# NORME INTERNATIONALE

# INTERNATIONAL STANDARD

**CEI**
**IEC**
**880**

Première édition
First edition
1986



## Logiciel pour les calculateurs utilisés dans les systèmes de sûreté des centrales nucléaires

## Software for computers in the safety systems of nuclear power stations


Numéro de référence
Reference number
CEI/IEC 880: 1986

## 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 auprès du Bureau Central de la CEI.

Les renseignements relatifs à ces révisions, à l'établissement des éditions révisées et aux amendements peuvent être obtenus auprès des Comités nationaux de la CEI et dans les documents ci-dessous:

- **Bulletin de la CEI**
- **Annuaire de la CEI**
  Publié annuellement
- **Catalogue des publications de la CEI**
  Publié annuellement et mis à jour régulièrement

## Terminologie

En ce qui concerne la terminologie générale, le lecteur se reportera à la CEI 50: *Vocabulaire Electrotechnique International* (VEI), qui se présente sous forme de chapitres séparés traitant chacun d'un sujet défini. Des détails complets sur le VEI peuvent être obtenus sur demande. Voir également le dictionnaire multilingue de la CEI.

Les termes et définitions figurant dans la présente publication ont été soit tirés du VEI, soit spécifiquement approuvés aux fins de cette publication.

## Symboles graphiques et littéraux

Pour les symboles graphiques, les symboles littéraux et les signes d'usage général approuvés par la CEI, le lecteur consultera:

– la CEI 27: *Symboles littéraux à utiliser en électro-technique;*

– la CEI 417: *Symboles graphiques utilisables sur le matériel. Index, relevé et compilation des feuilles individuelles;*

– la CEI 617: *Symboles graphiques pour schémas;*

et pour les appareils électromédicaux,

– la CEI 878: *Symboles graphiques pour équipements électriques en pratique médicale.*

Les symboles et signes contenus dans la présente publication ont été soit tirés de la CEI 27, de la CEI 417, de la CEI 617 et/ou de la CEI 878, soit spécifiquement approuvés aux fins de cette publication.

## Publications de la CEI établies par le même comité d'études

L'attention du lecteur est attirée sur les listes figurant à la fin de cette publication, qui énumèrent les publications de la CEI préparées par le comité d'études qui a établi la présente publication.

## 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 from the IEC Central Office.

Information on the revision work, the issue of revised editions and amendments may be obtained from IEC National Committees and from the following IEC sources:

- **IEC Bulletin**
- **IEC Yearbook**
  Published yearly
- **Catalogue of IEC publications**
  Published yearly with regular updates

## Terminology

For general terminology, readers are referred to IEC 50: *International Electrotechnical Vocabulary* (IEV), which is issued in the form of separate chapters each dealing with a specific field. Full details of the IEV will be supplied on request. See also the IEC Multilingual Dictionary.

The terms and definitions contained in the present publication have either been taken from the IEV or have been specifically approved for the purpose of this publication.

## Graphical and letter symbols

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications:

– IEC 27: *Letter symbols to be used in electrical technology;*

– IEC 417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets;*

– IEC 617: *Graphical symbols for diagrams;*

and for medical electrical equipment,

– IEC 878: *Graphical symbols for electromedical equipment in medical practice.*

The symbols and signs contained in the present publication have either been taken from IEC 27, IEC 417, IEC 617 and/or IEC 878, or have been specifically approved for the purpose of this publication.

## IEC publications prepared by the same technical committee

The attention of readers is drawn to the end pages of this publication which list the IEC publications issued by the technical committee which has prepared the present publication.

# NORME INTERNATIONALE

# INTERNATIONAL STANDARD

**CEI**
**IEC**
**880**

Première édition
First edition
1986



## Logiciel pour les calculateurs utilisés dans les systèmes de sûreté des centrales nucléaires

## Software for computers in the safety systems of nuclear power stations



Bureau central de la Commission Electrotechnique Internationale 3, rue de Varembé Genève Suisse

IEC

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

CODE PRIX
PRICE CODE
XB

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

# SOMMAIRE

# CONTENTS

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

# LOGICIEL POUR LES CALCULATEURS UTILISÉS DANS LES SYSTÈMES DE SÛRETÉ DES CENTRALES NUCLÉAIRES

## PRÉAMBULE

1) Les décisions ou accords officiels de la CEI en ce qui concerne les questions techniques, préparés par des Comités d'Etudes où sont représentés tous les Comités nationaux s'intéressant à ces questions, expriment dans la plus grande mesure possible un accord international sur les sujets examinés.

2) Ces décisions constituent des recommandations internationales et sont agréées comme telles par les Comités nationaux.

3) Dans le but d'encourager l'unification internationale, la CEI exprime le vœu que tous les Comités nationaux adoptent dans leurs règles nationales le texte de la recommandation de la CEI, dans la mesure où les conditions nationales le permettent. Toute divergence entre la recommandation de la CEI et la règle nationale correspondante doit, dans la mesure du possible, être indiquée en termes clairs dans cette dernière.

## PRÉFACE

La présente norme a été établie par le Sous-Comité 45A: Instrumentation des réacteurs, du Comité d'Etudes n° 45 de la CEI: Instrumentation nucléaire.

Le texte de cette norme est issu des documents suivants:

| Règle des Six Mois | Rapport de vote |
|---|---|
| 45A(BC)88-I/II/III | 45A(BC)90 |

Pour de plus amples renseignements, consulter le rapport de vote mentionné dans le tableau ci-dessus.

*Les publications suivantes de la CEI sont citées dans la présente norme:*

Publications n$^{os}$ 557 (1982): Terminologie CEI sur les réacteurs nucléaires.
639 (1979): Réacteurs nucléaires. Utilisation du système de protection à d'autres fins que la sécurité.
643 (1979): Application des calculateurs numériques à l'instrumentation et à la conduite des réacteurs nucléaires.
671 (1980): Essais périodiques et surveillance du système de protection des réacteurs nucléaires.

*Autres publications citées:*

Norme ISO 2382/1 (1984): Traitement des données — Vocabulaire — Partie 01: Termes fondamentaux.
Guide de l'AIEA 50-SG-D3 (1980): Guide de sûreté.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

# SOFTWARE FOR COMPUTERS IN THE SAFETY SYSTEMS OF NUCLEAR POWER STATIONS

## FOREWORD

1) The formal decisions or agreements of the IEC on technical matters, prepared by Technical Committees on which all the National Committees having a special interest therein are represented, express, as nearly as possible, an international consensus of opinion on the subjects dealt with.

2) They have the form of recommendations for international use and they are accepted by the National Committees in that sense.

3) In order to promote international unification, the IEC expresses the wish that all National Committees should adopt the text of the IEC recommendation for their national rules in so far as national conditions will permit. Any divergence between the IEC recommendation and the corresponding national rules should, as far as possible, be clearly indicated in the latter.

## PREFACE

This standard has been prepared by Sub-Committee 45A: Reactor Instrumentations, of IEC Technical Committee No. 45: Nuclear Instrumentation.

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

| Six Months' Rule | Report on Voting |
|---|---|
| 45A(CO)88-I/II/III | 45A(CO)90 |

Further information can be found in the Report on Voting indicated in the table above.

*The following IEC publications are quoted in this standard:*

Publications Nos. 557 (1982): IEC Terminology in the Nuclear Reactor Field.
639 (1979): Nuclear Reactor. Use of the Protection System for Non-safety Purposes.
643 (1979): Application of Digital Computers to Nuclear Reactor Instrumentation and Control.
671 (1980): Periodic Tests and Monitoring of the Protection System of Nuclear Reactors.

*Other publications quoted:*

ISO standard 2382/1 (1984): Data processing — Vocabulary — Part 01: Fundamental terms.
IAEA Guide 50-SG-D3 (1980): Safety guide.

# LOGICIEL POUR LES CALCULATEURS UTILISÉS DANS LES SYSTÈMES DE SÛRETÉ DES CENTRALES NUCLÉAIRES

---

INTRODUCTION

Les principes de base pour la conception de l'instrumentation nucléaire appliqués en particulier aux systèmes de sûreté des centrales nucléaires ont été traités dans les normes existantes, se référant aux systèmes câblés tel le «Guide de sûreté 50-SG-D3» de l'AIEA.

La présente norme a été développée pour interpréter ces principes lors de l'utilisation de systèmes programmés — systèmes multiprocesseurs distribués aussi bien que gros processeurs centraux — dans les systèmes de sûreté des centrales nucléaires. Elle traite des principes et prescriptions relatives au logiciel du système et il convient qu'elle soit employée en association avec les normes appropriées traitant du matériel et de l'intégration du système.

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

Les aspects qui ont été spécialement envisagés, vu le caractère particulier des systèmes programmés et de leur logiciel, sont:

*a)* les critères auxquels doit répondre le matériel pour autant qu'ils affectent le logiciel, en tenant compte du haut degré d'interdépendance entre le matériel et le logiciel;

*b)* une approche générale du développement du logiciel pour assurer la production d'un logiciel de grande fiabilité;

*c)* une approche générale de la vérification du logiciel et de la validation du système programmé;

*d)* les procédures relatives à la maintenance du logiciel, la modification, la gestion de la configuration.

## 1. Domaine d'application et objet

Cette norme est applicable au logiciel de haute fiabilité exigé pour les équipements programmés devant être utilisés dans les systèmes de sûreté des centrales nucléaires pour les fonctions liées à la sûreté — fonctions de classe 1, selon la Publication 643 de la CEI: Application des calculateurs numériques à l'instrumentation et à la conduite des réacteurs nucléaires. Cela inclut le système de commande, les systèmes auxiliaires des systèmes de sauvegarde et les systèmes de protection.

Pour les fonctions qui ne sont pas de sûreté, les principes de la Publication 639 de la CEI: Réacteurs nucléaires — Utilisation du système de protection à d'autres fins que la sûreté, sont applicables dans le cas d'utilisation de systèmes programmés.

Cette norme fournit un ensemble de prescriptions pour chaque étape de l'élaboration du logiciel, comprenant la conception, le développement, la qualification et la mise en œuvre, ainsi que la documentation accompagnant chaque étape de l'élaboration du logiciel dans le but d'obtenir un logiciel de haute fiabilité.

Les principes appliqués dans la mise en œuvre de ces prescriptions sont les suivants:

— méthodes les plus avancées;
— conception descendante;
— modularité;
— vérification de chaque phase;
— documentation claire;
— documents pouvant être soumis à audit;
— contrôle de validation.

# SOFTWARE FOR COMPUTERS IN THE SAFETY SYSTEMS OF NUCLEAR POWER STATIONS

---

## INTRODUCTION

The basic principles for the design of nuclear instrumentation as specifically applied to the safety systems of nuclear power plants have been interpreted in existing standards with reference to hard-wired systems as the "Safety Guide 50-SG-D3" of the IAEA.

This standard has been developed to interpret these principles for the utilization of digital systems — multiprocessor distributed systems as well as larger scale central processor systems — in the safety systems of nuclear power plants. It discusses the software system principles and requirements and should be read in association with appropriate standards on computer hardware and system integration.

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

Aspects for which special recommendations have been produced, due to the unique nature of computer systems and their software are:

*a)* established hardware criteria as far as they affect the software, taking careful account of the high degree of interdependency between hardware and software;

*b)* a general approach to software development to assure the production of the highly reliable software required;

*c)* a general approach to software verification and computer system validation;

*d)* procedures for software maintenance, modification and configuration control.

## 1. Scope and object

This standard is applicable to highly reliable software required for computers to be used in the safety systems of nuclear power plants for safety functions — Class 1 functions according to IEC Publication 643: Application of Digital Computers to Nuclear Reactor Instrumentation and Control. This includes the safety actuation systems, the safety system support features and the protection systems.

For the utilization of computer systems for non-safety functions the principles of IEC Publication 639: Nuclear Reactors — Use of the Protection System for Non-safety Purposes, are applicable.

This standard provides requirements for each stage of software generation, including design, development, qualification and operation as well as the documentation for each stage of the software generation for the purpose of achieving highly reliable software.

The principles applied in developing these requirements include:

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

Des directives et informations complémentaires sur la manière de satisfaire aux prescriptions de la partie principale de cette norme sont données dans les annexes A à F. Les références à ces annexes sont données entre parenthèses.

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

## 2. Termes et définitions

Les termes utilisés sont conformes à la Norme ISO 2382 (jusqu'à 1984), sauf en ce qui concerne les termes suivants:

2.1 *Programme d'application*

Programme d'ordinateur accomplissant une tâche liée au processus à commander plutôt qu'au fonctionnement de l'ordinateur lui-même.

2.2 *Compactage du code*

Réduction réfléchie de la taille mémoire nécessitée par un programme obtenue par élimination des instructions redondantes ou sans rapport avec ledit programme.

2.3 *Ordinateur ou calculateur (Réf. Norme ISO 2382/1)*

Unité fonctionnelle programmable se composant d'une ou de plusieurs unités centrales associées et de périphériques, qui est commandée par des programmes rangés en mémoire interne et qui est capable d'effectuer des calculs importants, comportant de nombreuses opérations arithmétiques ou des opérations logiques, sans intervention humaine pendant l'exécution.

*Note.* — Un ordinateur peut se composer d'une seule ou de plusieurs unités interconnectées.

2.4 *Programme d'ordinateur*

Ensemble ordonné d'instructions et de données qui spécifient des opérations sous une forme acceptable pour exécution par un système programmé.

2.5 *Système programmé*

Ensemble fonctionnel constitué d'une ou de plusieurs unités fonctionnelles et du logiciel associé qui utilise des éléments de stockage communs pour tout ou partie d'un programme ainsi que pour tout ou partie des données nécessaires à l'exécution du programme.

2.6 *Données*

Représentation des faits, concepts ou instructions selon une manière formalisée convenable pour la communication, l'interprétation ou le traitement par un ordinateur.

2.7 *Défense en profondeur*

Mise en œuvre, vis-à-vis d'un seuil, de plusieurs barrières successives se recouvrant de telle manière que ce seuil ne puisse être franchi que dans le cas où toutes les barrières sont défaillantes.

2.8 *Diversité*

Existence de différents moyens de réaliser une fonction requise (par exemple autres principes physiques, autres manières de résoudre la même tâche).

2.9 *Tolérances aux fautes*

Capacité intrinsèque d'un système de fonctionner correctement de manière continue en présence d'un nombre limité de fautes concernant le matériel ou le logiciel.

Additional guidance and information on how to comply with the requirements of the main part of this standard is given in Appendices A to F. References to those appendices are given in brackets.

If practices differing from those of the appendices are used, they shall be documented and auditable according to the requirements of the main part of this standard.

## 2. Terms and definitions

Terms are used as defined in ISO Standard 2382 (up to 1984), except as given below.

2.1 *Application program*

A computer program that performs a task related to the process being controlled rather than to the functioning of the computer itself.

2.2 *Code compaction*

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

2.3 *Computer (Ref. ISO Standard 2382/1)*

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

*Note.* — A computer may be a stand-alone unit or may consist of several interconnected units.

2.4 *Computer program*

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

2.5 *Computer system*

A functional unit, consisting of one or more computers and associated software, that uses common storage for all or part of a program and also for all or part of the data necessary for the execution of the program.

2.6 *Data*

A representation of facts, concepts or instructions in a formalized manner suitable for communication, interpretation or processing by a computer.

2.7 *Defence in depth*

Provision of several overlapping subsequent limiting barriers with respect to one threshold, such that the threshold can only be surpassed, if all barriers have failed.

2.8 *Diversity*

Existence of different means of performing a required function (e.g. other physical principles, other ways of solving the same task).

2.9 *Fault tolerance*

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

2.10 *Marche dégradée*

Réduction par étapes des fonctions du système en réponse à une panne détectée alors que les fonctions essentielles sont maintenues.

2.11 *Initialiser*

Forcer les compteurs, clés, adresses ou contenus des organes de stockage à des valeurs prédéterminées au début ou en certains points prescrits du fonctionnement d'un programme.

2.12 *Tests d'intégration*

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

2.13 *Procédure*

Partie de programme d'ordinateur qui est désignée par un nom et qui réalise une tâche spécifique.

2.14 *Redondance*

Présence d'éléments ou systèmes (identiques ou différents) de rechange, de sorte que l'un d'entre eux puisse remplir la fonction requise quel que soit l'état de fonctionnement ou malgré la défaillance d'un autre (Réf. AIEA 50-SG-D8).

2.15 *Critère de défaillance unique (Publication 557 de la CEI: Terminologie CEI sur les réacteurs nucléaires)*

Critère appliqué à un système tel que celui-ci soit capable de remplir sa propre tâche de sécurité lorsqu'il est soumis à un seul défaut.

2.16 *Logiciel*

Programmes, procédures, règles et tout document associé, liés à la mise en œuvre du système programmé.

2.17 *Cycle de vie du logiciel*

Période de temps qui débute lors de la conception d'un produit logiciel et qui s'achève lorsque le produit n'est plus utilisable. Le cycle de vie du logiciel comprend une phase de spécification, une phase de conception, une phase de réalisation, une phase de test, une phase d'installation et de validation et une phase d'exploitation et de maintenance.

2.18 *Modification du logiciel*

Changement de documents déjà approuvés conduisant à un changement du code exécutable ou de ses données.

2.19 *Modularité du logiciel*

Caractéristique du logiciel qui présente une structure en unités de programme fortement indépendantes les unes des autres, autonomes et identifiables pour la compilation, le test et l'association avec d'autres unités.

2.20 *Validation*

Test et évaluation du système de traitement intégré (matériel et logiciel) pour s'assurer de sa conformité aux spécifications fonctionnelles, ainsi qu'aux spécifications de performances et d'interfaces.

2.21 *Vérification*

Processus consistant à déterminer lors de chaque phase du développement du système de traitement programmé si le produit élaboré répond aux spécifications issues de la phase précédente.

2.10 *Graceful degradation*

Stepwise reduction of system functions in response to detected failures while essential functions are maintained.

2.11 *Initialize*

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

2.12 *Integration tests*

Tests performed during the hardware/software integration process prior to computer system validation to verify compatibility of the software and the computer system hardware.

2.13 *Procedure*

A portion of a computer program which is named and which performs a specific task.

2.14 *Redundancy*

Provision of alternative (identical or diverse) elements or systems so that any one can perform the required function regardless of the state of operation or failure of any other (Ref. IAEA 50-SG-D8).

2.15 *Single failure criterion (IEC Publication 557: IEC Terminology in the Nuclear Reactor Field)*

A criterion applied to a system such that it is capable of performing its safety task in the presence of any single failure.

2.16 *Software*

Programs, procedures, rules and any associated documentation pertaining to the operation of a computer system.

2.17 *Software life cycle*

The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The software life cycle typically includes a requirements phase, design phase, implementation phase, test phase, installation and check-out phase, operation and maintenance phase.

2.18 *Software modification*

Changes of already agreed documents leading to a change to the executable code or its data.

2.19 *Software modularity*

The software attribute that provides a structure of highly independent computer program units that are discrete and identifiable with respect to translating, testing and combining with other units.

2.20 *Validation*

The test and evaluation of the integrated computer system (hardware and software) to ensure compliance with the functional, performance and interface requirements.

2.21 *Verification*

The process of determining whether or not the product of each phase of the digital computer system development process fulfils all the requirements imposed by the previous phase.

## 3. Structure du projet

En règle générale, un projet doit être découpé en phases. Chaque phase constitue une entité mais est interdépendante des autres phases, qui elles-mêmes en dépendent. Ces phases sont identifiables de manière informelle par les activités spécifiques qui s'y rapportent.

Pour les applications liées à la sûreté, ces phases doivent être formalisées, et aucune des phases identifiées ne doit être omise.

### 3.1 *Généralités*

Les facteurs généraux suivants déterminent les activités dans l'exécution d'un projet:

3.1.1 L'ensemble du cycle de vie du logiciel doit être considéré.

3.1.2 Chaque phase du cycle de vie du logiciel doit être divisée en tâches élémentaires dont l'activité doit être bien définie.

3.1.3 Chaque produit doit être systématiquement contrôlé après chaque phase (B4.g).

3.1.4 Les tâches d'assurance qualité doivent en règle générale se dérouler en parallèle avec les autres tâches du cycle de vie du logiciel.

3.1.5 Chaque phase doit inclure l'élaboration des documents appropriés (F3).

3.1.6 Chaque phase doit systématiquement se terminer par une revue critique. Les revues critiques constituent une part importante du processus de vérification du logiciel (voir article 6).

3.1.7 Chaque vérification ou revue critique doit donner lieu à un rapport sur les analyses réalisées, les conclusions et les décisions acceptées. Ce rapport doit être inclus dans la documentation.

3.1.8 Une liste des documents requis au minimum durant la vie du logiciel est donnée dans l'annexe F.

### 3.2 *Plan d'assurance qualité pour le logiciel*

Un plan d'assurance qualité doit exister ou être établi au début du projet soit en tant que chapitre dans les spécifications du système programmé (voir article 4), soit en tant que document d'accompagnement.

Bien que l'ensemble de cette norme et de ses annexes puisse être considéré comme un plan d'assurance qualité pour le logiciel relatif aux systèmes de sûreté, des plans de qualité spéciaux peuvent être adoptés pour les phases liées à des produits individuels pour des composants logiciels particuliers selon les normes nationales ou les règlements propres aux sociétés.

Il convient que ces plans traitent de toutes les procédures d'assurance qualité requises durant toutes les phases du cycle de vie du logiciel.

## 4. Spécifications du logiciel

Les spécifications du logiciel (F.M2) sont issues des spécifications des systèmes de sûreté et constituent une partie des spécifications du système programmé. Les spécifications du système programmé présentent une description à la fois du matériel et du logiciel et établissent les objectifs et les fonctions du système programmé.

Les exigences relatives au logiciel décrivent le produit, et non son développement. Elles doivent décrire ce qui doit être fait et non la manière de le faire. Une approche satisfaisante pour la définition du contenu et l'établissement des spécifications du logiciel, qui est en accord avec le présent article, est donnée dans l'annexe A.

Bien que le but principal de cette analyse fonctionnelle du logiciel soit de constituer la base de développement du logiciel, les aspects liés aux organismes de sûreté ne doivent pas être

## 3. Project structure

Any project will normally be divided up into a number of phases. Each phase is to some extent self-contained but will depend on other phases and will, in its turn, be depended on by others. These phases are informally recognizable by the specific activities pertinent to them.

For safety related applications, these phases shall be formalized and none of the identified phases shall be omitted.

### 3.1 *General*

The following general factors determine the activities in implementing a project:

3.1.1 The whole software life cycle shall be considered.

3.1.2 Each phase of the software life cycle shall be divided into elementary tasks with a well-defined activity for each of them.

3.1.3 Every product shall be systematically checked after each phase (B4.g).

3.1.4 The tasks of quality assurance should be run generally in parallel with the other tasks of the life cycle.

3.1.5 Each phase shall include generation of the appropriate documents (F3).

3.1.6 Each phase shall be systematically terminated by a critical review. Critical reviews form a major part of the verification process of the software project (see Clause 6).

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

3.1.8 A list of documents required as a minimum through the software life cycle is given in Appendix F.

### 3.2 *Software quality assurance plan*

A quality assurance plan shall exist or be established at an early stage either as a part of the computer system specification (see Clause 4) or as a companion document.

Although the whole of this standard and its appendices can be considered as a quality assurance plan for safety related software, special quality assurance plans may be adopted for individual product phases or particular software components according to national standards or company regulations.

These plans should address all quality assurance procedures required during all phases of the software life cycle.

## 4. Software requirements

The software requirements (F.M2) shall be derived from requirements of the safety systems and are part of the computer system specification. The computer system specification is a description of the combined hardware/software system and states the objectives and functions assigned to the computer system.

The software requirements describe the product, not the project. They shall describe what has to be done and not how it has to be done. An acceptable approach to the development and content of the software requirements, which is consistent with this clause, is given in Appendix A.

Whereas the main purpose of the software requirements document is to form the basis for software development, the licensing aspects should not be neglected. Therefore, it may

négligés. En conséquence, il peut contenir des éléments moins importants pour la conception du logiciel qui sont cependant des informations de base pour les organismes de sûreté. Ce sont par exemple:

— les considérations liées aux risques;

— des recommandations pour les fonctions des systèmes de sauvegarde et de sûreté;

— d'autres éléments qui fournissent des bases pour les exigences spécifiques.

Les exigences fonctionnelles relatives au logiciel détaillent les fonctions du système qui seront mises en œuvre par le système programmé.

Les exigences en matière de fiabilité du logiciel détaillent, en ce qui concerne le logiciel, les exigences en fiabilité du système programmé. Elles doivent en être issues de la même manière que les exigences fonctionnelles.

4.1 Les spécifictions du système programmé doivent donner une vue externe et synthétique des fonctions qui doivent être réalisées par le logiciel du système programmé (A2.1).

4.2 La configuration du système programmé doit être décrite en prenant en compte ses exigences de fiabilité et son environnement, étant donné que les caractéristiques de fiabilité et la configuration du système sont intimement liées (A2.2).

4.3 Les principes du Guide 50-SG-D3 de l'AIEA, et spécialement ceux figurant aux paragraphes 7.12 et 7.3.2, sont applicables en ce qui concerne l'interaction entre le système programmé et l'opérateur ou toute autre personne.

Ce dialogue homme-machine doit être spécifié en tenant compte:

— de la recette du système et de sa mise en service;

— de l'exploitation du système;

— de la maintenance du système et des essais après mise en service (A2.3).

4.4 L'interface entre le système programmé et tout autre système à l'intérieur ou à l'extérieur de la centrale nucléaire, qu'une connexion directe existe ou soit prévue, doit être identifiée et documentée en indiquant les interfaces spécifiques et les exigences pour le logiciel qui en découlent (Réf. Publication 639 de la CEI) (A2.4).

4.5 Les fonctions destinées à prévenir une émission de radioactivité pour les conditions accidentelles (A2.1) et les fonctions associées doivent être décrites (A2.5). Il convient de décrire les fonctions à réaliser plutôt que la manière de les mettre en œuvre.

4.6 Les contraintes entre matériel et logiciel doivent être décrites (A2.6). La référence au document de spécifications du matériel doit être indiquée.

4.7 Les conditions particulières de fonctionnement, tels le démarrage de la centrale et le rechargement en combustible, doivent être décrites (A2.7).

4.8 Le logiciel du système programmé doit posséder des fonctions d'autocontrôle couvrant le logiciel lui-même et le matériel hôte (A2.8). Cela est considéré comme un facteur essentiel pour atteindre l'objectif de fiabilité de l'ensemble du système.

4.8.1 L'autocontrôle doit remplir les exigences suivantes, à moins qu'il ne puisse être prouvé que d'autres moyens procurent le même degré de sûreté:

*a)* aucune panne unique ne doit pouvoir bloquer directement ou indirectement une fonction quelconque destinée à prévenir un rejet de radioactivité;

*b)* les zones mémoires qui contiennent les paramètres ou les constantes doivent être protégées en écriture de manière à prévenir tout changement de leur contenu.

4.8.2 Le logiciel du système de sûreté doit être conçu de manière à ce que les fonctions essentielles de l'ensemble du système de sûreté puissent être testées durant le fonctionnement de la centrale nucléaire.

contain aspects of minor importance to software design which are, however, a background for licensing. Such aspects may be:

— risk considerations;
— recommendations for functions or engineered safety features;
— other items that provide the background for specific requirements.

The software functional requirements represent an expansion of the functions which are assigned to the computer system and which are to be implemented by the computer system.

The software reliability requirements represent an expansion of the reliability requirements of the computer system. They shall be derived in a similar way to the functional requirements.

4.1 The computer system specification shall give an external and synthetic view of the functions to be implemented by the software of the computer system (A2.1).

4.2 The computer system configuration shall be described using as a background the reliability requirements and the environment of the system, since reliability properties and system configuration are closely connected (A2.2).

4.3 For the interaction between the computer system and the plant operator or any other person, the principles of IAEA "Guide 50-SG-D3" in general and especially those of Sub-clauses 7.12 and 7.3.2 are applicable.

This man-machine dialogue shall be specified with regard to:

— commissioning of the system and start up;
— system operation;
— system maintenance, and testing after commissioning (A2.3).

4.4 The interface between the computer system and any other system either within or outside the nuclear plant, wherever a direct connection exists or is planned, shall be identified and documented indicating the specific interfaces and the related software requirements (Ref. IEC Publication 639) (A2.4).

4.5 Functions dedicated to the prevention of the release of radioactivity under accident conditions (A2.1) and their related system functions shall be described (A2.5). The functions to be performed should be defined rather than the means of implementation.

4.6 The constraints between hardware and software shall be described (A2.6). A reference to the hardware requirements document shall be made.

4.7 Special operating conditions such as plant commissioning and refuelling shall be described (A2.7).

4.8 The computer system software shall continuously supervise both itself and the hardware (A2.8). This is considered a primary factor in the achievement of the overall system reliability requirements.

4.8.1 Self-supervision shall meet the following requirements, unless it is proved that other means provide the same degree of safety:

*a)* no single failure shall be able to block directly or indirectly any function dedicated to the prevention of the release of radioactivity;

*b)* those parts of the memory that contain code or invariable data shall be monitored to prevent any changes.

4.8.2 The safety system software shall be designed in such a manner that essential functions of the whole safety system be testable during the operation of the nuclear power generating station.

4.8.3 Si une panne quelconque du système de sûreté est détectée durant le fonctionnement de la centrale, les actions automatiques appropriées doivent être prises en tenant compte des exigences temporelles. Cela peut impliquer de prendre en considération des dispositions pour éviter des déclenchements intempestifs.

4.8.4 Les autocontrôles du système ne doivent pas dégrader les fonctions qu'il doit remplir.

4.8.5 Le logiciel du système de sûreté doit être conçu pour satisfaire aux exigences du test périodique qui a lieu à des intervalles de temps maximaux spécifiés (par exemple pendant les périodes d'arrêt).

*a)* Toutes les fonctions élémentaires doivent être couvertes par le test périodique.

*b)* Toute dégradation dans l'exécution des fonctions de sûreté doit être détectée.

*c)* Les autocontrôles de base doivent être vérifiables durant les tests périodiques.

Il convient que le logiciel soit capable d'acquérir automatiquement toutes les informations obtenues durant le test périodique. Voir le paragraphe 10.3 pour d'autres détails sur le test périodique.

4.9 Les exigences fonctionnelles pour le logiciel doivent être présentées selon un modèle dont le formalisme ne doit pas, en principe, nuire à la compréhension (A2.9). Les exigences doivent être non équivoques, testables ou vérifiables, et réalisables.

4.10 L'utilisation d'un langage formel de spécification peut être une aide pour démontrer la cohérence et l'exhaustivité des exigences fonctionnelles pour le logiciel. Des outils automatiques peuvent être utilisés à cet effet.

4.11 Les exigences fonctionnelles pour le logiciel (analyse fonctionnelle) doivent être fournies:

— aux auteurs du document spécifiant les exigences de l'ensemble du système programmé, pour évaluation et accord avant programmation;

— au responsable de l'équipe de conception du logiciel pour accord, en ce qui concerne les aspects faisabilité et lisibilité;

— aux responsables de la sûreté pour autorisation d'exploitation, en ce qui concerne le respect de l'ensemble des exigences de sûreté du réacteur.

## 5. Développement (conception et codage) du logiciel des systèmes de sûreté

Les recommandations ci-après fournissent un guide de bonne pratique pour écrire un logiciel aussi exempt d'erreur que possible dès le début de la conception et pouvant être aisément vérifié.

Il convient que la spécification des exigences fonctionnelles à remplir par le logiciel soit disponible avant les phases de conception et de codage du développement des programmes.

### 5.1 *Principes du développement (conception et codage)*

5.1.1 Les principes généraux de développement ci-après sont basés sur l'expérience dans la production de logiciels exempts d'erreur et compréhensibles.

— La conception du logiciel doit inclure l'autosurveillance des flux de contrôle et de données. La détection d'une erreur doit entraîner l'action appropriée, conformément au paragraphe 4.8.

— Il convient que la structure du programme soit basée sur une décomposition en modules.

— Il convient que la structure du programme soit simple et facile à comprendre, à la fois dans sa conception générale et dans ses détails. Il est recommandé de bannir les «astuces», les structures récursives et la réduction de code non nécessaires.

— Il convient que le programme source résultat soit compréhensible dans sa totalité.

— Une bonne documentation doit être fournie.

4.8.3 If any failure is detected during plant operation, appropriate and timely automatic actions shall be taken. This may require giving due consideration to avoiding spurious trips.

4.8.4 System self-checking shall not adversely affect the intended system functions.

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

*a)* Every single function shall be coverable by periodic testing.

*b)* Any degradation of the execution of safety functions shall be detected.

*c)* The basic self-checking functions shall also be testable during periodic tests.

The software should be able to collect automatically all information gained during periodic testing. Further details for periodic testing are given in Sub-clause 10.3.

4.9 Software functional requirements shall be presented according to a standard whose formality should not preclude readability (A2.9). The requirements shall be unequivocal, testable or verifiable, and realizable.

4.10 The use of a formal specification language may be a help to show coherence and completeness of the software functional requirements. Automatic tools may be used for this purpose.

4.11 The software functional requirements shall be provided:

— to the authors of the computer system requirements documents for assessment and approval prior to programming;

— to the software design team manager for approval and with respect to feasability and readability;

— to the licensers with respect to compliance with the overall plant safety requirements for licensing approval.

## 5. Development (design and coding) of safety system software

The following recommendations provide a guide to good practice for writing software which is as error-free as possible from the very beginning and which can easily be verified.

The software functional requirements specification should be available, before the design and coding phase of program development begins.

### 5.1 *Principles for development (design and coding)*

5.1.1 The following general principles for development are based on experience in producing error-free and understandable software.

— The software design shall include self-supervision of control flow and data. On failure detection, appropriate action shall be taken in accordance with Sub-clause 4.8.

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

— The program structure should be simple and easy to understand, both in its overall design and in its details. Tricks, recursive structures and unnecessary code compaction should be avoided.

— The final source program should be readable from start to end.

— Good documentation shall be provided.

5.1.2 Les recommandations suivantes dérivent de ces principes.

Il convient que:

*a)* les mesures à prendre pour obtenir la fiabilité requise, y compris l'autocontrôle, soient choisies au début du développement (B3);

*b)* une approche descendante soit préférée à une approche ascendante dans le développement du logiciel (B1);

*c)* un modèle conceptuel de la structure du système soit adopté au début de chaque projet logiciel (B2);

*d)* le programme soit écrit pour être facilement testable (B4, B5);

*e)* dans le cas d'utilisation d'un logiciel commercialisé, la démonstration de son bon fonctionnement soit réalisée (B2.c).

5.2 *Langage, traducteur et autres outils*

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

Bien qu'un langage spécifique ne puisse être requis, ce qui suit peut être considéré comme les règles de base communes pour les langages de programmation de système de sûreté.

5.2.1 Il convient d'utiliser des langages disposant d'un traducteur complètement testé. Si un traducteur partiellement testé est utilisé, une vérification supplémentaire devra montrer que le résultat de la traduction est correct.

5.2.2 Il convient que le langage soit défini de manière complète et non ambiguë, à défaut, son emploi devra être limité aux seuls éléments définis de manière complète et non ambiguë. Cela s'applique également quand il existe un doute dans la traduction d'un élément particulier du langage ou une combinaison particulière d'éléments.

5.2.3 Les langages orientés «application» sont fortement préférables aux langages orientés «machine».

5.2.4 De même que les points spécifiques mentionnés dans l'annexe D, il est recommandé qu'un langage de programmation d'un système de sûreté et son traducteur n'interdisent pas par leur conception:

— les constructions limitant les erreurs;

— la vérification des types lors de la traduction;

— la vérification des types et des limites de validité des pointeurs de tables, ainsi que la vérification des paramètres lors de l'exécution.

5.2.5 Il convient que des outils automatiques de test soient disponibles.

5.2.6 Il est recommandé d'utiliser des outils automatiques.

5.3 *Recommandations détaillées*

5.3.1 *Développement*

Dans l'annexe B figure une série de recommandations qui spécifient en détail les aspects du paragraphe 5.1.

Les têtes de chapitre des recommandations individuelles de l'annexe B, ainsi que les articles appropriés de la présente norme, peuvent aussi être affectés aux deux parties essentielles du développement d'un programme, la conception et le codage, suivant le tableau ci-après.

5.1.2 From these principles the following recommendations are derived:

*a)* measures to obtain the required reliability including self-supervision should be chosen at the outset of the development (B3);

*b)* a top down approach to software development should be preferred to a bottom up approach (B1);

*c)* a conceptual model of system structure should be adopted at the beginning of each software project (B2);

*d)* the program should be written to allow easy testing (B4, B5);

*e)* where standard software from a manufacturer or supplier is used, it should be shown to have operated satisfactorily (B2.c).

5.2 *Language, translator and other tools*

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

Even though a specific programming language cannot be required, the following may be considered as common basic rules for safety system programming languages.

5.2.1 Languages with a thoroughly tested translator should be used. If no thoroughly tested translator is employed, additional verification shall show that the result of the translation is correct.

5.2.2 The language should be completely and unambiguously defined, otherwise the use of the language shall be restricted to completely and unambiguously defined features. This applies in a similar way if there is any doubt about the correct translation of a specific language feature or a particular combination of such features.

5.2.3 Problem oriented languages are strongly preferred to machine oriented ones.

5.2.4 As well as the specific points mentioned in Appendix D, a programming language for safety systems and its translator should not prevent by their design:

— error-limiting constructs;
— translation-time type checking;
— run-time type and array bound check, and parameter checking.

5.2.5 Automatic testing aids should be available.

5.2.6 The use of automatic tools is recommended.

5.3 *Detailed recommendations*

5.3.1 *Development*

In Appendix B a set of recommendations is given which specifies in detail the aspects identified in Sub-clause 5.1.

The headings of the individual recommendations of Appendix B as well as relevant clauses of this standard can also be attributed to the two major parts of program development, design and coding, according to the following table.

*Aspects, procédures et structures de conception et de codage de programmes d'application pour les systèmes de sûreté*

| | Aspects procédures | Rubrique/ Article | Aspects structures | Rubrique/ Article |
|---|---|---|---|---|
| Conception | Adaptabilité | 1.*a*/4. | Structure de contrôle et d'accès | 2.*a* |
| | Approche descendante | 1.*b* | Modules | 2.*b* |
| | Vérification intermédiaire | 1.*c*/6. | Systèmes d'exploitation | 2.*c* |
| | Modifications en cours de développement | 1.*d*/9. | Temps d'exécution | 2.*d*/4. |
| | Reconfiguration du système | 1.*e*/9. | Interruptions | 2.*e* |
| | | | Expressions arithmétiques | 2.*f* |
| | | | Contrôle de vraisemblance | 3.*a*/4. |
| | | | Sûreté des sorties | 3.*b*/4. |
| | | | Branchements et boucles | 4.*a* |
| | | | Sous-programmes et procédures | 4.*b* |
| | | | Structures imbriquées | 4.*c* |
| | | | Structures des données | 4.*e* |
| Codage | Vérification intermédiaire | 1.*c*/6. | Modules | 2.*b* |
| | Modification en cours de développement | 1.*d*/9. | Temps d'exécution | 2.*d*/4. |
| | Tests intermédiaires | 4.*g*/6. | Interruptions | 2.*e* |
| | Règles de codage | 5.*d* | Expressions arithmétiques | 2.*f* |
| | | | Contrôle de vraisemblance | 3.*a*/4. |
| | | | Sûreté des sorties | 3.*b*/4. |
| | | | Contenu mémoire | 3.*c* |
| | | | Contrôle des erreurs | 3.*d* |
| | | | Branchements et boucles | 4.*a* |
| | | | Sous-programmes et procédures | 4.*b* |
| | | | Structures imbriquées | 4.*c* |
| | | | Adressage et zones de données | 4.*d* |
| | | | Structures des données | 4.*e* |
| | | | Modifications dynamiques | 4.*f* |
| | | | Organisation du programme | 5.*a* |
| | | | Commentaires | 5.*b* |
| | | | Assembleur | 5.*c* |



5.3.2 *Utilisation des recommandations*

Il convient que les recommandations applicables au programme à développer soient choisies et notées dès le début de tout projet de réalisation de logiciel concernant un système de sûreté. Ce choix peut inclure une modification dans l'ordre des priorités des recommandations ainsi qu'une subdivision en détail des recommandations individuelles.

En liaison avec certains problèmes particuliers, il peut être raisonnable de modifier les recommandations contenues dans l'annexe B afin d'atteindre l'objectif de simplicité et de compréhension de l'ensemble. Ces modifications doivent être documentées, en tenant compte de ce qui suit:

*a)* durant la procédure de sélection des recommandations, il convient de considérer les parties du logiciel relevant particulièrement de la sûreté;

*b)* pour les sections du logiciel les plus critiques du point de vue de la sûreté du système, il y a lieu de sélectionner et d'appliquer les recommandations les plus contraignantes;

*c)* il est recommandé que les sections de programme relevant particulièrement de la sécurité soient clairement identifiées dans la structure du système et des données utilisées;

*Procedural and structural aspects of design and coding of user program for safety system*

| | Procedural aspects | Item/ Clause | Structural aspects | Item/ Clause |
|---|---|---|---|---|
| Design | Changeability | 1.*a*/4. | Control and access structures | 2.*a* |
| | Top-down approach | 1.*b* | | |
| | Intermediate verification | 1.*c*/6. | Modules | 2.*b* |
| | Changes during development | 1.*d*/9. | Operating system | 2.*c* |
| | | | Execution time | 2.*d*/4. |
| | System reconfiguration | 1.*e*/9. | Interrupts | 2.*e* |
| | | | Arithmetic expressions | 2.*f* |
| | | | Plausibility check | 3.*a*/4. |
| | | | Safe output | 3.*b*/4. |
| | | | Branches and loops | 4.*a* |
| | | | Subroutines and procedures | 4.*b* |
| | | | Nested structures | 4.*c* |
| | | | Data structures | 4.*e* |
| Coding | Intermediate verification | 1.*c*/6. | Modules | 2.*b* |
| | Changes during development | 1.*d*/9. | Execution time | 2.*d*/4. |
| | | | Interrupts | 2.*e* |
| | Intermediate tests | 4.*g*/6. | Arithmetic expressions | 2.*f* |
| | Coding rules | 5.*d* | Plausibility check | 3.*a*/4. |
| | | | Safe output | 3.*b*/4. |
| | | | Memory contents | 3.*c* |
| | | | Error checking | 3.*d* |
| | | | Branches and loops | 4.*a* |
| | | | Subroutines and procedures | 4.*b* |
| | | | Nested structures | 4.*c* |
| | | | Addressing and arrays | 4.*d* |
| | | | Data structures | 4.*e* |
| | | | Dynamic changes | 4.*f* |
| | | | Sequences and arrangements | 5.*a* |
| | | | Comments | 5.*b* |
| | | | Assembler | 5.*c* |



5.3.2 *Use of the recommendations*

At the outset of any safety related programming project those recommendations should be selected and recorded which apply to the intended program development. The selection may include the change of priorities and further subdivide into details of individual recommendations.

In connection with particular problems it may be reasonable to modify the recommendations selected from Appendix B, in order to meet the goal of overall simplicity and understandability. These modifications shall be documented, taking account of the following:

*a)* during the selection procedure of the recommendations, the safety relevance of particular software parts should be considered;

*b)* the more limiting recommendations should be selected and applied to those program sections which are more critical to the safety of the system;

*c)* program sections of outstanding safety relevance should be clearly identifiable from the system and data structure used;

*d)* il y a lieu de tenir compte des facilités de test disponibles et de la stratégie de validation prévue lors de la procédure de sélection. Si des parties importantes sont programmées de manière diversifiée, on peut utiliser pour ces parties une stratégie différente de vérification;

*e)* si des difficultés apparaissent durant la phase de vérification, des modifications rétrospectives du style de programmation peuvent être nécessaires.

5.4 *Documentation*

5.4.1 Lors du développement du programme, la fin de la phase de conception doit donner lieu à un document formalisé, l'analyse organique (F.M3). Ce document est utilisé comme référence pour la revue d'étude de conception et le codage ultérieur.

Des détails suffisants doivent y être inclus pour permettre le codage sans clarification supplémentaire. Il est recommandé de structurer le document pour permettre son développement parallèlement au processus de conception. Un résumé du contenu proposé est donné dans l'annexe C.

5.4.2 En plus de l'analyse organique, d'autres documents peuvent être requis. Ils sont utilisés à la fois pour les étapes de vérifications intermédiaires et finales. Certains documents se rapportent à la conception initiale. Ils sont dérivés des spécifications fonctionnelles et constituent la base de l'analyse organique. D'autres sont issus de ce document particulier et servent de référence au codage ultérieur.

Si les recommandations relatives à la procédure de développement donnée dans l'annexe B sont suivies, la documentation appropriée est élaborée automatiquement.

5.4.3 Le but est d'obtenir un ensemble intégré de documents. Chaque document doit avoir une relation définie avec les autres documents et traiter d'un sujet bien délimité.

5.4.4 Il est recommandé de choisir la présentation de la documentation selon le besoin particulier, incluant:

— la description narrative;
— les expressions arithmétiques;
— toute forme de diagrammes et dessins.

En règle générale, il est préférable de choisir une représentation graphique. La documentation elle-même doit suivre les normes nationales.

## 6. Vérification

6.1 *Processus de vérification*

Après établissement des exigences fonctionnelles du logiciel et avant que ne commence la phase suivante, la vérification consiste à contrôler l'adéquation entre les exigences fonctionnelles pour le logiciel et les exigences du système de sûreté assignées au logiciel par les spécifications de l'équipement programmé.

Après la phase de conception et avant le début de la phase suivante, la vérification consiste à contrôler l'adéquation de la conception du logiciel, incluse dans l'analyse organique, avec les exigences fonctionnelles pour le logiciel (analyse fonctionnelle).

Après la phase de codage et avant le début de la phase suivante, la vérification consiste à contrôler la conformité du code avec l'analyse organique issue de la phase de conception.

Chaque phase doit être complétée par une vérification (A1).

*d)* the selection procedure should take into account the available testing facilities and the intended validation strategy. If important parts are programmed diversely, a different verification strategy may be used for those parts;

*e)* if difficulties arise during verification, a retrospective change of programming style may be necessary.

5.4 *Documentation*

5.4.1 During program development the end of the design phase shall be marked by production of a formal document, the software performance specification (F.M3). This document serves as the basis for the formal design review and the subsequent program coding.

Sufficient detail shall be included so that program coding can proceed without further design clarification. The document should be structured for continued expansion in parallel to the design process. An outline of the suggested content is given in Appendix C.

5.4.2 In addition to the software performance specification, other documents may be required. They are used for both intermediate and final program verification steps. Some of these documents relate to the first design steps. They are derived from the functional requirements specification and provide the basis for the software performance specification. Others are derived from this particular document and represent the basis for later coding.

If the recommendations for the development procedure according to Appendix B are met, appropriate documentation is provided as a byproduct of program development.

5.4.3 The aim is an integrated set of documents. Each document shall have a defined relationship to the other document and contain a well bounded subject-matter.

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

— narrative description;
— arithmetic expressions;
— appropriate diagrams and drawings.

As a general rule it is preferable to choose a graphical representation. The documentation itself shall follow national standards.

## 6. Verification

6.1 *Software verification process*

After the software functional requirements have been established and before the next phase begins, verification addresses the adequacy of the software functional requirements in fulfilling the safety system requirements assigned to the software by the computer system specification.

After the design phase and before the next phase begins, verification addresses the adequacy of computer system software design as documented in the software performance specification to the software functional requirements.

After the coding phase and before the next phase begins, verification addresses the compliance of the coded computer system software to the software performance specification as derived by the design phase.

Each phase shall be completed by verification (A1).

6.2 *Activités de vérification du logiciel*

6.2.1 *Plan de vérification*

En parallèle avec les phases de développement du logiciel décrites ci-dessus, un plan de vérification doit être établi. Ce plan doit expliciter tous les critères, les techniques et les outils à utiliser dans le processus de vérification. Il doit décrire les activités à réaliser pour évaluer chaque élément du logiciel et pour contrôler à chaque phase le respect des exigences fonctionnelles et de fiabilité. Le niveau de détail doit être tel qu'un groupe indépendant puisse exécuter ce plan de vérification et émettre un jugement objectif sur la capacité ou non du logiciel à satisfaire aux exigences de performances.

*Note.* — L'indépendance implique que la vérification soit effectuée par un individu ou une organisation séparée de l'individu ou de l'organisation réalisant le développement. La meilleure manière est la mise en place d'une équipe de vérification.

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

*a)* la sélection des stratégies de vérification (systématique et/ou aléatoire avec choix de jeux d'essais selon les fonctions requises et/ou les caractéristiques de la structure du programme, voir annexe E);

*b)* la sélection et l'utilisation d'un outil de test du logiciel;

*c)* l'exécution de la vérification;

*d)* la documentation des activités de vérification;

*e)* l'évaluation des résultats de la vérification obtenus directement à partir des outils de vérification et des tests, évaluation du respect des exigences de fiabilité.

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

6.2.2 *Vérification de la conception, revues critiques*

La vérification de la conception consiste à examiner:

*a)* l'adéquation de l'analyse organique avec les spécifications fonctionnelles du logiciel (analyse fonctionnelle) en ce qui concerne leur cohérence et leur exhaustivité, et cela jusqu'au niveau des modules;

*b)* la décomposition de l'analyse organique en modules fonctionnels et la manière selon laquelle ils sont spécifiés, pour les aspects:

— faisabilité des traitements requis;
— testabilité pour la vérification ultérieure;
— lisibilité par les équipes de développement et de vérification;
— aptitude à la maintenance pour permettre des évolutions ultérieures;

*c)* le respect des exigences de qualité.

Le résultat de la vérification de la conception doit être explicité dans un rapport (F.M5).

Ce rapport doit indiquer:

— les éléments qui ne sont pas conformes aux exigences fonctionnelles du logiciel;
— les éléments qui ne sont pas conformes aux normes de conception;
— les modules, données, structures et algorithmes peu adaptés au problème.

6.2.3 *Phase de vérification du codage*

La vérification du codage commence avec le test des modules et prend en charge le logiciel par une approche ascendante. Au niveau module, le logiciel n'est pas encore intégré dans le système, il peut donc être testé en grande partie. Le but du test des modules est de montrer que chaque module réalise exclusivement la fonction attendue. Un test d'intégration des modules

## 6.2 *Software verification activities*

### 6.2.1 *Verification plan*

Concurrently with the phases of the software development cycle described above a software verification plan shall be established. The plan shall document all the criteria, the techniques and tools to be utilized in the verification process. It shall describe the activities to be performed to evaluate each item of software and each phase to show whether the functional and reliability requirements specification is met. The level of detail shall be such that an independent group can execute the verification plan and reach an objective judgement on whether or not the software meets its performance requirements.

*Note.* — The requirements for an independent group implies verification either by an individual or an organization which is separate from the individual or organization developing the software. The most appropriate way is to engage a verification team.

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

*a)* selection of verification strategies, either systematic, random or both, with test case selection according to either required functions, special features of program structure, or both, see Appendix E;

*b)* selection and utilization of the software test equipment;

*c)* execution of verification;

*d)* documentation of verification activities;

*e)* evaluation of verification results gained from verification equipment directly and from tests, evaluation of whether the reliability requirements are met.

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

### 6.2.2 *Design verification, critical reviews*

The design verification addresses:

*a)* the adequacy of the software performance specification for the software functional requirements with respect to consistency and completeness down to and including the modular level;

*b)* the decomposition of the design into functional modules and the manner of specification with reference to:

— feasibility of the performance required;

— testability for further verification;

— readability by the development and verification team;

— maintainability to permit further evolution;

*c)* the respect of quality requirements.

The result of the design verification shall be documented in a report (F.M5).

This report shall include:

— items which do not conform to the software functional requirements;

— items which do not conform to the design standards;

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

### 6.2.3 *Coding phase verification*

The code verification activities begin with module testing and go up through the software by a bottom-up strategy. At the module level, the software is not yet integrated into the system, therefore it can be extensively tested. The purpose of module testing is to show that each module performs its intended function and does not perform unintended functions. A

doit alors être effectué pour montrer au plus tôt l'interaction correcte de tous les modules pour une fonction donnée.

La spécification de test du logiciel constitue des directives pour les activités de vérification du code.

6.2.3.1 *Spécification de test du logiciel*

La spécification de test du logiciel est un des principaux documents du plan de vérification pour la phase de codage. Ce document doit être fondé sur un examen détaillé des exigences fonctionnelles pour le logiciel et doit donner des informations détaillées sur les tests à réaliser pour chaque élément du logiciel (les modules et leurs constituants).

La spécification de test du logiciel s'appuie sur le descriptif général du logiciel à tester. Cette description est fournie par l'équipe de développement.

La spécification de test du logiciel doit inclure:

*a)* l'environnement dans lequel s'effectue le test;

*b)* les procédures de test;

*c)* les critères d'acceptation, c'est-à-dire la définition détaillée des critères à remplir pour accepter les modules et les composants importants du logiciel au niveau sous-système et système;

*d)* la détection d'erreurs et les procédures de correction;

*e)* une liste de tous les documents qu'il est recommandé de produire lors de la phase de vérification du codage.

6.2.3.2 *Compte-rendu de test du logiciel*

Il doit présenter les résultats du programme de test décrit dans la spécification de test du logiciel en indiquant s'il satisfait ou non aux exigences données dans l'analyse fonctionnelle. Ce document (F.M4) doit permettre un diagnostic complet des anomalies de conception et de fonctionnement. Il doit aussi comprendre la correction des anomalies et des divergences, entre résultats attendus et résultats obtenus, découvertes lors du test.

Le compte rendu de test du logiciel doit inclure les rubriques suivantes à la fois au niveau du module et des niveaux plus élevés:

*a)* configuration du matériel utilisé pour le test;

*b)* moyen de stockage et conditions d'accès au code final à tester;

*c)* liste des éléments en entrée du test;

*d)* liste des éléments en sortie du test;

*e)* données supplémentaires en ce qui concerne la chronologie, la séquence d'événements, etc.;

*f)* conformité avec les critères d'acceptation donnés dans la spécification du test;

*g)* relevé des erreurs décrivant les caractéristiques de chaque erreur et les remèdes mis en œuvre par l'équipe de développement.

## 7. Intégration matériel/logiciel

Le processus d'intégration du matériel et du logiciel est l'association du matériel contrôlé et du logiciel vérifié dans un système capable de réaliser les fonctions spécifiées. Ce processus comprend quatre parties:

*a)* assemblage des sous-ensembles du matériel par câblage selon les plans du système;

*b)* assemblage des modules logiciels par un éditeur de liens;

module integration test shall be performed to show at the earliest stage of development that all modules interact correctly to perform the intended function.

Guidance for code verification activities is given in the software test specification.

6.2.3.1 *Software test specification*

The software test specification is one of the principal documents of the verification plan for the coding phase. This document shall be based upon a detailed examination of the software functional requirements and shall give detailed information on the tests to be performed addressing each of the components of the software (modules and their constituents).

The software test specification is based on a general description of the software being tested. The description is supplied by the design team.

The software test specification shall include:

*a)* the environment in which the tests are run;

*b)* the test procedures;

*c)* acceptance criteria i.e. a detailed definition of the criteria to be fulfilled in order to accept modules and major software components on subsystem and system level;

*d)* error detection and corrective action procedures;

*e)* a list of all documents that should be produced during the code verification phase.

6.2.3.2 *Software test report*

The software test report shall present the results of the test program described in the software test specification stating whether or not the software has achieved the performance requirements given in the software functional requirements. This document (F.M4) shall allow the complete diagnosis of design and performance deficiencies. It shall also include the resolution of all software deficiencies and test discrepancies discovered during the test.

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

*a)* hardware configuration used for the test;

*b)* storage medium used and access requirements of the final code tested;

*c)* input test listing;

*d)* output test listing;

*e)* additional data regarding timing, sequence of events, etc.;

*f)* conformance with acceptance criteria given in the test specification;

*g)* error incident log which describes the character of the error and the remedies taken by the design team.

## 7. Hardware/software integration

The process of hardware/software integration is the combining of verified hardware and software modules into a system that is capable of performing specified functions. This process consists of four parts:

*a)* assembling hardware modules by interconnecting wiring according to the system design drawings;

*b)* assembling software modules by a linkage processor;

*c)* chargement du logiciel dans le matériel;

*d)* vérification par le test que les spécifications d'interface matériel/logiciel sont satisfaites et que le logiciel est capable de fonctionner dans cet environnement matériel.

7.1 *Plan d'intégration du système*

Ce plan doit être préparé et explicité dans les spécifications d'intégration (A1), et vérifié par rapport aux spécifications du système de sûreté. Le plan d'intégration du système (F.M2) décrit les aspects relatifs à l'organisation et aux procédures pour l'intégration matériel/logiciel.

Ce plan doit spécifier les normes et les procédures à suivre lors de l'intégration matériel/logiciel et doit comprendre les dispositions du plan général d'assurance qualité appliquables.

Le plan d'intégration du système doit traiter de toutes les contraintes liées à une conception spécifique de matériel et de logiciel. Il doit inclure les exigences pour les procédures et les méthodes de contrôle comprenant:

— la gestion de configuration du système;

— l'intégration du système;

— le test du système intégré;

— le traitement des erreurs.

Lors du processus de test des modules matériels et logiciels, certains aspects de conception de ces modules peuvent être mieux testés au niveau du système intégré. Mais s'il n'est pas possible de tester complètement à ce niveau, toutes les spécifications fonctionnelles de conception relatives aux modules individuels doivent être testées par d'autres moyens. Toutes les interdépendances entre la vérification des modules individuels et la vérification du système intégré doivent être documentées dans le plan d'intégration du système.

7.2 *Relations entre l'intégration du système et le développement du matériel et du logiciel*

Le plan d'intégration du système doit être préparé suffisamment tôt dans le processus de développement pour permettre la prise en compte des exigences liées à l'intégration, lors de la conception du matériel et du logiciel.

7.3 *Gestion de configuration du système*

Le plan d'intégration du système doit établir une base de données de modules logiciels et matériels, en tant qu'outil de gestion de configuration du système. Cet outil doit permettre la gestion de versions de tous les modules matériels et logiciels utilisés dans le système.

Des procédures adéquates doivent être établies pour mettre en œuvre cette base de données et réaliser les tâches suivantes:

— la procédure doit s'assurer qu'aucune insertion ou révision de module puisse être introduite si elle n'a pas été vérifiée;

— la procédure doit s'assurer que l'intégration du système est réalisée en utilisant les bonnes versions des modules individuels;

— la procédure doit fournir l'indice d'utilisation de chaque module du système pour permettre l'évaluation de l'impact des révisions futures;

— la procédure doit être conduite de façon que le plan général d'assurance qualité soit respecté.

7.4 *Phase d'intégration du système*

Les procédures spécifiques d'intégration du matériel et du logiciel dépendent nécessairement de l'architecture du système et ne peuvent être incluses dans cette norme. Cependant,

*c)* loading the software into the hardware;

*d)* verifying by testing that the hardware/software interface requirements have been satisfied and that the software is capable of operating in that particular hardware environment.

7.1 *System integration plan*

The system integration plan shall be prepared and documented in the integration requirements (A1) and verified against the safety system requirements. The system integration plan (F.M2) describes the organizational and procedural aspects of the hardware/software integration.

This plan shall specify the standards and procedures to be followed in the hardware/software integration and shall document those provisions of the overall quality assurance plan that are applicable to the system integration.

The system integration plan shall discuss any constraints made by the specific design of the hardware and the software. The plan shall include the requirements for procedures and control methods covering:

— system configuration control;

— system integration;

— integrated system verification testing;

— error resolution.

In the process of verifying the individual hardware and software modules, certain aspects of the design of these modules may be better tested at the integrated system level. But if it is not feasible to test fully at this level, then all functional requirements of the individual module design shall be tested by other means. All interdependencies between the verification of the individual modules and the verification testing of the integrated system shall be documented in the system integration plan.

7.2 *Relationship of system integration to hardware and software development*

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

7.3 *System configuration control*

The system integration plan shall establish a library of software and hardware modules as the means for system configuration control. This library shall provide revision control for all hardware and software modules to be used in the system.

Adequate procedures shall be established to implement this library function and shall provide for the following tasks of the library function:

— the procedure shall ensure that no module or revision is entered into the library until it has been verified;

— the procedure shall ensure that the system integration is performed using the proper revision levels of the individual modules;

— the procedure shall provide for an index of the use of each module in the system so that the impact of future revisions to the modules can be adequately assessed;

— the procedure shall be conducted in such a way that the overall quality assurance plan is met.

7.4 *System integration phase*

The specific procedures for the hardware/software integration necessarily depend on the nature of the system design and cannot be included in this standard. However, such pro-

de telles procédures doivent être établies et documentées dans le plan d'intégration et doivent couvrir les activités suivantes:

— l'acquisition de modules adéquats à la bonne version à partir de la base de données conformément aux procédures du paragraphe 7.3;
— l'intégration des modules matériels dans le système (par exemple position des modules, adresse mémoire, adresse module, câblage);
— l'édition de liens des modules logiciels et le chargement du logiciel dans le matériel;
— le test fonctionnel préliminaire du système intégré (voir dispositions ci-après);
— la documentation du processus d'intégration et de la configuration du système qui seront soumis à vérification;
— la mise à disposition formelle du système intégré pour permettre le test de vérification.

Le test réalisé au cours de cette phase d'intégration matériel/logiciel est le test fonctionnel du concepteur du système. C'est l'équivalent au niveau système de la mise au point faite par un programmeur avant vérification. Toutes les erreurs détectées durant ce test, qui sont strictement des erreurs liées à l'intégration elle-même et qui n'affectent aucun document du projet, peuvent être corrigées sans rapport formel de l'erreur. Cependant, si la résolution d'une erreur exige une modification du logiciel vérifié ou de tout autre document de conception, cette erreur doit donner lieu à un rapport selon la procédure du paragraphe 7.6

7.5 *Vérification du système intégré*

La vérification du système est le processus consistant à déterminer si les modules matériels et logiciels vérifiés ont été correctement intégrés dans le système et si le matériel et le logiciel sont compatibles et réalisent un système conformément à la spécification d'intégration.

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

Les jeux d'essais choisis pour la vérification du système doivent activer toutes les interfaces des modules ainsi que les fonctions de base des modules eux-mêmes. La simulation d'une partie du système ou de ses interfaces doit être justifiée dans le plan d'intégration du système.

La vérification du système intégré doit être réalisée selon un plan de test formalisé. Ce plan doit identifier les tests pour chaque exigence relative à l'interface du système programmé. Le test du système intégré doit être réalisé et les résultats évalués par un personnel connaissant les spécifications du système mais n'ayant pas participé à son développement.

Le niveau d'indépendance existant entre les concepteurs et les vérificateurs du système doit être suffisant pour amener toute communication entre les deux parties — soit pour demande de clarification, soit pour constat d'erreur — à se faire uniquement de manière formelle par écrit, afin de pouvoir être vérifié par des personnes qui ne sont pas engagées dans le développement du système.

Les matériels utilisés pour la vérification du système doivent être étalonnés. Des mesures d'assurance qualité doivent être établies pour les outils logiciels utilisés dans la vérification, en rapport avec l'importance de ces outils pour cette action.

7.6 *Procédures de résolution des erreurs*

Des procédures spécifiques pour rendre compte des erreurs et de leur correction doivent être établies en tant qu'éléments du plan d'intégration du système.

Ces procédures doivent être appliquées à toutes les erreurs trouvées au cours de la vérification du système ainsi qu'à celles trouvées pendant les tests fonctionnels ayant nécessité des modifications du logiciel vérifié ou des documents de conception du système. La procédure

cedures shall be established and documented by the integration plan, and shall cover the following activities:

— the acquisition of the proper modules using the library and procedures of Sub-clause 7.3;

— the integration of the hardware modules into the system (e.g. module position, memory address, selection, interconnection wiring);

— the linkage of software modules and the loading of the software into the hardware;

— the preliminary functional test of the integrated system function (see requirements below);

— the documentation of the integration process and the system configuration that will be subjected to verification testing;

— the formal release of the integrated system to verification testing.

The testing performed in this phase of the hardware/software integration is the designer's functional test of the system. It is the system analogy to the debugging done by a programmer before he releases his software to be verified. Any errors detected during the test that are strictly mistakes in the integration itself, and do not affect any project document, may be corrected without formal error report. However, if the resolution of any error requires a change to verified software or any design document, that error shall be reported according to the procedures established by Sub-clause 7.6.

7.5 *Integrated system verification*

The system verification is the process of determining whether or not the verified hardware and software modules have been properly integrated into the system and that the hardware and software are compatible and perform as a system as required by the integration requirements.

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

The test cases selected for system verification shall exercise all module interfaces as well as the basic operation of the modules themselves. Justification shall be provided in the system integration plan for the simulation of any part of the system or its interfaces.

Integrated system verification shall be conducted in accordance with a formal test plan. The plan shall identify the tests for each computer system interface requirement. The integrated system test shall be developed and the test results evaluated by individuals who are familiar with the system specification and who did not participate in the development of the system.

The level of independence provided between the designers and the system verifiers shall be sufficient to ensure that all communication between the two parties, whether for clarification or error reporting, is only conducted formally in writing at a level of detail which may be audited by persons not involved in the development of the system.

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

7.6 *Error resolution procedures*

Specific procedures for the reporting and resolution of errors shall be prepared as a part of the system integration plan.

These procedures shall apply to all errors found during the system verification as well as those found during the integration functional test that require changes to verified software or system design documents. The procedure shall ensure that any required reverification of

doit donner l'assurance qu'une revérification des modules matériels et/ou logiciels est réalisée et que la nouvelle version des modules a été prise en compte dans la base de données selon les procédures du paragraphe 7.3.

Une évaluation de chaque erreur relevée doit être faite afin de déterminer si elle aurait pu être détectée dans une phase précédente de vérification (par exemple lors du test d'un module). Si cela est le cas, une investigation de cette phase précédente doit être conduite pour déterminer s'il existe une défaillance systématique dans la vérification.

7.7 *Vérification du système intégré, compte rendu de test*

Les résultats de la vérification du système intégré doivent être inclus dans un compte rendu de test (F.M5). Celui-ci doit identifier le matériel et le logiciel utilisés, l'équipement d'essai utilisé et son réglage, la simulation de parties du système ou des interfaces, les résultats incorrects rencontrés en cours de vérification avec les actions correctives entreprises conformément au paragraphe 7.6. Les résultats d'essais doivent être consignés sous une forme permettant l'audit par des personnes qui ne sont pas directement impliquées dans le programme d'essai.

La correction de toutes les erreurs consignées et les résultats de l'évaluation qui en découlent doivent être documentés avec suffisamment de détails et sous une forme permettant l'audit par des personnes qui ne sont pas directement impliquées dans le développement du système et le programme de vérification.

## 8. Validation du système programmé

Des essais doivent être réalisés pour valider le matériel et le logiciel en tant que système par rapport aux spécifications du système de sûreté que doit satisfaire le système programmé. La validation du système programmé doit être conduite en accord avec un plan formalisé de validation (F.M5). Ce plan doit expliciter la validation dans les conditions statiques et dynamiques.

Le système programmé doit être sollicité par la simulation statique et dynamique des signaux d'entrée présents durant le fonctionnement normal, ainsi que lors des incidents d'exploitation et des conditions d'accident entraînant une action du système programmé. Il est recommandé que chaque fonction de sécurité du réacteur soit contrôlée par des essais représentatifs de chaque variable intervenant dans l'arrêt d'urgence ou la protection, prise isolément et en combinaison.

Il est recommandé que les essais:

— couvrent la plage de variation des signaux et les gammes des variables élaborées ou calculées d'une manière pleinement représentative;

— couvrent les circuits de vote et les autres circuits logiques le plus complètement possible;

— soient effectués pour tous les signaux d'arrêt d'urgence ou de protection dans la configuration finale du système;

— donnent l'assurance que la précision et les temps de réponse sont respectés et qu'une action correcte est effectuée dans le cas d'une panne quelconque de l'équipement ou d'une combinaison de pannes.

De plus, les valeurs des signaux d'entrée requis, les signaux de sortie attendus et les critères d'acceptation doivent être spécifiés dans le plan de validation. Celui-ci doit être mis en œuvre et les résultats de la validation évalués par des personnes qui n'ont pas participé à la phase de développement décrite dans l'article 5.

Les équipements utilisés pour la validation doivent être étalonnés. Les outils matériels et logiciels utilisés pour cette validation ne nécessitent pas de vérification spéciale. Il convient cependant de pouvoir montrer qu'ils remplissent leur rôle.

hardware or software modules is performed and that the revision to the modules is carried out through the library procedures of Sub-clause 7.3.

An evaluation of each error reported shall be made to determine whether the error was of such a nature that it should have been detected at an earlier phase (such as module testing) of the verification. If this is found to be the case, then an investigation of that earlier stage of the verification shall be conducted to determine whether any systematic deficiency of the verification exists.

7.7 *Integrated system verification test report*

The results of the integrated system verification shall be documented in a test report (F.M5). This report shall identify the hardware and software used, the test equipment used and its calibration, the simulation of system or interface components, and any test results discrepancy found along with the corrective actions taken according to Sub-clause 7.6. The test results shall be retained in a form that is auditable by persons not directly engaged in the test program.

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

## 8. Computer system validation

Testing shall be performed to validate the hardware and software as a system in accordance with the safety systems requirements to be satisfied by the computer system. The computer system validation shall be conducted in accordance with a formal validation plan (F.M5). The plan shall identify the validation for static and dynamic cases.

The computer system shall be exercised by static and dynamic simulation of input signals present during normal operation, anticipated operational occurrences and accident conditions requiring computer system action. Each reactor safety function should be confirmed by representative tests of each trip or protection parameter singly and in combination.

It is recommended that the tests:

— cover all signal ranges, and the ranges of computed or calculated parameters in a fully representative manner;

— cover the voting and other logic and logic combinations comprehensively;

— be made for all trip or protective signals in the final assembly configuration;

— ensure that accuracy and response times are confirmed, and that correct action is taken for any equipment failure or failure combination.

In addition, the required input signals and their values, the anticipated output signals and the acceptance criteria shall be stated in the validation plan. The computer system validation plan shall be developed and the results of the validation evaluated by individuals who did not participate in the development phase of Clause 5.

Equipment used for validation shall be calibrated. Hardware and software tools used for this validation need no special verification. They should, however, be shown to be suited to their purpose.

8.1 *Compte rendu de validation du système programmé*

Les résultats de la validation du système programmé doivent être inclus dans un compte rendu (F.M6). Ce compte rendu doit identifier le matériel et le logiciel utilisés, les outils employés et leur étalonnage, les modèles de simulation utilisés, les anomalies et les actions correctives entreprises selon la procédure de modification de l'article 9.

Le compte rendu doit résumer les résultats de la validation du système programmé et doit montrer comment le système correspond aux spécifications.

Les résultats doivent être présentés de sorte qu'ils puissent être vérifiés par les personnes qui n'ont pas été directement impliquées dans la validation. Il est recommandé de citer dans le compte rendu les outils logiciels utilisés dans le processus de validation. Toute simulation de la centrale et de ses systèmes, utilisée pour la validation, doit être mentionnée.

## 9. Maintenance et modification

Une procédure formelle de gestion des modifications doit être établie, incluant la vérification et la validation.

9.1 *Procédure et demande de modification*

Pour qu'une modification puisse être considérée, la procédure suivante doit être respectée.

9.1.1 Toute demande de modification doit être établie en indiquant:

— sa raison d'être;
— son but;
— son domaine fonctionnel;
— l'identité du demandeur.

La modification peut être demandée pendant la phase de développement à la suite d'un changement dans les spécifications fonctionnelles causé par:

— une modification fonctionnelle;
— une évolution technologique;
— un changement dans les conditions d'exploitation.

La modification peut être demandée à la suite d'un compte rendu d'anomalie consécutif à des essais. La modification due à ces causes peut occasionner un changement dans l'analyse fonctionnelle du logiciel. La modification peut aussi être demandée après la livraison de l'équipement, il s'agit alors de maintenance de logiciel.

9.1.2 La demande de modification doit être évaluée de manière indépendante, il peut en résulter:

— le rejet de la demande; dans ce cas, elle est retournée avec des commentaires;
— l'acceptation d'une demande de modification mineure et sa mise en œuvre immédiate si elle intervient lors du développement initial du logiciel et cela après analyse;
— la demande d'une analyse détaillée des incidences conduisant à l'élaboration d'un compte rendu d'analyse des incidences sur le logiciel. Ce compte rendu doit être rédigé par un spécialiste du logiciel du système;
— l'acceptation de la demande.

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

— la faisabilité;
— les conséquences sur le matériel (par exemple extension mémoire) et/ou sur d'autres équipements (par exemple équipements de tests). Dans ce cas, la demande correspondante de modification doit être explicitée au niveau équipement;
— les conséquences sur le logiciel, comprenant une liste des modules affectés;

8.1 *Computer system validation report*

The results of the computer system validation shall be documented in a report (F.M6). This report shall identify the hardware and software used, the equipment used and its calibration, the simulation models used, and any discrepancies and corrective actions according to the modification procedure of Clause 9.

The report shall summarize the results of the computer system validation and shall assess the system compliance with all requirements.

The results shall be retained in a form that is auditable by persons not directly engaged in the validation. Software tools used in the validation process should be identified as an item in the validation report. Simulations of the plant and its systems used for the validation shall be documented.

## 9. Maintenance and modification

A formal modification control procedure shall be established including verification and validation.

9.1 *Modification request procedure*

For a modification to be considered, the following procedure shall be followed.

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

- reason for request;
- aim;
- functional scope;
- originator.

The modification may be requested during the development phase because of a change of functional requirements caused by:

- functional modification;
- technological evolution;
- changes of operating conditions.

The modification may be requested because of an anomaly report as a result of tests. Modification for these reasons may cause a change in the software requirements document. The modification may also be requested after delivery; it is then considered to be maintenance.

9.1.2 The modification request shall be evaluated independently, the result being either:

- to reject the request; in this case it is sent back with comments;
- to approve a request of minor importance and impact immediately if it is made during initial development after analysis;
- to require a detailed analysis resulting in a software modification analysis report. This report shall be written by software personnel knowledgeable in the system software;
- to accept the request.

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

- technical feasibility;
- impact upon hardware (e.g. memory extension) or upon other equipment (e.g. test systems) in which case the request for modification addressing this impact area must be documented for the equipment;
- impact upon software including a list of affected modules;

— les conséquences sur les performances (temps de calcul, précision, etc.);

— l'effort nécessaire de vérification et de validation; l'analyse de la vérification ultérieure nécessaire du logiciel doit être explicitée de manière à permettre un audit;

— la liste des documents à revoir.

9.1.4 La demande d'une modification de logiciel reste en suspens jusqu'à ce qu'une décision soit prise pour:

— l'accepter (immédiatement ou après l'examen du compte rendu d'analyse des incidences) et la mettre en œuvre, ou

— la rejeter en le justifiant.

9.2 *Procédure d'exécution d'une modification*

9.2.1 Pour les modifications sur le site, il convient que le fournisseur du logiciel ait accès à une configuration d'essai identique au système réel dans tous ses aspects significatifs (matériel, compilateur, outils de tests, simulateur de la centrale, etc.) pour assurer la validité des modifications.

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

— un changement de spécification fonctionnelle exige un nouvel examen de toute la procédure de développement du logiciel, en ce qui concerne la partie du système touchée par la modification;

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

— les modifications doivent être traitées selon les règles figurant dans l'article 5.

9.2.3 Après réalisation d'une modification, le processus complet ou partiel de vérification et de validation décrit au paragraphe 6.1 et dans l'article 8 doit être effectué selon l'analyse des incidences de la modification (voir paragraphe 9.1.3).

9.2.4 Tous les documents concernés par la modification doivent être corrigés et faire référence à l'identification de la demande de modification logicielle. Un compte rendu de modification logicielle doit résumer toutes les actions faites à son sujet.

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

9.3 *Maintenance*

Les raisons d'une maintenance peuvent être:

— un compte rendu d'anomalie;

— une modification des spécifications fonctionnelles après livraison;

— une évolution technologique;

— un changement dans les conditions d'exploitation.

9.3.1 En cas d'anomalie, un compte rendu d'anomalie doit être établi donnant les symptômes, l'environnement et l'état du système lors de la découverte de l'incident, ainsi que les causes suspectées. La correction de l'anomalie requiert l'élaboration d'une demande de modification qui doit suivre la procédure décrite au paragraphe 9.1.

9.3.2 Dans le cas de modification des spécifications fonctionnelles ou des conditions d'exploitation, tout le processus de développement du logiciel doit être repris, en ce qui concerne la partie du système touchée par la modification.

9.3.3 De nouvelles exigences et possibilités du matériel doivent être évaluées en ce qui concerne leur impact potentiel sur le logiciel. Pour cette évaluation, il convient de prendre en considération les caractéristiques du matériel examinées lors de la conception initiale du logiciel. S'il

— impact upon performance (including speed, accuracy, etc.);
— necessary effort for verification and validation; the analysis of the software reverification needed shall be documented in an auditable form;
— the set of documents to be reviewed.

9.1.4 The software modification request is pending until the decision is made:

— to accept it (immediately, or after examination of the software modification analysis report) and to execute it, or
— to reject it and justify the rejection.

9.2 *Procedure for executing a modification*

9.2.1 For modifications on site, the software supplier should have access to a test configuration which is identical to the real system in all relevant aspects (including installed machine, translator, testing tools, plant simulator, etc.) to ensure the validity of the modifications.

9.2.2 A modification procedure depends upon the level at which it is introduced:

— for a change of the software functional specification, the whole software development process for any part of the system impacted by the change shall be re-examined;
— a change during the development phase shall be reviewed in terms of its potential impact upon corresponding lower levels;
— the modification shall be carried out according to the rules given in Clause 5.

9.2.3 After implementation of the modification, the whole or part of the verification and validation process described in Sub-clause 6.1 and Clause 8, shall be performed again according to the software modification analysis (see Sub-clause 9.1.3).

9.2.4 All the documents affected by the modification shall be corrected and refer to the identification of the software modification request. A software modification report shall sum up all the actions made for modification purposes.

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

9.3 *Maintenance*

The reasons for maintenance include:

— an anomaly report;
— a functional requirements change after delivery;
— technological evolution;
— a change in operating conditions.

9.3.1 In case of an anomaly, an anomaly report shall be written giving the symptoms, the system environment and system status at the time at which the anomaly was discovered and the suspected causes. Anomaly correction requires the generation of a software modification request, the execution of which shall follow the procedure described in Sub-clause 9.1.

9.3.2 In case of a change in software functional requirements or in operating conditions, the whole software development process shall be re-examined for that part of the system impacted by the change.

9.3.3 Any new hardware requirements and capabilities shall be examined with respect to their potential impact on the software systems. This evaluation should include all hardware considerations reviewed in the original software design. If it can be shown that the new system

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

9.3.4 Dans tous les cas, après implantation de la modification sur l'équipement du site, on doit établir un «document site» donnant des informations sur la date de l'implantation et les résultats des observations effectuées. Ce document doit être classé dans l'historique de modification du logiciel relatif au projet.

## 10. Exploitation

Le terme exploitation se rapporte aux activités concernant la mise à disposition et le fonctionnement du système. Il recouvre le système programmé et le personnel d'exploitation.

### 10.1 *Système programmé*

#### 10.1.1 *Essais de mise à disposition*

Un programme d'essai doit être fourni pour vérifier l'intégrité du système de sûreté programmé, en ce qui concerne les temps de réponse, les réglages, les opérations fonctionnelles et les interactions avec les autres systèmes.

Un programme d'essais de mise à disposition comprenant les critères d'acceptation, les jeux d'essais et l'environnement des essais doit être établi (F.M6).

Un compte rendu de mise à disposition présentant les résultats d'essai et attestant le respect des critères d'essai doit être établi (F.M7).

#### 10.1.2 *Interaction homme-machine*

Pour permettre l'interaction homme-machine, des fonctions de commande opérateur du système programmé peuvent être requises. Les périphériques utilisés à cet effet ne doivent pas fournir la possibilité d'altérer le programme logique mémorisé du système programmé.

Une procédure appropriée et/ou un verrouillage doivent être établis pour prévenir le changement par inadvertance par l'opérateur de paramètres qui peuvent affecter les points de consigne du système de sécurité.

Toutes les actions réalisées par l'opérateur à travers les commandes opérateur doivent être consignées par des moyens appropriés.

### 10.2 *Formation des opérateurs*

#### 10.2.1 *Programme de formation*

Afin d'obtenir une exploitation sûre de la centrale, le comportement de l'exploitant est aussi important que la fiabilité de l'équipement. Un programme de formation des opérateurs doit donc être fourni à la fois aux exploitants de la centrale et aux spécialistes de l'instrumentation et de la commande, formation en rapport avec la complexité des systèmes de protection mis en œuvre. Le programme de formation doit traiter des conditions de fonctionnement aussi bien normales qu'anormales. Toutes les interfaces du système de protection programmé avec les exploitants doivent être incluses dans la formation. Il est recommandé qu'une formation spécifique à la reconnaissance des anomalies matériel et logiciel soit aussi incluse dans le programme.

#### 10.2.2 *Plan*

Un plan de formation homogène avec les principes du programme de formation doit être établi (F.M6).

Un manuel utilisateur doit être disponible pour l'exploitant du réacteur. Il convient que ce manuel définisse chaque élément d'interface opérateur. Chaque fonction de chaque dispositif doit être expliquée et illustrée selon sa complexité.

does not impact the software requirements, a simplified procedure may be used to implement the modification either at the design or coding phase.

9.3.4 In all cases, after implementation of the modification upon the in-the-field equipment, a field document shall be issued which gives the date of the implementation and the result of the specified observations. This document shall be filed in the software modification control history for the project.

## 10. Operation

The term 'operation' is used to describe all activities regarding the commissioning and the operation of the system. It concerns the computer system and the operating personnel.

### 10.1 *Computer system*

#### 10.1.1 *Commissioning tests*

A test program shall be provided to verify the integrity of the installed computer-based safety system with respect to response, calibration, functional operation and interaction with other systems.

A commissioning test plan, consisting of acceptance criteria, test cases and test environment shall be established (F.M6).

A commissioning test report presenting test results and conformity to acceptance criteria shall be established (F.M7).

#### 10.1.2 *Man–machine interaction*

To allow man–machine interaction, operator control of computer functions may be required. These devices shall not provide the capability to alter the stored computer program logic.

An appropriate procedure and/or locking device shall be established to prevent the operator inadvertently changing parameters that can affect the set points of the safety system.

All the actions performed by the operator through the operator control shall be documented by suitable means.

### 10.2 *Operator training*

#### 10.2.1 *Training program*

In order to achieve safe plant operation, operator behaviour is as important as equipment reliability. Therefore an operator training program shall be provided both for plant operators and instrumentation and control specialists consistent with the complexity of the protective functions implemented. The training program shall provide for normal and abnormal reactor operation. All operator interfaces of the computer system shall be included in the training program. Specific training in the recognition of hardware and software abnormalities should also be included in the program.

#### 10.2.2 *Training plan*

A training plan consistent with the principles of the training program shall be established (F.M6).

A user manual shall be provided for the reactor operator (F.M6). The user manual should define each operator interface device. Each function of each device shall be explained and illustrated in accordance with its complexity.

10.2.3 *Système de formation*

La formation des exploitants doit être conduite sur un système de formation qui est équivalent au véritable système matériel/logiciel. La simulation du réacteur pour ce système doit être réalisée par un système de test capable de simuler les conditions normales et anormales du réacteur.

Il convient de plus que ce système permette la formation à chaque interface opérateur. Un simulateur programmé peut être utilisé à cet effet.

10.3 *Spécifications du logiciel relatives aux essais périodiques*

Les principes généraux de la Publication 671 de la CEI: Essais périodiques et surveillance du système de protection des réacteurs nucléaires, sont applicables pour les tests périodiques.

De plus, les exigences suivantes non couvertes par le paragraphe 4.8 doivent être respectées.

10.3.1 *Programme d'essais*

Un programme pour les essais périodiques des systèmes de sûreté doit être défini et inclure les essais fonctionnels applicables, les appareils, les tests, la vérification des réglages et des temps de réponse.

10.3.2 *Spécification des essais*

Les essais doivent vérifier les aptitudes fonctionnelles de base du logiciel, comprenant:

— toutes les fonctions de sûreté de base devant être périodiquement testées;

— des essais spéciaux prévus pour détecter les défauts qui ne peuvent être décelés par les dispositifs d'auto-test incorporés dans le système de sûreté ou par les indicateurs d'anomalies ou d'alarmes;

— les fonctions importantes ne se rapportant pas à la sûreté et qu'il convient de tester périodiquement. Il convient d'effectuer un essai capable de détecter une dégradation de ces fonctions.

10.3.3 *Exigences relatives aux systèmes auxiliaires*

Les logiciels des systèmes auxiliaires utilisés pour les essais périodiques des systèmes de sûreté ne sont pas tenus de satisfaire à toutes les exigences du logiciel de sûreté. Il est recommandé que leur qualité corresponde à celle des outils de vérification mentionnée dans l'article 8. Il convient que les données des tests soient choisies selon le tableau E4.2 de l'annexe E.

10.2.3 *Training system*

Operator training shall be conducted on a training system which is equivalent to the actual hardware/software system. The plant stimulus to this system shall be provided by a test system capable of simulating normal and abnormal reactor conditions.

Furthermore the training system should provide training facilities for each operator interface device. A computerized simulator may be used for this purpose.

10.3 *Software requirements for periodic testing*

For periodic testing, the general principles of IEC Publication 671: Periodic Tests and Monitoring of the Protection System of Nuclear Reactors, are applicable.

In addition, the following requirements not covered by Sub-clause 4.8 shall be met.

10.3.1 *Test program*

A program for periodic tests of the safety systems including applicable functional tests, instruments, checks, verification of proper calibration and response time tests shall be defined.

10.3.2 *Test requirements*

The tests shall verify the basic functional capabilities of the software periodically including:

— test of all basic safety functions;

— special testing to be used to detect failures that cannot be revealed by self-checking provisions of the safety systems or by alarm or anomaly indications;

— tests of the major non-safety related functions. A test should detect degradations of those functions.

10.3.3 *Auxiliary device requirements*

The software dedicated to auxiliary devices for periodic testing of the safety systems need not be designed to comply with all software safety requirements. Their quality should correspond to the quality of verification tools as mentioned in Clause 8. Test data should be selected according to Table E4.2 of Appendix E.

# ANNEXE A

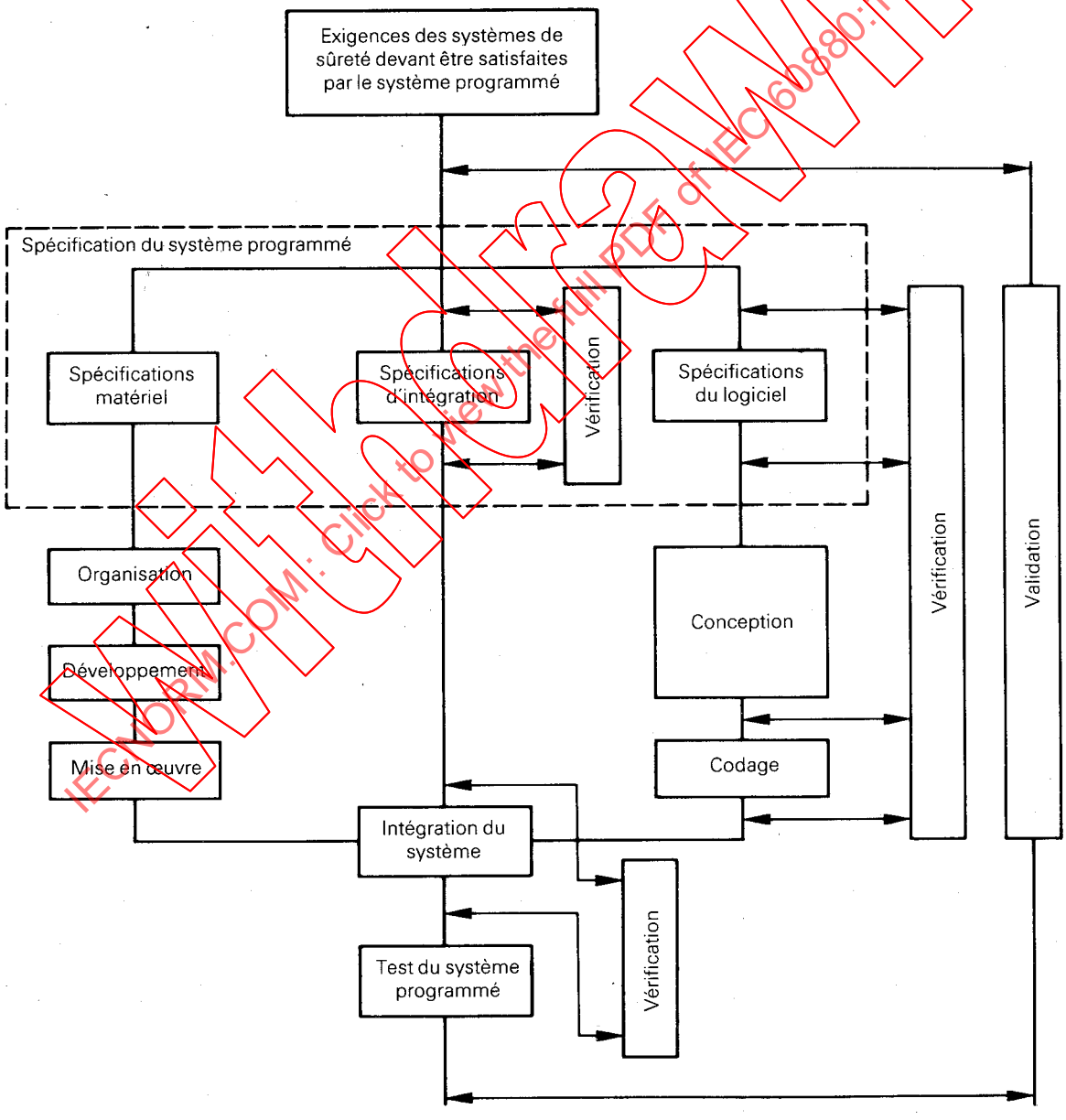
## SYNOPTIQUE DU CYCLE DE DÉVELOPPEMENT DU SYSTÈME ET DÉTAILS DES SPÉCIFICATIONS DU LOGICIEL

### INTRODUCTION

Le cycle de vie concernant le développement du système est illustré par la figure de l'article A1, montrant les relations entre les activités de conception et de réalisation et les activités de vérification et de validation.

Les détails des prescriptions du système sont donnés dans l'article A2.

### A1. **Synoptique du cycle de développement d'un système**



# APPENDIX A

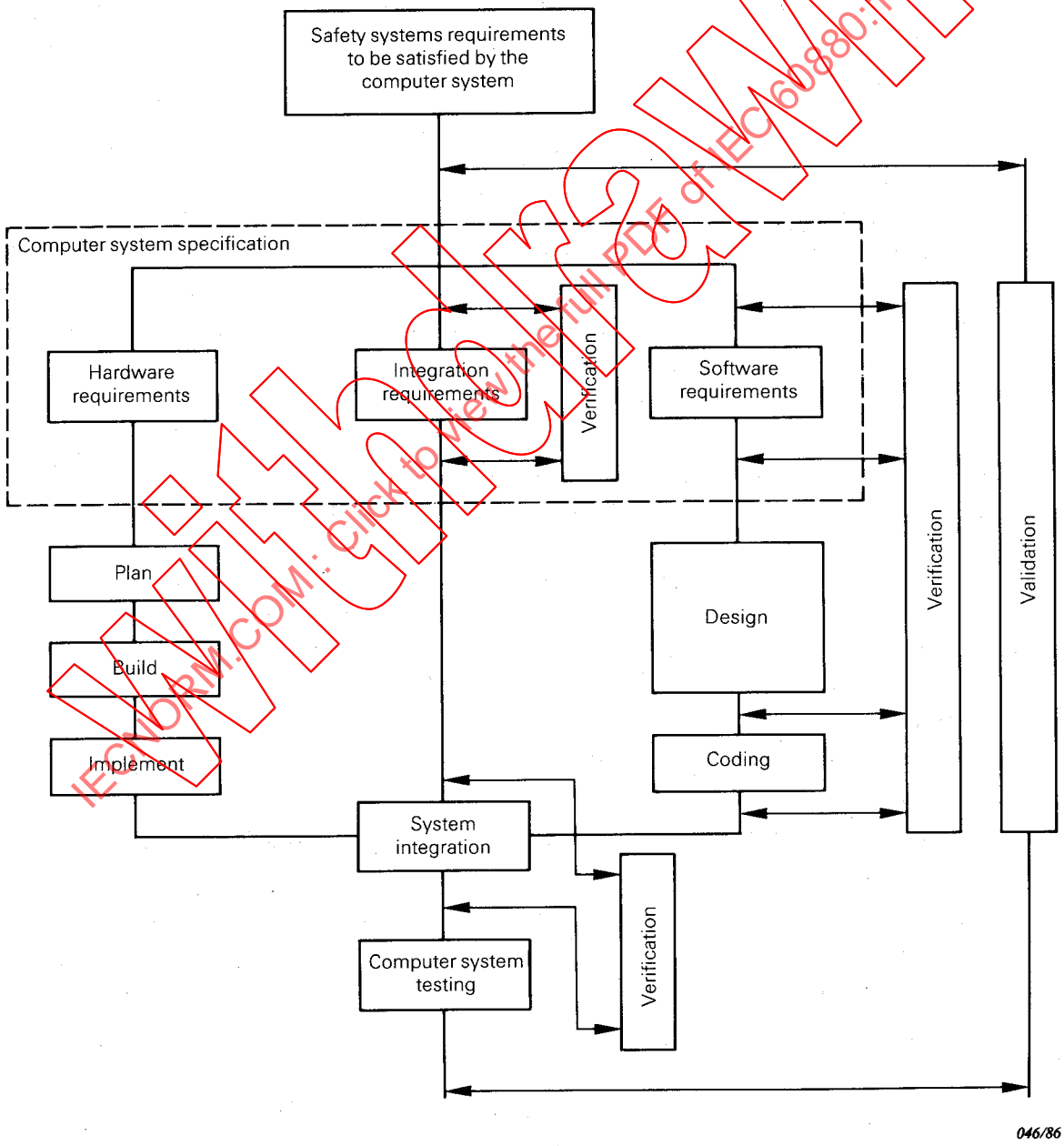
## SYSTEM DEVELOPMENT LIFE CYCLE AND DETAILS OF SOFTWARE REQUIREMENTS

### INTRODUCTION

The system development life cycle is illustrated by the figure of Clause A1, showing the relationship of the designed implementation activities to the verification and validation activities.

The details of system requirements are described in Clause A2.

### A1. System development life cycle



## A2. Détails des spécifications du logiciel

### A2.1 *Spécifications du système programmé*

A2.1.1 L'objet de la spécification du système programmé est:

- d'énoncer l'ensemble des finalités du logiciel avec une définition des limites explicites et l'exposé de ce qu'il ne doit pas faire;
- de lister les volumes et les flots attendus du logiciel et énoncer explicitement ceux qui sont seulement des objectifs à atteindre et ceux qui sont absolument nécessaires pour le système programmé;
- d'énoncer les résultats qualitatifs attendus du système (par exemple précision requise), séparer les objectifs précis des objectifs approximatifs;
- d'énoncer les objectifs relatifs à la stratégie, la tradition, la pratique industrielle et les consignes du management;
- de classer les objectifs selon leur priorité.

A2.1.2 La spécification décrit les fonctions destinées à prévenir une émission de radioactivité sous des conditions accidentelles, ainsi que les conditions dans lesquelles elles sont initiées. Il convient d'inclure dans le document les fonctions et leurs conditions d'initiation pour:

- l'arrêt d'urgence;
- les autres fonctions de sûreté, les calculs nécessaires, leur environnement physique;
- les fonctions destinées à prévenir une détérioration de la centrale, les calculs nécessaires, l'environnement physique;
- les fonctions qui ont une influence mineure sur la sûreté.

Il convient que la description établisse comment ces fonctions interviennent dans la conception générale de la centrale et dans la fonction de commande exécutée par les autres systèmes de conduite.

### A2.2 *Configuration du système programmé*

Il convient de considérer les exigences quantitatives en matière de fiabilité pour les actions individuelles, les fonctions et les séquences d'action ainsi que la manière de traiter les perturbations particulières, les transitoires et les situations accidentelles, en tenant compte des classes de sûreté correspondantes.

Le matériel et le logiciel doivent respecter le critère de défaillance unique pour le matériel.

Les facteurs suivants sont à considérer pour éviter les défaillances de mode commun et pour améliorer la fiabilité du système:

- défense en profondeur;
- marche dégradée;
- traitement des défaillances en général;
- diversité fonctionnelle et, si nécessaire, diversité du logiciel;
- séparation géographique et modularisation, découplage, séparation logique.

La conception doit tenir compte de:

- l'interaction entre les différents blocs et les unités séparées;
- la hiérarchie des fonctions;
- le critère de structuration du système;
- la configuration du logiciel du système;
- les principes de conception des interfaces;
- les possibilités d'adaptation et de modification du système et les capacités d'extension des interfaces.

## A2. Details of software requirements

### A2.1 *Computer system specification*

A2.1.1 The purpose of the computer system specification is to:

- state the overall purpose of the software with a definition of explicit limits and with a statement of what the software must not do;
- list volume and throughput expectations of the software and state explicitly which are merely target figures or goals and which are absolutely necessary for the software system;
- state qualitative expectations of the system (e.g. required accuracy) separated into absolute and approximate objectives;
- state objectives relative to policy, tradition, industrial practices and management directives;
- classify the objectives according to priority.

A2.1.2 The specification describes functions dedicated to the prevention of the release of radioactivity under accident conditions, and their initiation conditions. The document should include function and initiation conditions for:

- emergency shut down;
- other safety functions, the necessary calculations, their physical background;
- functions that prevent plant damage, the necessary calculations, their physical background;
- functions that have minor safety relevance.

The description should state the relationship of these functions to the total plant concept and the control function executed by other systems.

### A2.2 *Computer system configuration*

The quantitative reliability requirements for individual actions, functions and sequences of action or for handling particular disturbances, transients and accident situations should be considered, taking account of classes of safety relevance.

The hardware and software shall comply with the single failure criterion for the hardware.

To avoid common-mode failure effects and to increase system reliability, the following factors are relevant:

- defence-in-depth;
- graceful degradation;
- management of failures in general;
- functional diversity and if necessary software diversity;
- spatial separation and modularization, decoupling, logical separation.

The design shall take account of:

- interaction between the different blocks and separate units;
- hierarchy of functions;
- system structuring criteria;
- system software configuration;
- interface design principles;
- system adaptability, changeability and the expansion capability of interfaces.

A2.3 *Dialogue homme-machine*

A2.3.1 Les principes de base comprennent les directives suivantes:

— aucune panne du système programmé ne doit inhiber les commandes manuelles appropriées;

— les interfaces homme-machine doivent être identifiées;

— les procédures formelles et une syntaxe claire doivent être définies pour les interactions entre l'opérateur et le système programmé;

— les procédures manuelles qui sont susceptibles de provoquer des blocages ou d'être la cause de problèmes indus, telles que les opérations manuelles pouvant introduire des erreurs, doivent être identifiées;

— le système programmé doit tester toutes les entrées manuelles tant sur le respect de la syntaxe que sur la plausibilité de la sémantique;

— il convient que toute commande anormale de l'opérateur soit signalée;

— les facteurs d'ergonomie significatifs sur les performances de l'opérateur doivent être déduits des interfaces identifiées et des problèmes d'implantation;

— les procédures manuelles qui requièrent une formation significative ou des aides élaborées doivent être identifiées;

— le système complet doit être examiné et analysé pour s'assurer que les parties automatisées du système sont conçues pour assister les parties manuelles, et non l'inverse.

A2.3.2 Exigences des actions manuelles sur le système programmé:

— l'opérateur doit être capable de tester les fonctions de base du système en service;

— aucune modification du logiciel ne peut être faite durant le fonctionnement du réacteur, sans une procédure strictement contrôlée;

— les procédures pour introduire, modifier et afficher des paramètres doivent être définies avec précision, d'une manière simple et facile à comprendre;

— les techniques de «menu» appropriées doivent être fournies;

— les interactions manuelles ne doivent pas retarder les actions de sécurité au-delà de limites spécifiées.

A2.3.3 Exigences pour les affichages du système programmé:

— le système programmé ne doit pas induire en erreur l'opérateur;

— le système programmé doit signaler ses propres défauts et pannes à l'opérateur;

— l'utilisation de couleurs, de signaux clignotants, de signaux d'alerte, etc., doit respecter une convention claire;

— les informations affichées et leur présentation doivent suivre des principes ergonomiques.

A2.4 *Eléments relatifs aux interfaces*

Il convient d'examiner les rubriques suivantes:

— indépendance et découplage;

— prévention des erreurs de transmission et de propagation des pannes;

— contrôle strict des interactions entre les systèmes ou sous-systèmes de différents niveaux de sûreté entre eux et avec les autres systèmes:

- mécanismes et protocole de communication;
- test des pannes;
- format des messages;
- flot de données;

A2.3 *Man–machine dialogue*

A2.3.1 The basic principles include:

- no computer system failure shall inhibit appropriate human control actions;
- man–machine interfaces shall be identified;
- formal procedures and a clear syntax for human interactions with the computer system shall be defined;
- the human procedures within the system which are likely to be bottlenecks or are likely to cause undue problems with the system such as human operation which could enter errors shall be identified;
- the computer system shall check any manual input for syntactic correctness and semantic plausibility;
- inappropriate operator control actions should be signalled;
- from identified interfaces and problems areas, a description of human engineering factors which could have a significant effect on human performance shall be derived;
- the human procedures that will require significant training or elaborate aids shall be identified;
- the entire system shall be reviewed and analysed to ensure that the automated portions of the system are designed to aid and support the human portions and not the reverse.

A2.3.2 The requirements for human actions to the computer system include:

- the operator shall be able to check basic system functions on-line;
- no modification for software shall be made during plant operation, without following a strictly controlled procedure;
- the procedures for introduction, modification and display of parameters shall be exactly defined, simple and easy to comprehend;
- appropriate "menu" techniques shall be provided;
- manual interaction shall not delay basic safety actions beyond specified limits.

A2.3.3 The requirements for computer system displays include:

- the computer system shall not mislead the operator;
- the computer system shall report its own defects and failures to the operator;
- the use of colours, flashing signals, alerting signals etc. shall follow a clear scheme;
- the information displayed and its format shall follow ergonomic principles.

A2.4 *Items relating to interfaces*

The following items should be considered:

- independency and decoupling;
- prevention of transmission failures and of failure propagation;
- strict control of interactions between systems or subsystems of different safety relevance, including:
  - handshake mechanisms, communication protocols,
  - failure checks,
  - message formats,
  - message throughput;

— contraintes de ressources;

— références appropriées à des documents plus détaillés.

L'incorporation de systèmes programmés dans les systèmes de sûreté et leurs relations avec les systèmes d'actionneurs de sûreté doivent être décrites avec précision.

A2.5 *Descriptions des fonctions du système*

A2.5.1 La liste complète de ces fonctions doit être donnée. La quantité d'éléments descriptifs nécessaires dépend de la complexité de la fonction. Ce sont au minimum:

— le but de chaque fonction;

— son lien avec la fiabilité du système;

— les variables d'entrées/sorties.

A2.5.2 Toutes les variables nécessaires à l'exécution de la fonction doivent être décrites en ce qui concerne les éléments suivants:

— domaine d'entrée et domaine de sortie;

— relations entre la représentation interne de la variable et sa valeur correspondante en unités physiques;

— précision d'entrée et limite de bruit;

— précision de sortie.

A2.5.3 Une importance particulière doit être donnée à la description des fonctions en liaison avec la sûreté, à savoir celles requérant une conception logicielle complexe, ainsi que celles destinées à commander ou gouverner des mécanismes physiques complexes.

A2.5.4 Une description détaillée de toutes les fonctions du système doit être fournie, indiquant les relations entre elles et avec le système d'entrées/sorties. Des diagrammes décrivant les relations fonctionnelles et entrées/sorties doivent être proposés.

La description de telles fonctions doit inclure, si applicables:

— les raisons ayant nécessité ces fonctions particulières;

— les conditions déterminant l'activation de chaque fonction (détection d'accident);

— les séquences de tâches, d'actions et d'événements;

— les conditions de démarrage, l'état du système à l'initialisation des fonctions;

— les extensions possibles de cette fonction;

— les détails de la procédure de vérification.

A2.5.5 Les performances du système doivent être établies, en incluant:

— le cas le plus défavorable, le cas le plus favorable, le niveau de performance attendu pour tous les aspects, y compris la précision;

— la chronologie;

— les autres contraintes existantes et conditions obligatoires;

— tous les calculs particuliers à l'application ou les manipulations de données dépendant de l'application qui doivent être faites sur les données;

— les fonctions de manipulation des entrées/sorties, protocole de synchronisation et de communication;

— les fonctions de validation des entrées telles que validation de format, validation de champs, test logique et validation d'origine.

A2.5.6 Il convient de décrire la structure de données et leurs relations, en incluant:

— les caractéristiques de la base donnée, sa maintenance et sa mise à jour;

— toutes les fonctions permettant d'extraire l'information;

— resource constraints;

— appropriate reference to more detailed documents.

The incorporation of the computer systems in the safety systems and their relationship with the safety actuation systems shall be precisely described.

A2.5 *Description of system functions*

A2.5.1 The complete list of system functions shall be given. The number of items to be described depends on the complexity of the function. The description shall include as a minimum:

— the aim of each function;

— the relevance to the system reliability;

— all input/output variables.

A2.5.2 All variables necessary to execute the function shall be stated with regard to the following items:

— input domain and output range;

— relationship between the internal representation of the variable and its corresponding engineering units value;

— input accuracy and noise limits;

— output accuracy.

A2.5.3 Particular emphasis shall be placed on the description of functions with regard to safety, and those requiring a complex software design, and those needed to control or govern complex physical mechanisms.

A2.5.4 A detailed description of all system functions shall be provided relating them to each other and to system inputs and outputs. Diagrams shall be provided depicting functional and input/output relationships.

The description of such functions shall include, as far as applicable:

— reasons for particular functions;

— conditions which cause each function to operate (accident detection);

— sequences of tasks, actions and events;

— starting conditions, systems status at initiation of function;

— further possible extensions of that function;

— details of the verification procedure.

A2.5.5 The performance of the system shall be stated, including:

— worst case, best case, planned level of performance in any respect, including accuracy;

— timing;

— other existing constraints and compulsory conditions;

— all calculations particular to the application or application-dependent data manipulations that must be made on the data;

— input/output handling functions, synchronization communication protocol;

— input validation functions such as format validation, field validation, logic checks and source validation.

A2.5.6 The system data structure and relationships should be described including:

— data base characteristics, maintenance and updating;

— all information retrieval functions;

— l'identification de toutes les données qui sont requises pour arriver à une sortie spécifiée;

— la classification des données interdépendantes en groupes;

— l'activité de relations entre les groupes de données, les relations entre chacun des groupes de données et les entrées et sorties du système;

— la classification des groupes de données qui sont plus critiques que d'autres en matière d'exigence d'accès.

A2.6 *Description des contraintes entre le matériel et le logiciel*

A2.6.1 Les éléments suivants doivent être décrits:

— caractéristiques de fonctionnement en général (longueur du mot, type d'échange, vitesse, etc.); dans beaucoup de cas, une référence au manuel du fabricant de l'équipement est suffisante;

— caractéristiques de fonctionnement des équipements spéciaux (coupleurs particuliers, équipement de transmission de données, etc.);

— contraintes arithmétiques;

— exigences du logiciel standard;

— exigences de l'autocontrôle du matériel;

— au minimum une référence au document des exigences pour le matériel doit être faite.

A2.6.2 Il est recommandé d'examiner soigneusement les causes de pannes admises du matériel pour déterminer où le principe de diversité peut être efficacement appliqué pour éviter les pannes de mode commun. Cette diversité peut être:

— une diversité fonctionnelle dans laquelle des moyens différents sont utilisés pour accomplir une tâche particulière;

— une diversité d'équipements dans laquelle le matériel utilisé provient de fournisseurs différents.

A2.6.3 Les exigences réciproques entre le matériel et le logiciel doivent être déterminées pour la détection de pannes et la réaction sur les indications de pannes.

Les critères établis dans le guide 50-SG-D3 de l'AIEA ne nécessitent pas de développement complémentaire pour le matériel d'un système numérique. Cependant, une documentation de toutes les exigences du matériel, ayant un impact sur le logiciel, est nécessaire pour fournir la base de l'intégration du matériel et du logiciel et de la validation du système programmé. Les interfaces entre logiciels ou matériels d'un niveau de sûreté donné ainsi qu'avec celles de niveau plus élevé, doivent être décrites.

A2.6.4 Dans certains systèmes programmés, il peut être nécessaire de décrire les éléments suivants en plus de ceux déjà mentionnés:

— implantation séparée du logiciel et modularisation due à la structure choisie pour le matériel;

— effets de la structure multiprocesseurs sur le logiciel, effets sur les liens entre processeurs;

— influence du temps de traitement numérique;

— moniteur temps réel;

— capacité d'extension des systèmes et adaptabilité du système.

A2.7 *Description des conditions spéciales de fonctionnement*

Au minimum, les éléments suivants doivent être considérés:

— mise en route de la centrale et rechargement;

— identification of all data elements that will be required to arrive at the specified output;
— classification of related data elements into groups;
— data group activity relationships, relationship between each data group and the inputs and outputs of the system;
— classification of data groups which are more critical than others in terms of access requirements.

A2.6 *Description of constraints between hardware and software*

A2.6.1 The following items shall be described:

— operating characteristics in general (word length, exchange types, speed etc.); in many cases a reference to the equipment manufacturer manuals is sufficient;
— special equipment operating characteristics (particular drivers, data-communications equipment, etc.);
— arithmetic constraints;
— requirements of standard software packages;
— requirements of hardware self-supervision;
— at least a reference to the hardware requirements document shall be made.

A2.6.2 The postulated failure causes of the hardware should be carefully examined to determine where the principle for diversity could be used effectively to avoid common-mode failures. This diversity may be:

— functional diversity in which several different means are used to accomplish a particular task;
— equipment diversity in which hardware from different suppliers is used.

A2.6.3 The reciprocal requirements between hardware and software shall be determined with respect to failure detection and reaction on failure indications.

The criteria established in IAEA document 50-SG-D3 require no additional amplification for computer system hardware. However, documentation of all hardware requirements which impact software is necessary to provide the basis for hardware/software integration and computer system validation. The interface between software or hardware at one level of safety significance and at higher levels shall be described.

A2.6.4 In some software systems it may be advisable to describe the following items in addition to those already mentioned:

— spatial separation of software and modularization due to the hardware structure chosen;
— effects of the multiprocessor structure on software, effects of the linkage between the processors;
— time effects of digital processing;
— real time monitors;
— system expansion capabilities and system adaptability.

A2.7 *Description of special operating conditions*

At least the following conditions shall be considered:

— plant commissioning and refuelling;

— initialisation du système, incluant la mise en route du système en fournissant des valeurs de démarrage nécessaires, etc.;

— arrêt du système, mise hors service;

— procédure de reprise mise en œuvre dans le système;

— marche dégradée dans le cas où une erreur dans le logiciel est reconnue ou si une panne du matériel est détectée, en particulier si certains périphériques d'entrées ou de sorties ne sont pas disponibles;

— reconfiguration du système et réinitialisation après que l'ensemble du système ou une partie du système a été mis hors service;

— référence aux actions nécessaires de l'opérateur.

A2.8 *Autocontrôle*

A2.8.1 Il convient de déclencher des actions automatiques appropriées dans le cas de pannes, en considérant les points suivants:

— les pannes doivent être identifiées avec un degré raisonnable de précision et localisées dans un environnement le plus restreint possible;

— une sortie orientée dans le sens de la sécurité doit être garantie autant que possible;

— si une telle garantie ne peut être donnée, les sorties du système ne doivent violer que les exigences de sûreté les moins essentielles;

— les conséquences des pannes doivent être minimisées;

— il convient d'inclure les procédures de réparation, telles que retour en arrière, reprise, redémarrage du système;

— la reconstitution de données effacées ou incorrectement modifiées peut être essayée;

— une information sur les défaillances doit être fournie pour le personnel de conduite.

A2.8.2 Le système doit être conçu de telle façon qu'un autotest approprié soit réalisable. De tels principes de conception sont:

— la modularisation;

— le test intermédiaire de vraisemblance;

— l'utilisation de la redondance et de la diversité; la diversité peut être réalisée comme une diversité fonctionnelle ou une diversité logicielle;

— les réserves de temps d'exécution suffisant et d'espace mémoire pour l'implémentation des autotests;

— l'incorporation de facilités de tests permanentes;

— la simulation de défauts peut être utilisée pour confirmer l'adéquation des autotests.

A2.8.3 Les autotests du système ne doivent en aucune circonstance empêcher les réactions du système dans les conditions de temps imparties.

A2.9 *Présentation des exigences fonctionnelles du logiciel*

A2.9.1 Les exigences fonctionnelles du logiciel doivent être présentées de manière claire pour tous les groupes d'utilisateurs. La présentation doit être suffisamment détaillée, sans contradictions, et, autant que possible, sans redondances.

Le document doit être exempt de détails de réalisation, complet, cohérent et mis à jour.

A2.9.2 Le document des exigences du logiciel doit faire une distinction claire entre les exigences essentielles et les objectifs moins prioritaires.

- system initialization, including setting the system into operation, providing the necessary starting values, etc.;
- system switch-off, removal from service;
- re-try procedures implemented in the system;
- graceful degradation if errors in the software are recognized and if hardware failures are detected, in particular if certain input or output devices are not available;
- system re-configuration and re-initialization after the whole system or some parts have been out of service;
- reference to necessary operator actions.

A2.8 *Self-supervision*

A2.8.1 Appropriate automated actions should be taken at failures considering the following factors:

- failures shall be identified to a reasonable degree of detail and isolated to the most narrow environment;
- fail-safe output shall be guaranteed as far as possible;
- if such a guarantee cannot be given, system output shall violate only less essential safety requirements;
- the consequences of failures shall be minimized;
- remedial procedures, such as fall back, re-try, system recovery, should be considered for inclusion;
- reconstruction of obliterated or incorrectly altered data may be tried;
- information on failures shall be provided for the operating staff.

A.2.8.2 The system shall be designed such that appropriate self-supervision is feasible. Design principles used to assist this include:

- modularization;
- intermediate plausibility checks;
- use of redundancy and diversity; diversity may be implemented as functional diversity or software diversity;
- provision of sufficient execution time and memory space for self-checking purposes;
- incorporation of permanent test facilities;
- failure simulation may be used to confirm the adequacy of self-supervision features.

A2.8.3 System self-checking shall not prevent timely system reactions in any circumstance.

A2.9 *Presentation of software functional requirements*

A2.9.1 The software functional requirements shall be presented in a manner which is easy to understand for all user groups. The presentation shall be sufficiently detailed, free from contradiction, and non-redundant as far as possible.

The document shall be free of implementation details, complete, consistent and up-to-date.

A2.9.2 The software requirements document shall distinguish clearly between essential performance requirements and less stringent targets.

A2.9.3 Le document indiquant les exigences pour le logiciel est destiné à être utilisé par:

— ses auteurs, c'est-à-dire des spécialistes de centrales;
— le client ou l'utilisateur final;
— l'équipe de développement du logiciel du système;
— l'équipe de vérification du logiciel;
— les contrôleurs et les autorités de sûreté.

A2.9.3 The software requirements document is intended to be used by:

- its authors, i.e. plant specialists;
- the customer, client or final user;
- the software system development team;
- the software system verification team;
- assessors and licensing personnel.

# ANNEXE B

## RECOMMANDATIONS DÉTAILLÉES POUR LE DÉVELOPPEMENT (CONCEPTION ET PROGRAMMATION) DES LOGICIELS DE SÛRETÉ

Les recommandations sont énumérées dans les tableaux suivants. Les indications de priorité indiquées donnent une idée de l'importance de chaque recommandation. «1» veut dire importance maximale, «3» minimale. Dans le cadre d'un projet, une sélection des critères recommandés doit être, en principe, faite selon les priorités indiquées dans les en-têtes puis dans les rubriques associées.

Si la réalisation d'une recommandation particulière peut être facilitée par l'utilisation d'un matériel, d'un programme spécial ou d'un système d'exploitation, ce fait est d'ordinaire mentionné par «×». Si, de plus, on peut obtenir pour cette réalisation une aide de moindre importance, celle-là est indiquée par un «○» additionnel.

La colonne «contrôle» comprend une référence aux éléments qui permettent de confirmer qu'une recommandation a été respectée, comme indiqué ci-dessous:

— «D» indique la documentation;
— «C» indique les lignes de code;
— «H» indique que la vérification est à peine possible;
— «T» indique que la vérification est possible pendant le test des programmes.

# APPENDIX B

## DETAILED RECOMMENDATIONS FOR THE DEVELOPMENT (DESIGN AND CODING) OF SAFETY RELATED SOFTWARE

The recommendations are listed in the following tables. The indicated priority or importance shows the significance of the individual requirements. "1" means highest significance, "3" lowest. A project should select recommended criteria according to the priority indicated for each main heading and sub-heading in order.

If the implementation of a specific requirement can be facilitated by computer hardware, a special program or the operating system, this is usually marked by "×". If in addition to this a minor help can be used this is marked additionally by "○".

A reference is given in the "Check" column to the material which allows confirmation that a recommendation has been met, as follows:

- "D" stands for documentation;
- "C" for written code;
- "H" signifies verification is hardly possible;
- "T" means verification is possible during program testing.

B1 **Procédures de conception**
B1.*a* *Adaptabilité*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *a* | La conception du logiciel doit permettre des évolutions aisées | 2 | | | × | | | D | Nécessité d'une reprise complète des études à un stade avancé du développement / |
| *aa* | Les caractéristiques du logiciel à développer, ainsi que les exigences fonctionnelles susceptibles d'évoluer pendant le cycle de vie devraient être identifiées au début des études | 2 | | | × | | | D | / Les extensions |
| *ab* | Pendant les étapes suivantes de l'étude, les modules devraient être choisis pour que les modifications les plus probables se traduisent par des changements dans un ou deux modules seulement | 2 | | | × | | | D | / La facilité des modifications |
| *ac* | L'adaptabilité devrait être soigneusement mise en balance avec l'accroissement du temps d'exécution et de l'espace mémoire qui peut en résulter | 1 | | | × | | | D | L'obtention de systèmes trop volumineux / |
| | B1.*b* *Approche descendante par raffinements successifs* | | | | | | | | |
| *b* | Une approche descendante doit être utilisée lors de la conception | 2 | | | | × | | D | Erreurs de conception en général / Etude complète |
| *ba* | Les aspects généraux devraient être prioritaires | 2 | | | | × | | D | Cohérence du système / Etude complète |
| *bb* | A chaque niveau de raffinement, l'ensemble du système devrait être entièrement décrit et contrôlé | 2 | | | | × | | D | / Cohérence du système par l'identification des problèmes le plus tôt possible |
| *bc* | Les zones sensibles devraient être identifiées autant que possible au début du processus de conception | 3 | | | | × | | D | / Coordination de la programmation |

B1 **Design procedures**
B1.*a* *Changeability*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *a* | Software design shall easily allow changes | 2 | | | × | | | D | Necessity of complete redesign at late development stages / |
| *aa* | At an early design stage it should be identified which characteristics of the software to be developed and its functional requirements are likely to change during its life cycle | 2 | | | × | | | D | / Reaching flexibility |
| *ab* | During further design stages modules should be chosen such that the most probable modifications result in the change of one or two modules only | 2 | | | × | | | D | / Ease of modifications |
| *ac* | Modifiability should be carefully counterbalanced with resulting overhead in run time and memory space | 1 | | | × | | | D | Getting too bulky systems / |
| | B1.*b* *Top-down approach* | | | | | | | | |
| *b* | A top-down approach shall be used in design | 2 | | | | × | | D | Design errors generally / Completeness of design |
| *ba* | General aspects should precede specific ones | 2 | | | | × | | D | Consistency of the system / Completeness of design |
| *bb* | At each level of refinement the whole system should be completely described and verified | 2 | | | | × | | D | / Consistency of the system finding problem areas as early as possible |
| *bc* | Areas of difficulty should be identified as far as possible at an early stage in the design procedure | 3 | | | | × | | D | / Coordination of programming |

B1 **Procédures de conception**

B1.*b* *Approche descendante par raffinements successifs*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *bd* | Les décisions importantes devraient être discutées et documentées le plus tôt possible | 3 | | | | × | | D | Rencontrer des difficultés en fin de développement / |
| *be* | Après toute décision importante affectant d'autres parties du système, les alternatives devraient être envisagées et leurs risques documentés | 3 | | | | × | | D | Travail fait plusieurs fois / Conception en connaissance de cause |
| *bf* | Les conséquences sur les autres parties du système qui découlent des décisions individuelles devraient être identifiées | 3 | | | | × | | D | Travail fait plusieurs fois / Conception en connaissance de cause |
| *bg* | L'intervalle entre les niveaux devrait être suffisamment petit pour permettre une compréhension claire du processus de décision impliquée par l'étape | 3 | | | | × | | D | Erreurs dues à des étapes trop grandes / |
| *bh* | La conception et le développement des programmes devraient être faits en utilisant un ou plusieurs formalismes de description de haut niveau, tels que ceux utilisés en logique mathématique, théorie des ensembles, pseudo code, tables de décisions, diagrammes logiques ou autres outils graphiques ou langages orientés application | 2 | | | | × | | D | Erreurs d'interprétation ou imprécision / |
| *bi* | Autant que possible, des outils de développements automatisés devraient être utilisés | 3 | | × | | × | | D | Erreurs humaines / Economie d'efforts |
| *bk* | La documentation devrait être telle que l'auteur des spécifications puisse comprendre et contrôler la conception et le programme | 2 | | | | × | | D C | / Maintenance et cohérence avec les spécifications |
| *bl* | Le codage devrait être parmi les dernières étapes de la réalisation | 2 | | | × | ○ | | H | Incohérences / |
| | B1.*c* *Vérification intermédiaire* | | | | | | | | |
| *c* | La production intermédiaire doit être constamment vérifiée (article A1.) | 1 | | | | × | | D | / Trouver les erreurs le plus tôt possible, étude complète |

B1 **Design procedures**
B1.*b* *Top-down approach*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *bd* | Principal decisions should be discussed and documented as soon as possible | 3 | | | | × | | D | Running into difficulties at the end of the development / |
| *be* | After any major decision affecting other system parts, the alternatives should be considered and their risks be documented | 3 | | | | × | | D | Duplicating work / Careful design |
| *bf* | Consequences for other system parts which are implied by individual decisions should be identified | 3 | | | | × | | D | Duplicating work / Careful design |
| *bg* | The interval between levels should be small enough to permit a clear understanding of the decision process involved within the step | 3 | | | | × | | D | Errors due to too large steps / |
| *bh* | The program design and development should proceed using one or more higher level descriptive formalism, such as used in mathematical logic, set theory, pseudo code, decision tables, logic diagrams or other graphic aids or problem oriented languages | 2 | | | | × | | D | Misinterpretation or inaccuracy / |
| *bi* | As far as possible automatic development aids should be used | 3 | | × | | × | | D | Human mistakes / Saving effort |
| *bk* | Documentation should be such that the specifier is able to understand and check both the design and the program | 2 | | | | × | | D C | / Maintenance and consistency with specification |
| *bl* | Coding should be among the final steps taken | 2 | | | × | ○ | | H | Inconsistencies / |
| | B1.*c* *Intermediate verification* | | | | | | | | |
| *c* | The intermediate product shall be continuously verified (Clause A1) | 1 | | | | × | | D | / Finding errors as soon as possible, completeness of design |

B1 **Procédures de conception**
B1.*c* *Vérification intermédiaire*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *ca* | On devrait démontrer que chaque niveau constitue un tout complet et cohérent | 1 | | | | × | | D | Nécessité de modifications au dernier moment / |
| *cb* | On devrait démontrer que chaque niveau de raffinement est cohérent avec le niveau précédent | 1 | | | | × | | D | Oubli de certains aspects / |
| *cc* | Les contrôles des cohérences devraient être effectués par des personnes neutres | 3 | | | | × | | H | Erreurs communes de programmation / |
| *cd* | Ces personnes devraient seulement indiquer les déficiences sans recommander aucune action | 3 | | | | × | | H | Identification personnelle avec le programme / Maintenir une attitude critique |
| | B1.*d* *Modifications en cours de développement* | | | | | | | | |
| *d* | Les modifications nécessaires en cours de développement doivent commencer à l'étape la plus en amont encore en rapport avec la modification | 1 | | | | × | | D | Introduction de nouvelles erreurs du fait de la modification / Eviter les effets induits à distance |
| *da* | Les modifications devraient s'effectuer en allant des étapes générales vers les étapes plus spécifiques | 2 | | ○ | | × | | D | / Identification de tous les effets de la modification |
| *db* | Si un module est modifié, il devrait être retesté conformément aux principes décrits précédemment, avant d'être réintégré dans le système | 1 | | × | × | × | | D<br>C | Erreurs cachées induites dans le module par la modification / |
| *dc* | De plus, l'environnement du module modifié devrait être retesté, dans la mesure où il est affecté par la modification | 1 | | ○ | × | | | D | Erreurs cachées induites dans l'environnement du module par la modification / |
| *dd* | La documentation des modifications importantes devrait comprendre les éléments du cahier des charges, du code, des zones de données, du flot de contrôle et de la chronologie affectés ou non affectés par la modification | 2 | | | | × | | D | / Suivi des effets de la modification |

B1 **Design procedures**
B1.*c* *Intermediate verification*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *ca* | It should be shown that each level of refinement is complete and consistent in itself | 1 | | | | × | | D | Necessity of later changes / |
| *cb* | It should be shown that each level of refinement is consistent with the previous level | 1 | | | | × | | D | Forgetting aspects / |
| *cc* | Consistency checks should be made by neutral personnel | 3 | | | | × | | H | Common errors with programming / |
| *cd* | These personnel should only mark deficiencies and not recommend any action | 3 | | | | × | | H | Personal identification with the program / Maintaining critical attitude |
| | B1.*d* *Changes during development* | | | | | | | | |
| *d* | Changes which are necessary during program development shall begin at the earliest design stage which is still relevant to the change | 1 | | | | × | | D | Introducing new errors due to changes / Avoiding hidden far distant effects |
| *da* | Changes should proceed from the general stage to the more specific stages | 2 | | ○ | | × | | D | / Recognizing all effects of the change |
| *db* | If any module is changed, it should be retested according to the principles described earlier, before it is reintegrated into the system | 1 | | × | × | × | | D C | Hidden errors in module due to change / |
| *dc* | In addition the environment of the changed module should be retested, as far as it is affected by the change | 1 | | ○ | × | | | D | Hidden errors in module environment due to change / |
| *dd* | Documentation of major changes should include the requirements, code parts, data areas, control flow characteristics and time aspects that were affected or not affected | 2 | | | | × | | D | / Traceability of change effects |

B1 **Procédures de conception**
B1.*d* *Modifications en cours de développement*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *de* | Les modifications affectant des parties déjà testées ou le travail d'autres personnes devraient être évaluées et revues avant leur incorporation | 2 | | | | × | | D | Effets cachés induits à distance / |

*Note.* — Cette procédure est valable pour les modifications qui affectent le travail d'une seule personne ainsi que pour les modifications qui affectent l'ensemble du système. Dans ce cas, les recommandations de l'article 10 s'appliquent en plus.

B1.*e* *Reconfiguration du système*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *e* | L'expérience obtenue lors de l'adaptation de tout système à de nouvelles applications doit être prise en compte | 1 | ○ | | × | ○ | | D | / Vérification statistique |
| *ea* | L'état du système devrait être établi avant l'adaptation | 1 | | ○ | × | × | | D | Négliger des parties réutilisables / Identifier des parties défectueuses |
| *eb* | On devrait plutôt envisager d'utiliser des parties de systèmes ou des modules éprouvés plutôt que d'en vérifier de nouveaux | 2 | | | × | | | D | Efforts supplémentaires non nécessaires / |
| *ec* | Les modifications ou améliorations décidées devraient être conduites selon les recommandations données dans les tableaux B1.*b* et B1.*d* ainsi que dans l'article 9 | 1 | | | × | × | | D | / Obtenir un système propre |
| *ed* | Au niveau codage, chaque modification devrait être faite séparément | 2 | | | × | | | D | / Identification précoce des erreurs |
| *ee* | Les parties ou modules déjà vérifiés devraient être laissés inchangés | 2 | | | × | | | C | Produire de nouvelles erreurs / Economiser des efforts de vérification |

*Note.* — Toute amélioration d'un système qui ne vient pas d'une modification effectuée selon le tableau B1.*d* va contre sa clarté.

B2 **Structure logicielle**
B2.*a* *Structure de contrôle et d'accès*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *a* | Les programmes et les parties du programme doivent être groupés de façon systématique | 1 | ○ | | × | | ○ | D | / Structure système claire |

B1 **Design procedures**
B1.*d* *Changes during development*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *de* | Changes affecting already tested parts or the work of other persons should be evaluated and reviewed prior to incorporation | 2 | | | | × | | D | Hidden far distant effects / |

*Note.* — This procedure is valid for changes that affect the work of only one person and for modifications that affect the whole system. In the latter case the recommendations of Chapter 10 apply additionally.

B1.*e* *System reconfiguration*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *e* | The experience gained with any system to be adapted for new applications shall be taken into account | 1 | ○ | | × | ○ | | D | / Statistical verification |
| *ea* | The state of the system should be assessed before adaptation | 1 | | ○ | × | × | | D | Neglecting usable parts / Recognizing failed parts |
| *eb* | One should rather try to use system parts or modules with extensive operating experience than to formally verify new ones | 2 | | | × | | | D | Unnecessary additional effort / |
| *ec* | Decided changes or amendments should proceed as recommended in Tables B1.*b* and B1.*d* and Chapter 9 | 1 | | | × | × | | D | / Getting a clean system |
| *ed* | At code level each change should be made separately | 2 | | | × | | | D | / Identifying errors soon |
| *ee* | Already verified parts or modules should be left unchanged | 2 | | | × | | | C | Producing new errors / Saving verification effort |

*Note.* — Any amendment of a system that does not result in a change according to Table B1.*d* spoils its clarity.

B2 **Software structure**
B2.*a* *Control and access structure*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *a* | Programs and program parts shall be grouped systematically | 1 | ○ | | × | | ○ | D | / Clear system structure |

B2 **Structure logicielle**

B2.*a* *Structure de contrôle et d'accès*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *aa* | Les opérations spécifiques du système devraient être réalisées dans des parties spécifiques | 1 | ○ | ○ | × | | | D<br>C | / Testabilité |
| *ab* | Le logiciel devrait être découpé de façon que les parties qui gèrent des fonctions telles que<br>— interfaces externes au calculateur (par exemple gestions de périphérique, et d'interruption)<br>— signaux temps réel (par exemple horloge)<br>— gestion du parallélisme (par exemple gestions multitâches)<br>— allocation mémoire<br>— fonctions spéciales (par exemple utilitaires)<br>— adéquation de «fonctions» standard à la configuration matérielle spécifique du calculateur,<br>soient séparées des programmes «d'application» avec des interfaces bien définies | 2 | ○ | | × | | ○ | D<br>C | / Structure système claire, testabilité |
| *ac* | La structure des programmes devrait permettre d'effectuer les modifications à venir avec un minimum d'efforts (voir aussi tableau B1.*a*) | 2 | | | | × | | D | / Evolutivité du système |
| *ad* | Les méthodes de structuration utilisées devraient apparaître clairement | 1 | | | | × | | D | / Compréhension |
| *ae* | Dans un processeur, le système programmé devrait travailler séquentiellement autant que possible | 1 | × | | × | | | C<br>D | Confusion due à des problèmes de synchronisation ou à différentes séquences d'interruption / |
| *af* | Un programme ou une partie de programme devrait être divisé en modules clairs et intelligibles quand il contient plus de 100 instructions exécutables | 1 | ○ | | × | | | C<br>D | / Compréhension |
| | B2.*b* *Modules* | | | | | | | | |
| *b* | Les modules doivent être clairs et intelligibles | 1 | ○ | | × | ○ | | D, C | / Compréhension |

B2 **Software structure**
B2.*a* *Control and access structure*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *aa* | Specific system operations should be performed by specific parts | 1 | ○ | ○ | × | | | D<br>C | / Testability |
| *ab* | The software should be partitioned so that aspects which handle such functions as<br>— computer external interfaces (e.g. device driving, interrupt handling);<br>— real time signals (e.g. clock);<br>— parallel processing (e.g. scheduler);<br>— store allocation;<br>— special functions (e.g. utilities);<br>— mapping standard "functions" on to the particular computer hardware;<br>are separated from "application" programs with well defined interfaces between them | 2 | ○ | | × | | ○ | D<br>C | / Clear system structure, testability |
| *ac* | The program structure should permit the implementation of anticipated changes with a minimum of effort (see also Table B1.*a*) | 2 | | | | × | | D | / System adaptability |
| *ad* | It should be made clear which structuring methods are being used | 1 | | | | × | | D | / Understandability |
| *ae* | In one processor the program system should work sequentially, as far as possible | 1 | × | | × | | | C<br>D | Confusion due to timing problems or different interrupt sequences / |
| *af* | A program or a program part should be broken down into clear and intelligible modules when it contains more than 100 executable statements | 1 | ○ | | × | | | C<br>D | / Understandability |
| | B2.*b* *Modules* | | | | | | | | |
| *b* | Modules shall be clear and intelligible | 1 | ○ | | × | ○ | | D, C | / Understandability |

B2 **Structure logicielle**
*B2.b Modules*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *ba* | Chaque module devrait correspondre à une fonction spécifique | 1 | | | × | | | D<br>C | / Testabilité |
| *bb* | Un module devrait avoir un seul point d'entrée. Bien que plusieurs points de sortie soient parfois nécessaires, un seul point de sortie est recommandé | 1 | | | × | | | C | / Facilité de vérification |
| *bc* | Aucun module ne devrait dépasser la limite spécifiée pour un système donné (par exemple 50 ou 100 instructions, ou le volume de programme qui tient sur une page. Des modules plus longs ne sont permis que sous conditions spéciales) | 2 | | | × | | | C | Gros modules / Conditions ergonomiques |
| *bd* | Les interfaces entre les modules devraient être aussi simples que possible, cohérentes dans tout le système et entièrement documentées | 1 | | | × | | | C<br>D | Erreurs dans les interfaces qui sont normalement difficiles à détecter / |
| *be* | Le nombre de paramètres d'entrée et de sortie devrait être limité au minimum | 1 | | | × | | | C<br>D | Erreurs dans les interfaces qui sont normalement difficiles à détecter / |
| *bf* | On devrait établir de façon claire quelles variables d'interfaces sont seulement utilisées, seulement définies ou redéfinies | 2 | | | | × | | D<br>C | / Compréhension de la signification des paramètres |
| | B2.*c* *Système d'exploitation* | | | | | | | | |
| *c* | L'utilisation du système d'exploitation doit être réduit | 1 | × | | × | | × | D | Pannes dues aux erreurs du système d'exploitation / Facilité de vérification du système |
| *ca* | Seuls des systèmes d'exploitation soigneusement testés devraient être utilisés; il est préférable de connaître quantitativement l'expérience existante de l'exploitation du système | 1 | × | | | | | D | Erreurs cachées / |

B2 **Software structure**
B2.*b* *Modules*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *ba* | Each module should correspond to a specific function | 1 | | | × | | | D<br>C | / Testability |
| *bb* | A module should have only one entry. Although multiple exits may be sometimes necessary, single exits are recommended | 1 | | | × | | | C | / Ease of verification |
| *bc* | No module should exceed a limit specified for the particular system (e.g. 50 or 100 statements, or the amount of coding which can be placed on one page. Only under special circumstances are longer modules allowed) | 2 | | | × | | | C | Bulky modules / Meeting ergonomic facts |
| *bd* | The interfaces between modules should be as simple as possible, uniform throughout the system and fully documented | 1 | | | × | | | C<br>D | Errors in interfaces, which are normally difficult to detect / |
| *be* | The number of input and output parameters of modules should be limited to a minimum | 1 | | | × | | | C<br>D | Errors in interfaces, which are normally difficult to detect / |
| *bf* | It should be clearly stated which interface data are only used, only defined or redefined | 2 | | | | × | | D<br>C | / Understanding meaning of parameters |
| | B2.*c* *Operating system* | | | | | | | | |
| *c* | Operating system use shall be restricted | 1 | × | | × | | × | D | Failures due to operating system errors / Ease of system verification |
| *ca* | Only thoroughly tested operating systems should be used; preferably the existing operating experience should be quantitatively known | 1 | × | | | | | D | Hidden errors / |

B2 **Structure logicielle**
B2.*c* *Système d'exploitation*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *cb* | Si possible, on ne devrait pas utiliser de système d'exploitation | 1 | | | × | | × | C | / Vérification analytique |
| *cc* | Si un système d'exploitation était jugé nécessaire, son utilisation devrait se réduire à quelques fonctions simples | 1 | × | | × | | ○ | C | / Obtenir un système d'exploitation soigneusement testé |
| *cd* | Il ne devrait contenir que les fonctions nécessaires | 1 | × | | | | ○ | D<br>C | Code non utilisé / |
| *ce* | Une fonction donnée du système d'exploitation devrait être appelée toujours de la même façon | 2 | | | × | | ○ | C | / Vérification statistique due à des appels fréquents |
| *cf* | Les contrôleurs de périphériques devraient être pris dans le système d'exploitation plutôt que développés spécialement | 2 | × | | | | × | C | Programmation et test supplémentaires / |
| *cg* | Les fonctions du système d'exploitation devraient être définies de façon rigoureuse | 1 | | | ○ | × | | D | Problèmes de compréhension à l'utilisation / |
| *ch* | Si un système d'exploitation spécifique ou une partie spécifique était développé pour une application de sûreté particulière, ces recommandations devraient être suivies | 1 | × | | ○ | | × | D<br>C | / Facilité de vérification |
| *ci* | Les nouvelles versions devraient être traitées conformément aux tableaux B1.*d* et Bl.*e* | | × | | ○ | | | D<br>C | Introduction de nouvelles erreurs / |
| *ck* | Les autres programmes ou parties de programme normalisés devraient être traités comme des systèmes d'exploitation | 1 | | | × | | | D | Introduction de nouvelles erreurs / |
| | B2.*d* *Temps d'exécution* | | | | | | | | |
| *d* | L'influence du comportement du procédé technique sur le temps d'exécution doit rester faible | 1 | | | × | | | D<br>T | Problèmes de synchronisation non compréhensibles / |
| *da* | Le temps d'exécution de tout système ou partie de système en pointe de charge devrait être court comparé au temps d'exécution au-delà duquel les exigences du système de sûreté sont violées | 1 | | | × | | | T | Nécessite des changements tardifs dans la conception / |

B2 **Software structure**
B2.*c* *Operating system*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *cb* | If possible no operating system should be used | 1 | | | × | | × | C | / Analytical verification |
| *cc* | If an operating system is considered necessary its use should be restricted to a few simple functions | 1 | × | | × | | ○ | C | / Getting a thoroughly tested operating system |
| *cd* | It should contain only the necessary functions | 1 | × | | | | ○ | D<br>C | Dead code / |
| *ce* | One particular operating system function should be called always in a similar way | 2 | | | × | | ○ | C | / Statistical verification due to frequent call |
| *cf* | Device drivers should be taken from the operating system rather than specially developed | 2 | × | | | | × | C | Additional programming and test effort / |
| *cg* | Operating system functions should be rigorously defined and should have well defined interfaces | 1 | | | ○ | × | | D | Misunderstanding during using / |
| *ch* | If a dedicated operating system or a dedicated part is developed for a special safety application, these recommendations should be followed | 1 | × | | ○ | | × | D<br>C | / Ease of verification |
| *ci* | New releases should be treated according to Tables B1.*d* and B1.*e* | | × | | ○ | | | D<br>C | Introducing new errors / |
| *ck* | Other standardized programs or program parts should be treated as operating systems | 1 | | | × | | | D | / Statistical verification due to frequent call |
| | B2.*d* *Execution time* | | | | | | | | |
| *d* | Technical process behaviour influence on execution time shall be kept low | 1 | | | × | | | D<br>T | Unintelligible timing problems / |
| *da* | The execution time of any system or part of a system under peak load conditions should be short compared to the execution time beyond which the system safety requirements are violated | 1 | | | × | | | T | Necessity of late design changes / |

B2 **Structure logicielle**
B2.*d* *Temps d'exécution*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *db* | Le résultat d'un programme séquentiel ne doit dépendre<br>— ni du temps mis pour exécuter le programme<br>— ni de l'heure (en référence à une horloge indépendante) à laquelle l'exécution du programme est initialisée | 1 | | | × | | | D<br>T | Problèmes de synchronisation non compréhensibles / |
| *dc* | L'exécution de programmes séquentiels qui reçoivent ou transmettent des données à d'autres programmes séquentiels doit être synchronisée avec ces programmes | 1 | × | | × | | | D | Problèmes de synchronisation non compréhensibles / |
| *dd* | Les programmes devraient être conçus de façon à ce que les actions s'effectuent dans un ordre correct indépendamment de leur rapidité d'exécution | 1 | × | | × | | × | C<br>D | Problèmes de synchronisation et de temps d'exécution / Analyse |
| *de* | Les différences de durée d'exécution en fonction des paramètres d'entrée devraient être faibles | 1 | | | × | | | T | / Facilité d'estimation et de vérification du temps d'exécution |
| *df* | Les différences de durée d'exécution en fonction des paramètres d'entrée devraient être documentées | 2 | | | × | × | | D | / Facilité d'estimation et de vérification du temps d'exécution |
| *dg* | Les parties de code pour lesquelles la durée d'exécution dépend des paramètres d'entrée devraient être courtes | 3 | | | × | | | C | / Rejoint *de* |
| *dh* | La quantité de paramètres acquis pendant un cycle d'exécution devrait être constante | 3 | | | × | | | D<br>C<br>T | / Maintenir les différences de temps d'exécution faibles |
| | B2.*e* *Interruptions* | | | | | | | | |
| *e* | L'utilisation des interruptions doit être limitée | 1 | × | | × | | × | C<br>D | Problèmes de synchronisation et de temps d'exécution / Analyse |
| *ea* | Les interruptions peuvent être utilisées si elles simplifient le système | 1 | × | | × | | | C<br>D | / Facilité de compréhension de configurations spéciales |

B2 **Software structure**
B2.*d* *Execution time*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *db* | The results produced by a sequential program shall not be dependent on either<br>— the time taken to execute the program or<br>— the time (referenced to an independent "clock") at which execution of the program is initiated | 1 | | | × | | | D<br>T | Unintelligible timing problems / |
| *dc* | Execution of sequential programs which receive or transmit data from/to other sequential programs shall be synchronized with those other programs | 1 | × | | × | | | D | Unintelligible timing problems / |
| *dd* | The programs should be designed so that operations are performed in a correct sequence independent of their speed of execution | 1 | × | | × | | × | C<br>D | Synchronization problems and run time problems / Analysis |
| *de* | Run time differences depending on input parameters should be small | 1 | | | × | | | T | / Ease of run time estimation and run time verification |
| *df* | Run time differences depending on input parameters should be documented | 2 | | | × | × | | D | / Ease of run time estimation and run time verification |
| *dg* | Code parts for which execution time depends on input parameters should be short | 3 | | | × | | | C | / Reaching *de* |
| *dh* | The amount of parameters read during one computation cycle should be constant | 3 | | | × | | | D<br>C<br>T | / Keeping run time differences short |
| | B2.*e* *Interrupts* | | | | | | | | |
| *e* | The use of interrupts shall be restricted | 1 | × | | × | | × | C<br>D | Synchronization problems and run time problems / Analysis |
| *ea* | Interrupts may be used if they simplify the system | 1 | × | | × | | | C<br>D | / Ease of understanding of special configurations |

B2 **Structure logicielle**
B2.*e* *Interruptions*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *eb* | La gestion logicielle des interruptions doit être inhibée pendant les phases critiques (par exemple durée critique, modification de données critiques) de la fonction exécutée | 1 | × | | × | | × | C D | Problèmes de synchronisation et de temps d'exécution / Analyse |
| *ec* | Si les interruptions sont utilisées, les parties non interruptibles devraient avoir un temps d'exécution donné à ne pas dépasser de façon à ce que le temps maximal d'inhibition d'une interruption puisse être calculé | 1 | × | | × | | | C D T | Problèmes de temps d'exécution / |
| *ed* | La mise en œuvre des interruptions et leur inhibition doivent être soigneusement documentées | 1 | × | | | × | × | D | / Validation du système |
| | B2.*f* *Expressions arithmétiques* | | | | | | | | |
| *f* | On doit utiliser des expressions arithmétiques simples plutôt que des expressions compliquées | 2 | | | × | | | C D | Effets de bord / Facilité de test |
| *fa* | Les décisions ne devraient pas dépendre de calculs arithmétiques volumineux | 2 | | | × | | | C D | / Trouver la fonction inverse en calculant des expressions de chemin |
| *fb* | On devrait utiliser autant que possible des expressions arithmétiques simplifiées et vérifiées auparavant | 2 | | | × | | | C D | / Montrer les relations entre la fonction envisagée et le code |
| *fc* | Si des expressions arithmétiques volumineuses étaient utilisées elles devraient être codées de façon à ce que leur cohérence avec les expressions arithmétiques spécifiées puisse être montrée facilement par le code | 3 | | | × | | | C | / Analyse du programme |
| | B3 **Autosurveillance**<br>B3.*a* *Contrôle de vraisemblance* | | | | | | | | |
| *a* | Des contrôles de vraisemblance doivent être effectués (programmation défensive) | 2 | | | × | | | C D | Erreurs non encore détectées / Vérification statistique |

B2 **Software structure**
B2.*e* *Interrupts*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of: Operating system | Special program | User program | Program document | Hardware | Check | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| *eb* | Software handling of interrupts must be inhibited during critical parts (e.g. time critical, critical to data changes) of the executed function | 1 | × | | × | | × | C D | Synchronization problems and run time problems / Analysis |
| *ec* | If interrupts are used, parts not interruptable should have a defined maximum computation time, so that the maximum time for which an interrupt is inhibited can be calculated | 1 | × | | × | | | C D T | Run time problems / |
| *ed* | Interrupt usage and masking shall be thoroughly documented | 1 | × | | | × | × | D | / System validation |
| | B2.*f* *Arithmetic expressions* | | | | | | | | |
| *f* | Simple arithmetic expressions shall be used instead of complex ones | 2 | | | × | | | C D | Side effects / Easy testing |
| *fa* | Decisions should not depend on voluminous arithmetic calculations | 2 | | | × | | | C D | / Finding reverse function calculating path expressions |
| *fb* | As far as possible, simplified previously verified arithmetic expressions should be used | 2 | | | × | | | C D | / Showing relationship between intended function and code |
| *fc* | If voluminous arithmetic expressions are used, they should be coded so that their consistency with the specified arithmetic expression can be shown easily from the code | 3 | | | × | | | C | / Program analysis |
| | B3. **Self-supervision**<br>B3.*a* *Plausibility checks* | | | | | | | | |
| *a* | Plausibility checks shall be performed (defensive programming) | 2 | | | × | | | C D | Yet undetected errors / Statistical verification |

B3 **Autosurveillance**

B3.*a* *Contrôle de vraisemblance*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *aa* | La correction ou la vraisemblance des résultats intermédiaires devrait être contrôlée aussi souvent que possible, au moins dans la limite d'un petit pourcentage de la capacité du processeur (redondance) | 2 | | | × | | | C<br>D | Erreurs non encore détectées / Vérification statistique |
| *ab* | Quand des résultats relatifs à la sûreté sont impliqués, des résultats identiques devraient être évalués en utilisant des méthodes différentes (diversité) | 2 | | | × | | | C<br>D | Erreurs non encore détectées / Vérification statistique |
| *ac* | Des procédures de reprises devraient être utilisées | 3 | | | × | | | C<br>D<br>T | Pannes aléatoires du matériel / |
| *ad* | Les domaines de validité des:<br>— variables d'entrée<br>— variables de sortie<br>— paramètres intermédiaires<br>devraient être contrôlés, en incluant le contrôle des limites des tableaux | 2 | | | × | | | C<br>D | Erreurs non encore détectées / Vérification statistique |
| | B3.*b* *Sûreté des sorties* | | | | | | | | |
| *b* | Si une panne est détectée, le système doit donner une sortie bien définie | 1 | | | × | | | D<br>T | Propagation des erreurs / |
| *ba* | Si possible, des techniques de reprise correctes et complètes après erreurs devraient être utilisées | 3 | | | × | | | C<br>D<br>T | Arrêt système dû à des pannes mineures / |
| *bb* | Même si une reprise correcte après erreur ne peut être garantie, la détection d'une panne doit conduire à des sorties bien définies | 1 | | | × | | ○ | D<br>T | / Sûreté de l'équipement |
| *bc* | Si des techniques de reprise après erreur sont utilisées, l'occurrence de chaque erreur doit être signalée | 1 | × | | × | | | D<br>T | Accumulation d'erreurs / Suppression rapide des erreurs |
| *bd* | L'occurrence d'une erreur persistante affectant la fonction du système doit être enregistrée | 1 | × | | × | | | D<br>T | Accumulation d'erreurs / Suppression rapide des erreurs |

**B3 Self-supervision**

*B3.a Plausibility checks*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of: Operating system | Special program | User program | Program document | Hardware | Check | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| *aa* | The correctness or plausibility for intermediate results should be checked as often as possible, at least to within a small percentage of computer capacity (redundancy) | 2 | | | × | | | C D | Yet undetected errors / Statistical verification |
| *ab* | Where safety related results are involved identical results should be evaluated, using different methods (diversity) | 2 | | | × | | | C D | Yet undetected errors / Statistical verification |
| *ac* | Re-try procedures should be used | 3 | | | × | | | C D T | Sporadic hardware faults / |
| *ad* | The ranges of:<br>— input variables;<br>— output variables;<br>— intermediate parameters<br>should be checked, including array bound checking | 2 | | | × | | | C D | Yet undetected errors / Statistical verification |
| | B3.*b* *Safe output* | | | | | | | | |
| *b* | If a failure is detected the system shall produce a well defined output | 1 | | | × | | | D T | Fault propagation / |
| *ba* | If possible, complete and correct error recovery techniques should be used | 3 | | | × | | | C D T | System breakdown due to minor failures / |
| *bb* | Even if correct error recovery cannot be guaranteed, failure detection must lead to well-defined output | 1 | | | × | | ○ | D T | / Equipment safety |
| *bc* | If error recovery techniques are used the occurrence of any error shall be reported | 1 | × | | × | | | D T | Accumulation of errors / Early error removal |
| *bd* | The occurrence of a persistent error affecting the system function shall be recorded | 1 | × | | × | | | D T | Accumulation of errors / Early error removal |

B3 **Autosurveillance**
B3.*c* *Contenu mémoire*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *c* | Le contenu mémoire doit être protégé ou surveillé | 1 | ○ | × | | | ○ | D<br>T | Modifications non autorisées d'un système certifié / |
| *ca* | L'espace mémoire pour les constantes et les instructions doit être protégé ou surveillé contre les modifications | 1 | ○ | × | | | ○ | D<br>T | Propagation d'erreurs d'adressage ou de pannes matérielles incluant les pannes intermittentes / |
| *cb* | Les lectures et écritures non autorisées devraient être empêchées | 1 | ○ | × | | | ○ | D<br>T | Propagation d'erreurs d'adressage ou de pannes matérielles incluant les pannes intermittentes / |
| *cc* | Le système devrait être protégé contre des modifications de code ou de données par l'opérateur de la centrale | 1 | ○ | × | | | ○ | D | / Maintenir l'intégrité du système dans sa forme certifiée |
| | B3.*d* *Contrôle des erreurs* | | | | | | | | |
| *d* | Le contrôle des erreurs doit être effectué au niveau de l'analyse organique | 1 | | | × | | | C<br>T | Propagation des erreurs / |
| *da* | Des compteurs et des trappes de vraisemblances («relay runners») devraient assurer que le programme s'est déroulé correctement | 1 | | ○ | × | | | C<br>T | Erreurs spécifiques du flot de contrôle, erreurs matérielles intermittentes / |
| *db* | La correction de tous les types de transfert de paramètres devrait être contrôlée, y compris les vérifications de type de paramètre | 1 | ○ | | × | | × | C<br>D | Erreurs dans la conception d'interface, erreurs dans la conception du flot de données / |
| *dc* | Quand on adresse un tableau, ses limites devraient être contrôlées | 1 | ○ | | × | | × | C<br>D<br>T | Erreurs dans le flot de données — nombre d'itérations trop grand dans une boucle / |

B3 **Self-supervision**
B3.*c* *Memory contents*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *c* | Memory contents shall be protected or monitored | 1 | ○ | × | | | ○ | D<br>T | Unauthorized changes of a licensed system / |
| *ca* | Memory space for constants and instructions shall be protected or supervised against changes | 1 | ○ | × | | | ○ | D<br>T | Propagation of addressing errors or hardware faults, including intermittent faults / |
| *cb* | Unauthorized reading and writing should be prevented | 1 | ○ | × | | | ○ | D<br>T | Propagation of addressing errors or hardware faults, including intermittent faults / |
| *cc* | The system should be secure against code or data changes by the plant operator | 1 | ○ | × | | | ○ | D | / Maintaining system integrity in its licensed form |
| | B3.*d* *Error checking* | | | | | | | | |
| *d* | Error checking shall be performed at a detailed coding level | 1 | | | × | | | C<br>T | Failure propagation / |
| *da* | Counters and reasonableness traps (relay runners) should insure that the program structure has been run through correctly | 1 | | 0 | × | | | C<br>T | Specific control flow errors, intermittent hardware faults / |
| *db* | The correctness of any kind of parameter transfer should be checked, including parameters type verification | 1 | ○ | | × | | × | C<br>D | Errors in interface design, data flow design errors / |
| *dc* | When addressing an array its bounds should be checked | 1 | ○ | | × | | × | C<br>D<br>T | Data flow errors, too high loop repetition numbers / |

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| | **B3 Autosurveillance**<br>B3.*d* *Contrôle des erreurs* | | | | | | | | |
| *dd* | Les sous-programmes appelés devraient vérifier si le module appelant est autorisé à le faire | 3 | | | × | | | C<br>D<br>T | Erreurs dans la conception du flot de contrôle / |
| *de* | Le temps d'exécution de phases critiques devrait être surveillé (par exemple par une horloge chien de garde) | 2 | × | | × | | × | D<br>T | Erreurs dans la conception du flot de contrôle, nombre d'itérations trop grand dans une boucle / |
| *df* | Des assertions devraient être utilisées | 3 | | | × | | | C<br>D | / Vraisemblance des résultats intermédiaires |
| | **B4 Analyse organique et codage**<br>B4.*a* *Branchements et boucles* | | | | | | | | |
| *a* | Les branchements et les boucles devraient être employés avec précaution | 1 | | | × | | | C<br>D | / Compréhension et vérification du flot de contrôle |
| *aa* | Les branchements conditionnels ou inconditionnels devraient être évités dans la mesure où ils obscurcissent la relation entre la structure du problème et la structure du programme. On devrait utiliser le plus possible un code séquentiel | 2 | | | × | | | H | Difficultés dans l'analyse du flot de contrôle / Lisibilité |
| *ab* | Les branchements arrière devraient être évités au profit des instructions de boucle (pour langages évolués seulement) | 1 | | | × | | | C | / Compréhension et vérification du flot de contrôle |
| *ac* | Les branchements pénétrant dans des boucles, des modules ou des sous-programmes doivent être interdits | 1 | | | × | | | C | / Compréhension et vérification du flot de contrôle |
| *ad* | Les branchements vers l'extérieur des boucles devraient être évités s'ils n'arrivent pas exactement à la fin de la boucle. Exception: sortie en cas d'erreur | 3 | | | × | | | C | / Compréhension et vérification du flot de contrôle |
| *ae* | Dans les modules de structure complexe, les macros, les procédures ou les sous-programmes devraient être utilisés de façon que la structure ressorte clairement | 3 | | | × | | | D<br>C | / Compréhension et vérification du flot de contrôle |

B3 **Self-supervision**
B3.*d* *Error checking*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of: Operating system | Special program | User program | Program document | Hardware | Check | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| *dd* | Called routines should check whether the calling module is entitled to do so | 3 | | | × | | | C D T | Control flow design errors / |
| *de* | The run time of critical parts should be monitored (e.g. by a watchdog timer) | 2 | × | | × | | × | D T | Control flow design errors, too high loop repetition numbers / |
| *df* | Assertions should be used | 3 | | | × | | | C D | / Plausibility of intermediate results |

B4 **Detailed design and coding**
B4.*a* *Branches and loops*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *a* | Branches and loops should be handled cautiously | 1 | | | × | | | C D | / Understandable and verifiable control flow |
| *aa* | Conditional and unconditional branches should be avoided as far as they obscure the relationship between problem structure and program structure, as much sequential code as possible should be used | 2 | | | × | | | H | Difficulties with control flow analysis / Readability |
| *ab* | Backward going branches should be avoided, loop statements should be used instead (only for higher level languages) | 1 | | | × | | | C | Difficulties with control flow analysis / Readability |
| *ac* | Branches into loops, modules or subroutines must be barred | 1 | | | × | | | C | Difficulties with control flow analysis / Readability |
| *ad* | Branches out of loops should be avoided, if they do not lead exactly to the end of the loop. Exception: failure exit | 3 | | | × | | | C | Difficulties with control flow analysis / Readability |
| *ae* | In modules with complex structure, macros, procedures or subroutines should be used so that structure stands out clearly | 3 | | | × | | | D C | Difficulties with control flow analysis / Readability |

B4 **Analyse organique et codage**
B4.*a* *Branchements et boucles*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *af* | Comme mesure spéciale afin d'améliorer le contrôle et la vérification des programmes, les instructions de GOTO calculés ou d'aiguillage (SWITCH) de même que les variables étiquetées devraient être évitées | 2 | | | × | | | C | / Facilité de compréhension du fait de la proximité de la condition et du code associé |
| *ag* | Quand une liste de branchements logiques ou d'instructions contrôlées par un «CASE» sont utilisées, la liste des branchements ou des conditions du «CASE» doit être une liste exhaustive des possibilités. Le concept de «branche par défaut» devrait être réservé pour la gestion des erreurs | 2 | | | × | | | C | / Clarifier le «ou» exclusif |
| *ah* | Les boucles ne devraient être utilisées qu'avec des limites constantes d'évolution de l'indice de boucle | 1 | | | × | | | C | Problèmes de temps d'exécution, violation des frontières de tableau / Nombre de chemins contrôlables |
| | B4.*b* *Sous-programmes et procédures* | | | | | | | | |
| *b* | Les sous-programmes et procédures devraient être organisés aussi simplement que possible | 1 | | | × | | | C | Complexité inutile / |
| *ba* | Ils devraient avoir seulement un nombre minimal de paramètres | 3 | | | × | | | C | / Conserver des programmes et interfaces courts et simples |
| *bb* | Ils devraient communiquer avec leur environnement exclusivement via leurs paramètres | 1 | | | × | | | C | / Compréhension du flux de données, analyse du flux de données |
| *bc* | Tous les paramètres devraient avoir la même forme (par exemple pas de mélange d'appel par nom et par valeur) | 3 | | | × | | | C | Difficultés dans la signification des paramètres / |
| *bd* | Les sous-programmes devraient avoir un seul point d'entrée | 1 | | | × | | | C | / Flux de contrôle compréhensible, analyse du flux de contrôle |
| *be* | Le retour des sous-programmes devrait se faire seulement en un seul point pour chaque appel de sous-programme. Exception: sortie sur défaut | 1 | | | × | | | C | / Flux de contrôle compréhensible, analyse du flux de contrôle |

**B4 Detailed design and coding**

**B4.*a* *Branches and loops***

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *af* | As a special measure to support program proving and verification computed GOTO or SWITCH statements as well as label variables should be avoided | 2 | | | × | | | C | / Ease of understanding, because of near neighbourhood between condition and its code |
| *ag* | Where a list of alternative branches or case controlled statements are used, the list of branch or case conditions must be an exhaustive list of possibilities. The concept of a "default branch" should be reserved for failure handling | 2 | | | × | | | C | / Clarifying exclusive or |
| *ah* | Loops should only be used with constant maximum loop variable ranges | 1 | | | × | | | C | Run time problems, violating array boundaries / Surveyable path number |
| | **B4.*b* *Subroutines and procedures*** | | | | | | | | |
| *b* | Subroutines or procedures should be organized as simply as possible | 1 | | | × | | | C | Unnecessary complexity / |
| *ba* | They should have only a minimum number of parameters | 3 | | | × | | | C | / Keeping routines and interfaces short and simple |
| *bb* | They should communicate exclusively via their parameters to their environment | 1 | | | × | | | C | / Understandability of data flow, data flow analysis |
| *bc* | All parameters should have the same form (e.g. no mixing of call by name and value) | 3 | | | × | | | C | Difficulties with parameter significance / |
| *bd* | Subroutines should have only one entry point | 1 | | | × | | | C | / Understandable control flow, control flow analysis |
| *be* | Subroutines should return to only one point for each subroutine call. Exception: default exit | 1 | | | × | | | C | / Understandable control flow, control flow analysis |

**B4 Analyse organique et codage**
**B4.*b* *Sous-programmes et procédures***

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *bf* | Le point de retour devrait suivre immédiatement le point d'appel | 1 | | | × | | | C | / Flux de contrôle compréhensible, analyse du flux de contrôle |
| | B4.*c* *Structures imbriquées* | | | | | | | | |
| *c* | Les structures imbriquées doivent être manipulées avec soin | 2 | | | × | | | C | / Intelligibilité |
| *ca* | Les macros imbriquées devraient être évitées | 2 | | | × | | | C | Création de types supplémentaires de macros / |
| *cb* | Les procédures avec paramètres calculés devraient être évitées | 2 | | | × | | | C | / Accessibilité aux résultats intermédiaires |
| *cc* | La liaison de différents types d'action du programme au moyen de résultat de boucles imbriquées ou de procédures imbriquées devrait être évitée, si elle nuisait à la relation entre la structure du problème et la structure du programme | 2 | | | × | | | D<br>C | / Favoriser les structures séquentielles |
| *cd* | Des hiérarchies de procédures et de boucles devraient être utilisées, si elles clarifiaient la structure du système | 2 | | | × | | | D<br>C | / Indication des différents niveaux d'abstraction durant un développement à approche descendante |
| | B4.*d* *Adressage et zones de données* | | | | | | | | |
| *d* | Des techniques simples d'adressage doivent être utilisées | 1 | | | × | | | C | / Facilité d'analyse des flux de données |
| *da* | Un seul type d'adressage devrait être utilisé pour chaque type de donnée | 3 | | | × | | | C | / Interface uniforme à la base de données |
| *db* | Les calculs complexes d'index devraient être évités | 1 | | | × | | | C | / Flux de données compréhensible |
| *dc* | Les tableaux devraient avoir une longueur fixe, prédéfinie | 1 | | | × | | | C | Difficultés de temps d'exécution, flux de contrôle complexe / Facilité d'analyse des flux de données |

B4 **Detailed design and coding**
B4.*b* *Subroutines and procedures*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *bf* | The return point should immediately follow the point of call | 1 | | | × | | | C | / Understandable control flow, control flow analysis |
| | B4.*c* *Nested structures* | | | | | | | | |
| *c* | Nested structures shall be handled with care | 2 | | | × | | | C | / Intelligibility |
| *ca* | Nested macros should be avoided | 2 | | | × | | | C | Creating additional macro types / |
| *cb* | Procedure valued parameters should be avoided | 2 | | | × | | | C | / Accessibility of intermediate results |
| *cc* | Joining of different types of program action by means of nested loop statement or nested procedures should be avoided, if they obscure the relationship between problem structure and program structure | 2 | | | × | | | D<br>C | / Favouring sequential structures |
| *cd* | Hierarchies of procedures and loops should be used, if they clarify the system structure | 2 | | | × | | | D<br>C | / Indicating different levels of abstraction during top-down development |
| | B4.*d* *Addressing and arrays* | | | | | | | | |
| *d* | Simple addressing techniques shall be used | 1 | | | × | | | C | / Ease of data flow analysis |
| *da* | Only one addressing technique should be used for each data type | 3 | | | × | | | C | / Uniform interface to data base |
| *db* | Bulky computations of indexes should be avoided | 1 | | | × | | | C | / Understandable data flow |
| *dc* | Arrays should have a fixed, predefined length | 1 | | | × | | | C | Run time difficulties, complex control flow / Ease of data flow analysis |

B4 **Analyse organique et codage**
B4.*d* *Adressage et zones de données*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *dd* | Le nombre de dimensions dans toute référence à un tableau devrait être égal au nombre déclaré | 1 | | | × | | | C | / Processus d'adressage compréhensible |
| | B4.*e* *Structure des données* | | | | | | | | |
| *e* | Les structures des données et les conventions de nom doivent être utilisées uniformément d'un bout à l'autre du système | 1 | | | × | | | D<br>C | / Analyse des flux de données, compréhension intuitive de la signification des éléments |
| *ea* | Les variables, zones et cellules mémoires devraient avoir un seul objet et une seule structure. L'utilisation de techniques d'équivalence devrait être évitée | 1 | | | × | | | D<br>C | Erreurs dans l'utilisation des données, problèmes artificiels de temps / Traçabilité du flux de données |
| *eb* | Chaque nom de variable devrait refléter:<br>— son type (tableau, variable, constante);<br>— son domaine de validité (locale, globale programme, module);<br>— sa catégorie (entrée, sortie, interne, dérivée, comptage, taille de tableau);<br>— sa signification | 1 | | | × | | | D<br>C | / Compréhension intuitive de l'élément de donnée |
| *ec* | Les paramètres du système qui peuvent changer devraient être identifiés et leur valeur attribuée à une position bien définie du code | 1 | | | × | | | D<br>C | / Compréhension du système, modifications |
| *ed* | Les constantes et les variables devraient être placées dans des parties distinctes de la mémoire | 2 | | | × | | | C | Pollution des données et du code / Autotest matériel |
| | *Note.* — Beaucoup de systèmes temps réel utilisent une base de donnée accessible sans restriction ou une ressource de même type. Quand des structures globales de données de ce type sont utilisées, leur accès devrait s'effectuer via des procédures ou des tâches normales de manipulation de ressources. | | | | | | | | |
| | B4.*f* *Modifications dynamiques* | | | | | | | | |
| *f* | Les modifications dynamiques d'instruction doivent être évitées | 1 | | | × | | | C | / Analyse du flux de contrôle |

B4 **Detailed design and coding**
B4.*d* *Addressing and arrays*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of: Operating system | Special program | User program | Program document | Hardware | Check | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| *dd* | The number of dimensions in every array reference should equal the number of dimensions in its corresponding declaration | 1 | | | × | | | C | / Understanding addressing process |
| | B4.*e* *Data structures* | | | | | | | | |
| *e* | Data structures and naming conventions shall be used uniformly throughout the system | 1 | | | × | | | D C | / Data flow analysis, intuitive comprehension of the significance of data elements |
| *ea* | Variables, arrays and memory cells should have a single purpose and structure. The use of equivalence techniques should be avoided | 1 | | | × | | | D C | Errors in data use, artificial timing problems / Traceability of data flow |
| *eb* | Each variable's name should reflect:<br>— type (array, variable, constant);<br>— region of validity (local, global program, module);<br>— kind (input, output, internal, derived; count, array length);<br>— significance | 1 | | | × | | | D C | / Intuitive comprehension of data element |
| *ec* | System parameters that can change should be identified and their values assigned at a well defined outstanding code position | 1 | | | × | | | D C | / System understanding, changes |
| *ed* | Constants and variables should be located in different parts of the memory | 2 | | | × | | | C | Poisoning of data and code / Hardware self-supervision |
| | *Note.* — Many real time program systems make use of a universally accessible "data base" or similar resource. When such global data structures are used, they should be accessed via standard resource handling procedures or via communication with standard resource manipulating tasks. | | | | | | | | |
| | B4.*f* *Dynamic changes* | | | | | | | | |
| *f* | Dynamic instruction changes shall be avoided | 1 | | | × | | | C | / Control flow analysis |

**B4 Analyse organique et codage**
*B4.g Tests intermédiaires*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *g* | Des tests intermédiaires doivent être réalisés durant le développement du programme | 1 | | × | | ○ | | D | / Trouver les erreurs aussi tôt que possible |
| *ga* | L'approche de test devrait suivre l'approche de conception (i.e. durant la conception de haut en bas, le contrôle de la conception devrait être fait en simulant les parties non existantes du système; après la conception, un contrôle d'intégration de bas en haut devrait être suivi) | 1 | | × | | | | D | / Trouver les erreurs aussi tôt que possible |
| *gb* | Chaque module devrait être testé complètement avant son intégration dans le système et les résultats des tests documentés | 1 | | × | | × | ○ | D | Difficultés après intégration / Gestion des modifications |
| *gc* | Une description formelle des entrées du test et les résultats (protocole de test) devraient être produits | 1 | | ○ | | × | | D | Duplication du travail / Accélération de l'approbation |
| *gd* | Les erreurs de codage qui sont détectées lors du test devraient être notées et analysées | 3 | | | | × | | D | / Détection d'erreur de conception |
| *ge* | Les tests incomplets devraient être notés | 3 | | | | × | | D | / Clarté |
| *gf* | Dans le but de faciliter l'utilisation des résultats des tests intermédiaires, lors de la validation finale, le niveau de test atteint au préalable devrait être noté (par exemple tous les chemins testés dans un module) | 2 | | × | | | | D | Duplication du travail / |

**B5 Recommandations dépendant du langage**
*B5.a Organisation du programme*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *a* | Des règles détaillées doivent être élaborées pour la présentation des constructions des divers langages | 2 | | | | | | C | / Obtenir une forme unique et intelligible des listages de programme |

B4 **Detailed design and coding**
B4.*g* *Intermediate tests*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *g* | Intermediate tests shall be performed during the program development | 1 | | × | | 0 | | D | / Finding errors as soon as possible |
| *ga* | The approach to testing should follow the approach to design (e.g. during top down design testing should be made by using simulation of not yet existing system parts – stubs –; after completion of system development this should be followed by bottom up integration testing) | 1 | | × | | | | D | / Finding errors as soon as possible |
| *gb* | Each module should be tested thoroughly before it is integrated into the system and the test results documented | 1 | | × | | × | ○ | D | Difficulties after integration / Change management |
| *gc* | A formal description of the test inputs and results (test protocol) should be produced | 1 | | 0 | | × | | D | Duplication of work / Speeding up licensing |
| *gd* | Coding errors which are detected during program testing should be recorded and analyzed | 3 | | | | × | | D | / Detection of some design errors |
| *ge* | Incomplete testing should be recorded | 3 | | | | × | | D | / Clarity |
| *gf* | In order to facilitate the use of intermediate test results during final validation the former degree of testing achieved should be recorded (e.g. all paths through the module tested) | 2 | | × | | | | D | Duplication of work / |

B5 **Language dependent recommendations**
B5.*a* *Sequences and arrangements*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *a* | Detailed rules shall be elaborated for the arrangement of various language constructs | 2 | | | | | | C | / Getting a uniform and intelligible shape of program listings |

**B5 Recommandations dépendant du langage**

*B5.a Organisation du programme*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *aa* | Les recommandations devraient inclure la séquence de déclarations | 2 | | | | | | C | |
| *ab* | La séquence d'initialisation | 2 | | | | | | C | |
| *ac* | La séquence de code non exécutable / de code exécutable | 2 | | | | | | C | |
| *ad* | La séquence de types de paramètres | 2 | | | | | | C | |
| *af* | La séquence de formats | 2 | | | | | | C | |
| | *B5.b Commentaires* | | | | | | | | |
| *b* | Les relations entre les commentaires et le code exécutable ou non doivent être fixées par des règles détaillées | 3 | | | | | | C | Difficultés à la fois dans l'écriture et la compréhension des commentaires / Obtention de commentaires significatifs |
| *ba* | Ce qui doit être commenté devrait être clairement précisé | 3 | | | | | | C | |
| *bb* | La position des commentaires devrait être uniforme | 3 | | | | | | C | |
| *bc* | La forme et le style des commentaires devraient être uniformes | 3 | | | | | | C | |
| | *B5.c Assembleur* | | | | | | | | |
| *c* | Si un langage assembleur est utilisé, des règles de programmation plus complètes devront être suivies | 1 | | | × | | | C | Difficultés de programmation en assembleur / Eviter les astuces |
| *ca* | Les instructions de branchements faisant appel à une substitution d'adresse ne doivent pas être utilisées. Le contenu des tables de branchement devrait être constant | 1 | | | × | | × | C | Les branchements dont la destination ne peut être identifiée depuis le code au point de branchement / |

**B5 Language dependent recommendations**

*B5.a Sequences and arrangements*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *aa* | The recommendations should include sequence of declarations | 2 | | | | | | C | |
| *ab* | Sequence of initializations | 2 | | | | | | C | |
| *ac* | Sequence of non-executable code/executable code | 2 | | | | | | C | |
| *ad* | Sequence of parameter types | 2 | | | | | | C | |
| *af* | Sequence of formats | 2 | | | | | | C | |
| | *B5.b Comments* | | | | | | | | |
| *b* | Relations between comments and executable or non-executable code shall be fixed by detailed rules | 3 | | | | | | C | Difficulties with both writing and understanding comments / Getting meaningful comments |
| *ba* | It should be made clear what has to be commented | 3 | | | | | | C | |
| *bb* | The position of comments should be uniform | 3 | | | | | | C | |
| *bc* | Form and style of comments should be uniform | 3 | | | | | | C | |
| | *B5.c Assembler* | | | | | | | | |
| *c* | If an assembler language is used, extended programming rules shall be followed | 1 | | | × | | | C | Difficulties of assembler programming / Avoiding tricks |
| *ca* | Branching instructions using address substitution must not be used. Branch table contents should be constant | 1 | | | × | | × | C | Branches whose goal cannot be identified from the code at the branching point / |

B5 **Recommandations dépendant du langage**
B5.*c* *Assembleur*

| Art. | Recommandation | Priorité ou importance | Réalisé par ou dans, ou avec l'aide de | | | | | | Bon contre / Bon pour |
|---|---|---|---|---|---|---|---|---|---|
| | | | Système d'exploitation | Programme spécial | Programme d'application | Document | Matériel | Contrôle | |
| *cb* | Tous les adressages indirects devraient suivre le même cheminement | 1 | | | × | | | C | / Clarté de l'adresse mémoire finalement atteinte |
| *cc* | Les décalages indirects devraient être évités | 1 | | | × | | × | C | / Voir immédiatement la destination du décalage |
| *cd* | Les substitutions multiples ou les indexations multiples dans une même instruction machine devraient être évitées | 1 | | | × | | × | C | / Facilité pour trouver l'adresse finale |
| *ce* | La même macro devrait toujours être appelée par le même nombre de paramètres | 2 | | | × | | | C | / Compréhension de la fonction de la macro |
| *cf* | La référence aux étiquettes devrait être faite par des noms plutôt que par des adresses absolues ou relatives | 1 | | | × | | | C | / Association d'une signification à la destination du branchement |
| *cg* | Les conventions d'appel aux sous-programmes devraient être uniformes à l'intérieur du système et spécifiées par des règles additionnelles | 1 | | | × | | | C | Types et emplacements des paramètres arbitraires / |
| | B5.*d* *Règles de codage* | | | | | | | | |
| *d* | Des règles détaillées de codage doivent être publiées | 3 | | | | | | | / Forme du listage de programme uniforme et compréhensible |
| *da* | L'indentation des lignes de code doit être claire (où, comment) | 3 | | | | | | C | / Identification des blocs |
| *db* | L'arrangement des modules devrait être fixé uniformément | 3 | | | | | | C | / Compréhension de la structure en module |
| *dc* | Des détails supplémentaires devraient être formalisés par des règles selon les besoins | 1 | | | | | | H | |

**B5 Language dependent recommendations**

B5.*c* *Assembler*

| Item | Recommendation | Priority or importance | Realization by or in, or with the help of | | | | | | Good against / Good for |
|---|---|---|---|---|---|---|---|---|---|
| | | | Operating system | Special program | User program | Program document | Hardware | Check | |
| *cb* | All indirect addressing should follow the same scheme | 1 | | | × | | | C | / Clarity of ultimately addressed memory locations |
| *cc* | Indirect shifting should be avoided | 1 | | | × | | × | C | / Seeing immediately how far shift goes |
| *cd* | Multiple substitutions or multiple indexing within a single machine instruction should be avoided | 1 | | | × | | × | C | / Ease of finding ultimately addressed locations |
| *ce* | The same macro should always be called with the same number of parameters | 2 | | | × | | | C | / Understanding the macro's function |
| *cf* | Labels should be referred to by names rather than by absolute or relative addresses | 1 | | | × | | | C | / Associating a meaning with the branching goal |
| *cg* | Subroutine call conventions should be uniform throughout the system and specified by further rules | 1 | | | × | | | C | Arbitrary parameter types and locations / |
| | B5.*d* *Coding rules* | | | | | | | | |
| *d* | Detailed coding rules shall be issued | 3 | | | | | | | / Uniform and understandable shape of program listing |
| *da* | It should be made clear, where and how code lines are to be indented | 3 | | | | | | C | / Identification of blocks |
| *db* | Module layout should be fixed uniformly | 3 | | | | | | C | / Understanding module structure |
| *dc* | Further details should be regulated according to need | 1 | | | | | | H | |

# ANNEXE C

## GRANDES LIGNES DES SPÉCIFICATIONS DU LOGICIEL

C1. **Informations générales**

- C1.1 Sommaire et référence à la méthodologie de conception
- C1.2 Sommaire et réference aux exigences fonctionnelles du logiciel
- C1.3 Référence aux spécifications du matériel
- C1.4 Référence au programme de validation du logiciel

C2. **Niveau de conception du système**

- C2.1 Description générale du système
- C2.2 Flux de contrôle entre système et sous-systèmes
- C2.3 Interfaces externes
- C2.4 Définition des données globales
- C2.5 Diagnostic
- C2.6 Reprise
- C2.7 Référence au programme de test d'intégration

C3. **Niveau de conception des sous-systèmes**

Pour chaque sous-système:

- C3.1 Fonction
- C3.2 Traçabilité par rapport aux exigences fonctionnelles du logiciel
- C3.3 Flux de contrôle, interface avec les autres modules et limitation de branchement
- C3.4 Conditions de fonctionnement
  - Interruption
  - Possibilité de translation
- C3.5 Algorithmes
- C3.6 Exigences en matière de données
  - Entrée/sortie
  - Internes
- C3.7 Exigences en matière de performances
  - Précision
  - Exigences temporelles
  - Contraintes imposées par le matériel
  - Contrôle de sécurité et d'accès
- C3.8 Détection d'erreurs par codage
- C3.9 Initialisation
- C3.10 Référence au programme de test du module ou du sous-système
- C3.11 Exigence en matière d'interface avec la console d'opérateur

C4. **Spécification de la base de données**

- C4.1 Description
  - C4.1.1 Identification

# APPENDIX C

## OUTLINE FOR THE SOFTWARE PERFORMANCE SPECIFICATION

C1. **General information**

- C1.1 Summary and reference to design methodology
- C1.2 Summary and reference to software functional requirements
- C1.3 Reference to hardware performance specifications
- C1.4 Reference to software validation plan

C2. **System design level**

- C2.1 System survey
- C2.2 System/subsystem control flow
- C2.3 External interfaces
- C2.4 Global data definition
- C2.5 Diagnostics
- C2.6 Recovery
- C2.7 Reference to integrated testing plan

C3. **Subsystem design level**

For each subsystem:

- C3.1 Function
- C3.2 Traceability to software functional requirements
- C3.3 Control flow, interface to other modules and branch limitation
- C3.4 Operating conditions
  - Interruptability
  - Relocatability
- C3.5 Algorithm
- C3.6 Data requirements
  - Input/output
  - Internal
- C3.7 Performance requirements
  - Accuracy
  - Timing
  - Hardware imposed constraints
  - Security and access controls
- C3.8 Encoded error detection
- C3.9 Initialization
- C3.10 Reference to module or subsystem test plan
- C3.11 Requirements for interface with operator console

C4. **Data base specification**

- C4.1 Description
  - C4.1.1 Identification

C4.1.2 Convention

C4.1.3 Description de l'organisation

C4.2 Caractéristiques logiques

Pour chacun des éléments:

C4.2.1 Identification

C4.2.2 Définition

C4.2.3 Relations avec les autres éléments

C4.2.4 Contrôle d'accès et sécurité

C4.1.2 Conventions

C4.1.3 Survey of organization

C4.2 Logical characteristics

For each unique set:

C4.2.1 Identification

C4.2.2 Definition

C4.2.3 Relationship to other sets

C4.2.4 Access control and security

# ANNEXE D

## LANGAGE, TRADUCTEUR, ÉDITEUR DE LIENS, ETC.

Pour un langage lié à la sûreté, son traducteur et éditeur de liens, les recommandations détaillées suivantes sont données en plus de celles figurant dans la partie principale de cette norme. Elles sont aussi applicables aux autres programmes additionnels du système. Les recommandations pour les traducteurs s'appliquent également aux interpréteurs, compilateurs croisés et émulateurs. De la même façon, les aspects qui s'appliquent aux traducteurs et éditeurs de liens devraient être pris en compte lors de la sélection des outils de spécification formelle et de conception et lors de leur emploi. Dans le cadre d'un projet, une sélection des critères recommandés doit être faite selon les priorités indiquées.

### D1. **Généralités**

| Article | Recommandations | Priorité |
|---|---|---|
| *a* | Traducteur, éditeur de liens et chargeur devraient être largement testés avant leur utilisation; leur fonctionnement est considéré comme très important | 1 |
| *b* | Les données de fiabilité de qualité suffisante en ce qui concerne le traducteur, l'éditeur de liens et le chargeur, devraient être disponibles | 2 |
| *c* | Dans le cas où des programmes additionnels du système sont utilisés tels que des aides, des systèmes de documentation ou équivalent, ils devraient être testés de manière appropriée, avant utilisation | 1 |
| *d* | La syntaxe du langage devrait être définie complètement et de façon non ambiguë | 2 |
| *e* | La sémantique devrait être complètement et correctement spécifiée et compréhensible | 1 |
| *f* | Les langages de haut niveau devraient être utilisés de préférence aux langages orientés machine | 2 |
| *g* | La diffusion d'un langage et son adéquation avec le problème sont toutes deux considérées comme importantes | 2 |
| *h* | En général, les recommandations de l'annexe B devraient, autant que possible, être suivies | 1 |
| *i* | La lisibilité du code produit est plus importante que la facilité d'écriture durant la programmation | 2 |
| *k* | Les notations syntaxiques devraient être uniformes; pour un même concept, une seule notation devrait être permise | 2 |
| *l* | Le langage devrait éviter les caractéristiques susceptibles d'entraîner des erreurs | 2 |
| *m* | Les programmes produits devraient être faciles à maintenir | 2 |
| *n* | Les paramètres d'entrée, les paramètres de sortie et les paramètres intermédiaires devraient être distincts du point de vue de la syntaxe | 2 |
| *o* | Une sortie optionnelle pouvant être examinée, devrait être fournie à toutes les étapes du processus de traduction | 3 |

# APPENDIX D

## LANGUAGE, TRANSLATOR, LINKAGE EDITOR, ETC.

For a safety related application of a language, its translator and linkage editor, the following more detailed recommendations are given in addition to those from the main part of this standard. These recommendations also apply to any other auxiliary system program. Recommendations for translators apply likewise to interpreters, cross compilers and emulators. Similarly the aspects that apply to translators and linkage editors should be recognized during the selection and formal specification of design tools and their use. A project should select recommended criteria according to the priority indicated.

### D1. **General**

| Item | Recommendations | Priority |
|---|---|---|
| *a* | Translator, linkage editor and loader should be thoroughly tested prior to use; operation is considered very important | 1 |
| *b* | Reliability data of sufficient quality about translator, linkage editor and loader should be available | 2 |
| *c* | In cases where auxiliary system programs are used such as aids, documentation systems and the like, they should be appropriately tested before being employed | 1 |
| *d* | Language syntax should be completely and unambiguously defined | 2 |
| *e* | Semantics should be well and completely specified and understandable | 1 |
| *f* | High level languages should be used rather than machine-oriented ones . | 2 |
| *g* | Both the spread of a language and its problem adequacy are considered important | 2 |
| *h* | The recommendations of Appendix B should be supported in general and as far as possible | 1 |
| *i* | Readability of produced code is more important than writeability during programming | 2 |
| *k* | Syntactic notations should be uniform; for the same concept not more than one notation should be allowed | 2 |
| *l* | The language should avoid error prone features | 2 |
| *m* | Produced programs should be easy to maintain | 2 |
| *n* | Input parameters, output parameters and transient parameters should be syntactically distinct | 2 |
| *o* | An optional output that is reviewable should be provided at all stages of the translation process | 3 |

## D2. Traitement des erreurs

| Article | Recommandations | Priorité |
|---|---|---|
| *a* | Le traducteur de langage et l'éditeur de liens devraient avoir la possibilité de détecter autant d'erreurs de programmation que possible durant les phases de traduction ou d'exécution | 2 |
| *b* | Durant l'exécution, le traitement d'exception devrait être possible | 2 |
| *c* | Le langage devrait permettre les assertions | 3 |
| *d* | Les erreurs susceptibles de causer une exception durant l'exécution devraient inclure: | |
| | — les excès des limites de zone; | 1 |
| | — les excès des gammes de valeur; | 1 |
| | — l'accès à des variables non initialisées; | 3 |
| | — le non respect des assertions; | 2 |
| | — la troncature de chiffres significatifs des valeurs numériques; | 2 |
| | — le passage de paramètres de type erroné | 1 |
| *e* | Si une erreur quelconque est détectée durant la traduction ou l'édition de liens, elle devra être indiquée sans aucune tentative de correction | 2 |
| *f* | S'il existait des doutes qu'une règle quelconque ait été violée, un avertissement devrait être fourni | 3 |
| *g* | Durant la traduction, les types de paramètre devraient être testés | 3 |

## D3. Traitement des données et des variables

| Article | Recommandations | Priorité |
|---|---|---|
| *a* | Le domaine de chaque variable devrait être déterminé au moment de la traduction | 1 |
| *b* | La précision de chaque variable à virgule flottante et de chaque expression devrait être déterminée lors de la traduction | 2 |
| *c* | Aucune conversion implicite entre types ne devrait avoir lieu | 2 |
| *d* | Le type de chaque variable, zone, élément d'enregistrement, expression, fonction et paramètre devrait être déterminable au moment de la traduction | 2 |
| *e* | Les variables, zones, paramètres, etc., devraient être déclarés explicitement, en incluant leurs types | 1 |
| *f* | Les types de variables devraient être décomposés entre entrée, sortie, procédure intermédiaire et paramètres de sous-programme | 2 |
| *g* | Les noms de variables de longueur arbitraire devraient être permis | 2 |
| *h* | Autant que possible, les tests de types devraient avoir lieu durant la traduction plutôt que lors de l'exécution | 3 |
| *i* | Au moment de la traduction, il faudrait vérifier si une affectation était autorisée pour un élément particulier de donnée | 2 |

## D4. Aspects temps réel

| Article | Recommandations | Priorité |
|---|---|---|
| *a* | Durant l'évaluation d'expression, l'assignation externe d'une variable quelconque qui est accessible à l'intérieur de l'expression ne devrait pas être permise | 2 |
| *b* | Les temps de traitement devraient être examinables en direct | 3 |
| *c* | Il convient de prévoir la détection d'erreur en temps réel (D2.*d*) | 1 |

## D2. Error handling

| Item | Recommendations | Priority |
|---|---|---|
| *a* | Language translator and linkage editor should provide for detection of as many programming errors as possible during translation time or on-line execution | 2 |
| *b* | During on-line execution, exception handling should be possible | 2 |
| *c* | The language may provide assertions | 3 |
| *d* | Errors likely to cause an exception during execution time should include: | |
| | — exceeding of array boundaries; | 1 |
| | — exceeding of value ranges; | 1 |
| | — access to variables that are not initialized; | 3 |
| | — failing to satisfy assertions; | 2 |
| | — truncation of significant digits of numerical values; | 2 |
| | — passing of parameters of wrong type | 1 |
| *e* | If any error is detected during translation or linkage time, it should be reported without any attempt at correction | 2 |
| *f* | If it is not clear whether any rules have been violated, a warning should be issued | 3 |
| *g* | During translation time, parameter types should be checked | 3 |

## D3. Data and variable handling

| Item | Recommendations | Priority |
|---|---|---|
| *a* | The range of each variable should be determinable at translation time | 1 |
| *b* | The precision of each floating point variable and expression should be determinable at translation time | 2 |
| *c* | No implicit conversion between types should take place | 2 |
| *d* | The type of each variable, array, record component, expression, function and parameter should be determinable at translation time | 2 |
| *e* | Variables, arrays, parameters etc. should be explicitly declared, including their types | 1 |
| *f* | Variable types should distinguish between input, output, transient procedure and subroutine parameters | 2 |
| *g* | Variable names of arbitrary length should be allowed | 2 |
| *h* | As far as possible, type checking should take place during translation time rather than execution time | 3 |
| *i* | At translation time, it should be checked whether an assignment is allowed to any particular data item | 2 |

## D4. On-line aspects

| Item | Recommendations | Priority |
|---|---|---|
| *a* | During expression evaluation, external assignment should not be allowed to any variable that is accessible in the scope of the expression | 1 |
| *b* | Used computing time should be examinable on-line | 3 |
| *c* | On-line error capture should be provided (D2.*d*) | 1 |

# ANNEXE E

## TEST DU LOGICIEL

### E1. Activités liées au test du logiciel

Le but de cet article est de fournir des directives pour le test du logiciel. Selon le programme à tester, les différentes sortes de tests sont plus ou moins efficaces dans la détection des erreurs de programme. Un ensemble de tests complémentaires peut être utilisé comprenant une combinaison de tests statistiques et systématiques. Les données sélectionnées statistiquement pourraient être appliquées dans la recherche de cas oubliés.

Des techniques complémentaires, telles que l'inspection du code en manuel et l'utilisation de preuves mathématiques, peuvent être utilisées. Pour chaque application, le programme de vérification doit inclure ces différentes techniques et les mettre en œuvre dans l'ordre souhaitable.

Un examen des méthodes possibles est donné dans le tableau E4.1. Quand cela est nécessaire, des approches et des critères différents pour la sélection des données de tests à l'intérieur d'une approche devraient être choisis pour les différentes parties d'un programme, afin de pouvoir s'assurer qu'une partie est bien exempte d'erreur, ou que la probabilité qu'il puisse y avoir une erreur est en dessous d'un certain niveau d'acceptation. La sélection dépend de la structure interne de la partie du programme, du niveau de fiabilité exigé, des demandes qui lui sont faites durant le fonctionnement de la centrale et des outils de test disponibles.

Le test du logiciel devrait prendre en considération les différents niveaux de conception du logiciel (par exemple niveau module, niveau sous-système et niveau système).

### E2. Approches systématiques

Chaque module devrait être testé de manière systématique selon des critères de test bien définis spécifiant ce qui doit être testé (toutes les instructions, tous les arcs, tous les chemins). Les outils automatiques de tests devraient être utilisés autant que possible dans le déroulement des différents cas de tests. Les résultats des tests sont contrôlés par rapport à des résultats attendus déduits directement des spécifications du module de programme. Les paramètres devraient être les entrées et les sorties du module de la même façon que dans le système de sûreté.

Selon le tableau E4.2, des classes d'erreurs résiduelles possibles peuvent être déterminées et, en conséquence, il sera finalement possible de poursuivre le test en utilisant une approche différente.

Le tableau E4.2 est considéré comme une liste de vérification qui peut être interprétée selon les besoins de chaque cas particulier. A cause de la très grande variété de cas pratiques, il n'est pas possible de recommander une quelconque combinaison de tests pour une classe particulière d'applications. Il est cependant indiqué quels sont les tests qui devraient être réalisés dans tous les cas. D'autre part, il est évident que, ni toutes les combinaisons de tests, ni même tous les tests individuels ne peuvent être effectués dans une application particulière.

Au niveau sous-système, le logiciel est partiellement intégré dans le système. Les tests au niveau sous-système témoignent de l'intégration correcte des modules de logiciel. Une distinction doit être faite entre les cas où une dépendance du flot de données existe entre les modules logiciels et les cas où elle n'existe pas. Les essais doivent être faits pour fournir l'assu-

# APPENDIX E

## SOFTWARE TESTING

### E1. Software testing activities

This clause is intended to provide guidance for software testing. Depending on the program to be tested, the individual kinds of tests are more or less effective in finding program errors. A set of complementary testing approaches may be used, including a combination of statistical and systematic testing. Statistically selected data could be applied to search for forgotten cases.

Supplementary (manual) techniques, however, will be code inspection (desk checking) and use of mathematical proofs. For each application the verification plan may include these different techniques and put them into a suitable order.

A review of possible methods is given in Table E4.1. When necessary, different approaches and different criteria for test data selection within one approach should be chosen for different program parts in order to determine whether a particular part is free from errors or determine the probability that it fails is below a certain acceptance level. The selection depends on the internal structure of a program part, the level of reliability required, the demands imposed on it during plant operation and the available testing tools.

Software testing should give consideration to different levels of software design (for example module, subsystem and system levels).

### E2. Systematic approaches

Each module should be tested in a systematic way according to well-defined test criteria which specify what is to be tested (e.g. all statements, all arcs, all paths). Automatic testing tools should be used as much as possible in deriving test cases. Test results are checked with expected results derived from the program module specifications. Parameters should be input to and output from the module in the same manner as in the safety system.

According to Table E4.2 the classes of possibly remaining errors can be determined and as a consequence it will be possible eventually to proceed with the testing using a different approach.

Table E4.2 is a checklist, which can be interpreted according to the needs of each special case. Due to the enormous variety of possible practical cases it is not possible to recommend any combination of the tests indicated for a particular class of applications. It is, however, indicated which tests should be performed under all circumstances. On the other hand, it is evident that neither all combinations of all tests nor all individual tests can be executed in a particular application.

At the subsystem level, the software is partially integrated into the system. The subsystem level tests assure the proper integration of the software modules. A distinction must be made between the case in which data flow dependency exists among the software modules and the case in which it does not. Testing shall be done to provide assurance that all independent arcs

rance que tous les arcs indépendants dans le sous-système sont correctement suivis. Les tests aux frontières des domaines d'entrée et aux limites de la région opérationnelle du module devraient être effectués.

Le même jeu d'essais utilisé au niveau module devrait être effectué. Les résultats devraient être testés par rapport à des valeurs précalculées. Les paramètres devraient être les entrées et les sorties du sous-système de la même manière que dans le système de sûreté. Dans la mesure où cela est réalisable pratiquement, les tests au niveau sous-système devraient être effectués sur le matériel définitif.

Au niveau système, le logiciel est complètement intégré dans le matériel. Le test du système témoigne de l'intégration correcte des sous-systèmes. Le test devrait être réalisé en faisant fonctionner les programmes ensemble avec un simulateur du contexte, ou avec une version réaliste du système qui est à commander ou à contrôler par les systèmes de sûreté.

Ce test du système complet devra être réalisé en accord avec les exigences en matière de performances données dans les spécifications fonctionnelles. Au-delà des activités de test décrites ci-dessus, un test systématique des performances en temps devrait être réalisé.

E3. **Approches statistiques**

Les approches statistiques devraient être utilisées pour compléter les approches systématiques.

Les conditions générales préalables à des approches statistiques sont les suivantes:

— l'ordre et le nombre de tests réalisés ne doivent pas influencer le résultat donné par un seul test;

— le passage d'un test est supervisé de manière parfaite afin que toutes les fautes de déroulement ou les erreurs puissent être détectées;

— le nombre de tests est grand, par exemple plus de 1 000;

— pratiquement aucune erreur ou panne ne sont détectées.

Les approches statistiques sont réalisées pour:

— tester le corps des boucles, étant donné que la structure de la boucle a été systématiquement analysée;

— la vérification de fonctions standard fournies par des personnes ou par des organismes extérieurs à ceux qui ont la charge du développement du logiciel, si on dispose de suffisamment de données opérationnelles de fonctionnement;

— la vérification des systèmes d'exploitation qui sont par nature difficiles à analyser à cause de leur nature interactive et de leur conception optimisée, si on dispose de suffisamment de données opérationnelles de fonctionnement;

— la vérification de l'interface entre des programmes standard et des logiciels nouvellement développés, s'il n'est pas possible d'analyser systématiquement l'interface de manière exhaustive.

Dans le tableau E4.3, les exigences préliminaires repérées signifient que l'on suppose

[A1]: qu'aucune erreur ou panne n'a été détectée durant «n» tests;

[A2]: que la connaissance du contenu interne des programmes est nécessaire à partir des résultats de l'analyse du programme.

Selon les résultats des tests systématiques, les tests statistiques devraient être choisis avec soin. Normalement, un mélange approprié devra être la solution la meilleure pour chaque cas particulier.

in the subsystem are correctly followed. Test cases at the borders of input domains and at the limits of module operational region should be used.

The same test data set utilized at module level should be used. Results should be checked with pre-calculated values. Parameters should be input and output from the subsystem in the same manner as in the safety system. Where practical, the subsystem level testing should involve the actual hardware.

At the system level, the software is completely integrated in the hardware. The system test assures proper integration of subsystems. Testing should be performed by running programs together with a realistic simulator, or with a realistic version of the system to be monitored or controlled by the safety systems.

This test of the complete system shall be performed in accordance with the performance statements given in the functional requirements specification. Beyond the testing activities described above a systematic timing check should be performed.

E3. **Statistical approaches**

Statistical approaches should be utilized to complement systematic approaches.

The general assumptions behind statistical approaches are:

— sequence and number of test runs does not influence the result of a single test run;

— the test run is supervised so perfectly that each faulty run or failure is detected as such;

— number of test runs is large, for example more than 1 000;

— nearly no errors or failures are detected.

Statistical approaches are performed for:

— testing loop bodies, given that the loop structure has been systematically analyzed;

— the verification of standard functions provided by individuals or organizations other than the software developer, if sufficient operational data are available;

— the verification of operating systems which are inherently difficult to analyze due to their interactive nature and optimized design, if sufficient operational data are available;

— the verification of the interface between standard programming and newly developed software, if it is not possible to systematically analyze this interface exhaustively.

In Table E4.3, the prerequisites marked:

[A1] means it is assumed that no errors or failures are detected during "n" test runs;

[A2] means it is assumed that knowledge about program internals is needed from results of program analysis.

Dependent on the results of the systematic tests, the statistic tests should be selected carefully. Normally an appropriate mixture will be best for each specific case.

## E4. Méthodes de test

E4.1 *Synoptique des méthodes de vérification*

| N° | Méthode | But | Avantages principaux | Problèmes lors de l'utilisation, principaux désavantages | Coût et effort comparés au coût de développement | Résultats | Relations avec les autres méthodes | Evaluation |
|---|---|---|---|---|---|---|---|---|
| 1 | Surveillance de la procédure de test | Obtenir un profil de fiabilité sans efforts supplémentaires | — Efforts supplémentaires réduits<br>— Connaissance détaillée de l'objet soumis au test non nécessaire | — Profil de fiabilité obtenu insuffisant<br>— Les situations pratiques ne suivent que grossièrement la théorie | Faible | Informations statistiques par exemple: MTBF | Utilise la théorie des probabilités comme n° 2 | Difficile à utiliser pour les applications de sûreté |
| 2 | Test statistique | Obtenir un profil de fiabilité sans examiner les détails du code | Connaissance détaillée de l'objet soumis au test non nécessaire | | — Le nombre de cas de test est très grand pour obtenir des résultats significatifs<br>— Le coût des tests peut être très supérieur au coût du développement | | Même type de raisonnement que n° 1 | Peut être raisonnablement appliqué sous une forme réduite en complément à l'analyse du programme |
| 2.1 | Par rapport à la demande | | | — Fournir un profil de test qui représente les états d'entrée avec la même probabilité qu'en exploitation | | Informations statistiques, par exemple: disponibilité en fonction de la demande ou risque en fonction de la durée de vie du programme | | |
| 2.2 | Par rapport à la qualité | | | — Fournir un profil de test qui touche tous les aspects des programmes avec la même probabilité | | Informations statistiques, par exemple taux d'erreur résiduel | | |

## E4. Testing methods

E4.1 *Overview of verification method*

| No. | Method | Aim | Major advantages | Problems during application, major disadvantages | Cost and effort compared with development cost | Result | Relationship to other methods | Assessment |
|---|---|---|---|---|---|---|---|---|
| 1 | Supervision of testing procedure | To derive a reliability figure without additional effort | — Small additional effort for application<br>— No detailed knowledge from test object required | — Reliability figures to be gained not sufficient<br>— Practical cases behave only roughly according to theory | Small | Probabilistic figure, e.g. MTBF | Uses probability theory as No. 2 | Difficult to use for safety applications |
| 2 | Statistical testing | To derive a reliability figure without looking into the code's details | No detailed knowledge from test object required | | — Number of required test cases very high, if meaningful results are to be obtained<br>— Cost for test can be much higher | | Same kind of reasoning as No. 1 | Can be reasonably applied in a short form complementary to program analysis |
| 2.1 | According to demand | | | — To provide a test profile that represents input states with the same probability as the on line operation | | Probabilistic figure, e.g. availability per demand or risk per program life time | | |
| 2.2 | According to correctness | | | — To provide a test profile that hits any program property with equal probability | | Probabilistic figure, e.g. limit for still contained errors | | |

E4.1 *Synoptique des méthodes de vérification*

| N° | Méthode | But | Avantages principaux | Problèmes lors de l'utilisation, principaux désavantages | Coût et effot comparés au coût de développement | Résultats | Relations avec les autres méthodes | Evaluation |
|---|---|---|---|---|---|---|---|---|
| 3 | Preuve de programme | Fournir une base de comparaison avec les spécifications | Données rigoureuses sur la qualité de la réalisation | Pas de formalisme disponible permettant d'établir les assertions ou les invariants de boucle; opération manuelle soumise à un risque d'erreurs | Du même ordre de grandeur ou supérieure | Programmes corrects ou incorrects, dans la mesure où la preuve est correcte | Quelques aspects à utiliser lors de l'analyse | Probablement la seule solution raisonnable pour les boucles générales «TANT QUE» |
| 4 | Analyse de programme | | Donne de bons résultats avec des programmes simples | Certaines structures de programme sont difficiles à analyser | | Exempts de certains types d'erreurs spécifiques dans la mesure où l'analyse a été suffisamment poussée | Une démarche rappelant celle des preuves de programme est utilisée | |
| 4.1 | Analyse manuelle | Fournir une base pour des tests significatifs ou pour une comparaison avec les spécifications | Peut être effectuée sans outils supplémentaires | Opération manuelle soumise à un risque d'erreurs | Légèrement moins que le coût de développement | | | Devrait être utilisée si les outils automatiques ne sont pas disponibles |
| 4.2 | Analyse automatique | Comme ci-dessus parfois restreint à ne fournir que quelques vagues indications de qualité | — Travail manuel limité<br>— Résultats obtenus rapidement | — De nombreux outils demandent du travail manuel supplémentaire<br>— Certains résultats de tels outils sont difficiles à interpréter | Fonction des outils disponibles beaucoup plus faible que le coût de développement | | | Devrait être utilisée le plus largement possible<br>*Approche la plus prometteuse* |