studies on Systems which Incorporate a Programmable Electronic System". Ministry of Defence (UK). March 1995.

Hazard and Operability (HAZOP) studies applied to computer-controlled process plants. P. Chung and E. Broomfield. In "Computer Control and Human Error" by T. Kletz, Institution of Chemical Engineers, 165-189 Railway Terrace, Rugby, CV1 3HQ, UK, 1995, ISBN 0-85295-362-3.

Reliability and Hazard Criteria for Programmable Electronic Systems in the Chemical Industry. E. Johnson. Proc. of Safety and Reliability of PES, PES 3 Safety Symposium, B. K. Daniels (ed.), 28-30 May 1986, Guernsey Channel Islands, Elsevier Applied Science, 1986.

HAZOP and HAZAN. T. A. Kletz. Institution of Chemical Engineers, 165-189 Railway Terrace, Rugby, CV1 3HQ, UK, 3rd Edition, 1992, ISBN 0-85295-285-6.

A Guide to HAZOPS. Chemical Industries Association Ltd, 1977.

Reliability Engineering and Risk Assessment. E. J. Henlty and H. Kumamoto, Prentice-Hall, 1981.

Systems Reliability and Risk Analysis (Engineering Application of Systems Reliability and Risk Analysis), E. G. Frenkel, Kluwer Academic Pub., May 1988, ISBN 90-2473-665X.

Control Hazard Studies for Process Plants. K. Walters, in Integrated Risk Assessment – Current Practice and New Directions, edited by R. E. Melchers and M. G. Stewart, The University of Newcastle, NSW Australia. A. A. Balkema Publishers, Rotterdam Netherlands 1995, ISBN 90-5410-5550.

Description: A team of engineers, with expertise covering the whole system under consideration, participate in a structured examination of a design, through a series of scheduled meetings. They consider both the functional aspects of the design and how the system would operate in practice (including human activity and maintenance). A leader encourages team members to be creative in exposing potential hazards, and drives the procedure by presenting each part of the system in connection with several guide words: "none", "more of", "less of", "part of", "more than" (or "as well as") and "other than". Every applied condition or failure mode is considered for its feasibility, how it could arise, the possible consequences (is there a hazard?), how it could be avoided and if the avoidance technique is worth the expense.

At a later time, it is often necessary to carry out further hazard analysis (often referred to as probabilistic or quantitative risk assessment), to consider the major hazards in more detail.

Hazard studies may take place at many stages of project development, but are most effective when performed early enough to influence major design and operability decisions. It is helpful if a fixed time schedule is allocated within the project for the meetings; each one is scheduled for at least half a day; and no more than four per week are scheduled, so that the flow of accompanying documentation is maintained. Documentation from the meetings will form a substantial part of the system hazard/safety dossier.

The HAZOP technique evolved in the process industry and is difficult to apply without modification to the software element of PES. Different derivative methods for PES HAZOPs (or Computer HAZOPs – "CHAZOPs") have been proposed which in general introduce new guide words and/or suggest schemes for systematically covering the system and software architecture.

References:

Draft Interim Defence Standard 00-58/1: "A Guide to HAZOP Studies on Systems which Incorporate a Programmable Electronic System". Ministry of Defence (UK). March 1995.

Hazard and Operability (HAZOP) studies applied to computer-controlled process plants. P. Chung and E. Broomfield. In "Computer Control and Human Error" by T. Kletz, Institution of Chemical Engineers, 165-189 Railway Terrace, Rugby, CV1 3HQ, UK, 1995, ISBN 0-85295-362-3.

Reliability and Hazard Criteria for Programmable Electronic Systems in the Chemical Industry. E. Johnson. Proc. of Safety and Reliability of PES, PES 3 Safety Symposium, B. K. Daniels (ed.), 28-30 May 1986, Guernsey Channel Islands, Elsevier Applied Science, 1986.

HAZOP and HAZAN. T. A. Kletz. Institution of Chemical Engineers, 165-189 Railway Terrace, Rugby, CV1 3HQ, UK, 3rd Edition, 1992, ISBN 0-85295-285-6.

A Guide to HAZOPS. Chemical Industries Association Ltd, 1977.

Reliability Engineering and Risk Assessment. E. J. Henlty and H. Kumamoto, Prentice-Hall, 1981.

Systems Reliability and Risk Analysis (Engineering Application of Systems Reliability and Risk Analysis), E. G. Frenkel, Kluwer Academic Pub., May 1988, ISBN 90-2473-665X.

Control Hazard Studies for Process Plants. K. Walters, in Integrated Risk Assessment – Current Practice and New Directions, edited by R. E. Melchers and M. G. Stewart, The University of Newcastle, NSW Australia. A. A. Balkema Publishers, Rotterdam Netherlands 1995, ISBN 90-5410-5550.

C.6.3 Analyse des défaillances de cause commune

NOTE 1 Cette technique/mesure est référencée dans le tableau A.10 de la CEI 61508-3.

NOTE 2 Voir aussi l'annexe D de la CEI 61508-6.

But: Déterminer les défaillances potentielles des systèmes multiples ou sous-systèmes multiples susceptibles de réduire les avantages de la redondance par l'apparition simultanée des mêmes défaillances dans les multiples parties.

Description: Les systèmes chargés de la sécurité d'une installation utilisent la redondance en ce qui concerne les éléments matériels ainsi que le vote majoritaire. Cette technique est utilisée afin d'éviter les défaillances matérielles dans les composants ou les sous-systèmes qui tendraient à empêcher le traitement correct des données.

Toutefois, certaines défaillances peuvent être communes à plusieurs composants ou sous-systèmes. Par exemple, si un système est installé dans une seule pièce, les défauts du système de conditionnement d'air peuvent affecter les avantages de la redondance. La même chose se produit en ce qui concerne les autres effets extérieurs sur le système tels que le feu, l'inondation, les perturbations électromagnétiques, les accidents d'avions et les tremblements de terre. Le système peut être également affecté par des incidents liés à l'exploitation et à la maintenance. Il est donc essentiel que des procédures adéquates et bien documentées soient fournies pour l'exploitation et la maintenance et que le personnel d'exploitation et de maintenance soit parfaitement entraîné.

Les effets internes contribuent également de façon majeure à l'apparition des défaillances de cause commune. Ils peuvent être dus à des anomalies de conception des composants communs ou identiques et de leurs interfaces, ainsi qu'au vieillissement des composants. L'analyse des défaillances de cause commune à pour but la recherche des défaillances communes potentielles à l'intérieur du système. Les méthodes suivantes sont utilisées pour l'analyse des défaillances de cause commune: contrôle général de la qualité, revues de conception, vérification et test par une équipe indépendante et analyse des incidents réels avec retour d'expérience acquise sur systèmes similaires. Toutefois, l'analyse porte au-delà des éléments matériels. Même si une certaine diversité logicielle est utilisée pour les différentes voies d'un système redondant, il peut y avoir une certaine similitude dans les approches logicielles susceptibles d'occasionner une défaillance de cause commune comme, par exemple, les anomalies de la spécification commune.

Lorsque des défaillances de cause commune ne se produisent pas exactement en même temps, des précautions peuvent être prises en utilisant des méthodes de comparaison entre les voies multiples susceptibles de permettre la détection d'une défaillance avant que cette défaillance soit commune à toutes les voies. Il convient de prendre cette technique en compte dans l'analyse des défaillances de cause commune.

Références:

Review of Common Cause Failures. I. A. Watson, UKAEA, Centre for Systems Reliability, Wigshaw Lane, WA3 4NE, England, NCSR R 27, July 1981.

Common-Mode Failures in Redundancy Systems. I. A. Watson and G. T. Edwards. Nuclear Technology Vol. 46, December 1979.

Programmable Electronic Systems in Safety Related Applications. Health and Safety Executive, Her Majesty's Stationary Office, London, 1987.

C.6.4 Modèles de Markov

NOTE Voir en B.1 de la CEI 61508-6 une comparaison rapide de cette technique avec les diagrammes de blocs de fiabilité dans le contexte de l'analyse de l'intégrité de sécurité du matériel.

But: Evaluer la fiabilité, la sécurité ou la disponibilité d'un système.

C.6.3 Common cause failure analysis

NOTE 1 This technique/measure is referenced in table A.10 of IEC 61508-3.

NOTE 2 See also annex D of IEC 61508-6.

Aim: To determine potential failures in multiple systems or multiple subsystems which would undermine the benefits of redundancy, because of the appearance of the same failures in the multiple parts at the same time.

Description: Systems intended to take care of the safety of a plant often use redundancy in hardware and majority voting. This is to avoid random hardware failures in components or subsystems which would tend to prevent the correct processing of data.

However, some failures can be common to more than one component or subsystem. For example, if a system is installed in one single room, shortcomings in the air-conditioning, might reduce the benefits of redundancy. The same is true for other external effects on the system such as fire, flooding, electromagnetic interference, plane crashes, and earthquakes. The system may also be affected by incidents related to operation and maintenance. It is essential, therefore, that adequate and well- documented procedures are provided for operation and maintenance, and operating and maintenance personnel are extensively trained.

Internal effects are also major contributors to common cause failures. They can stem from design faults in common or identical components and their interfaces, as well as ageing of components. Common cause failure analysis has to search the system for such potential common failures. Methods of common cause failure analysis are: general quality control; design reviews; verification and testing by an independent team; and analysis of real incidents with feedback of experience from similar systems. The scope of the analysis, however, goes beyond hardware. Even if software diversity is used in different channels of a redundant system, there might be some commonality in the software approaches which could give rise to common cause failure, for example, faults in the common specification.

When common cause failures do not occur exactly at the same time, precautions can be taken by means of comparison methods between the multiple channels which should lead to detection of a failure before this failure is common to all channels. Common cause failure analysis should take this technique into account.

References:

Review of Common Cause Failures. I. A. Watson, UKAEA, Centre for Systems Reliability, Wigshaw Lane, WA3 4NE, England, NCSR R 27, July 1981.

Common-Mode Failures in Redundancy Systems. I. A. Watson and G. T. Edwards. Nuclear Technology Vol. 46, December 1979.

Programmable Electronic Systems in Safety Related Applications. Health and Safety Executive, Her Majesty's Stationary Office, London, 1987.

C.6.4 Markov models

NOTE See B.1 of IEC 61508-6 for a brief comparison of this technique against reliability block diagrams, in the context of analysing hardware safety integrity.

Aim: To evaluate the reliability, safety or availability of a system.

Description: On trace un graphe du système représentant l'état du système par rapport à ses états de défaillance (les états de défaillances sont représentés par les nœuds du graphe). Les arêtes entre les nœuds, qui représentent les défaillances ou les réparations, sont pondérées par les taux de défaillance et de réparation correspondants. On suppose qu'un changement d'état *N* vers l'état suivant *N*+1 est indépendant de l'état précédent *N*-1. A noter que les défaillances, états et taux peuvent être détaillés de manière à obtenir une description précise du système comme, par exemple, les défaillances détectées ou non détectées, la manifestation d'une défaillance plus importante, etc.

La technique de Markov est bien adaptée à la modélisation de systèmes multiples dont le niveau de redondance varie en fonction du temps à cause des pannes des composants et des réparations. D'autre méthodes classiques comme, par exemple, les méthodes FMEA (analyse des modes de défaillances et de leurs effets) et FTA (analyse au moyen d'arbre de panne), ne peuvent être facilement adaptées à la modélisation des effets des défaillances pendant le cycle de vie complet du système étant donné qu'aucune formule combinatoire simple n'est disponible pour le calcul des probabilités correspondantes.

Dans les cas les plus simples, les formules qui décrivent les probabilités du système sont disponibles dans la documentation ou peuvent être calculées manuellement. Dans les cas plus complexes, il existe certaines méthodes de simplification (permettant de réduire le nombre d'états). Pour les cas très complexes, les résultats peuvent être calculés par simulation informatique du graphe.

Références:

CEI 61165:1995, *Application des techniques de Markov*.

The Theory of Stochastic Processes. R. E. Cox and H. D. Miller, Methuen and Co. Ltd., London, UK, 1963.

Finite MARKOV Chains. J. G. Kemeny and J. L. Snell. D. Van Nostrand Company Inc, Princeton, 1959.

Reliability Handbook. B. A. Koslov and L. A. Usnakov, Holt Rinehart and Winston Inc, New York, 1970.

The Theory and Practice of Reliable System Design. D. P. Siewiorek and R. S. Swarz, Digital Press, 1982.

C.6.5 Diagrammes de blocs de fiabilité

NOTE Cette technique/mesure est référencée dans le tableau A.10 de la CEI 61508-3 et est utilisée en annexe B de la CEI 61508-6.

But: Modéliser sous forme de diagramme l'ensemble des événements qui doivent se produire et les conditions qu'il faut remplir pour obtenir le fonctionnement correct d'un système ou d'une tâche.

Description: Le but de l'analyse est représenté par un chemin efficace constitué de cases, de lignes et de jonctions logiques. Le chemin efficace débute d'un côté du diagramme et continue à travers les cases et les jonctions vers l'autre côté du diagramme. Une case représente une condition ou un événement et le chemin peut traverser la case si la condition est vraie ou si l'événement s'est produit. Lorsque le chemin arrive à une jonction, il continue si la logique de la jonction est satisfaite. Lorsqu'il atteint un sommet, le chemin peut continuer le long de toutes les lignes sortantes. Si au moins un chemin efficace traverse le diagramme, l'analyse est satisfaisante.

Description: A graph of the system is constructed. The graph represents the status of the system with regard to its failure states (the failure states are represented by the nodes of the graph). The edges between nodes, which represent the failure events or repair events, are weighted with the corresponding failure rates or repair rates. It is assumed that a change of state, *N*, to a subsequent state, *N*+1, is independent of the previous state, *N*-1. Note that the failure events, states and rates can be detailed in such a way that a precise description of the system is obtained, for example detected or undetected failures, manifestation of a larger failure, etc.

The Markov technique is suitable for modelling multiple systems in which the level of redundancy varies with time due to component failure and repair. Other classical methods, for example, FMEA and FTA, cannot readily be adapted to modelling the effects of failures throughout the lifecycle of the system since no simple combinatorial formulae exist for calculating the corresponding probabilities.

In the simplest cases, the formulae which describe the probabilities of the system are readily available in the literature or can be calculated manually. In more complex cases, some methods of simplification (i.e. reducing the number of states) exist. For very complex cases, results can be calculated by computer simulation of the graph.

References:

IEC 61165:1995, *Application of Markov techniques*.

The Theory of Stochastic Processes. R. E. Cox and H. D. Miller, Methuen and Co. Ltd., London, UK, 1963.

Finite MARKOV Chains. J. G. Kemeny and J. L. Snell. D. Van Nostrand Company Inc, Princeton, 1959.

Reliability Handbook. B. A. Koslov and L. A. Usnakov, Holt Rinehart and Winston Inc, New York, 1970.

The Theory and Practice of Reliable System Design. D. P. Siewiorek and R. S. Swarz, Digital Press, 1982.

C.6.5 Reliability block diagrams

NOTE This technique/measure is referenced in table A.10 of IEC 61508-3 and is used in annex B of IEC 61508-6.

Aim: To model, in a diagrammatic form, the set of events that must take place and conditions which must be fulfilled for a successful operation of a system or a task.

Description: The target of the analysis is represented as a success path consisting of blocks, lines and logical junctions. A success path starts from one side of the diagram and continues via the blocks and junctions to the other side of the diagram. A block represents a condition or an event, and the path can pass it if the condition is true or the event has taken place. If the path comes to a junction, it continues if the logic of the junction is fulfilled. If it reaches a vertex, it may continue along all outgoing lines. If there exists at least one success path through the diagram, the target of the analysis is operating correctly.

Références:

CEI 61078:1991, *Techniques d'analyse de la sûreté de fonctionnement – Méthode du diagramme de fiabilité*.

System Reliability Engineering Methodology: A Division of the State of the Art. J. B. Fussel and J. S. Arend, Nuclear Safety 20 (5), 1979.

Fault Tree Handbook. W. E. Vesely et al, NUREG-0942, Division of System Safety Office at Nuclear Reactor Regulation, US Nuclear Regulatory Commission, Washington, DC 20555, 1981.

C.6.6 Simulation de Monte-Carlo

NOTE Cette technique/mesure est référencée dans le tableau B.4 de la CEI 61508-3.

But: Simuler en logiciel des phénomènes du monde réel à l'aide de nombres aléatoires.

Description: Les simulations de Monte-Carlo sont utilisées pour résoudre deux types de problèmes:

– les problèmes probabilistes, où des nombres aléatoires sont utilisés pour générer des phénomènes stochastiques; et
– les problèmes déterministes, qui sont traduits mathématiquement en un problème probabiliste équivalent.

La simulation de Monte-Carlo injecte des flux de nombres aléatoires pour simuler du bruit sur un signal d'analyse ou pour ajouter des tolérances ou erreurs systématiques aléatoires. La simulation de Monte-Carlo est utilisée afin de produire un échantillon important à partir duquel les résultats statistiques sont obtenus.

Lors de l'utilisation de simulations de Monte-Carlo, il faut prendre soin de s'assurer que les valeurs d'erreurs systématiques, de tolérances ou de bruit sont raisonnables.

Un principe général des simulations de Monte-Carlo est de réénoncer et reformuler le problème de manière que les résultats obtenus soient aussi précis que possible plutôt que d'aborder le problème tel qu'il était initialement énoncé.

Référence: Monte Carlo Methods. J. M. Hammersley, D. C. Handscomb, Chapman & Hall, 1979.

References:

IEC 61078:1991, *Analysis techniques for dependability – Reliability block diagram method*.

System Reliability Engineering Methodology: A Division of the State of the Art. J. B. Fussel and J. S. Arend, Nuclear Safety 20 (5), 1979.

Fault Tree Handbook. W. E. Vesely et al, NUREG-0942, Division of System Safety Office at Nuclear Reactor Regulation, US Nuclear Regulatory Commission, Washington, DC 20555, 1981.

C.6.6 Monte-Carlo simulation

NOTE This technique/measure is referenced in table B.4 of IEC 61508-3.

Aim: To simulate real world phenomena in software using random numbers.

Description: Monte-Carlo simulations are used to solve two classes of problems:

– probabilistic, where random numbers are used to generate stochastic phenomena; and

– deterministic, which are mathematically translated into an equivalent probabilistic problem.

Monte-Carlo simulation injects random number streams to simulate noise on an analysis signal or to add random biases or tolerances. The Monte-Carlo simulation is run to produce a large sample from which statistical results are obtained.

When using Monte-Carlo simulations care must be taken to ensure that the biases, tolerances or noise have reasonable values.

A general principle of Monte-Carlo simulations is to restate and reformulate the problem so that the results obtained are as accurate as possible rather than tackling the problem as initially stated.

Reference: Monte Carlo Methods. J. M. Hammersley, D. C. Handscomb, Chapman & Hall, 1979.

Annexe D
(informative)

Une approche probabiliste pour déterminer l'intégrité de sécurité logicielle pour un logiciel prédéveloppé

D.1 Généralités

La présente annexe fournit les lignes directrices initiales concernant l'utilisation d'une approche probabiliste en vue de déterminer l'intégrité de sécurité logicielle d'un logiciel prédéveloppé à partir de l'expérience en exploitation. Cette approche est considérée comme particulièrement adaptée en tant que partie de la qualification des systèmes d'exploitation, des éléments de bibliothèque, des compilateurs et autres logiciels système. Cette annexe fournit une indication sur ce qui est possible, mais il convient que les techniques ne soient utilisées que par ceux qui sont compétents en analyse statistique.

NOTE La présente annexe utilise le terme niveau de confiance, qui est décrit dans l'IEEE 352. Un terme équivalent, niveau de signification, est utilisé dans la CEI 61164.

Les techniques pourraient également être utilisées pour démontrer l'amélioration du niveau d'intégrité de sécurité logicielle dans le temps. Par exemple, on peut montrer qu'un logiciel réalisé selon les prescriptions de la CEI 61508-3 de manière à atteindre le niveau d'intégrité de sécurité 1 (SIL1) peut, après une certaine période d'utilisation satisfaisante dans un grand nombre d'applications, atteindre le niveau SIL2.

Le tableau D.1 ci-dessous montre le nombre de demandes sans défaillance traitées ou le nombre d'heures d'exploitation nécessaire pour obtenir la qualification correspondant à un certain niveau d'intégrité de sécurité. Ce tableau est un résumé des résultats donnés en D.2.1 et D.2.3.

L'expérience en exploitation peut être traitée de façon mathématique comme présenté en D.2 ci-dessous pour compléter ou remplacer le test statistique, et l'expérience en exploitation venant de plusieurs sites peut être combinée (c'est-à-dire par addition du nombre de demandes traitées ou d'heures d'exploitation), mais seulement si

- la version logicielle à utiliser dans le système E/EP/PE relatif à la sécurité est identique à la version ayant servi à acquérir l'expérience en exploitation;
- le profil opérationnel de l'espace d'entrée est similaire;
- il y a un système effectif pour rapporter et documenter les défaillances; et
- les conditions préalables significatives (voir D.2. ci-dessous) sont remplies.

Tableau D.1 – Historique nécessaire pour s'assurer des niveaux d'intégrité de sécurité

SIL	Mode de fonctionnement faible demande	Nombre de demandes traitées		Mode de fonctionnement continu ou forte demande	Nombre total d'heures de fonctionnement	
	(Probabilité de défaillance concernant la réalisation de la fonction prévue à la demande)	$1-\alpha = 0,99$	$1-\alpha = 0,95$	(Probabilité de défaillance dangereuse par heure)	$1-\alpha = 0,99$	$1-\alpha = 0,95$
4	$\geq 10^{-5}$ à $< 10^{-4}$	$4,6 \times 10^5$	3×10^5	$\geq 10^{-9}$ à $< 10^{-8}$	$4,6 \times 10^9$	3×10^9
3	$\geq 10^{-4}$ à $< 10^{-3}$	$4,6 \times 10^4$	3×10^4	$\geq 10^{-8}$ à $< 10^{-7}$	$4,6 \times 10^8$	3×10^8
2	$\geq 10^{-3}$ à $< 10^{-2}$	$4,6 \times 10^3$	3×10^3	$\geq 10^{-7}$ à $< 10^{-6}$	$4,6 \times 10^7$	3×10^7
1	$\geq 10^{-2}$ à $< 10^{-1}$	$4,6 \times 10^2$	3×10^2	$\geq 10^{-6}$ à $< 10^{-5}$	$4,6 \times 10^6$	3×10^6
NOTE 1 $1-\alpha$ représente le niveau de confiance. NOTE 2 Voir D.2.1 et D.2.3 pour les prérequis et les détails concernant l'obtention de ce tableau.						

Annex D
(informative)

A probabilistic approach to determining software safety integrity for pre-developed software

D.1 General

This annex provides initial guidelines on the use of a probabilistic approach to determining software safety integrity for pre-developed software based on operational experience. This approach is considered particularly appropriate as part of the qualification of operating systems, library components, compilers and other system software. The annex provides an indication of what is possible, but the techniques should be used only by those who are competent in statistical analysis.

NOTE This annex uses the term confidence level, which is described in IEEE 352. An equivalent term, significance level, is used in IEC 61164.

The techniques could also be used to demonstrate an increase in the safety integrity level of software over time. For example, software built to the requirements of IEC 61508-3 to SIL1 may, after a suitable period of successful operation in a large number of applications, be shown to achieve SIL2.

Table D.1 below shows the number of failure-free demands experienced or hours of failure-free operation needed to qualify for a particular safety integrity level. This table is a summary of the results given in D.2.1 and D.2.3.

Operating experience can be treated mathematically as outlined in D.2 below to supplement or replace statistical testing, and operating experience from several sites may be combined (i.e. by adding the number of treated demands or hours of operation), but only if

– the software version to be used in the E/E/PE safety-related system is identical to the version for which operating experience is being claimed;

– the operational profile of the input space is similar;

– there is an effective system for reporting and documenting failures; and

– the relevant prerequisites (see D.2 below) are satisfied.

Table D.1 – Necessary history for confidence to safety integrity levels

SIL	Low demand mode of operation	Number of treated demands		High demand or continuous mode of operation	Hours of operation in total	
	(Probability of failure to perform its design function on demand)	1-α = 0,99	1-α = 0,95	(Probability of a dangerous failure per hour)	1-α = 0,99	1-α = 0,95
4	$\geq 10^{-5}$ to $< 10^{-4}$	$4{,}6 \times 10^{5}$	3×10^{5}	$\geq 10^{-9}$ to $< 10^{-8}$	$4{,}6 \times 10^{9}$	3×10^{9}
3	$\geq 10^{-4}$ to $< 10^{-3}$	$4{,}6 \times 10^{4}$	3×10^{4}	$\geq 10^{-8}$ to $< 10^{-7}$	$4{,}6 \times 10^{8}$	3×10^{8}
2	$\geq 10^{-3}$ to $< 10^{-2}$	$4{,}6 \times 10^{3}$	3×10^{3}	$\geq 10^{-7}$ to $< 10^{-6}$	$4{,}6 \times 10^{7}$	3×10^{7}
1	$\geq 10^{-2}$ to $< 10^{-1}$	$4{,}6 \times 10^{2}$	3×10^{2}	$\geq 10^{-6}$ to $< 10^{-5}$	$4{,}6 \times 10^{6}$	3×10^{6}
NOTE 1 1-α represents the confidence level.						
NOTE 2 See D.2.1 and D.2.3 for prerequisites and details of how this table is derived.						

D.2 Formules de tests statistiques et exemples d'utilisation

D.2.1 Test statistique simple en mode de fonctionnement faible demande

D.2.1.1 Conditions préalables

a) La distribution des données de test est égale à la distribution des demandes pendant l'exploitation en ligne.

b) Les passages de test sont statistiquement indépendants les uns des autres, par rapport à la cause de la défaillance.

c) Existence d'un mécanisme permettant la détection des défaillances susceptibles de se produire.

d) Nombre de cas de test $n > 100$.

e) Aucune défaillance pendant les n cas de test.

D.2.1.2 Résultats

La probabilité de défaillance p (par demande), au niveau de confiance 1-α, est donnée par

$$p \leq 1 - \sqrt[n]{\alpha} \qquad \text{ou} \qquad n \geq -\frac{\ln \alpha}{p}$$

D.2.1.3 Exemple

Tableau D.2 – Probabilités de défaillance en mode de fonctionnement faible demande

1-α	***p***
0,95	3/n
0,99	4,6/n

Pour une probabilité de défaillance sur demande de niveau SIL3 avec un niveau de confiance de 95 %, l'application de la formule donne 30 000 cas de test en tenant compte des conditions préalables. Le tableau D.1 résume les résultats pour chaque niveau d'intégrité de sécurité.

D.2.2 Test d'un espace (domaine) d'entrée pour un mode de fonctionnement faible demande

D.2.2.1 Conditions préalables

La seule condition préalable est que les données de test soient sélectionnées pour donner une distribution aléatoire uniforme sur tout l'espace (domaine) d'entrée.

D.2.2.2 Résultats

L'objectif est de trouver le nombre de tests n qui sont nécessaires, en se fondant sur le seuil de précision δ des entrées pour la fonction de faible demande à tester (par exemple un arrêt d'urgence de sécurité).

D.2 Statistical testing formulae and examples of their use

D.2.1 Simple statistical test for low demand mode of operation

D.2.1.1 Prerequisites

a) Test data distribution equal to distribution for demands during on-line operation.

b) Test runs are statistically independent from each other, with respect to the cause of a failure.

c) An adequate mechanism exists to detect any failures which may occur.

d) Number of test cases $n > 100$.

e) No failure occurs during the n test cases.

D.2.1.2 Results

Failure probability p (per demand), at the confidence level $1-\alpha$, is given by

$$p \leq 1 - \sqrt[n]{\alpha} \qquad \text{or} \qquad n \geq -\frac{\ln \alpha}{p}$$

D.2.1.3 Example

Table D.2 – Probabilities of failure for low demand mode of operation

$1-\alpha$	p
0,95	$3/n$
0,99	$4{,}6/n$

For a probability of failure on demand of SIL3 at 95 % confidence the application of the formula gives 30 000 test cases under the conditions of the prerequisites. Table D.1 summarises the results for each safety integrity level.

D.2.2 Testing of an input space (domain) for a low demand mode of operation

D.2.2.1 Prerequisites

The only prerequisite is that the test data is selected to give a random uniform distribution over the input space (domain).

D.2.2.2 Results

The objective is to find the number of tests, n, that are necessary based on the threshold of accuracy, δ, of the inputs for the low demand function (such as a safety shut-down) that is being tested.

Tableau D.3 – Distances moyennes de deux points de test

Dimension du domaine	Distance moyenne de deux points de test en direction d'un axe arbitraire
1	$\delta = 1/n$
2	$\delta = \sqrt[2]{1/n}$
3	$\delta = \sqrt[3]{1/n}$
k	$\delta = \sqrt[k]{1/n}$
NOTE *k* peut être n'importe quel entier positif. Les valeurs 1,2 et 3 sont juste des exemples.	

D.2.2.3 Exemple

Considérons un arrêt d'urgence de sécurité qui est dépendant de deux variables seulement, A et B. S'il est à vérifier que les seuils qui divisent la paire d'entrée des variables A et B sont traités correctement jusqu'à une précision de 1 % de la plage de mesure de A ou de B, le nombre de cas de test uniformément distribués requis dans l'espace de A et B est

$$n = 1/\delta^2 = 10^4$$

D.2.3 Test statistique simple en mode de fonctionnement continu ou forte demande

D.2.3.1 Conditions préalables

a) La distribution des données de test est égale à la distribution pendant l'exploitation en ligne.

b) La réduction relative de la probabilité d'absence de défaillance est proportionnelle à la longueur de l'intervalle de temps considéré ou est constante autrement.

c) Un mécanisme adéquat existe pour détecter toute défaillance qui peut se produire.

d) Le test dure pendant un temps *t*.

e) Aucune défaillance ne se produit pendant le temps *t*.

D.2.3.2 Résultats

La relation entre la probabilité de défaillance λ, le niveau de confiance 1-α et le temps de test *t* est

$$\lambda = -\frac{\ln \alpha}{t}$$

La probabilité de défaillance est indirectement proportionnelle au temps de fonctionnement moyen entre défaillances (MTBF):

$$\lambda = \frac{1}{MTBF}$$

NOTE La présente norme ne fait pas de distinction entre la probabilité de défaillances par heure, et le taux de défaillance en 1 h. Strictement, la probabilité de défaillance *F*, est liée au taux de défaillance *f*, par l'équation $F = 1\text{-}e^{-ft}$, mais le domaine d'application de la présente norme concerne les taux de défaillances de moins de 10^{-5} et pour ces valeurs $F \approx ft$.

Table D.3 – Mean distances of two test points

Dimension of the domain	Mean distance of two test points in direction of an arbitrary axis
1	$\delta = 1/n$
2	$\delta = \sqrt[2]{1/n}$
3	$\delta = \sqrt[3]{1/n}$
k	$\delta = \sqrt[k]{1/n}$
NOTE *k* can be any positive integer. The values 1, 2 and 3 are just examples.	

D.2.2.3 Example

Consider a safety shut-down that is dependent on just two variables, A and B. If it has been verified that the thresholds that partition the input pair of variables A and B are treated correctly to an accuracy of 1 % of A or B's measuring range, the number of uniformly distributed test cases required in the space of A and B is

$$n = 1/\delta^2 = 10^4$$

D.2.3 Simple statistical test for high demand or continuous mode of operation

D.2.3.1 Prerequisites

a) Test data distribution equal to distribution during on-line operation.

b) The relative reduction for the probability of no failure is proportional to the length of the considered time interval and constant otherwise.

c) An adequate mechanism exists to detect any failures which may occur.

d) The test extends over a test time *t*.

e) No failure occurs during *t*.

D.2.3.2 Results

The relationship between the probability of failure λ, the confidence level 1-α and the testing time *t* is

$$\lambda = -\frac{\ln \alpha}{t}$$

The probability of failure is indirectly proportional to the mean operating time between failures:

$$\lambda = \frac{1}{MTBF}$$

NOTE This standard does not distinguish between the probability of failure per hour and the rate of failures in 1 h. Strictly, the probability of failure, *F*, is related to the failure rate, *f*, by the equation $F = 1\text{-}e^{-ft}$, but the scope of this standard is for failure rates of less than 10^{-5}, and for values this small $F \approx ft$.

D.2.3.3 Exemple

Tableau D.4 – Probabilité de défaillance en mode de fonctionnement forte demande ou continu

$1-\alpha$	λ
0,95	$3/t$
0,99	$4,6/t$

Pour vérifier que le temps moyen entre défaillances est au moins 10^8 h avec un niveau de confiance de 95 %, un temps de test de 3×10^8 h est requis et les conditions préalables doivent être remplies. Le tableau D.1 résume le nombre de tests requis pour chaque niveau d'intégrité de sécurité.

D.2.4 Test complet

Le programme est considéré comme une urne contenant une nombre connu *N* de boules. Chaque boule représente une propriété du programme digne d'intérêt. Les boules sont tirées au hasard et replacées après inspection. Le test complet est effectué lorsque toutes les boules sont tirées.

D.2.4.1 Conditions préalables

a) La distribution des données de test est telle que chacun des *N* propriétés du programme soit testée avec une probabilité égale.

b) Les exécutions de test sont indépendantes les unes des autres.

c) Chaque défaillance qui arrive est détectée.

d) Le nombre de cas de test $n >> N$.

e) Aucune défaillance ne se produit pendant les *n* cas de test.

f) Chaque exécution de test concerne une propriété du programme (une propriété du programme est ce qui peut être testé au cours de l'exécution d'un test).

D.2.4.2 Résultats

La probabilité *p* de tester toutes les propriétés du programme est donnée par

$$p = \sum_{j=0}^{N-1} (-1)^j \binom{N}{j} \left(\frac{N-j}{N}\right)^n \quad \text{ou} \quad p = 1 + \sum_{j=1}^{N} (-1)^j \, C_{j,N} \left(\frac{N-j}{N}\right)^n$$

où

$$C_{j,N} = \frac{(N)(N-1)\ldots(N-j+1)}{j!}$$

En ce qui concerne l'évaluation de cette formule, en général seuls les premiers termes sont normalement importants puisque les cas réalistes sont caractérisés par $n >> N$. Le dernier facteur rend très petits les termes correspondant aux grands *j*. Cela est également visible sur le tableau D.5.

D.2.3.3 Example

Table D.4 – Probabilities of failure for high demand or continuous mode of operation

$1-\alpha$	λ
0,95	$3/t$
0,99	$4{,}6/t$

To verify that the mean time between failures is at least 10^8 h with a confidence level of 95 %, a test time of 3×10^8 h is required and the prerequisites must be satisfied. Table D.1 summarises the number of tests required for each safety integrity level.

D.2.4 Complete test

The program is considered as an urn containing a known number *N* of balls. Each ball represents a program property of interest. Balls are drawn at random and replaced after inspection. A complete test is achieved if all the balls are drawn.

D.2.4.1 Prerequisites

a) Test data distribution is such that each of the *N* program properties is tested with equal probability.

b) Test runs are independent from each other.

c) Each occurring failure is detected.

d) Number of test cases $n >> N$.

e) No failure occurs during the *n* test cases.

f) Each test run tests one program property (a program property is what can be tested during one run).

D.2.4.2 Results

The probability *p* to test all program properties is given by

$$p = \sum_{j=0}^{N-1} (-1)^j \binom{N}{j} \left(\frac{N-j}{N}\right)^n \quad \text{or} \quad p = 1 + \sum_{j=1}^{N} (-1)^j \, C_{j,N} \left(\frac{N-j}{N}\right)^n$$

where

$$C_{j,N} = \frac{N(N-1)\ldots(N-j+1)}{j!}$$

For evaluation of this formula usually only the first terms matter since realistic cases are characterised by $n >> N$. The last factor makes all terms for large *j* very small. This is also visible in table D.5.

D.2.4.3 Exemple

Considérons un programme qui a été utilisé dans plusieurs installations depuis plusieurs années. Au total, il a été exécuté au moins $7,5 \times 10^6$ fois. On estime alors que chaque centaine d'exécutions répond aux conditions préalables. Alors, on peut considérer que $7,5 \times 10^4$ exécutions représentent une évaluation statistique. On estime que 4 000 exécutions de test correspondront à un test exhaustif: les estimations sont conservatoires. Conformément au tableau D.5, la probabilité de ne pas avoir testé quelque chose est égale à $2,87 \times 10^{-5}$.

Pour N = 4 000, les valeurs des premiers termes qui dépendent de n sont les suivantes:

Tableau D.5 – Probabilité de test de toutes les propriétés du programme

n	***p***
5×10^4	$1 - 1,49 \times 10^{-2} + 1,10 \times 10^{-4} - \ldots$
$7,5 \times 10^4$	$1 - 2,87 \times 10^{-5} + 4 \times 10^{-10} - \ldots$
1×10^5	$1 - 5,54 \times 10^{-8} + 1,52 \times 10^{-15} - \ldots$
2×10^5	$1 - 7,67 \times 10^{-19} + 2,9 \times 10^{-37} - \ldots$

En pratique, de telles estimations devraient être faites de façon conservatoire.

D.3 Références

De plus amples informations sur les techniques ci-dessus peuvent être trouvées dans les documents suivants:

a) Verification and Validation of Real-Time Software, Chapter 5. W. J. Quirk (ed.). Springer Verlag, 1985, ISBN 3-540-15102-8.

b) Combining Probabilistic and Deterministic Verification Efforts. W. D. Ehrenberger, SAFECOMP 92, Pergamon Press, ISBN 0-08-041893-7.

c) Ingenieurstatistik. Heinhold/Gaede, Oldenburg, 1972, ISBN 3-486-31743-1.

d) IEEE 352:1987, IEEE Guide for general principles of reliability of nuclear power generating station safety systems.

e) CEI 61164:1995, *Croissance de la fiabilité – Tests et méthodes d'estimation statistiques*.

D.2.4.3 Example

Consider a program that has been used at several installations for several years. In total, at least $7{,}5 \times 10^6$ runs have been executed. It is estimated that each 100th run fulfils the above prerequisites. So $7{,}5 \times 10^4$ runs made can be taken for statistical evaluation. It is estimated that 4 000 test runs would perform an exhaustive test. The estimates are conservative. According to table D.5, the probability of not having tested everything equals $2{,}87 \times 10^{-5}$.

For N = 4 000 the values of the first terms depending on n are:

Table D.5 – Probability of testing all program properties

n	***p***
5×10^4	$1 - 1{,}49 \times 10^{-2} + 1{,}10 \times 10^{-4} - \ldots$
$7{,}5 \times 10^4$	$1 - 2{,}87 \times 10^{-5} + 4 \times 10^{-10} - \ldots$
1×10^5	$1 - 5{,}54 \times 10^{-8} + 1{,}52 \times 10^{-15} - \ldots$
2×10^5	$1 - 7{,}67 \times 10^{-19} + 2{,}9 \times 10^{-37} - \ldots$

In practice, such estimates should be made so that they are conservative.

D.3 References

Further information on the above techniques can be found in the following documents:

a) Verification and Validation of Real-Time Software, Chapter 5. W. J. Quirk (ed.). Springer Verlag, 1985, ISBN 3-540-15102-8.

b) Combining Probabilistic and Deterministic Verification Efforts. W. D. Ehrenberger, SAFECOMP 92, Pergamon Press, ISBN 0-08-041893-7.

c) Ingenieurstatistik. Heinhold/Gaede, Oldenburg, 1972, ISBN 3-486-31743-1.

d) IEEE 352:1987, IEEE Guide for general principles of reliability analysis of nuclear power generating station safety systems.

e) IEC 61164:1995, *Reliability growth – Statistical test and estimation methods*.

Bibliographie

CEI 60068-1:1988, *Essais d'environnement – Première partie: Généralités et guide*

CEI 60529:1989, *Degrés de protection procurés par les enveloppes (Code IP)*

CEI 60812:1985, *Techniques d'analyse de la fiabilité des systèmes – Procédure d'analyse des modes de défaillance et de leurs effets (AMDE)*

CEI 60880:1986, *Logiciel pour les calculateurs utilisés dans les systèmes de sûreté des centrales nucléaires*

CEI 61000-4-1:1992, *Compatibilité électromagnétique (CEM) – Partie 4: Techniques d'essai et de mesure – Section 1: Vue d'ensemble sur les essais d'immunité. Publication fondamentale en CEM*

CEI 61000-4-5:1995, *Compatibilité électromagnetic (CEM) – Partie 4: Techniques d'essai et de mesure – Section 5: Essai d'immunité aux ondes de choc*

CEI 61000-5-2:1997, *Compatibilité électromagnétique (CEM) – Partie 5: Guides d'installation et d'atténuation – Section 2: Mise à la terre et câblage*

CEI 61025:1990, *Analyse par arbre de panne (AAP)*

CEI 61069-5:1994, *Mesure et commande dans les processus industriels – Appréciation des propriétés d'un système en vue de son évaluation – Partie 5: Evaluation de la sûreté de fonctionnement d'un système*

CEI 61078:1991, *Techniques d'analyse de la sûreté de fonctionnement – Méthode du diagramme de fiabilité*

CEI 61131-3:1993, *Automates programmables – Partie 3: Langages de programmation*

CEI 61160:1992, *Revue de conception formalisée*
Amendement 1 (1994)

CEI 61163-1:1995, *Déverminage sous contraintes – Partie 1: Entités réparables fabriquées en lots*

CEI 61164:1995, *Croissance de la fiabilité – Tests et méthodes d'estimation statistiques*

CEI 61165:1995, *Application des techniques de Markov*

CEI 61346-1:1996, *Systèmes industriels, installations et appareils, et produits industriels – Principes de structuration et désignations de référence – Partie 1: Règles de base*

CEI 61506:1997, *Mesure et commande dans les processus industriels – Documentation des logiciels d'application*

CEI 61704: *Lignes directrices pour le choix de méthodes de test en vue de l'évaluation de la fiabilité des logiciels* [1)]

_____

1) A publier.

Bibliography

IEC 60068-1:1988, *Environmental testing – Part 1: General and guidance*

IEC 60529:1989, *Degrees of protection provided by enclosures (IP Code)*

IEC 60812:1985, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*

IEC 60880:1986, *Software for computers in the safety systems of nuclear power stations*

IEC 61000-4-1:1992, *Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 1: Overview of immunity tests. Basic EMC publication*

IEC 61000-4-5:1995, *Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 5: Surge immunity test*

IEC 61000-5-2:1997, *Electromagnetic compatibility (EMC) – Part 5: Installation and mitigation guidelines – Section 2: Earthing and cabling*

IEC 61025:1990, *Fault tree analysis (FTA)*

IEC 61069-5:1994, *Industrial-process measurement and control – Evaluation of system properties for the purpose of system assessment – Part 5: Assessment of system dependability*

IEC 61078:1991, *Analysis techniques for dependability – Reliability block diagram method*

IEC 61131-3:1993, *Programmable controllers – Part 3: Programming languages*

IEC 61160:1992, *Formal design review*
Amendment 1 (1994)

IEC 61163-1:1995, *Reliability stress screening – Part 1: Repairable items manufactured in lots*

IEC 61164:1995, *Reliability growth – Statistical test and estimation methods*

IEC 61165:1995, *Application of Markov techniques*

IEC 61346-1:1996, *Industrial systems, installations and equipment and industrial products – Structuring, principles and reference designation – Part 1: Basic rules*

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

IEC 61704: *Guide to the selection of software test methods for reliability assessment* [1)]

[1)] To be published.

ISO/CEI 5807:1985, *Traitement de l'information – Symboles de documentation et conventions applicables aux données, aux organigrammes de programmation et d'analyse, aux schémas des réseaux de programmes et des ressources de système*

ISO/CEI 7185:1990, *Technologies de l'information – Langages de programmation – Pascal*

ISO/CEI 8631:1989, *Technologies de l'information – Structures de programmes et normes pour leur représentation*

ISO/CEI 8652:1995, *Technologies de l'information – Langages de programmation – Ada*

ISO/CEI 8807:1989, *Systèmes de traitement de l'information – Interconnexion de systèmes ouverts – LOTOS – Technique de description formelle basée sur l'organisation temporelle de comportement observationnel*

ISO/CEI 9899:1990, *Langages de programmation – C*

ISO/CEI/TR 10206:1991, *Technologies de l'information – Langages de programmation – Pascal étendu*

ISO/CEI 10514-1:1996, *Technologies de l'information – Langages de programmation – Partie 1: Modula-2, langage de base*

ISO/CEI 10514-3:1998, *Technologies de l'information – Langages de programmation – Partie 3: Modula-2 orienté objet*

ISO/CEI 13817-1:1996, *Technologies de l'information – Langages de programmation, leurs environnements et interfaces logiciel système – Méthode de développement de Vienne – Langage de spécification – Partie 1: Langage de base*

ISO/CEI 14882:1998, *Langages de programmation – C++*

ISO/CEI 1539-1:1997, *Technologies de l'information – Langages de programmation – Fortran – Partie 1: Langage de base*

ISO/CEI/TR 15942, *Guidance for the use of the Ada programming language in high integrity systems*[1)]

1) A publier.

ISO/IEC 5807:1985, *Information processing – Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts*

ISO/IEC 7185:1990, *Information technology – Programming languages – Pascal*

ISO/IEC 8631:1989, *Information technology – Program constructs and conventions for their representation*

ISO/IEC 8652:1995, *Information technology – Programming languages – Ada*

ISO/IEC 8807:1989, *Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*

ISO/IEC 9899:1990, *Programming languages – C*

ISO/IEC/TR 10206:1991, *Information technology – Programming languages – Extended Pascal*

ISO/IEC 10514-1:1996, *Information technology – Programming languages – Part 1: Modula-2, Base Language*

ISO/IEC 10514-3:1998, *Information technology – Programming languages – Part 3: Object Oriented Modula-2*

ISO/IEC 13817-1:1996, *Information technology – Programming languages, their environments and system software interfaces – Vienna Development Method – Specification Language – Part 1: Base language*

ISO/IEC 14882:1998, *Programming languages – C++*

ISO/IEC 1539-1:1997, *Information technology – Programming languages – Fortran – Part 1: Base language*

ISO/IEC/TR 15942, *Guidance for the use of the Ada programming language in high integrity systems*[1)]

---

1) To be published.

Index

Index

———————

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.

A Prioritaire

Nicht frankieren
Ne pas affranchir

Non affrancare
No stamp required

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 2-8318-5151-3

ICS 25.040.40; 35.240.50

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND