REPUBLIQUE FRANÇAISE

AUTORITE DE SURETE NUCLEAIRE

Direction de la sûreté des installations nucléaires

Paris, le 15 MAI 2000

DSIN/GRE/SD2/N° 0088/2000
Affaire suivie par : A.-M. Lapassat
☎ 01 43 19 71 03

Objet : **Décision relative à l'adoption de la Règle fondamentale de sûreté RFS II.4.1.a relative aux logiciels des systèmes électriques classés de sûreté.**

La Règle fondamentale de sûreté RFS II.4.1.a, relative aux logiciels des systèmes électriques classés de sûreté, dont le texte est annexé à la présente décision est adoptée.

le Directeur de la Sûreté des Installations Nucléaires

A.C. ~~LACOSTE~~

99, rue de Grenelle - 75353 PARIS 07 SP Tél 01 43 19 36 36 Fax 01 43 19 48 69
MINISTERE DE L'ECONOMIE, DES FINANCES ET DE L'INDUSTRIE - SECRETARIAT D'ETAT A L'INDUSTRIE
MINISTERE DE L'AMENAGEMENT DU TERRITOIRE ET DE L'ENVIRONNEMENT

REPUBLIQUE FRANÇAISE

AUTORITE
DE SURETE
NUCLEAIRE

Direction de la sûreté
des installations
nucléaires

Paris, le 15 MAI 2000

REGLES FONDAMENTALES DE SURETE RELATIVES AUX REACTEURS A EAU SOUS PRESSION

REGLE II.4.1.a

Chapitre II : Conception des systèmes élémentaires

Objet : Logiciels des systèmes électriques classés de sûreté

Domaine d'application : Rénovation des systèmes de commande et de surveillance des réacteurs de puissance

99, rue de Grenelle - 75353 PARIS 07 SP Tél 01 43 19 36 36 Fax 01 43 19 48 69
MINISTERE DE L'ECONOMIE, DES FINANCES ET DE L'INDUSTRIE - SECRETARIAT D'ETAT A L'INDUSTRIE
MINISTERE DE L'AMENAGEMENT DU TERRITOIRE ET DE L'ENVIRONNEMENT

RÈGLE FONDAMENTALE DE SÛRETÉ II.4.1.a

relative aux

LOGICIELS DES SYSTÈMES ÉLECTRIQUES CLASSÉS DE SÛRETÉ

Novembre 1999

NOTE PRÉLIMINAIRE

Les règles fondamentales de sûreté applicables à certains types d'installations nucléaires de base sont destinées à expliciter les conditions dont le respect est, pour le type considéré d'installations et pour l'objet dont elles traitent, jugé comme valant conformité avec la pratique réglementaire technique française.

Ces règles devraient faciliter les analyses de sûreté et la bonne compréhension entre les personnes intéressées aux questions relatives à la sûreté nucléaire.

Elles ne diminuent en rien la responsabilité de l'exploitant, et ne font pas obstacle aux dispositions réglementaires en vigueur.

Pour tout objet auquel une règle fondamentale de sûreté est applicable, il est possible de ne pas s'y conformer si la preuve est apportée que les objectifs de sûreté visés par la présente règle sont atteints par d'autres moyens, proposés dans le cadre des procédures réglementaires.

La direction de la sûreté des installations nucléaires se réserve à tout moment la faculté de modifier, si cela lui apparaît nécessaire, toute règle fondamentale de sûreté, le cas échéant en précisant les conditions d'applicabilité.

Le groupe permanent d'experts placé auprès du directeur de la sûreté des installations nucléaires, et chargé des réacteurs nucléaires, a été consulté sur la présente règle.

TABLE DES MATIÈRES

Règle Fondamentale de Sûreté

LOGICIELS DES SYSTÈMES ÉLECTRIQUES CLASSÉS DE SÛRETÉ

OBJET :

Principes et exigences à prendre en compte pour la conception, la réalisation, la mise en œuvre et l'exploitation des logiciels des systèmes électriques classés de sûreté.

Domaine d'application :

Tranches nucléaires comportant un réacteur à eau sous pression.

1. Rappel de la pratique réglementaire française

La pratique réglementaire française prévoit que des dispositions appropriées soient prises pour permettre l'arrêt sûr du réacteur, le refroidissement à long terme du combustible et le maintien du confinement des produits radioactifs, dans toutes les conditions de fonctionnement considérées comme plausibles. En particulier, la mise en œuvre du concept de défense en profondeur conduit à établir des classifications des matériels mécaniques, des systèmes électriques, des structures et ouvrages de génie civil en fonction de la sévérité des exigences à respecter pour ceux-ci depuis leur conception jusqu'à leur exploitation.

La règle fondamentale de sûreté IV.1.a (21 décembre 1984) définit les classes qui sont utilisées dans le cadre de la pratique réglementaire française en matière de sûreté. Les systèmes électriques « classés 1E » sont ceux qui, s'agissant des conditions de fonctionnement de dimensionnement :

- d'une part, sont nécessaires pour accomplir les fonctions suivantes :
  - arrêt d'urgence du réacteur,
  - isolement de l'enceinte de confinement,
  - refroidissement de secours du cœur,
  - extraction de la chaleur du réacteur et du bâtiment réacteur,
- d'autre part, permettent de prévenir les accidents ou d'en limiter les conséquences radiologiques.

Dans l'ensemble des systèmes électriques classés de sûreté et non classés 1E, il est courant de distinguer deux sous-ensembles :

- ceux classés au titre des conditions de fonctionnement de dimensionnement,
- ceux classés au titre des conditions de fonctionnement complémentaires.

La règle fondamentale de sûreté IV.2.b (31 juillet 1985) énonce les règles à respecter pour que, s'agissant de l'ensemble des conditions de fonctionnement considérées comme plausibles, les matériels appartenant aux systèmes électriques classés de sûreté soient en mesure d'assurer leurs fonctions de sûreté.

2. Objet de la règle : principes et exigences associés aux logiciels utilisés dans les systèmes classés de sûreté

Les systèmes électriques, au sens de la règle fondamentale de sûreté IV.1.a (§3.1.3.a) peuvent utiliser des logiciels qui réalisent des fonctions de contrôle-commande de ces systèmes. Ils sont alors appelés systèmes programmés. Ces systèmes font l'objet d'exigences en liaison avec les fonctions nécessaires à l'exploitation et à la sûreté de la tranche. Ces exigences doivent faire l'objet d'une répartition entre le matériel et le logiciel du système programmé considéré.

On distingue les étapes suivantes :
- l'expression des besoins, dont le résultat est appelé cahier des charges,
- la spécification du système,
- la conception du système, qui conduit à définir les exigences relatives au matériel et celles relatives au logiciel, sous formes d'une spécification du matériel et d'une spécification du logiciel.

Les documents, associés à ces étapes, servent de référence pour le développement du logiciel dont il est question dans la présente règle fondamentale de sûreté.

Celle-ci a pour objet de préciser les principes et les exigences à respecter pour la conception, la réalisation, la mise en œuvre et l'exploitation des logiciels des systèmes programmés classés de sûreté. Cette règle s'inscrit dans le plan de principe des règles fondamentales de sûreté.

3. Introduction

Des termes, utilisés dans cette règle, font l'objet d'une définition qui est donnée en annexe. Ils sont repérés par un astérisque lors de leur première utilisation.

La règle énoncée au chapitre suivant vise à s'assurer que le logiciel*, une fois assemblé avec le matériel du système programmé, conduira à un comportement* du système conforme aux besoins définis dans son cahier des charges tout en prenant en compte les aspects nécessaires à l'obtention d'un haut degré de fiabilité*. De plus, elle établit les conditions d'une maintenance à même de conforter la qualité initiale obtenue lors du développement du logiciel.

3.1 Particularités d'un logiciel

Le logiciel est un des éléments constitutifs d'un système programmé*. Il est composé :

- de la documentation qui décrit les spécifications, l'architecture du logiciel, le programme* écrit en langage simple (assembleur) ou évolué (langage C par

exemple), les vérifications* et validations*, les outils de développement, les dispositions d'organisation (Plan Qualité Logiciel*) adoptées pour le développement et la maintenance ;
- du programme exécutable, qui est une suite d'éléments binaires (bits) compréhensible uniquement par l'ordinateur pour lequel il a été produit.

L'architecture du logiciel conduit à définir des modules* de logiciel qui réalisent un traitement* spécifique.

Le cycle de développement* d'un logiciel comprend, dans le cas général, les activités suivantes accompagnées de vérifications :
- spécifications,
- conception,
- programmation,
- essais*.

De plus, les modifications, liées aux évolutions des fonctions* d'un logiciel comme à la correction d'erreurs* mises en évidence lors de l'exploitation, sont réalisées par la maintenance de ce logiciel.

Le logiciel ne peut être examiné que sous deux aspects obligatoirement indirects :
- les documents;
- le comportement du programme exécutable, ce qui nécessite un support capable d'interpréter et d'exécuter ce programme.

3.2 Particularités d'un logiciel pour un système programmé classé de sûreté

Comme tout composant, le logiciel d'un système programmé classé de sûreté concourt à l'obtention d'une fiabilité appropriée pour ce système.

Dans le cas général, des essais sont réalisés pour s'assurer que le logiciel répond à ses spécifications. Cet objectif ne peut pas être atteint uniquement par des essais, en dépit de son importance pour les systèmes programmés classés de sûreté. En effet, ceux-ci devraient solliciter le logiciel suivant son profil opérationnel*, c'est-à-dire pour toutes les séquences de combinaisons de valeurs issues des spécifications du logiciel. En pratique, ceci est rarement réalisable, ce qui conduit à élargir la vérification à :
- l'évitement d'erreurs pendant le développement du logiciel, qui conduit à réaliser des efforts de rigueur (équipes indépendantes, formalisation des phases du cycle de développement, revues, vérifications, ...) afin de limiter le nombre d'erreurs;
- l'élimination des erreurs, par exemple par des essais que ce soit au niveau du logiciel (essai de validation d'un logiciel) ou en association avec le matériel pour constituer le système programmé (essai de validation du système);
- la tolérance* des erreurs résiduelles éventuelles, qui a pour objectif de forcer le logiciel à donner une réponse prévue pour toute erreur qui induirait un fonctionnement dangereux du système.

Ces trois concepts fondamentaux ont conduit à l'élaboration des principes et exigences énoncés ci-après.

4. Enoncé de la règle

Les principes, repérés par les lettres Pa (ou Pb), et les exigences, repérées par un numéro précédé respectivement des lettres Ea ou Eb, sont applicables aux logiciels des systèmes programmés classés 1E ou aux logiciels des systèmes programmés classés de sûreté et non classés 1E qui sont nécessaires à la démonstration de sûreté dans les conditions de dimensionnement du réacteur.

Les systèmes programmés classés au titre des conditions complémentaires de fonctionnement doivent faire l'objet d'exigences à définir et à justifier cas par cas.

4.1 Dispositions de principe

Les dispositions de développement et de maintenance concourent à l'obtention d'un programme ayant un comportement vérifiable (tant du point de vue des résultats des traitements que du temps de réponse) et compatible avec les études de sûreté de l'installation, associé à un haut degré de fiabilité. Ceci conduit à adopter l'un des deux principes de conception suivants, en fonction de la classe du système programmé considéré.

4.1.1 Logiciels des systèmes programmés classés 1E

S'agissant de logiciels des systèmes programmés classés 1E, le principe à satisfaire est le déterminisme*.

Pa. Le principe de conception des logiciels des systèmes programmés classés 1E doit permettre de connaître, d'une part la séquence des traitements parcourue pour chaque situation prévue pour le logiciel, d'autre part l'utilisation des ressources*.

Une pratique acceptable pour un logiciel d'un système programmé classé 1E est de le concevoir de telle sorte que, par exemple, il n'utilise aucun mécanisme d'interruption*, que le nombre d'itérations d'un calcul soit fixé de façon constante et qu'un emplacement de la mémoire du système programmé contienne, au cours du temps, une seule information (allocation statique de la mémoire).

4.1.2 Logiciels des systèmes programmés classés de sûreté et non classés 1E

Les systèmes programmés classés de sûreté et non classés 1E sont utilisés notamment pour assurer des fonctions de contrôle-commande pendant une phase post-accidentelle. Ils traitent un nombre important d'informations. Le concepteur de ces systèmes peut alors être amené à choisir pour les réaliser des mécanismes de traitement temps réel multitâches* ou à base d'interruptions. Une telle solution conduit à ne plus pouvoir déterminer une séquence de traitements et un temps de réponse pour chaque situation d'exploitation de l'installation.

Dans le cas où le concepteur choisit des mécanismes de traitement temps réel multitâches ou à base d'interruptions, pour réaliser des logiciels de systèmes programmés classés de sûreté et non classés 1E, le principe à satisfaire est alors la prédictibilité*.

Pb. Le principe de conception des logiciels des systèmes programmés classés de sûreté et non classés 1E doit être fondé sur des règles validées, reposant sur des hypothèses ou des modèles* mathématiques relatifs à ces logiciels. La prédictibilité résulte de la démonstration du caractère déterministe des dits hypothèses et modèles.

Ces principes nécessitent la caractérisation correcte du comportement du logiciel dans son environnement* réel, notamment aux conditions limites d'emploi.

4.2 Exigences

Les exigences auxquelles doivent satisfaire les logiciels utilisés dans les systèmes programmés classés de sûreté, définis au début de ce chapitre, sont groupées selon cinq rubriques : les fonctions, la fiabilité, la qualité, l'expérience antérieure* et la maintenance.

4.2.1 Exigences applicables aux logiciels des systèmes programmés classés 1E

4.2.1.1 Fonctions

Ea 1.1. L'exhaustivité des spécifications du logiciel, par rapport aux fonctions du système programmé assignées au logiciel dans la conception de ce système, doit être établie.

L'utilisation de références croisées entre les spécifications du logiciel et les fonctions qui lui ont été attribuées est une pratique acceptable.

Ea 1.2. Les spécifications du logiciel doivent énoncer les missions* du logiciel, qui sont déduites des fonctions du système programmé assignées au logiciel, et les fonctions assurant la permanence des fonctionnements requis pour ce système, qui sont déduites de la tolérance aux défaillances* (d'origine matérielle ou logicielle) et de la robustesse* nécessaires à l'accomplissement de certaines fonctions de ce système.

Ea 1.3. L'absence de contradiction entre le programme exécutable et les spécifications du logiciel doit être établie.

Une pratique acceptable pour cette validation est de réaliser des essais sur la version du programme incluse dans le système programmé.

Ea 1.4. Chaque énoncé de spécification doit être suffisamment précis afin qu'un essai soit réalisable pour contrôler que cet énoncé est correctement traduit dans le programme.

4.2.1.2 Fiabilité

La fiabilité est ici considérée sous un aspect qualitatif.

Ea 2.1. La conception des modules de logiciel et leur documentation doivent permettre l'utilisation de méthodes de vérification et de validation ayant pour objectif de montrer que chaque module fait ce pour quoi il a été spécifié et uniquement cela. Le principe est d'éviter la présence de parties de programme inutilisées, sauf exception justifiée. Dans ce cas, celles-ci doivent être identifiées. Elles doivent être spécifiées, programmées, vérifiées et validées avec le reste du programme.

La vérification s'appuie sur une combinaison adéquate des méthodes et techniques d'analyse statique* par une équipe indépendante de celle chargée du développement. Parmi ces méthodes et techniques, on peut citer :

- la revue critique* de document,
- la relecture de programme,
- l'analyse syntaxique du programme, ...

Une pratique acceptable, pour ce qui concerne les méthodes et techniques de vérification, est présentée dans les chapitres 6 (vérification) et 7 (intégration matériel/logiciel) de la publication 60880 (1986) de la Commission Electrotechnique Internationale (CEI).

De même, la réalisation d'essais est une technique acceptable de validation du programme exécutable, notamment du point de vue de ses performances temporelles. En particulier, l'utilisation de cette technique peut s'appuyer sur les dispositions prévues au chapitre 8 de la publication CEI 60880 (1986).

Les deux pratiques de vérification et de validation citées ci-dessus sont complémentaires.

Ea 2.2. Les règles d'élaboration des essais et des valeurs attendues doivent permettre de montrer que l'étendue des essais est suffisante et que les observations réalisées sont pertinentes.

Pour déterminer l'étendue des essais, une pratique acceptable peut être l'identification des parties structurelles et fonctionnelles activées par les essais. En ce qui concerne la pertinence des observations, l'application des dispositions prévues au chapitre 8 de la publication CEI 60880 (1986) représente une pratique acceptable.

Ea 2.3. Une étude concernant les défaillances potentielles du système programmé ayant pour cause une erreur logicielle et leurs conséquences pour la sûreté doit être réalisée. L'objectif de cette étude est de vérifier que les défaillances du système causées par des erreurs logicielles n'ont pas de conséquence pour la sûreté.

Une pratique acceptable consiste à :

- identifier les différents types d'anomalies à considérer pour le logiciel, quelles qu'en soient les causes (données hors des plages admissibles, débordements de tableaux, divisions par zéro, ...) ;

- localiser les menaces internes au logiciel, c'est-à-dire les possibilités d'occurrence de ces types d'anomalie, compte tenu de la programmation défensive* mise en œuvre ;
- étudier les conséquences de ces anomalies potentielles du logiciel sur le comportement du système programmé (modes de défaillance) ;
- vérifier que ces modes de défaillance n'ont pas de conséquence sur la sûreté du système élémentaire.

Ea 2.4. Les principes mis en œuvre pour assurer la détection et la tolérance des défaillances doivent être spécifiés. Les traitements correspondants doivent être documentés dans le programme.

Ea 2.5. La conception du logiciel doit limiter la propagation d'une anomalie aussi près que possible de sa cause.

Une pratique acceptable en la matière est l'usage pertinent (c'est-à-dire limité par la complexité des traitements à ajouter) de la programmation défensive.

Ea 2.6. Chaque anomalie détectée doit conduire à un fonctionnement du logiciel prévu dans la conception du système programmé.

4.2.1.3 Qualité

Les exigences suivantes précisent, pour ce qui concerne les logiciels, les prescriptions de l'arrêté du 10 août 1984 et de sa circulaire relatifs à la qualité de la conception, de la construction et de l'exploitation des installations nucléaires de base.

Ea 3.1. Les différentes activités du cycle de développement, leurs relations, les produits de chaque activité, leur enchaînement, doivent être définis et leurs modalités pratiques doivent être décrites dans le Plan Qualité Logiciel ou dans tout autre document complémentaire qui doit être mis en référence dans le Plan Qualité Logiciel (par exemple les règles de programmation manuelle).

Ea 3.2. Une revue critique doit terminer chaque étape du développement du logiciel, les étapes faisant l'objet de ces revues étant définies par le Plan Qualité Logiciel.

Ea 3.3. Chaque revue critique doit donner lieu à un rapport.

Ea 3.4. Les rapports de revue critique doivent être archivés pendant une durée qui doit être justifiée.

Ea 3.5. Il doit être établi que les méthodes utilisées pour contrôler les résultats des activités de développement sont suffisantes.

Ea 3.6. Les méthodes de développement doivent s'appuyer sur des règles reconnues pour concourir à la meilleure qualité possible.

Les méthodes et techniques citées dans la publication 60880 (1986) de la CEI constituent une pratique acceptable pour répondre à cette exigence.

Ea 3.7. Dans le cas où des outils sont utilisés comme aide au développement ou au contrôle du logiciel, certaines exigences applicables au logiciel peuvent être remplacées par des exigences sur les fonctionnalités correspondantes des outils. Ces fonctionnalités doivent être définies et validées.

Une pratique acceptable consiste à s'appuyer sur des combinaisons appropriées :

- d'expériences antérieures documentées (cf. § 4.2.1.4) ;
- de tests des outils ;
- de règles d'utilisation des outils ;
- et de vérifications et / ou tests des résultats.

Ea 3.8. La documentation et le programme exécutable du logiciel doivent permettre le contrôle indépendant de la cohérence de chaque représentation du logiciel et de l'absence de contradiction entre les différentes représentations.

Ea 3.9. La documentation concernant chaque activité du cycle de développement du logiciel doit comporter les relations (traçabilité*) entre les éléments décrits par cette documentation et ceux contenus dans les documents des activités situées en amont. Le niveau de traçabilité doit être apprécié en regard de l'exigence Ea 3.8.

Une pratique acceptable peut être de mentionner dans chaque partie d'un document les références au(x) paragraphe(s) du (des) document(s) des activités situées en amont nécessaires à l'établissement des relations visées par cette exigence.

Ea 3.10. Un document de synthèse doit être constitué pour établir la validation du logiciel en regard de ses spécifications.

Ea 3.11. L'archivage de la documentation du logiciel doit permettre la maintenance et l'utilisation du retour d'expérience ultérieure du logiciel, tant par son contenu que par la durée de conservation des documents.

4.2.1.4 Expérience antérieure

L'utilisation de l'expérience antérieure apporte des informations qui complètent la confiance* attribuée au logiciel.

Ea 4.1. Le fabricant d'un logiciel doit faire la preuve d'expériences réussies dans le développement et la maintenance de logiciels répondant à des exigences similaires.

Ea 4.2. Les informations recueillies pour l'utilisation de l'expérience antérieure d'un logiciel doivent provenir de conditions d'utilisation comparables à celles visées.

Ea 4.3. Les informations recueillies pour l'utilisation de l'expérience antérieure doivent être pertinentes pour la démonstration considérée.

Ea 4.4. Les informations recueillies pour l'utilisation de l'expérience antérieure doivent permettre de préciser la nature et les conséquences des défaillances lorsqu'il s'en est produit.

Ea 4.5. La collecte d'informations pour l'utilisation de l'expérience antérieure doit être organisée entre les acteurs (utilisateurs, fournisseurs, fabricants, sous-traitants, ...) et formalisée.

Une pratique acceptable est de constituer un formulaire contenant les différentes natures d'information à recueillir.

4.2.1.5 Maintenance

La maintenance comprend toutes les modifications liées aux évolutions des fonctions confiées aux logiciels et à la correction d'erreurs mises en évidence lors de l'exploitation. Cette activité ne doit pas dégrader la sûreté obtenue avant modification, et doit être réalisée conformément aux exigences des chapitres relatifs aux fonctions, à la qualité et à la fiabilité.

Ea 5.1. Toute modification d'un logiciel doit respecter les exigences relatives à la qualité.

Toutefois, des dispositions particulières peuvent être adoptées, sous réserve de justification, pour la maintenance des systèmes programmés dont les logiciels n'ont pas été développés selon les exigences de la présente règle. Dans ce cas (modification mineure au sens du § 4.3), une pratique acceptable est l'utilisation d'un Plan Qualité Logiciel fondé sur les mêmes principes que celui utilisé lors du développement, avec, le cas échéant, les améliorations issues de la progression de l'état de l'art en génie logiciel, associé à un protocole entre les acteurs de la maintenance (utilisateur du logiciel, fournisseur, fabricant, ...).

Ea 5.2. Il doit être montré, après modification du logiciel, que chaque fonction réalisée correspond à sa spécification (essais de non-régression et de validation).

Ea 5.3. Toute modification d'un logiciel déjà installé et en exploitation ne doit pas conduire à une tolérance aux défaillances et à une robustesse plus faibles que celles existant avant modification.

Une pratique acceptable pour vérifier cette exigence est la réalisation d'une étude des conséquences pour la sûreté de la modification envisagée (cf. Ea 2.3).

4.2.2 Exigences applicables aux logiciels des systèmes programmés classés de sûreté et non classés 1E

Un système programmé classé de sûreté et non classé 1E peut être construit à partir d'un produit existant configurable (appelé Système Numérique de Contrôle-Commande*, ou SNCC) qui comprend lui-même du matériel et du logiciel. Les fonctions du SNCC sont utilisées par des logiciels spécifiques des systèmes élémentaires, appelés logiciels d'application.

Le choix du SNCC est effectué par rapport au cahier des charges du SNCC établi par l'exploitant. Des règles d'utilisation de ce SNCC sont ensuite définies et mises en œuvre pour bâtir une architecture matérielle et développer les logiciels d'application.

Un système programmé classé de sûreté et non classé 1E peut aussi être développé spécifiquement, sans utilisation d'un SNCC. Un tel système programmé peut être réalisé à l'aide de logiciels existants.

Dans le texte qui suit, on entendra :
- par système programmé, soit un SNCC avec l'ensemble de ses logiciels d'application, soit un système programmé développé spécifiquement ;
- par logiciel, soit le logiciel du SNCC avec l'ensemble de ses logiciels d'application, soit le logiciel d'un système programmé développé spécifiquement.

S'il est fait appel à un logiciel existant, il est à considérer dans la suite du texte au même titre que le logiciel d'un SNCC.

4.2.2.1 Fonctions

Eb 1.1. L'exhaustivité des spécifications du logiciel, par rapport aux fonctions du système programmé assignées au logiciel dans la conception de ce système, doit être établie.

Dans le cas où on utilise un SNCC, une pratique acceptable consiste à, successivement, établir :
- au moyen d'une documentation technique suffisante, que le SNCC répond aux besoins exprimés par l'exploitant dans son cahier des charges de SNCC;
- l'exhaustivité des spécifications du logiciel d'application par rapport aux fonctions du système programmé qui lui sont assignées lors de sa conception.

Eb 1.2. Les spécifications du logiciel doivent énoncer les missions du logiciel, qui sont déduites des fonctions du système programmé assignées au logiciel, et les fonctions assurant la permanence des fonctionnements requis pour ce système, qui sont déduites de la tolérance aux défaillances (d'origine matérielle ou logicielle) et de la robustesse nécessaires à l'accomplissement de certaines fonctions de ce système.

Dans le cas où l'on utilise un SNCC, une pratique acceptable consiste à identifier, dans le cahier des charges du SNCC, les caractéristiques attendues de celui-ci pour assurer la permanence des fonctions importantes pour la sûreté.

Eb 1.3. L'absence de contradiction entre le programme exécutable et les spécifications du logiciel doit être établie.

Dans le cas où on utilise un SNCC, une pratique acceptable consiste à établir la cohérence:
- entre le SNCC et sa spécification, en se basant sur les essais menés par le fournisseur, l'expérience antérieure (cf. 4.2.2.4) et, si nécessaire, les essais complémentaires effectués par l'exploitant;
- entre le logiciel d'application et sa spécification, en se basant sur des essais de celui-ci installé sur le SNCC.

Eb 1.4. Chaque énoncé de spécification doit être suffisamment précis afin qu'un essai soit réalisable pour contrôler que cet énoncé est correctement traduit dans le programme.

Dans le cas où on utilise un SNCC, cette exigence s'applique au cahier des charges du SNCC et à la spécification du logiciel d'application.

4.2.2.2 Fiabilité

La fiabilité est ici considérée sous un aspect qualitatif.

Eb 2.1. La conception des logiciels d'application (relatifs à un système élémentaire) et leur documentation doivent permettre l'utilisation de méthodes de vérification et de validation ayant pour objectif de montrer que chaque logiciel fait ce pour quoi il a été spécifié.

Une pratique acceptable consiste à obtenir la confiance, d'une part sur le travail des concepteurs (vérifications des programmes sources), d'autre part sur les processus de traduction en programme exécutable. Cette pratique de vérification est complémentaire de celle préconisée par l'exigence Eb 1.3.

Eb 2.2. Les règles d'élaboration des essais et des valeurs attendues doivent permettre de montrer que l'étendue des essais est suffisante et que les observations réalisées sont pertinentes.

Une pratique acceptable, dans le cas où l'on utilise un SNCC, est la vérification que l'étendue des essais et la pertinence des observations sont suffisantes :

- par l'analyse du plan de validation du SNCC (cf. Eb 1.3),
- et, pour les applications, par un processus d'essais fonctionnels et, si nécessaire, structurels.

Eb 2.3. La prédiction du comportement* du système programmé doit être réalisée (cf. concept de prédictibilité Pb).

Une pratique acceptable est l'application de règles permettant de garantir que le système programmé effectue les actions qui lui sont assignées dans le logiciel d'application, dans les temps requis.

Dans le cas où l'on utilise un SNCC, ces règles s'appliquent à l'architecture du système programmé et aux logiciels d'application. Elles sont basées sur des hypothèses ou sur un modèle de comportement global du SNCC.

Eb 2.4. La représentativité du modèle et la validité des hypothèses (cf. concept de prédictibilité Pb) concernant le logiciel, doivent être établies au regard des éléments du comportement du système programmé considérés comme essentiels pour la sûreté.

Eb 2.5. L'analyse des conséquences des défaillances, effectuée sur le contrôle-commande du réacteur, doit tenir compte des défaillances dont le système programmé peut être l'origine.

Eb 2.6. Les principes mis en œuvre pour assurer la détection et la tolérance des défaillances doivent être décrits.

4.2.2.3 Qualité

Les exigences suivantes précisent, pour ce qui concerne les logiciels, les prescriptions de l'arrêté du 10 août 1984 et de sa circulaire relatifs à la qualité de la conception, de la construction et de l'exploitation des installations nucléaires de base.

Eb 3.1. Les différentes activités de développement, leurs relations, les produits de chaque activité, leur enchaînement, doivent être définis et leurs modalités pratiques doivent être décrites dans le Plan Qualité Logiciel ou dans tout autre document complémentaire qui doit être mis en référence dans le Plan Qualité Logiciel (par exemple les règles de programmation manuelle).

Eb 3.2. Une revue critique doit terminer chaque étape du développement du logiciel, les étapes faisant l'objet de ces revues étant définies par le Plan Qualité Logiciel.

Dans le cas où on utilise un SNCC, une pratique acceptable consiste à :
- tenir une revue critique au terme des processus d'expression des besoins relatifs au SNCC, d'évaluation (conformité aux bonnes pratiques de développement et de vérification, fonctions offertes, programmation défensive, règles de mise en oeuvre, organisation du retour d'expérience, ...) et de choix du SNCC ;
- effectuer des contrôles techniques des logiciels d'application au terme des principales activités de développement et de mise en service.

Eb 3.3. Chaque revue critique doit donner lieu à un rapport.

Eb 3.4. Les rapports de revue critique doivent être archivés pendant une durée qui doit être justifiée.

Eb 3.5. Il doit être établi que les méthodes utilisées pour contrôler les résultats des activités de développement sont suffisantes.

Dans le cas où on utilise un SNCC, une pratique acceptable consiste à :
- statuer, en revue critique (cf. Eb 3.2), sur le caractère suffisant tant des méthodes d'évaluation du SNCC que de ses règles de mise en œuvre ;
- établir que les méthodes de contrôle des résultats des activités de développement du logiciel d'application sont suffisantes.

Eb 3.6. Les méthodes de développement doivent s'appuyer sur des règles reconnues pour concourir à la meilleure qualité possible.

Eb 3.7. Dans le cas où des outils sont utilisés comme aide au développement ou au contrôle du logiciel, certaines exigences applicables au logiciel peuvent être remplacées par des exigences sur les fonctionnalités correspondantes des outils. Ces fonctionnalités doivent être définies et validées.

Une pratique acceptable consiste à s'appuyer sur des combinaisons appropriées :

- d'expériences antérieures documentées (cf. § 4.2.2.4) ;
- de tests des outils ;
- de règles d'utilisation des outils ;
- et de vérifications et / ou tests des résultats.

Eb 3.8. La documentation doit permettre la maintenance du logiciel.

Eb 3.9. La documentation concernant chaque activité du cycle de développement du logiciel doit comporter les relations (traçabilité) entre les éléments décrits par cette documentation et ceux contenus dans les documents des activités situées en amont. Le niveau de traçabilité doit être apprécié en regard de l'exigence Eb 3.8.

Une pratique acceptable peut être de mentionner dans chaque document les références au(x) document(s) des activités situées en amont nécessaires à l'établissement des relations visées par cette exigence. Dans le cas d'un SNCC, le fournisseur est évalué sur son organisation de gestion de configurations, incluant l'aspect documentaire.

Eb 3.10. Un document de synthèse doit être constitué pour établir la validation de chaque système programmé en regard de sa spécification.

Dans le cas où l'on utilise un SNCC, une pratique acceptable est :
- la constitution d'un dossier de validation du SNCC basé sur l'expérience antérieure, les essais du fabricant et les éventuels essais complémentaires menés par l'exploitant,
- la constitution, système élémentaire par système élémentaire, d'un document de validation de chaque logiciel d'application, installé sur le SNCC, en regard de sa spécification.

Eb 3.11. L'archivage de la documentation doit permettre la maintenance et l'utilisation du retour d'expérience ultérieure du logiciel, tant par son contenu que par la durée de conservation des documents.

4.2.2.4 Expérience antérieure

L'utilisation de l'expérience antérieure apporte des informations confortant la confiance mise dans le choix des produits existants et de leur mise en œuvre.

Eb 4.1. Le fabricant d'un logiciel doit faire la preuve d'expériences réussies dans le développement et la maintenance de logiciels répondant à des exigences similaires.

Eb 4.2. Les informations recueillies pour l'utilisation de l'expérience antérieure doivent provenir de conditions d'utilisation comparables à celles visées.

Eb 4.3. Les informations recueillies pour l'utilisation de l'expérience antérieure doivent permettre de préciser la nature et les conséquences des défaillances lorsqu'il s'en est produit.

Eb 4.4. La collecte d'informations recueillies pour l'utilisation de l'expérience antérieure doit avoir été organisée et formalisée.

4.2.2.5 Maintenance

La maintenance comprend toutes les modifications liées aux évolutions des fonctions confiées aux logiciels et à la correction d'erreurs mises en évidence lors de l'exploitation. Cette activité ne doit pas dégrader la sûreté obtenue avant modification, et doit être réalisée conformément aux exigences des chapitres relatifs aux fonctions, à la qualité et à la fiabilité.

Eb 5.1. Toute modification d'un logiciel doit respecter les exigences relatives à la qualité. Chaque modification du logiciel doit être identifiée.

Toutefois, des dispositions particulières peuvent être adoptées, sous réserve de justification, pour la maintenance des systèmes programmés dont les logiciels n'ont pas été développés selon les exigences de la présente règle. Dans ce cas (modification mineure au sens du § 4.3), une pratique acceptable est l'utilisation d'un Plan Qualité Logiciel fondé sur les mêmes principes que celui utilisé lors du développement, avec, le cas échéant, les améliorations issues de la progression de l'état de l'art en génie logiciel.

Dans le cas où l'on utilise un SNCC, une pratique acceptable concernant l'identification des modifications est :

- d'identifier pour le logiciel du SNCC chaque modification réalisée par rapport à la version en exploitation, quelle qu'en soit l'origine,
- de répertorier pour le logiciel d'application chaque modification réalisée par l'exploitant.

Eb 5.2. Il doit être montré, après modification du logiciel, que chaque fonction réalisée correspond à sa spécification (essais de non-régression et de validation).

Eb 5.3. Toute modification d'un logiciel déjà installé et en exploitation ne doit pas conduire à une tolérance aux défaillances et à une robustesse plus faibles que celles existant avant modification.

Dans le cas de l'utilisation d'un SNCC, une pratique acceptable consiste à réaliser :

- une étude des conséquences de la modification envisagée vis-à-vis des mécanismes ou fonctions assurant la tolérance aux défaillances et la robustesse du SNCC;
- une étude des conséquences pour la sûreté de la modification envisagée sur les logiciels d'application.

4.3 Champ d'application

4.3.1 Conditions générales d'application

La présente règle fondamentale de sûreté s'applique à toutes les tranches comportant un réacteur à eau sous pression qui disposent de systèmes programmés classés de sûreté, tels que définis au début du chapitre 4, dont le décret d'autorisation de création est postérieur de plus d'un an à la publication de la présente règle.

Elle s'applique également à la première mise en œuvre ou à la modification de logiciels de systèmes électriques classés de sûreté sur les tranches nucléaires en exploitation comportant un réacteur à eau sous pression, dans la mesure où cette première mise en œuvre ou cette modification fait l'objet d'une réalisation postérieure de plus d'un an à la publication de la présente règle.

4.3.2 Conditions particulières d'application aux logiciels mis en œuvre antérieurement à l'entrée en vigueur de la présente règle

Une modification d'un système électrique classé de sûreté, dont le logiciel n'a pas été programmé suivant les exigences de la présente règle fondamentale de sûreté, n'a pas à répondre aux exigences de celle-ci, lorsque cette modification ne concerne :

- ni le support matériel du système (par exemple le microprocesseur),
- ni l'architecture du logiciel,
- ni les méthodes de développement du logiciel.

Dans le cas d'une modification concernant une ou deux des trois caractéristiques citées plus haut (modification dite mineure), l'exploitant appliquera le chapitre Maintenance de la présente règle fondamentale de sûreté, sauf justifications.

Dans le cas où une modification d'un système électrique concerne les trois caractéristiques citées plus haut (modification dite majeure), la présente règle fondamentale de sûreté est applicable dans sa totalité au logiciel de ce système électrique.

Annexe

Glossaire

Analyse statique (§ 4.2.1.2) :

Etude d'un logiciel basée sur l'examen de sa forme, de sa structure, de son contenu et de sa documentation. Cette notion est définie par opposition à l'analyse dynamique. Cette dernière représente l'étude d'un logiciel basée sur l'examen du comportement du programme exécutable correspondant afin de mettre en évidence les fonctionnements effectivement obtenus pour certaines conditions de l'environnement et de les comparer à ceux qui sont attendus. Un des moyens utilisés couramment est la réalisation d'essais.

Comportement (§ 3) :

Ensemble des fonctionnements d'une entité (système programmé, ...).

Confiance (en un logiciel) (§ 4.2.1.4) :

Conviction que le logiciel accomplit exactement ce pour quoi il a été spécifié et qu'il a la capacité de subir des évolutions sans diminution de cette assurance.

Cycle de développement d'un logiciel (§ 3.1) :

Ensemble des phases correspondant aux activités de développement d'un logiciel (spécifications, conception, programmation, essais), incluant des vérifications.

Défaillance (§ 3.2) :

Cessation de l'aptitude d'une entité à accomplir une fonction requise.

Déterminisme (§ 4.1.1) :

Principe selon lequel l'ordre des faits fixe absolument les conditions d'existence d'un phénomène de telle façon que, ces conditions étant posées, le phénomène ne peut pas ne pas se produire.

Environnement (§ 4.1.2) :

Conditions extérieures susceptibles d'agir sur le comportement d'un logiciel ou d'un système programmé.

Les informations venues du procédé, les paramètres qui fixent des valeurs de constantes d'un logiciel sont des exemples de conditions extérieures qui constituent une part de l'environnement d'un logiciel.

Erreur (§ 3.1) :

Cause d'une défaillance d'un logiciel.

Essai (§ 3.1) :

Procédé d'évaluation quantitative, par rapport à des valeurs attendues, des caractéristiques d'un logiciel ou d'un système programmé.

L'essai se pratique en soumettant le logiciel ou le système programmé à un environnement défini et en comparant les valeurs des sorties à celles attendues.

Un essai structurel est conçu pour solliciter une partie de programme dépendant de choix successifs dans un traitement.

Un essai fonctionnel est conçu pour solliciter une fonction du logiciel.

Expérience antérieure (§ 4.2) :

Ensemble de connaissances qui peuvent être obtenues préalablement à la conception, et qui peuvent participer à la démonstration des objectifs de qualité et de fiabilité attendus pour le logiciel. Ces connaissances peuvent être relatives, aussi bien aux résultats de fonctionnement de logiciels similaires qu'aux résultats obtenus par les équipes et outils retenus pour le développement.

En phase de maintenance, l'expérience antérieure agrège l'ensemble des résultats du fonctionnement en exploitation du logiciel, que l'on appelle habituellement « retour d'expérience ».

Fiabilité (§ 3) :

Assurance qu'un logiciel ou qu'un système réalise les fonctions qui lui ont été attribuées dans des conditions spécifiées.

Cette caractéristique peut être qualitative, lorsqu'elle est déduite de propriétés assurant la permanence des fonctionnements, ou quantitative lorsqu'elle est le résultat d'un calcul fondé sur des probabilités et des fréquences d'occurrence de défaillances.

Fonction (§ 3.1) :

Propriété active d'un logiciel ou d'un système programmé (qui peut regrouper plusieurs traitements).

Une fonction peut être une mission.

Interruption (§ 4.1.1) :

Suspension de l'exécution d'un programme, causée par un événement externe, qui se fait de telle sorte que le déroulement du programme puisse reprendre correctement.

Logiciel (§ 3) :

Le logiciel est un des éléments constitutifs d'un système programmé. Il est composé :

- de la documentation qui décrit les spécifications, l'architecture du logiciel, le programme écrit en langage simple (assembleur) ou évolué (langage C par exemple), les vérifications et validations, les moyens de développement, les dispositions d'organisation (Plan Qualité Logiciel) adoptées pour le développement et la maintenance ;
- du programme exécutable, qui est une suite d'éléments binaires (bits) compréhensible uniquement par l'ordinateur pour lequel il a été produit.

Mission (§ 4.2.1.1) :

Propriété active (traitement) d'un logiciel ou d'un système programmé qui lui permet d'accomplir l'objectif qui lui a été attribué.

Modèle (§ 4.1.2) :

Représentation simplifiée d'un processus ou d'un système (éventuellement sous la forme d'un modèle mathématique).

Module (de logiciel) (§ 3.1) :

Partie de logiciel, identifiée dans l'architecture de ce logiciel, qui réalise un traitement spécifique.

Multitâche (§ 4.1.2) :

Mode de fonctionnement d'un système programmé permettant l'exécution imbriquée dans le temps de plusieurs programmes avec le même microprocesseur.

Plan Qualité Logiciel (§ 3.1) :

Document énonçant les pratiques, les moyens et la séquence des activités liées à la qualité, spécifiques à un logiciel.

Prédictibilité (§ 4.1.2) :

Principe selon lequel le comportement d'un logiciel ou d'un système programmé en relation avec son environnement peut être établi au moyen d'un modèle.

Prédiction du comportement (§ 4.2.2.2) :

Etablissement du comportement du système programmé ou du logiciel, en relation avec son environnement, à partir d'un modèle déterministe.

Profil opérationnel (§ 3.2) :

Ensemble des combinaisons quantitatives et temporelles des informations provenant du procédé à l'entrée d'un logiciel ou d'un système programmé, issues des spécifications de son cahier des charges.

Programmation défensive (d'un logiciel) (§ 4.2.1.2) :

Technique de réalisation des traitements assurant la détection d'une ou plusieurs défaillances pour permettre la continuation des missions (tolérance aux défaillances) ou mettre le système programmé dans un état prédéterminé (robustesse).

Programme (d'un logiciel) (§ 3.1) :

Le programme représente l'expression en langage informatique des traitements attribués au logiciel. Le programme exécutable prend la forme d'une suite d'éléments binaires, de valeur un ou zéro, compréhensible uniquement par l'ordinateur pour lequel il a été produit.

Ressources (§ 4.1.1) :

Ensemble des moyens dont dispose un ordinateur pour exécuter un programme ou plusieurs programmes simultanément.

Revue critique (§ 4.2.1.2) :

Activité qui consiste à vérifier que les résultats de chaque phase du développement d'un logiciel remplissent les exigences ou les conditions qui leur sont imposées dans les étapes précédentes.

Robustesse (§ 4.2.1.1) :

Principe qui guide la spécification d'un logiciel pour qu'il se mette dans un état prédéterminé en présence d'une défaillance « non tolérée ».

Système numérique de contrôle-commande (SNCC) (§ 4.2.2) :

Produit existant configurable à partir duquel peut être construit un système programmé classé de sûreté et non classé 1E.

Système programmé (§ 3.1) :

Système électrique qui comprend un logiciel.

Tolérance aux défaillances (§ 4.2.1.1) :

Principe qui guide la spécification d'un logiciel ou d'un système programmé de telle sorte que ce logiciel ou ce système programmé soit capable d'accomplir ses missions correctement et de façon continue en présence d'un nombre limité de défaillances.

Tolérance aux erreurs (§ 3.2) :

Concept qui conduit à requérir qu'un logiciel ait un fonctionnement déterminé en présence d'une erreur.

Ce concept se concrétise par l'association des principes de tolérance aux défaillances et de robustesse.

Traçabilité (§ 4.2.1.3) :

Principe qui conduit à établir et maintenir les liens de filiation entre différentes parties de documents.

Traitement (§ 3.1) :

Suite d'opérations logiques et mathématiques effectuées sur des informations par le logiciel ou le matériel d'un système programmé.

Validation (§ 3.1) :

Action de contrôle de la cohérence entre le système programmé (ou le logiciel) réalisé et ses spécifications.

Vérification (§ 3.1) :

Action de contrôle de la cohérence entre les résultats d'une phase de développement du logiciel ou du système programmé et ceux de la phase précédente.