**Institut National Polytechnique de Lorraine**
**Centre de Recherche en Automatique de Nancy**

INRS

# THÈSE

Pour l'obtention du grade de

**DOCTEUR de l'Institut National Polytechnique de Lorraine**

Spécialité **: Automatique**

Présentée et soutenue publiquement

Par

**Philippe CHARPENTIER**

**Ingénieur de l'École Nationale Supérieure d'Électricité et de Mécanique**

**Le 10 juillet 2002**

# ARCHITECTURE D'AUTOMATISME EN SECURITE DES MACHINES : Étude des conditions de conception liées aux défaillances de mode commun

Directeur de thèse : **Jean-François AUBRY**

Membres du jury :

Rapporteurs :

| | | |
|---|---|---|
| **Mme Mireille** | **BAYART** | Université des Sciences et Technologies, Lille |
| **M. Eric** | **NIEL** | Institut National des Sciences Appliquées, Lyon |

Examinateurs :

| | | |
|---|---|---|
| **M. Jean-Claude** | **ANDRE** | Institut National de Recherche et de Sécurité, Nancy |
| **M. Jean-François** | **AUBRY** | Université Henry Poincaré, Nancy |
| **M. Daniel** | **NOYES** | Ecole Nationale d'Ingénieur, Tarbes |

# *REMERCIEMENTS*

Ces travaux ont été réalisés dans le cadre des Études et Recherches du département Ingénierie des Équipements de Travail de l'Institut National de Recherche et de Sécurité, en partenariat avec le Centre de Recherche de Automatique de Nancy.

Je tiens à remercier Monsieur Jean-François AUBRY, Professeur à l'École Nationale Supérieure d'Électricité et de Mécanique et Directeur de l'Institut de Sûreté Industrielle pour la confiance qu'il m'a témoigné en acceptant d'encadrer ces travaux. Ses conseils et son soutien m'ont été précieux et ont contribué à l'amélioration de ce travail. Ces années passées à échanger des points de vue scientifiques ont été très enrichissantes et j'espère que d'autres actions communes feront suite à cette thèse.

Que Madame Mireille BAYART, Professeur à l'École Universitaire D'Ingénieur de Lille et Monsieur Éric NIEL, Professeur à l'Institut National de Sciences Appliquées de Lyon, soient assurés de ma gratitude pour avoir accepté d'examiner attentivement ce mémoire en qualité de rapporteurs.

Je remercie également Monsieur Daniel NOYES, Professeur à l'École Nationale d'Ingénieurs de Tarbes pour l'intérêt qu'il a bien voulu porter à ces travaux en acceptant de participer à ce jury en tant qu'examinateur.

J'adresse mes sincères remerciements à Monsieur Jean-Claude ANDRÉ, Directeur Scientifique de l'INRS, pour m'avoir donné l'occasion d'effectuer ces travaux dans le cadre de mes activités professionnelles et pour avoir accepté d'en évaluer les résultats en me faisant l'honneur de participer à ce jury.

Je tiens à remercier ceux qui m'ont aidé à approfondir certains aspects spécifiques de ces travaux : Monsieur Albert EMOND et Madame Céline NEUGNOT de l'INRS, les sociétés VERIDATAS et SURLOG, les partenaires européens du projet **ST**andards for **SA**fety **R**elated **C**omplex **E**lectronic **S**ystems, ainsi que Christophe BLANCHET, stagiaire à l'INRS.

Que Monsieur Guy LOVAT, chef du département IET, soit remercié pour m'avoir permis de mener ces travaux dans les meilleures conditions possibles. Je tiens aussi à remercier le personnel du laboratoire Sûreté des Systèmes Électroniques de l'INRS, et plus généralement de l'ensemble de mes collègues du département IET, pour m'avoir soutenu dans ces travaux.

J'adresse mes remerciements aux membres du CRAN, plus particulièrement ceux du personnel administratif et ceux du groupe du groupe Conception de Systèmes Sûrs de Fonctionnement. Je remercie les premiers de leur disponibilité et j'espère avoir de nouveau l'occasion de travailler avec les seconds.

Enfin, je remercie ma famille qui m'a soutenu et qui a accepté mes indisponibilités lors des moments cruciaux de ces travaux.

# SOMMAIRE

**ANNEXES**

**Table des figures**

## Tableaux

# *INTRODUCTION GÉNÉRALE*

Des progrès considérables ont été accomplis pour améliorer la sécurité des opérateurs côtoyant des machines ou des lignes de production automatisées. Les causes en sont les évolutions technologiques et méthodologiques réalisées par les constructeurs, mais aussi la vigilance constante des partenaires impliqués dans la sécurité : préventeurs, normalisateurs, ministères, organismes de contrôle, utilisateurs, … . Cette vigilance doit être maintenue pour si possible améliorer encore les niveaux de sécurité atteints, ou à minima les conserver.

Elle se traduit par la nécessité de considérer les risques d'une machine dès la conception, pour en déduire les actions adéquates de réduction des risques : mesures intrinsèques de conception, implantation de "barrières" de sécurité, informations des utilisateurs sur les risques résiduels de la machine. La première de ces actions influence la conception du circuit de commande de la machine. Les concepteurs sont alors conduits à respecter les exigences formulées par la Directive Machine (89/392/CEE codifiée 98/37/CE), en particulier celle qui stipule : "*qu'un défaut affectant la logique du circuit de commande ou une défaillance ou une détérioration du circuit de commande ne doit pas créer de situations dangereuses"*.

Il y a encore 20 ans, les circuits de commande d'une machine étaient réalisés à l'aide de logiques basées sur des technologies électromécaniques ou électroniques faiblement intégrées. Les problèmes de sécurité liés aux défaillances de composants étaient alors correctement maîtrisés, puisqu'il était possible de simuler ou d'injecter les fautes susceptibles d'affecter les différents composants et de s'assurer que le comportement global de la machine était sûr. Les progrès de l'électronique, en particulier les avancées en termes d'intégration, ont fait que les composants électroniques programmables fortement intégrés ont maintenant supplanté ces technologies. Les modes de défaillances de ces composants n'étant plus connus, d'autres méthodes ont dû être envisagées, regroupées sous le terme générique de Sûreté de Fonctionnement. La démarche proposée, novatrice dans les années 1990 en sécurité des machines, est issue des travaux effectués dans les domaines d'activités tels que le nucléaire, les transports ou l'aéronautique.

Le questionnement qu'elle a suscité a conduit l'Institut National de Recherche et de Sécurité (INRS) à mener des Études et Recherches pour y apporter des éléments de réponses. Les travaux présentés dans ce document sont une contribution à la construction d'un système de commande pour machine, en suivant une démarche de sûreté et en respectant les prescriptions de sécurité en vigueur. Leur finalité est de concevoir une architecture de sécurité, basée sur deux sous-systèmes standards différents, en l'occurrence deux Automates Programmable Industriels. Ils ont été réalisés dans le cadre d'une coopération entre le Centre de Recherche en Automatique de Nancy, en particulier le groupe Conception de Systèmes Sûrs de Fonctionnement et le département Ingénierie des Équipements de Travail de l'INRS.

D'un point de vue réglementaire, seul le respect des exigences essentielles de sécurité fixées par la Directive Machine est obligatoire pour concevoir le circuit de commande d'une machine. En pratique, un référentiel normatif a été élaboré pour venir en appui à la Directive. Les normes qu'il contient sont d'application volontaire, mais leur respect vaut présomption de conformité à la Directive. Nous décrirons dans le ***premier chapitre*** les normes EN 954 et CEI 61508 / CD CEI 62061, qui fixent aux concepteurs le cadre pour concevoir des systèmes de commande capables de traiter les signaux de sécurité d'une machine. En parallèle à cette description, nous présenterons le référentiel relatif à la Sûreté de Fonctionnement des systèmes informatiques établi par le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) de Toulouse. Son domaine d'application est plus vaste que celui des référentiels précédents puisque, en plus de la sécurité, il traite aussi de la fiabilité, la disponibilité, la maintenabilité. La démarche proposée pour construire la Sûreté de Fonctionnement d'une installation fait appel à la tolérance, à l'évitement et à la prévision des fautes. Nous l'avons retenue pour structurer les travaux décrits dans ce document. La prise en compte des normes étant incontournable en sécurité des machines, nous avons établi des liens entre les principales définitions et les principaux concepts utilisés par le LAAS et par les normes.

Quel que soit le référentiel choisi, on constate que la Sûreté de Fonctionnement d'un système électronique à composants programmables complexes doit être construite dès la phase de spécification, en suivant une démarche réfléchie. Celle-ci s'attache à mettre en place les moyens nécessaires pour lutter contre les différents types de fautes (matérielles comme logicielles) susceptibles d'affecter les composants du système. Nous avons choisi de présenter nos travaux en distinguant les fautes indépendantes, attribuées à des causes différentes, des fautes dépendantes attribuées à une même cause. La préoccupation vis-à-vis des fautes indépendantes est un premier niveau d'action nécessaire pour construire un système sûr de fonctionnement. Elle permet de "blinder" les sous-systèmes ou modules élémentaires. Les fautes dépendantes doivent être considérées lorsque ces modules sont regroupés dans des architectures redondantes pour améliorer la tolérance aux fautes de ces systèmes. Nous présenterons des moyens relatifs à la tolérance, l'évitement et la prévision pour chaque type de fautes.

Le ***second chapitre*** considère uniquement les fautes indépendantes. Nous avons étudié différentes techniques applicables à :

- La tolérance aux fautes. Nous avons étudié deux techniques d'autotests destinées à détecter les fautes d'un système à microprocesseur. Nous avons tout d'abord effectué une étude comparative des performances de deux méthodes de détection des fautes d'une mémoire programme. Pour des capacités courantes de mémoires, on montre que la méthode la plus simple à implanter, à savoir une somme de contrôle avec retenue, est plus efficace qu'un contrôle à l'aide d'un générateur de CRC (Cyclic Redundancy Check), méthode préconisée par la norme CEI 61508. Nous avons ensuite étudié l'implantation d'un chien de garde. Ce mécanisme est quasi-systématiquement implanté sur tout système à microprocesseur pour détecter les modifications de la durée d'exécution de ses

instructions, le plus souvent les allongements. Une évaluation des performances par injection de fautes sur deux applications différentes montre qu'il est intéressant de détecter aussi les raccourcissements de la durée d'exécution.

- L'évitement des fautes. Nous avons abordé les problèmes liés à l'évitement des fautes logicielles, qui prennent une part prépondérante dans les systèmes programmés actuels. En coopération avec des sociétés spécialisées en Sûreté de Fonctionnement du logiciel, nous avons établi un référentiel destiné à assurer la qualité et la sûreté du logiciel embarqué d'un microprocesseur. Ce référentiel est maintenant intégré au projet de norme CD CEI 62061. Il a été complété par un guide pour effectuer la vérification d'un logiciel.

- La prévision des fautes a été essentiellement abordée via l'utilisation d'un dispositif d'injection de fautes via la mémoire programme d'un microprocesseur. Différents systèmes ont été soumis à cette injection et les résultats semblent corrélés avec les mesures prises par les concepteurs pour détecter les fautes.

Le ***troisième chapitre*** est relatif à la Sûreté de Fonctionnement vis-à-vis des *fautes dépendantes*. Dans un premier paragraphe, nous nous sommes attachés à définir et à caractériser ce que sont réellement les fautes dépendantes et les défaillances de mode commun qui leur sont associées. Comme pour les fautes indépendantes, les résultats sont présentés en suivant la démarche d'obtention de la Sûreté proposée par le LAAS. Ces fautes étant introduites par le recours à une technique particulière de tolérance (la redondance), il est difficile de répertorier des techniques pour tolérer les fautes dépendantes.

- L'évitement des fautes fait en grande partie appel à des techniques de diversité. Une étude bibliographique a été menée pour en déterminer les différentes déclinaisons. L'accent a été mis sur la diversité logicielle, pour constater que cette technique en théorie efficace, ne peut résoudre à elle seule tous les problèmes de défaillances de mode commun. Les constats opérés nous ont conduit à élaborer des "check listes" destinées essentiellement à guider les concepteurs dans la prise en compte des défaillances de mode commun.

- Le principal problème lié aux défaillances de mode commun est relatif à la prévision des fautes. Les fautes logicielles étant encore difficilement appréhendées, les efforts de recherche ont jusqu'à présent porté sur la quantification des défaillances de mode commun dues aux fautes matérielles. Une étude bibliographique met en évidence plusieurs modèles, généralement issus de travaux réalisés dans le domaine de la sécurité des installations nucléaires. Nous avons plus particulièrement analysé la modélisation par le facteur β. Une étude de l'utilisation de ce modèle montre que des approximations sont généralement faites, qui imposent des conditions sur les taux de défaillances des modules et sur les temps d'observation. D'autre part, l'étude montre que ce modèle n'est pas utilisable pour des structures redondantes

hétérogènes. Nous avons alors envisagé la décomposition d'une telle structure en éléments homogènes et les calculs correspondants ont été effectués.

Nous avons analysé dans le ***quatrième chapitre*** les principales architectures utilisées pour tolérer les fautes d'un système électronique : architectures mono canal (1oo1, 1oo1D) et architectures redondantes d'ordre 2 (1oo2, 1oo2D, 2oo2, 2oo2D). Les degrés de redondance supérieurs n'ont pas été abordés car n'étant pas indispensables pour assurer la sécurité du système de commande d'une machine. L'analyse qualitative a consisté à envisager les fautes des différents canaux composant l'architecture et à déterminer le comportement prévisible de la sortie du système. L'évaluation quantitative a consisté à calculer la probabilité de défaillance dangereuse de chaque architecture en considérant ou non les défaillances de mode commun. Les hypothèses utilisées pour simplifier les équations ont été recensées. Ces analyses qualitatives et quantitatives montrent que les structures 1oo2 et 1oo2D sont les mieux adaptées à la sécurité des systèmes de commande de machines. Les calculs probabilistes effectués sur ces deux structures font apparaître l'influence de la valeur du facteur β modélisant les défaillances de mode commun ainsi que le taux de couverture des tests de diagnostics implantés sur chaque canal de la structure redondante. La difficulté pratique à évaluer ces paramètres relativise la portée des calculs probabilistes relatifs aux défaillances dangereuses.

Les résultats des chapitres précédents montrent que le principal moyen d'éviter les défaillances de mode commun est de recourir à la diversité. En vue d'optimiser le degré de diversité, nous avons envisagé dans le ***cinquième chapitre*** les conditions de conception d'une structure redondante hétérogène composée de deux Automates Programmables Industriels standards de deux constructeurs différents. Nous avons plus particulièrement étudié les problèmes liés à la synchronisation des deux voies, ainsi qu'un comparateur permettant de fournir une information sûre à l'aide des sorties délivrées par chaque automate.

Une ***conclusion*** est donnée, qui propose différentes perspectives de recherche, tant en ce qui concerne la conception des sous-systèmes que de leur intégration pour assurer la Sûreté de Fonctionnement d'un système de commande de machine.

# 1. CONTEXTE

La sûreté de fonctionnement touche un grand nombre de domaines industriels : nucléaire, aéronautique, spatial, transport, médical, …, domaines dans lesquels se posent à des degrés divers des problèmes de fiabilité, de disponibilité, de maintenabilité ou de sécurité des biens et des personnes[Geoffry & Motet, 1998], [Sourisse et Boudillon, 1996a], [Villemeur, 1988]. D'un domaine à l'autre, les grandes lignes de la démarche suivie pour traiter les problèmes sont identiques. Cependant, des spécificités existent, qui justifient les choix ou les priorités opérés. C'est la raison pour laquelle il est important de présenter le contexte dans lequel ont été effectués les travaux de recherche décrits dans ce document.

Le premier chapitre décrira le contexte général de la sécurité des machines tel qu'abordé par l'INRS, ses principaux partenaires institutionnels (Caisse Nationale d'Assurance Maladie, Caisses Régionales d'Assurance Maladie, Ministère des affaires sociales, …) et européens (BIA[1] allemand, HSE[2] anglais, …). Les principales définitions seront rappelées et la démarche d'identification des parties à considérer dans les analyses de sûreté sera donnée. Le second chapitre donnera le contexte normatif à considérer pour réaliser les analyses. Dans le domaine de la sécurité des machines, il est en effet difficilement envisageable de faire abstraction du système normatif mis en place suite à l'apparition de la directive européenne relative aux machines.

En parallèle à la description du contexte normatif, le référentiel établit par le LAAS[3] pour Sûreté de Fonctionnement (SdF) des systèmes informatiques sera donné. On établira ensuite une comparaison des principales définitions des deux référentiels ainsi que les liens entre les démarches proposées. Un choix sera fait, qui sera utilisé dans la suite du document.

## *1.1. LA SÉCURITÉ DES MACHINES*

### 1.1.1. Le contexte normatif

La fin des années 1980 a vu le passage d'une réglementation nationale à une réglementation européenne destinée à harmoniser les règlements des différents états membres, par l'intermédiaire de directives européennes. C'est ainsi que pour la sécurité des machines, la Directive du 14 juin 1989 concernant le rapprochement des législations des États membres relatives aux machines (89/392/CEE codifiée 98/37/CE) [98/37/CE, 1998] stipule au paragraphe 1.2.7 "défaillance du circuit de commande", *qu'un défaut affectant la logique du circuit de commande ou une défaillance ou une détérioration du circuit de commande ne doit pas créer de situations dangereuses*.

[1] BIA : Berufsgenossenschaftliches Institut für Arbeitsicherheit
[2] HSE : Health and Safety Executive
[3] LAAS : Laboratoire d'Analyse et d'Architecture des Systèmes de Toulouse

De même au paragraphe 1.4.3 : "Exigences particulières pour les dispositifs de protection", il est spécifié que "*les dispositifs de protection doivent être conçus et insérés dans le système de commande de sorte que l'absence ou la défaillance de l'un de leurs organes empêche la mise en marche ou provoque l'arrêt des éléments mobiles*".

La conception des machines, systèmes et dispositifs de protection doit donc se préoccuper non seulement de leur aptitude fonctionnelle à assurer la sécurité mais encore deleur comportement effectif, d'une part, dans l'environnement industriel (mécano-climatique et électrique) et ,d'autre part, en présence de défauts de composants [Sourisse et Boudillon, 1996b].

Des normes européennes viennent en appui des directives. Elles sont rédigées par des groupes d'experts (fabricants, utilisateurs, organismes de contrôle, pouvoirs publics, …) chargés d'établir un consensus. Leur respect n'est pas obligatoire, seul celui des exigences contenues dans la directive est obligatoire. Par contre, satisfaire aux prescriptions contenues dans une norme permet de s'assurer que les objectifs de la directive sont atteints [Lacore, 1993]. Le référentiel normatif ainsi constitué comporte trois niveaux :

- Des normes de type A, précisant les notions fondamentales, les principes de conception et des aspects généraux valables pour tout type de machines. On trouve ainsi des normes donnant les principes pour l'estimation des risques [EN1050, 1991] ou encore définissant la terminologie et spécifiant les méthodes générales de conception [EN292, 1997].

- Des normes de type B, traitant d'un aspect de la sécurité ou d'un type de dispositif conditionnant la sécurité valable pour une large gamme de machines. La norme EN 954 [EN954-1, 1996] sur les parties des systèmes de commande relatives à la sécurité est représentative des normes de type B.

  - *Normes de type B1*, qui donnent des spécificités techniques relatives à des aspects spécifiques applicables à un grand nombre de machines ;

  - *Normes de type B2*, qui donnent des spécifications techniques relatives à des composants de systèmes de sécurité pouvant être utilisés pour différents types de machines

- Des normes de type C, indiquant des prescriptions de sécurité détaillées s'appliquant à une machine particulière ou à un groupe de machines : machines-outils, machines papetières, machines pour l'industrie textile …

Le diagramme suivant donne une vue générale de "l'univers normatif" élaboré par le CEN (Comité Européen de Normalisation) [Lacore, 1998], [Jouffroy, 1999].

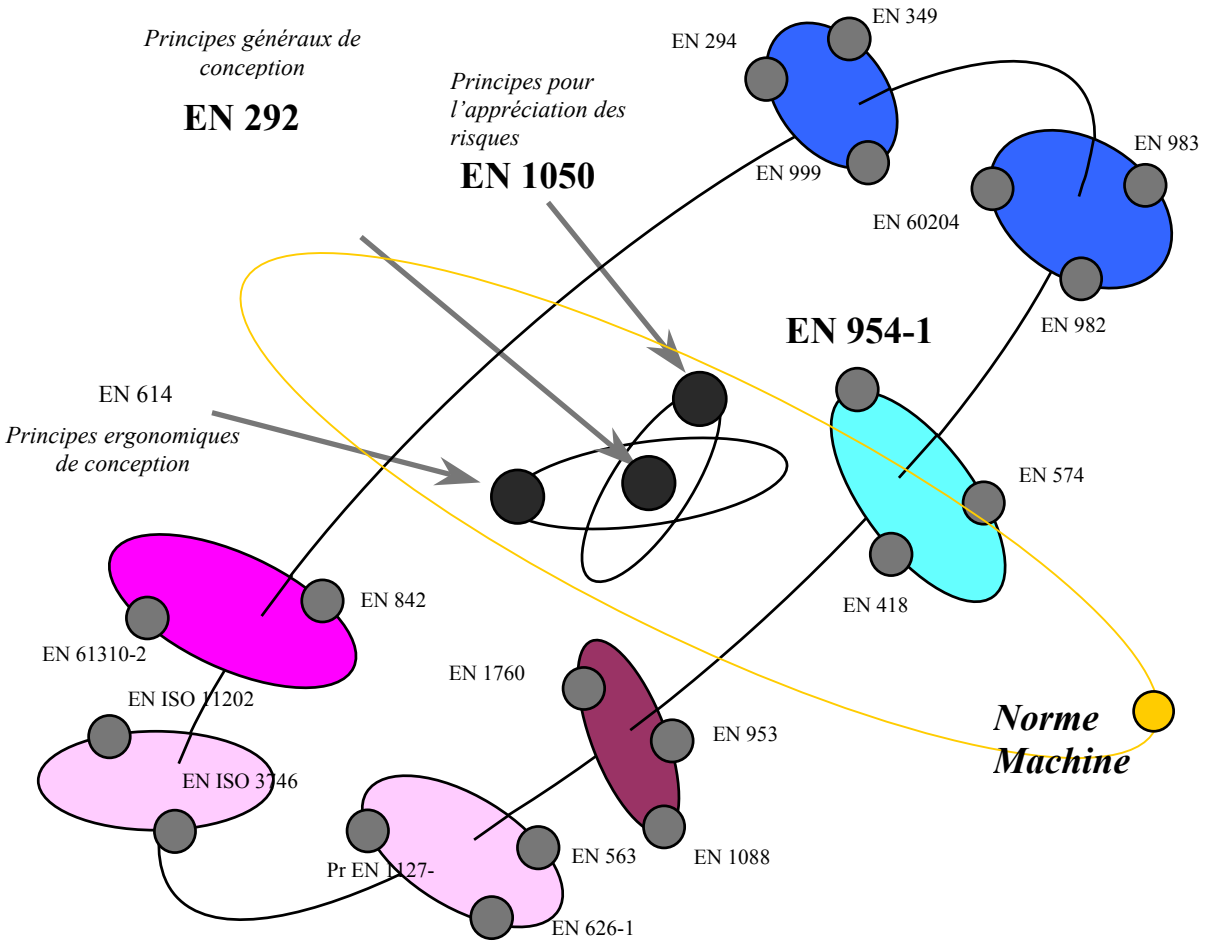



**Figure 1-1 : Représentation du système normatif européen en sécurité des machines**

L'ensemble des normes gravite autour de la norme fondamentale EN 292, de type A. La première orbite est composée des normes de type B, par exemple la norme EN 1050. Les seconde et troisième orbites sont composées des normes de type B : B1 pour la norme EN 954-1 par exemple, et B2 pour les normes relatives aux composants de sécurité (EN 61496 pour les barrages immatériels). La dernière orbite est composée des normes machines (de type C), comme par exemple la norme EN 12417 relative aux centres d'usinages.

Compte tenu des travaux présentés dans ce document, nous nous intéresserons essentiellement à la norme EN 954. Cette norme dédiée à la sécurité des machines contient des prescriptions déterministes applicables à toutes les technologies de réalisation : pneumatique, hydraulique, électromécanique, électrique. Par son caractère multi-technologie, cette norme s'avère insuffisante pour aborder les problèmes induits par l'utilisation de systèmes à technologies E/E/PE[4].

Une norme spécifique est en cours de développement pour pallier ces insuffisances. Il s'agit de la norme CEI 62061 [CEI62061, 2002] qui traite de la sécurité fonctionnelle des systèmes de commande de machines à technologies

[4] E/E/PE : Electrical / Electronic / Programmable Electronic – Electrique / Electronique / Electronique Programmable

E/E/PE. C'est une adaptation de la norme CEI 61508 [CEI61508, 2000] applicable à tous les domaines d'activité : nucléaire, aéronautique, spatial, transport, … Une partie de ses prescriptions porte sur des aspects probabilistes. A terme, en sécurité des machines, la norme CEI 61508 devrait s'appliquer au développement de 'composants' ou 'sous-systèmes' (Automates Programmables Industriels dédiés à la Sécurité, Capteurs, Actionneurs, …) alors que la CEI 62061 s'appliquera à la conception et à la validation d'un système de commande intégrant des 'composants' ou 'sous-systèmes' conformes à la CEI 61508.

Seules les normes en relation directe avec la sécurité des machines seront exploitées dans ce document. Compte tenu des technologies mises en œuvre dans les systèmes automatisés actuels, les définitions seront recherchées dans le référentiel international CEI[5] plutôt que dans le référentiel lié à la directive européenne.

La suite de ce chapitre donnera tout d'abord les définitions puis les démarches proposées par les normes EN 954 et CEI 61508 / CEI 62061 [Paques et al., 1999].

### 1.1.2. Description d'une machine

#### *1.1.2.1. Définition*

La notion de **machine** est clairement définie dans la norme EN 292. C'est un ensemble de pièces ou d'organes liés entre eux, dont au-moins un est mobile et, le cas échéant, d'actionneurs, de circuits de commande et de puissance, etc., réunis de façon solidaire en vue d'une application définie, notamment pour la transformation, le traitement, le déplacement et le conditionnement d'un matériau [Sourisse et Boudillon, 1997].

Un ensemble de machines disposées et commandées de manière à être solidaires dans leur fonctionnement est aussi considéré comme une machine.

La définition couvre donc un domaine très large allant d'un simple tour ou d'une presse à une ligne automatisée (construction automobile, imprimerie, …) en passant par des chariots élévateurs ou des marteaux piqueurs.

Cette définition est largement utilisée et est identique pour tout le système normatif européen en sécurité des machines.

#### *1.1.2.2. Système de commande / partie opérative*

L'annexe 1 montre de façon schématique les différentes composantes d'une machine qui sont classiquement regroupées en un **système de commande** et une **partie opérative** (PO). Ces deux parties sont reliées, d'une part, par les capteurs et les dispositifs de sécurité (sens PO vers système de commande) et, d'autre part, par les préactionneurs (sens système de commande vers PO).

[5] CEI : Commission Électronique Internationale

Le **système de commande de la machine** est défini comme étant le système qui répond à des signaux d'entrées en provenance du processus, des autres éléments de la machine, de l'opérateur, d'un équipement de commande externe ou de toute autre combinaison de ces signaux et qui génère des signaux de sortie permettant à la machine de se comporter de la façon attendue [CEI62061, 2002].

Le système de commande d'une machine peut être réalisé en différentes technologies : pneumatique, hydraulique, électrique. Nous ne considérerons dans ce document que les parties électriques et, plus particulièrement, celles basées sur des technologies programmables.

#### *1.1.2.3. Parties des systèmes de commande relatives à la sécurité*

La directive européenne requiert de s'assurer qu'aucune défaillance du circuit de commande ne soit dangereuse pour les personnes côtoyant la machine. Nous ne nous intéresserons donc qu'aux **parties des systèmes de commande relatives à la sécurité**, sans traiter les parties réalisant le fonctionnel.

Ces parties sont définies par la norme CEI 62061 comme étant celles dont les défaillances augmentent la probabilité que la machine provoque des blessures ou des atteintes à la santé. Elles répondent à des signaux d'entrées pour générer en sortie des signaux de sécurité. Elles débutent aux points où les signaux relatifs à la sécurité sont initiés et finissent à la sortie des éléments de contrôle de la puissance [EN954-1, 1996].

### 1.1.3. Processus de réduction des risques d'une machine

L'identification des parties électriques des systèmes de commande relatives à la sécurité est réalisée par une analyse globale des risques de la machine.

L'objet de ce paragraphe est de donner les principales étapes d'une démarche largement préconisée en sécurité des machines. Il s'agit du processus de réduction du risque proposé par la norme EN 292, en combinaison avec l'estimation du risque décrite dans la norme EN 1050 (cf. annexe 2).

Après avoir spécifié les limites de la machine et ses conditions d'utilisation, le concepteur doit suivre les étapes suivantes :

- *Identification des dangers et des situations dangereuses*

On décrit à cette étape les dangers 'de base' que la machine considérée est susceptible de générer, ainsi que les dangers associés à l'environnement dans lequel la machine est utilisée.

- *Estimation des risques pour chaque danger ou situation dangereuse identifiée*

Le concepteur de la machine doit estimer les risques relatifs aux différents dangers identifiés à l'étape précédente, que ces dangers soient permanents ou qu'ils apparaissent de façon inattendue.

Les risques sont par exemple : risque mécanique, électrique, thermique, risques dus aux vibrations, aux bruits, aux radiations, à l'environnement, …

- *Évaluation des risques et actions pour la réduction des risques*

Cette étape consiste essentiellement à identifier les actions à entreprendre pour réduire les risques identifiés précédemment, en prenant en compte les différents modes opérationnels. Les actions de suppression ou de réduction du risque seront envisagées dans l'ordre chronologique suivant :

- Mesures intrinsèques de conception

On entend par mesures intrinsèques de réduction des mesures qui réduisent les risques associés aux dangers sans recourir à des barrières ou à des dispositifs de protection. Le paragraphe 1.1.4 décrit deux des mesures proposées par la norme : l'utilisation de composants fiables et la duplication de composants "critiques".

- Implantation de 'barrières de sécurité'

C'est le recours à des barrières ou à des systèmes de protection, lorsque les mesures intrinsèques de conception se sont avérées insuffisantes pour réduire les risques de la machine. Ces dispositifs peuvent être fixes ou mobiles, matériels (barrières) ou immatériels (barrière lumineuse).

- Information des utilisateurs sur les risques résiduels de la machine

Cette action n'est envisagée que si les précédentes s'avèrent insuffisantes pour réduire les risques à un niveau tolérable. Elle consiste à mettre à disposition des utilisateurs des informations sur le fonctionnement de la machine, par exemple sous forme de signaux visuels, de diagrammes ou de notices écrites

Ce processus est itératif. Il faut en particulier s'assurer que les actions prises pour réduire les risques ne génèrent pas de dangers nouveaux. Ce n'est que lorsque l'analyste juge que la réduction du risque a atteint un niveau satisfaisant que ce processus est stoppé .

### 1.1.4. Réduction du risque par des mesures intrinsèques de conception

Les deux dernières actions proposées au paragraphe précédent n'entrent pas dans le cadre de cette étude. Seules les mesures intrinsèques de conception ont une influence sur les parties des systèmes de commandes relatives à la sécurité. La norme EN 292-2 fournit plusieurs voies pour réduire les risques à la conception d'un système de sécurité, généralement issues des technologies électromécaniques ou électroniques faiblement intégrées. Elles s'appliquent bien sûr aux technologies à composants électroniques complexes, mais ne dispensent en aucun cas de recourir à une démarche complète de Sûreté de Fonctionnement.

#### *1.1.4.1. Utilisation de composants fiables*

L'utilisation de composants fiables est indiquée comme étant un des moyens à mettre en oeuvre pour faire face aux différentes perturbations et stress associés à l'utilisation de l'équipement dans ses conditions d'utilisation.

Cette mesure concourt à la sécurité puisqu'en minimisant les défaillances des composants dues en particulier à l'environnement, elle minimise l'éventualité de créer des comportements dangereux de la machine. Elle permet au système "d'absorber" des perturbations thermiques (chaud, froid), mécaniques (vibrations, chocs, …) ou électriques (électricité statique, champ électromagnétique, …).

Elle n'est bien entendu pas suffisante puisque, même avec une fiabilité accrue, les composants seront toujours susceptibles d'être défaillants.

#### *1.1.4.2. Redondance des composants critiques*

Une solution couramment préconisée en sécurité des machines pour réduire les risques est de concevoir des architectures redondantes. La redondance est alors utilisée comme un moyen de détection de fautes permettant au système d'adopter une position de sécurité (généralement le repli). La norme indique la possibilité de recourir à la diversité (de conception ou de technologie) pour s'affranchir des éventuelles défaillances de mode commun.

Cette préconisation est une des origines des travaux présentés dans ce document. Pour concevoir une structure redondante performante, il est en effet important de connaître les origines potentielles des défaillances de mode commun ainsi que les divers moyens à disposition des concepteurs pour traiter ces défaillances (la diversité n'étant qu'un des moyens envisageables).

## *1.2. LA SÉCURITÉ DES SYSTÈMES DE COMMANDE À E/E/PE*

Nous allons exposer dans ce paragraphe les grandes lignes du processus de construction de la Sûreté de Fonctionnement des parties des systèmes de commande relatives à la sécurité d'une machine. Ce processus sera abordé en suivant les contraintes imposées par la normalisation en cours de développement, en établissant des liens avec les travaux menés par le LAAS [Laprie et al., 1995].

### 1.2.1. Définitions

#### *1.2.1.1. Référentiel "normatif"*

**Sécurité** (Réf. CD CEI 62061)

C'est l'absence de risque non acceptable.

**Sécurité fonctionnelle** (Réf. CD CEI 62061)

Ce terme est propre au référentiel international. Il est défini comme étant la capacité de la machine à fonctionner en sécurité. C'est la partie de la sécurité globale d'un équipement qui dépend du fonctionnement correct des systèmes relatifs à la sécurité.

**Fiabilité** (Réf. EN 292)

Seule la fiabilité est définie dans les référentiels normatifs liés à la machine. C'est la capacité de la machine ou de ses composants à accomplir sans défaillance la fonction requise dans les conditions spécifiées pendant une période donnée de temps.

**Intégrité de sécurité** (Réf. CD CEI 62061)

C'est la probabilité que le système de commande relatif à la sécurité exécute de manière satisfaisante les fonctions de sécurité requises, dans toutes les conditions spécifiées et pendant une période de temps spécifiée. C'est en fait la notion de fiabilité appliquée à une fonction de sécurité

**Faute** (Réf. CEI 61508), (Réf. EN 292)

La définition donnée par la norme CEI 61508 est la suivante : c'est une condition anormale qui peut entraîner une réduction ou une perte de capacité d'une unité fonctionnelle à accomplir une fonction requise. Dans ce sens, il est traduit par le terme anomalie.

Ce terme est aussi défini par la norme EN 292 : c'est l'état d'une entité caractérisé par son incapacité à réaliser la fonction prescrite.

**Erreur** (Réf. CEI 61508)

Une erreur est l'écart ou la discordance entre une valeur ou une condition calculée, observée ou mesurée et la valeur ou condition vraie, prescrite ou théoriquement correcte. Cette définition (erreur de mesure) relie la notion d'erreur à la mesure d'une grandeur physique plutôt qu'au processus conduisant à un dysfonctionnement du système.

**Défaillance** (Réf. CEI 61508), (Réf. EN 292)

La norme CEI 61508 définit une défaillance comme étant la cessation de l'aptitude d'une unité fonctionnelle à accomplir une fonction requise. C'est donc le passage d'un état de bon fonctionnement à un état d'inaptitude à accomplir la fonction.

La norme EN 292 définit quant à elle une défaillance comme étant la fin de la capacité d'une entité à accomplir sa fonction. La défaillance est un événement, qui place l'entité en état de faute. Cependant, des fautes peuvent exister sans qu'il y ait eu défaillance.

Elle indique que les termes 'faute' et 'défaillance' sont souvent synonymes. Cette confusion révèle un manque de rigueur certain dans la caractérisation des

phénomènes à l'origine d'une altération de la sécurité puisqu'il amène à confondre un événement et un état.

**Évitement des fautes (ou anomalies)** (Réf. CEI 61508)

C'est l'utilisation de techniques et de procédures destinées à éviter l'apparition de fautes durant chacune des phases du cycle de vie de sécurité d'un système relatif à la sécurité.

**Tolérance aux fautes (ou anomalies)** (Réf. CEI 61508)

C'est l'aptitude d'une unité fonctionnelle à continuer à accomplir une fonction requise en présence de fautes.

**Défaillance systématique** (Réf. CD CEI 62061)

Ce sont des défaillances reliées de façon déterministe à une certaine cause, qui ne peuvent être éliminées que par une modification de la conception, du processus de fabrication, des procédures d'exploitation ou de la documentation. Cette liste met en évidence toutes les sources potentielles d'introduction de ces fautes systématiques, une des principales étant les erreurs de conception du logiciel.

Ces défaillances, introduites lors d'une des phases du cycle de vie du système, existent à l'état latent dans le système. Elles se révèlent lors du fonctionnement du système et ne peuvent généralement être éliminées que par une modification de la conception ou du processus de fabrication. Des exemples typiques de ces défaillances sont les fautes de conception du matériel et du logiciel, ces dernières représentant la majeure partie des défaillances systématiques d'un système. Les perturbations environnementales et les erreurs commises par l'opérateur sont aussi considérées comme des défaillances systématiques.

**Défaillance aléatoire du matériel** (Réf. CEI 61508)

Ces défaillances surviennent de manière aléatoire et résultent de divers mécanismes de dégradation du matériel.

**Intégrité de sécurité du matériel** (Réf. CD CEI 62061)

C'est la partie de l'intégrité de sécurité du système de commande relatif à la sécurité liée aux défaillances aléatoires du matériel en mode de défaillance dangereux.

**Intégrité de sécurité systématique** (Réf. CD CEI 62061)

C'est la partie de l'intégrité de sécurité du système de commande relatif à la sécurité qui se rapporte aux défaillances systématiques en mode de défaillance dangereux.

#### *1.2.1.2. Référentiel LAAS*

Ce référentiel est issu des travaux de synthèse réalisés par J.C. Laprie et son équipe, exposés dans le guide de la sûreté de fonctionnement [Laprie et al., 1995]

**Sûreté de fonctionnement**

C'est la propriété d'un système qui permet à ses utilisateurs de placer une confiance justifiée dans le service qu'il leur délivre. La Sûreté de Fonctionnement se compose de 4 composantes : la fiabilité, la disponibilité, la maintenabilité et la sécurité.

**Sécurité**

On distingue deux types de sécurité : la sécurité-confidentialité, qui s'applique au traitement de l'information, et la sécurité-innocuité, qui est la composante de la Sûreté de Fonctionnement relative à la non-occurrence de défaillances catastrophiques.

**Faute**

C'est la cause adjugée ou supposée d'une erreur, mais c'est aussi la conséquence de la défaillance d'un composant pour le système qui le contient.

**Erreur**

C'est la partie de l'état d'un système qui est susceptible d'entraîner une défaillance, mais une erreur peut aussi être considérée comme la manifestation d'une faute dans un système.

**Défaillance**

C'est un événement survenant lorsque le service délivré dévie de l'accomplissement de la fonction du système. C'est la transition du service correct vers un service incorrect. Dans ce sens, la défaillance à un niveau N équivaut à une faute considérée au niveau d'observation N+1.

**Tolérance aux fautes**

Ce sont les méthodes et techniques destinées à fournir un service à même de remplir la, ou les fonctions du système en dépit des fautes.

**Élimination des fautes**

Ce sont les méthodes et techniques destinées à réduire la présence des fautes, en nombre et en sévérité. Ces méthodes s'appliquent au système développé.

**Prévention des fautes**

Ce sont les méthodes et techniques destinées à empêcher l'occurrence ou l'introduction de fautes au sein d'un système. Ces méthodes s'appliquent au processus de développement du système.

**Évitement des fautes**

Prévention et élimination des fautes concourent à la même finalité qui est d'éviter les fautes. Pour la première de ces techniques, on peut considérer que l'on agit en amont : le produit n'est pas encore développé (ou la phase du cycle de vie n'est pas exécutée) et on applique des méthodes pour ne pas introduire de fautes au sein du système. Pour la seconde, le produit est développé et il

s'agit de rechercher l'existence éventuelle de fautes au sein du système avant sa mise en exploitation.

La finalité commune de ces techniques conduit le plus souvent à les regrouper sous le vocable "évitement des fautes".

**Prévision des fautes**

Ce sont les méthodes et techniques destinées à estimer la présence, la création et les conséquences des fautes.

### 1.2.2. Démarche de la norme EN 954 [Vautrin et al., 1996], [Charpentier, 2002]

#### *1.2.2.1. Domaine d'application*

La norme EN 954-1 publiée en 1996 s'applique aux parties du système de commande relatives à la sécurité. C'est le cas par exemple de la fonction d'arrêt d'urgence, du point où le signal d'arrêt d'urgence est généré jusqu'à la sortie du préactionneur, via les parties de traitement gérant cette fonction. On conçoit que ce qui est relatif au fonctionnel même de la machine n'est pas concerné. Par contre, les fonctions de sécurité directe (dont le dysfonctionnement augmente immédiatement le risque) et indirecte (dont la défaillance n'engendre pas immédiatement un risque) rentrent parfaitement dans le champ d'application de cette norme.

Cette norme décrit les principes de conception des parties de systèmes de commande relatives à la sécurité. Elle s'applique à toutes les machines à usage professionnel ou privé. Elle peut également être une base lors de la conception de composants dits de sécurité au sens de la Directive Machines [98/37/CE, 1998] : dispositifs électro-sensibles conçus pour la détection des personnes, notamment barrages immatériels, tapis sensibles, détecteurs électromagnétiques ; blocs logiques pour commandes bimanuelles....

#### *1.2.2.2. Le concept de catégorie*

La norme EN 954 est basée jusqu'à présent sur une approche principalement déterministe. La cause d'une altération de la sécurité d'un système quelconque, plus précisément de son circuit de commande, est l'existence ou l'apparition de défauts (un défaut correspond à l'état d'une entité inapte à accomplir une fonction requise). La conséquence peut être un dysfonctionnement du circuit de commande qui pilotera la partie opérative de la machine de façon erronée, avec les conséquences possibles d'accidents.

La résistance aux défauts ou aux défaillances est une exigence de la directive machine qu'il est impératif de prendre en compte. Elle doit être adaptée aux conséquences de ces défauts. En effet, si l'on envisage une mission à haut risque (en nucléaire par exemple où des centaines ou des milliers de personnes sont impliquées), la résistance aux défauts exigée sera sans commune mesure avec celle concernant par exemple une machine pour laquelle la gravité d'un accident possible serait limitée à un simple pincement.

Le concept de catégorie est fondé sur cette considération et module la résistance aux défauts en fonction des résultats de l'évaluation du risque. Un guide informatif permet de choisir raisonnablement la catégorie des parties d'un système de commande relatives à la sécurité en fonction de cette évaluation. La méthode proposée est une méthode simplifiée reposant sur le projet de norme EN 1050 "appréciation du risque", qui considère les 3 éléments suivants : la gravité de la lésion, la fréquence et/ou la durée d'exposition au phénomène dangereux, la possibilité d'éviter le phénomène dangereux.

Une correspondance a été établie entre niveaux de risque et catégories : une ou deux catégories préférentielles sont données pour chaque niveau. D'autres catégories sont envisageables qui sont soit sur-dimensionnées, soit sous-dimensionnées. Cette latitude est destinée, sous réserve de justification, à faciliter l'adaptation des prescriptions à différentes technologies.

#### *1.2.2.3. Les différentes catégories*

Le tableau donné à l'annexe 3 résume les cinq catégories décrites par la norme EN 954-1. Le classement proposé est cohérent avec les recommandations de la norme EN 292 pour réduire les risque par des mesures intrinsèques à la conception : une première distinction est faite par la fiabilité des composants (pour les deux premières catégories), puis par la structure (pour les catégories 2, 3 et 4).

La hiérarchie apparente entre les différents niveaux n'est applicable que pour une technologie donnée : un système réalisé par exemple en technologie électronique procurera un meilleur niveau de sécurité s'il est de catégorie 4 que s'il est de catégorie 3. Par contre, un système de catégorie 3 réalisé en technologie mécanique pourra procurer un meilleur niveau de sécurité que s'il avait été réalisé en technologie électrique.

Apparaît clairement dans ce tableau le caractère déterministe des prescriptions : le niveau de sécurité d'un système est déterminé par son comportement en présence de fautes. Des listes par technologie sont données dans une annexe à la partie 2 de cette norme. Ce caractère déterministe fait qu'il est théoriquement impossible de démontrer l'appartenance d'un système électronique programmé à une catégorie. Il est en effet impossible de lister l'ensemble des fautes susceptibles d'affecter ces technologies, puis d'analyser le comportement du système en présence de ces fautes.

### 1.2.3. Démarche de la norme CEI 62061 [Charpentier, 2002]

La norme EN 954 n'étant pas adaptée aux technologies complexes, une nouvelle norme (CD CEI 62061) a été mise en chantier pour adapter les prescriptions et la démarche de la norme CEI 61508 à la sécurité des machines. Elle traite de la **sécurité fonctionnelle** des systèmes relatifs à la sécurité basés sur des technologies Électriques / Électroniques / Électroniques Programmées.

La norme CEI 62061 donne une méthode pour identifier la contribution à la réduction du risque apportée par le système électrique de commande, allouer

les niveaux d'intégrité de sécurité à chaque fonction de sécurité, concevoir le système puis vérifier qu'il satisfait aux performances prescrites (voir annexe 4).

Cette méthodologie repose sur trois concepts clés qui sont :

- le management de la sécurité fonctionnelle tout au long du cycle de vie du produit,
- la quantification du niveau de sécurité obtenu à l'aide de la probabilité de défaillances dangereuses du système,
- le recours à des méthodes déterminées pour la conception et la vérification de ces systèmes.

Nous allons nous intéresser dans la suite de ce paragraphe à la démarche proposée pour construire la sécurité d'un système de commande à E/E/PE. Elle consiste à décomposer le système à concevoir en un certain nombre de sous-systèmes. Chacun doit posséder des caractéristiques définies en terme d'intégrité de sécurité du matériel, d'intégrité de sécurité systématique et de comportement lors de détection de fautes.

#### *1.2.3.1. Intégrité de sécurité du matériel*

Contraintes d'architecture

Ces contraintes ont été inclues pour concevoir une architecture suffisamment robuste vis-à-vis des fautes matérielles. Elles limitent le niveau d'intégrité de sécurité pouvant être atteint en considérant, d'une part, le niveau de tolérance aux fautes de l'architecture et, d'autre part, la fraction de défaillances sûres des sous-systèmes qui réalisent la fonction de sécurité.

Prescriptions probabilistes

La norme distingue deux cas en fonction du type de sollicitation de la fonction de sécurité :

- faible demande, lorsque le système de sécurité n'est pas sollicité plus d'une fois par an ou plus de deux fois la fréquence des proof tests (tests hors ligne exécutés périodiquement pour remettre le système dans un état 'comme neuf')de preuve. Dans ce cas, on évalue la probabilité moyenne de défaillance à réaliser la fonction sur demande ;
- forte demande, qui correspond à un système de sécurité sollicité plus d'une fois par an ou plus de deux fois la fréquence des tests de preuve. Dans ce cas, on évalue la probabilité de défaillance dangereuse par heure.

Remarque : A terme, la norme CEI 62061 ne devrait retenir que le mode à forte demande.

#### *1.2.3.2. Intégrité de sécurité systématique*

Évitement des défaillances systématiques

La norme fournit des mesures et techniques destinées à prévenir l'introduction de fautes lors de la conception et du développement du matériel et du logiciel des sous-systèmes. Parmi ces mesures on trouve notamment le respect de normes ou de guides, une conception modulaire et structurée, l'utilisation de composants éprouvés, la simulation, ….

Contrôle des défaillances systématiques

Comme précédemment, la norme donne des mesures destinées à contrôler les fautes systématiques lors de leur apparition en fonctionnement. Parmi ces mesures on trouve notamment la surveillance de la séquence du programme, la détection de fautes par des tests en ligne, la diversité logicielle et matérielle ainsi que des mesures contre les perturbations environnementales (ex de la température, de l'humidité ou des perturbations électromagnétiques).

#### *1.2.3.3. Comportement suite à la détection de fautes*

Dans le cas de la sécurité des machines, la détection des fautes dangereuses (par des tests ou par tout autre moyen) doit résulter en une action spécifique destinée à atteindre ou maintenir l'état de sécurité.

### 1.2.4. Démarche proposée par le LAAS

La sécurité est considérée comme une des composantes de la sûreté de fonctionnement, les autres composantes étant la fiabilité, la disponibilité et la maintenabilité. Sa prise en compte nécessite des actions au niveau de la tolérance aux fautes, de l'élimination des fautes et de la prévision des fautes. C'est une démarche globale de construction qui est donnée pour agir sur l'ensemble des composantes de la sûreté. Nous ne retiendrons dans la suite de ce paragraphe que les aspects relatifs à la sécurité.

#### *1.2.4.1. Tolérance aux fautes*

La tolérance aux fautes est mise en œuvre par le traitement d'erreurs (conséquences des fautes), qui élimine les erreurs si possible avant qu'une défaillance ne survienne, et par le traitement de fautes, qui évite qu'une ou plusieurs fautes ne soient activées à nouveau [Avizienis, 1967].

Dans le premier cas, des mécanismes de détection d’erreurs sont implantés pour empêcher que l'activation d'une faute ne conduise à la défaillance du système. Une fois l'erreur détectée, on procède à son recouvrement. Pour la sécurité des machines, le recouvrement se limite généralement à un passage en position de repli. Le second cas est typique de la tolérance aux fautes à des fins de disponibilité. On cherche alors à localiser et à typer les fautes, pour les traiter et être capable de les passiver, c'est à dire de faire en sorte qu'elles ne puissent plus être activées.

Il est donc proposé de travailler sur la tolérance aux fautes en s'appliquant à détecter et à traiter leurs conséquences (erreurs). Cette démarche s'explique en grande partie par la difficulté, voire l'impossibilité, de caractériser l'ensemble des fautes pouvant affecter un système électronique à composants complexes. Il est dans ce cas plus aisé de s'attacher aux conséquences des fautes, c'est à dire aux manifestations perceptibles de ces fautes sur le fonctionnement du système.

#### *1.2.4.2. Prévention des fautes*

La prévention des fautes est peu développée. Elle touche le processus de conception d'un système. En fonction du niveau d'observation, on peut considérer qu'elle s'attache soit à éviter les erreurs de conception (par exemple l'erreur du programmeur), soit à éviter que l'erreur ne se traduise par une défaillance du système (par exemple une défaillance logicielle due à une erreur de programmation).

#### *1.2.4.3. Élimination des fautes*

La première étape consiste à rechercher les fautes "injectées" au sein du système lors des différentes phases de son cycle de vie. C'est l'objet de la vérification, qui consiste à déterminer si le système satisfait des propriétés, appelées conditions de vérification. En cas de non satisfaction, il faut diagnostiquer puis corriger la faute originelle. Suite à la correction, une approche rigoureuse implique une nouvelle vérification pour s'assurer de la non-régression du système.

Le test est la principale technique de vérification de la conception d'un système.

#### *1.2.4.4. Prévision des fautes*

La prévision des fautes consiste à évaluer le comportement du système en présence de fautes. On peut distinguer deux types d'évaluation : l'évaluation ordinale ou qualitative, qui cherche à connaître ce comportement à partir de listes de fautes prédéterminées et l'évaluation probabiliste ou quantitative, qui détermine certains des attributs de la sûreté de fonctionnement en terme de probabilité. Dans le cas des applications de sécurité, on détermine généralement la probabilité de défaillance dangereuse du système.

La première approche est aisément applicable aux technologies pour lesquelles on connaît les modes de défaillances de composants ou pour lesquelles il est possible de faire des hypothèses réalistes sur ces modes de défaillances. La seconde nécessite la connaissance des taux de pannes des composants élémentaires du système analysé.

### 1.2.5. Comparatif des deux référentiels

#### *1.2.5.1. Comparaison des définitions*

**Sûreté de fonctionnement / sécurité**

Il n'y a pas de divergence de fond entre les définitions proposées pour le terme de sécurité. Seule la sécurité-innocuité est traitée en sécurité des machines. On remarquera que la sûreté de fonctionnement n'est pas définie dans les normes relatives à la sécurité des machines (elle l'est dans d'autres normes internationales génériques rédigées sous couvert de la CEI). Seule la fiabilité l'est, car utilisée pour justifier la notion de composant éprouvé.

**Faute / erreur / défaillance**

La différence entre les définitions des différents référentiels se situe dans la description de l'enchaînement des fautes.

La norme EN 292 caractérise l'incapacité d'une entité à réaliser la fonction prescrite comme un état de faute provoqué par une défaillance (considérée comme un événement)

La norme CEI 62061 considère la faute comme l'événement (condition anormale) à l'origine d'une défaillance (passage de l'aptitude à l'inaptitude à assurer la fonction).

Le LAAS introduit la notion intermédiaire d'erreur. L'enchaînement suivant est proposé : faute → erreur → défaillance. Cet enchaînement étant cyclique et dépendant du niveau d'observation des phénomènes, la défaillance d'une entité peut être considérée comme la cause de la faute du maillon suivant de la chaîne. En se plaçant au niveau du fonctionnement d'un système, la défaillance d'un système est la phase finale du processus de propagation d'une faute.

Fautes : Les définitions données par le LAAS et la CEI 62061 pour le terme "faute" sont sensiblement les mêmes. Il s'agit dans un cas comme dans l'autre, d'un événement à l'origine d'une défaillance : clairement dans le référentiel du LAAS et de façon moins explicite dans le référentiel normatif, où il est question d'une condition anormale qui peut entraîner une réduction ou une perte de capacité d'une unité fonctionnelle à accomplir une fonction requise (donc une défaillance).

Compte tenu de cette convergence, nous utiliserons dans la suite de ce document le terme de faute pour décrire l'événement à l'origine d'une défaillance. La définition de la norme EN 292 (faute = état) ne sera pas retenue.

Défaillances : Dans tous les cas, la défaillance est un événement qui place l'entité considérée dans l'incapacité à accomplir sa fonction. Il y a donc de ce point de vue convergence entre les différents référentiels.

Remarque : lorsqu'il sera utilisé, le terme erreur sera pris dans le sens défini par le LAAS.

**Classification**

Le LAAS envisage les fautes selon 5 points de vue distincts :

- La cause phénoménologique : fautes physiques dues à des phénomènes physiques adverses et fautes dues à des imperfections humaines.
- La nature : fautes accidentelles, crées de manière fortuite et fautes intentionnelles, commises délibérément. On ne traitera pas dans ce document des fautes intentionnelles, qui n'entrent pas dans le champ de la sécurité-innocuité.
- La phase de création : fautes de développement, jusqu'à la phase de modification et fautes opérationnelles, en exploitation.
- La situation : fautes internes et fautes externes dues aux interférences entre le système et son environnement physique.
- La persistance : fautes permanentes et fautes temporaires, qui sont liées à des conditions ponctuelles.

La norme EN 954 quant à elle ne fait pas de distinction entre les fautes considérer lors des analyses, la CEI 62061 (comme la CEI 61508) classe les défaillances en :

- Défaillances matérielles aléatoires : défaillances survenant à un instant aléatoire, dues à un mécanisme de dégradation du matériel. Les taux de défaillances des composants sont prévisibles mais l'instant d'apparition de ces défaillances ne l'est pas.
- Défaillances systématiques : défaillances reliées de façon déterministe à une certaine cause, qui ne peuvent être éliminées que par une modification de la conception, de la fabrication, des procédures opérationnelles ou encore de la documentation. On trouve dans cette catégorie les défaillances dues à des erreurs humaines dans les spécifications, la conception du matériel et du logiciel, …

La différence principale entre ces deux types de défaillance réside dans la possibilité de prédire les taux des défaillances matérielles aléatoires et pas ceux des défaillances systématiques (car on n'arrive pas à prédire précisément les événements qui les génèrent).

Cette typologie est par nature moins précise que la précédente. Elle correspond essentiellement au premier point de vue envisagé par le LAAS.

Le Tableau 1-1 donne la correspondance entre ces défaillances et les fautes recensées par le LAAS.

| **Causes**<br>**Réf. LAAS** | **Fautes**<br>**Réf. LAAS** | **Défaillances**<br>**Réf. CEI 62061** |
|---|---|---|
| Cause Phénoménologique | Physique | Matérielles aléatoires |
| | Dues à l'homme | Systématiques |
| Nature | Accidentelles | Matérielles aléatoires |
| | Intentionnelles sans intention de nuire | Systématiques |
| | Intentionnelles nuisibles | Non considérées |
| Phase de création ou d'occurrence | De développement | Systématiques |
| | Opérationnelles | Matérielles aléatoires et systématiques |
| Frontières du système | Internes | Systématiques |
| | Externes | Matérielles aléatoires |
| Persistance | Permanentes | Systématiques et matérielles aléatoires |
| | Temporaires | Matérielles aléatoires |

**Tableau 1-1 : Liens fautes / défaillances**

#### *1.2.5.2. Comparaison des démarches*

Les démarches proposées sont toutes destinées à s'affranchir des entraves à la Sûreté de Fonctionnement ou plus spécifiquement à la sécurité. Dans le premier cas le LAAS propose une action sur les fautes susceptibles d'affecter le fonctionnement du système alors que la CEI 62061 considère les défaillances.

Le tableau suivant (cf. Tableau 1-2) donne les liens entre les démarches universitaire et normative. Il met clairement en évidence l'inadaptation de la norme EN 954 aux systèmes à composants complexes, en particulier par l'absence de la composante "Évitement des fautes".

**Les problèmes seront abordés dans la suite de ce document en suivant la démarche proposée par le LAAS.**

| Actions de la démarche LAAS | | Actions de la démarche CEI 62061 | Actions de la démarche EN 954 |
|---|---|---|---|
| Tolérance aux fautes | | Intégrité de sécurité du matériel / *Contraintes d'architecture*<br>Intégrité de sécurité du matériel / *Contrôle des défaillances systématiques*<br>Comportement suite à la détection d'une faute | Contraintes d'architectures implicitement imposées par les catégories 2, 3 et 4. |
| Élimination des fautes<br>Prévention des fautes | Évitement des fautes | Intégrité de sécurité systématique / *Évitement des défaillances systématiques* | Aucune action proposée |
| Prévision des fautes | | Intégrité de sécurité du matériel / *Prescriptions probabilistes* | Analyse 'déterministe' du comportement en présence de fautes |

**Tableau 1-2 : Liens Démarches LAAS et normatives**

# 2. SDF DES SYSTÈMES ÉLECTRONIQUES PROGRAMMABLES ET DÉFAILLANCES INDÉPENDANTES

La sûreté de fonctionnement d'un système informatique complexe part de l'analyse des risques [Mortureux, 2001] pour identifier les parties critiques d'un système. Ceux-ci doivent ensuite être développés en abordant la tolérance, l'évitement et la prévision des fautes [Charpentier, 1992], [Chevance, 1999], [Zwingelstein, 1999].

## *2.1. TOLÉRANCE AUX FAUTES*

La tolérance aux fautes est définie par le LAAS comme étant l'ensemble des méthodes et techniques constructives destinées à fournir un service à même de remplir la ou les fonctions du système, en dépit des fautes pouvant affecter ses composants, sa conception ou ses interactions avec des hommes ou d'autres systèmes. Ces méthodes concernent essentiellement la phase descendante du cycle de vie, de la spécification à la conception. Les aspects interactions avec l'environnement dépassent le cadre des travaux présentés dans ce document.

La tolérance aux fautes est obtenue par la mise en oeuvre (à la conception / a priori) de moyens capables de détecter les fautes pouvant affecter le système (en fonctionnement / a posteriori). Une fois les fautes détectées, le système doit adopter

un comportement sûr, soit par reprise, poursuite ou compensation d'erreur, soit par repli.

Les premiers types de comportements assurent une continuité de la mission, souvent en mode dégradé. Les industries du process (chimie ou autres) ainsi que le domaine du transport (automobile, ferroviaire ou aéronautique) y ont recours. Le repli, utilisé dans le domaine de la sécurité des machines, conduit quant à lui à une position sûre par coupure (plus ou moins instantanée suivant les applications) de la puissance des actionneurs.

Attention à la définition de la tolérance aux fautes. En toute rigueur, les mesures que l'on applique en sécurité des machines pour assurer la sécurité rendent le système fail-safe, mais pas tolérant aux fautes. La disponibilité n'est pas le critère de sûreté principal à considérer pour construire l'architecture. Malgré tout, on répertorie ces mesures dans la catégorie 'tolérance aux fautes'.

Plusieurs solutions sont envisageables pour détecter les fautes dans un système électronique programmé :

- ✓ Comparaison par rapport à une référence prédéterminée. C'est le cas typique des contrôles des mémoires mortes d'un microprocesseur (cf. paragraphe 2.1.1) ou du contrôle par un chien de garde de la durée d'exécution du programme déroulé par un microprocesseur (cf. paragraphe 2.1.2). D'autres techniques peuvent être utilisées, qui s'attachent à détecter les erreurs de la séquence des instructions exécutées par un microprocesseur. C'est le cas de la détection par analyse de signature [Kacprzak et al., 1994a], [Kacprzak et al., 1994b]. Bien que performante, cette technique est en pratique peu utilisée car contraignante à mettre en œuvre. Dans tous les cas, une référence est prédéterminée à la conception du dispositif (somme de référence pour une mémoire morte, temps maximum autorisé pour exécuter une partie de programme pour un chien de garde). La détection de faute se fait en comparant périodiquement le résultat du contrôle en ligne à la valeur de la référence préétablie à la conception.
- ✓ Comparaison par rapport à une référence établie en ligne par le microprocesseur. C'est le cas du contrôle des sorties par relecture. Le microprocesseur mémorise une référence de bon fonctionnement (l'état commandé pour la sortie concernée), relit l'état effectif de la sortie, compare les deux valeurs pour détecter la présence éventuelle d'erreurs dans la chaîne de sortie. La détection est d'autant plus performante que le signal retourné au microprocesseur est proche de l'actionneur.
- ✓ Comparaison entre deux canaux. La comparaison porte sur les résultats dynamiques des calculs effectués par les deux voies de traitement. On ne cherche plus à détecter les défaillances de composants, mais leurs conséquences sur le résultat des calculs. La capacité de détection dépendra des informations comparées. Elle sera élevée si des données intermédiaires de calculs (pré-traitement des entrées, traitement, sorties) sont comparées suite à des échanges inter-processeurs. Elle sera beaucoup plus faible dans le cas où seules les sorties des deux voies sont comparées. Ce mode de détection agissant sur les conséquences des défaillances de composants, il est très souvent difficile de diagnostiquer précisément l'origine exacte de la faute.

Du fait de la liaison physique imposée par les échanges, la mise en œuvre de ce type de détection devra être réalisée en s'assurant qu'aucune défaillance de mode commun supplémentaire n'est introduite.

**Important : Dans les deux premiers cas, les diagnostics sont exécutés par le microprocesseur lui-même, ce qui pose le problème du contrôle d'une entité par elle-même. Jusqu'à présent, ce problème n'a été que très peu traité. Seul [Kanoun & Ortalo-Borrel, 2000] propose des éléments de réponse sur ce sujet.**

### 2.1.1. Détection des erreurs des mémoires programmes par checksum et CRC

Le fonctionnement correct d'un système à microprocesseur repose en premier lieu sur l'intégrité de son logiciel : une défaillance de la mémoire programme contenant ce logiciel provoquera une altération des instructions exécutées par le microprocesseur et conduira à un comportement erratique du système, préjudiciable à son bon fonctionnement. C'est pourquoi, un dispositif à microprocesseur doit tester, pour des applications à niveau de sécurité élevé, l'intégrité des informations contenues dans sa mémoire programme.

Plusieurs méthodes sont envisageables pour effectuer ce test [CEI 61508, 2000] et l'INRS a mené une étude en vue de déterminer la méthode la mieux adaptée à son domaine d'activité [Charpentier et al., 1991], [Charpentier, 1991]. La démarche suivie a consisté à évaluer, pour les deux méthodes les plus utilisées (Checksum et CRC), les caractéristiques en terme :

- De détection d'erreurs. Le pouvoir de détection théorique Pd doit être élevé vis-à-vis des fautes mémoire les plus probables. Pd est le rapport, exprimé en pourcentage, du nombre de fautes détectées par le contrôle sur le nombre total de fautes pouvant survenir sur la mémoire.

- De facilité de mise en œuvre. Le contrôle doit être simple à mettre en œuvre et n'utiliser qu'un nombre restreint de ressources matérielles et logicielles.

- De rapidité d'exécution, pour ne pas pénaliser le fonctionnement du dispositif par des durées de tests trop élevées ou pour ne pas retarder la détection d'une faute.

Le premier critère énoncé fait référence aux modes de défaillance d'une mémoire programme. En règle générale, tous les bits altérés lors d'une défaillance de la mémoire le sont dans le même sens, c'est-à-dire d'un état '0' vers un état '1' ou inversement [Charpentier et al., 1991]. Il existe très peu de cas pour lesquels, au sein d'un même boîtier, certains bits sont altérés dans un sens et d'autres en sens contraire. En outre, la probabilité d'apparition d'une défaillance sur ce type de composants est faible, certains fabricants donnant des taux de pannes inférieurs à 200 défaillances pour $10^9$ heures de fonctionnement.

Les méthodes disponibles pour le test périodique d'une mémoire programme peuvent être classées en deux catégories :

- Adjonction d'informations à chaque octet mémoire. Selon la méthode retenue, on associe un bit représentant la parité de cet octet, 4 bits représentant le code de Hamming de l'octet ou 8 bits représentant le complément de l'octet. Il y a donc redondance partielle ou totale de l'information de chaque octet. Pour être efficace, les méthodes de ce type nécessitent l'utilisation d'un comparateur capable de détecter en temps réel d'éventuelles divergences entre l'octet mémoire programme et l'information supplémentaire qui l'accompagne.

- Adjonction d'information au bloc mémoire à contrôler. Ces méthodes consistent à affecter à tout ou partie du contenu d'une mémoire programme un ou plusieurs mots la caractérisant. Ce ou ces mots sont le plus souvent la signature sur 8 ou 16 bits du contenu de la mémoire obtenue par la méthode de CRC (Cyclic Redundancy Check) ou le résultat de la somme des octets de la mémoire programme (checksum). La sommation peut être effectuée avec ou sans propagation de la retenue.

Il existe d'autres méthodes pouvant être utilisées pour contrôler l'intégrité de la mémoire programme mais la complexité des algorithmes à mettre en œuvre est disproportionnée par rapport au problème posé.

La suite de ce paragraphe décrit le calcul du pouvoir de détection d'un CRC sur 8 bits et d'un checksum avec propagation de la retenue (aucune perte d'information lors de l'addition).

#### *2.1.1.1. Signature sur 16 bits générée par CRC*

Une signature est l'expression condensée d'une séquence d'information numérique transitant sur une ligne pendant un temps défini. Si le flux d'information est reproductible, la signature sera toujours la même. On associe à un flux d'informations numériques un polynôme E (x). La signature est le résultat de la division modulo 2 de la valeur de ce polynôme par un polynôme générateur P(x). L'obtention d'une signature peut être modélisée par un registre à décalage ou certains bits sont bouclés sur l'entrée par l'intermédiaire de la fonction logique OU Exclusif. La signature correspond au contenu final du registre après passage en série des informations. La longueur du registre ainsi que les bouclages sont fonction du polynôme P(x) qui, pour des applications usuelles, est généralement de degré 8 ou 16.

La figure représente le modèle associé au polynôme

$P(x)=x^{16}+a_{15}x^{15}+a_{14}x^{14}+ \ldots +a_1x+a_0$ avec $a_{15}$, $a_{14}$ , … , $a_1$ , $a_0$ (le bouclage i existe si $a_i=1$)

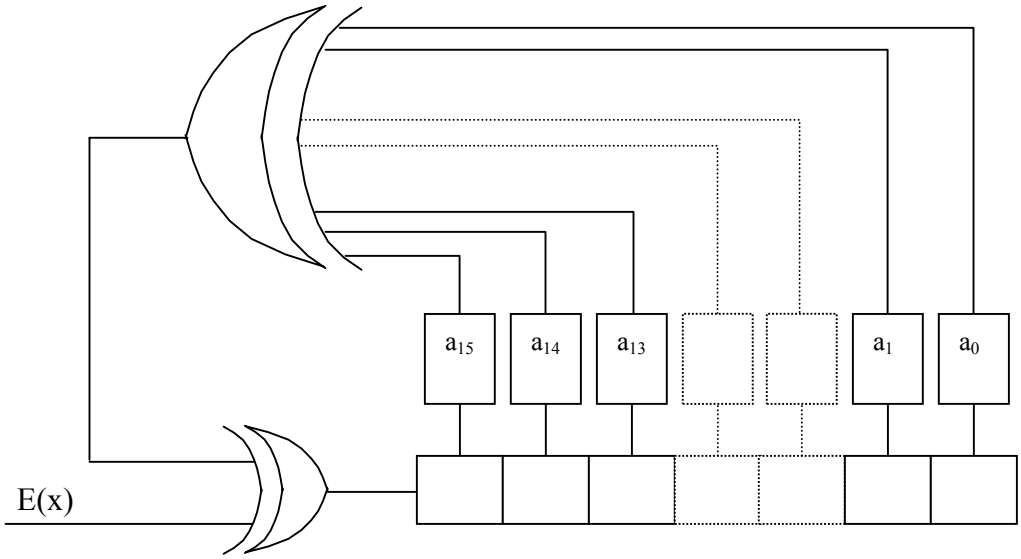



**Figure 2-1 : Schéma de principe de l'élaboration d'un CRC 16 bits**

Si m est le nombre de bits introduit dans un polynôme générateur de degré n, le pouvoir de détection Pd (%) est :

$$Pd\% = 100 \cdot (1 - \frac{2^{m-n} - 1}{2^m - 1})$$

et, si m est grand $Pd\% = 100 \cdot (1 - \frac{1}{2^n})$

ce qui donne Pd% = 99,998% pour n=16.

#### *2.1.1.2. Somme de contrôle avec retenue*

Les octets de la mémoire sont additionnés en propageant la retenue, la "signature" est la valeur de la somme de l'ensemble des N octets de la mémoire contrôlée. Le pouvoir de détection de cette méthode est de 100% vis-à-vis de toutes les erreurs affectant un seul octet de la mémoire à contrôler, ainsi que vis-à-vis des erreurs mémoire ayant pour conséquence une altération des bits dans le même sens (de 0 vers 1 ou inversement) [Charpentier et al., 1991]. Pour être en mesure de comparer cette méthode au CRC, il est nécessaire d'évaluer son pouvoir de détection vis-à-vis d'erreurs aléatoires. Deux méthodes ont été envisagées [Charpentier et al., 1991].

**Dénombrement exact.**

Cette première méthode consiste à dénombrer toutes les combinaisons de N octets d'une mémoire conduisant à une somme identique à la somme S obtenue sur les N octets de la mémoire contenant l'information originale.

Le nombre total d'erreurs $N_{TOT}$ est représenté par le nombre de combinaisons que peuvent prendre les 8xN bits de la mémoire à contrôler (si celle-ci contient N octets), ce qui donne $N_{TOT}=2^{8N}$.

Le problème essentiel réside dans la détermination du nombre $N_{DET}$ des erreurs détectées par le contrôle, ou encore de son complément, le nombre des erreurs non détectées $N_{NDET}$, sachant que $N_{DET} + N_{NDET} = 2^{8N}$. $N_{NDET}$ correspond au nombre de combinaisons des N octets mémoire dont la somme est identique à la somme S obtenue sur les octets de la mémoire en l'absence d'erreurs. On peut donc exprimer le pouvoir de détection Pd de la façon suivante :

$$Pd = 100 \cdot \frac{N_{DET}}{2^{8N}} = 100 \cdot \left(1 - \frac{N_{NDET}}{2^{8N}}\right) = 100 \cdot \left(1 - \frac{C_S^N}{2^{8N}}\right)$$

Les calculs exacts donnent les égalités suivantes :

Pour $S = 0 \;\; C_0^N = 1 \;\; \forall N \;\; on\ a\ Pd = 100 \cdot \left(1 - 1/2^{8N}\right) \approx 100\%$

Pour $S \in \left]2^8 k, 2^8 \cdot (k+1)\right] \; avec \;\; k \;\; entier \;\; et \;\; k \in \left[0, N - 1 - ENT\left(\frac{N}{2^8}\right)\right]$

on a $C_S^N = \sum_{i=0}^{k} (-1)^i \cdot \binom{N}{i} \cdot \binom{N + S - 1 - i \cdot 2^8}{N-1}$, d'où la valeur de Pd correspondante.

Le problème est de calculer la valeur numérique de ces expressions. Cette méthode trouve ses limites pour des valeurs de N supérieures à 40.

**Approche probabiliste**

Avec cette méthode, une configuration donnée d'une mémoire de N octets est considérée comme étant la réalisation des N variables aléatoires correspondantes. Si on considère que la loi de distribution est la même pour chaque octet mémoire (moyenne μ, variance σ), lorsque N est grand (N>50), la distribution de la somme des variables aléatoires est très sensiblement normale, de moyenne Nμ et de variance Nσ, d'où la probabilité P(S=s) que la somme des N variables aléatoires soit égale à une valeur s :

$$P(S = s) \approx \frac{1}{\sigma \cdot \sqrt{2\pi N}} \cdot e^{\left[-\frac{1}{2N} \cdot \frac{(s - N\mu)^2}{\sigma^2}\right]}$$

Avec $\mu = \frac{2^8 - 1}{2} \;\; et \;\; \sigma^2 = \frac{1}{8} \sum_{i=0}^{2^8 - 1} (i^2 - \mu^2)$

Le pouvoir de détection est donné par : $Pd = 100 \cdot \left(1 - P(S = s)\right)$

Ces équations permettent de tracer des courbes donnant, pour une capacité mémoire donnée, le pouvoir de détection en fonction de la valeur de la somme des octets de la mémoire contrôlée (Figure 2-2, Figure 2-3, Figure 2-4).

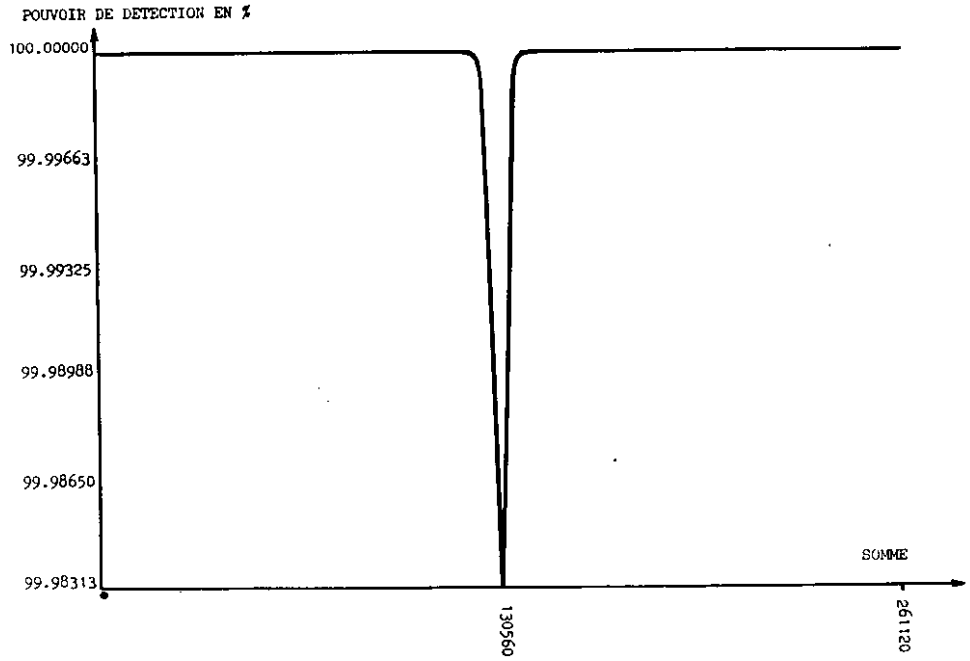



**Figure 2-2 : Pouvoir de détection d'un checksum d'une mémoire de 1 koctets**

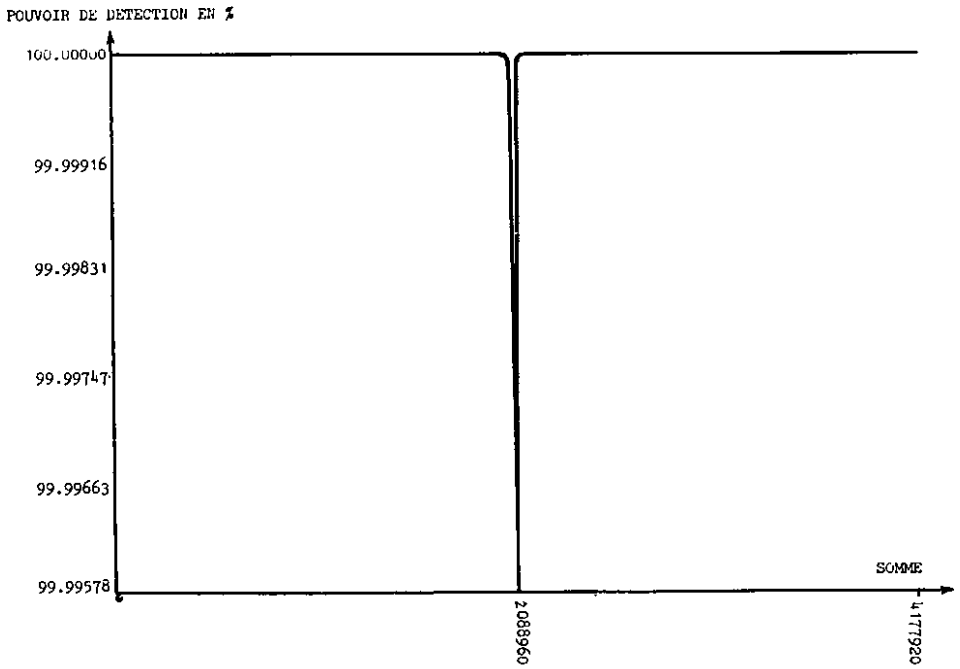



**Figure 2-3 : Pouvoir de détection d'un checksum d'une mémoire de 16 koctets**

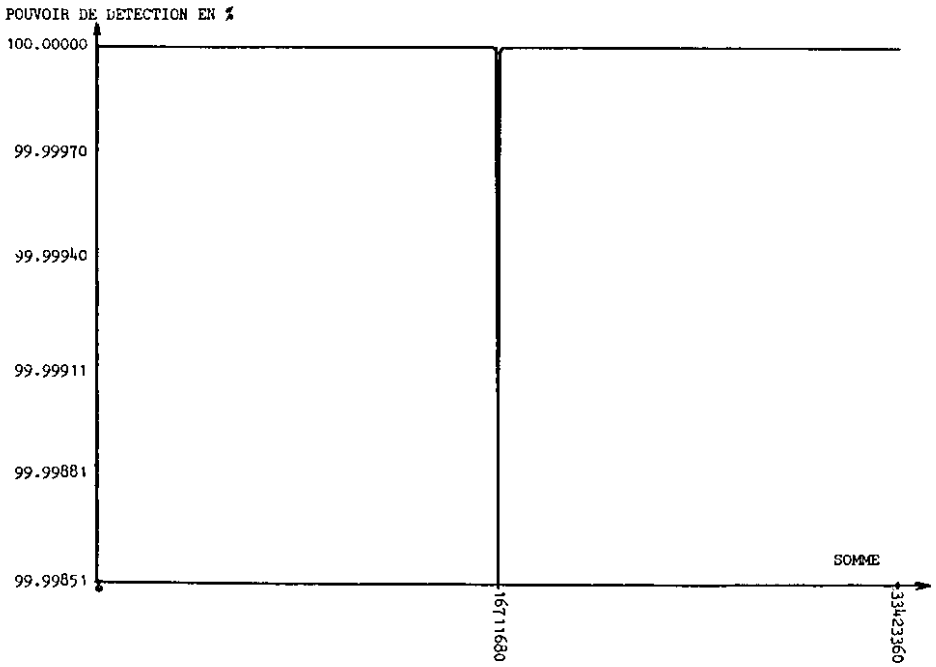



**Figure 2-4 : Pouvoir de détection d'un checksum d'une mémoire de 128 koctets**

De ces courbes, on constate :

- La valeur minimum du pouvoir de détection théorique croît avec le nombre N d'octets à contrôler.
- Il n'y a que peu de valeurs de la somme S pour lesquelles le pouvoir de détection théorique est inférieur à 99,998 % (pouvoir de détection du CRC). Le nombre des valeurs étant d'ailleurs proportionnellement moins élevé pour des grandes capacités mémoires (N élevé).
- La probabilité de détection de la somme de contrôle avec retenue devient rapidement très proche de 100%, dès que la somme S des N octets de la mémoire non erronée s'éloigne de la valeur moyenne (255xN)/2.

#### *2.1.1.3. Comparaison des deux méthodes*

- Les pouvoirs théoriques de détection des deux méthodes sont du même ordre de grandeur et que pour les tailles mémoire supérieures à 128 koctets, la somme de contrôle avec retenue a un pouvoir de détection supérieur à celui de la génération d'un signature sur 16 bits par CRC, et ce quelle que soit la somme d'origine.
- Il faut noter que la différence entre la valeur maximale du pouvoir de détection de la somme de contrôle et celui de la génération d'une signature est minime (0,015% pour une mémoire de 1 koctets).

- Les erreurs de même sens seront toujours détectées par la somme de contrôle avec retenue, alors qu'il existe des configurations erronées de la mémoire résultant de ce type d'erreurs qui ne seront pas détectées par la génération d'une signature par CRC [Charpentier et al., 1991].
- La somme de contrôle avec retenue ayant recours à des opérations simples et rapides est bien sûr plus aisée à mettre en œuvre. Elle permet d'obtenir une durée d'autotests bien inférieure à celui réalisé par la génération d'une signature par CRC [Charpentier et al., 1991].

Compte tenu des tailles de mémoire programme actuelles, ces constats font que la somme de contrôle avec retenue peut être préférée au CRC pour réaliser l'autotest d'une mémoire programme.

### 2.1.2. Détection des modifications des durées d'exécution par chien de garde

#### *2.1.2.1. Généralités*

La détection d'un chien de garde temporel appliqué à un dispositif à microprocesseur repose sur l'observation suivante : un grand nombre de fautes pouvant affecter ces dispositifs se traduisent par une altération de la séquence des instructions exécutées par le microprocesseur, et ont de ce fait une forte probabilité d'en modifier la durée d'exécution. Le chien de garde ne va donc pas rechercher à détecter la faute, mais sa conséquence. Pour cela, il mesure en temps réel la durée d'exécution d'un ensemble d'instructions, compare cette durée à une durée de référence préétablie lors de la conception et signale l'apparition de la faute lorsqu'il y a divergence entre la durée mesurée et la durée de référence [Charpentier, 1993].

Il doit donc exister un lien entre le logiciel et le matériel, le premier signalant son passage en des points précis prédéfinis, le second mesurant l'intervalle de temps séparant ces points de passage. Comme le montre la Figure 2-5, le contrôle de l'ensemble d'un logiciel nécessite le plus souvent de diviser ce logiciel en des groupes d'instructions de durées d'exécution connues et prédéterminées.

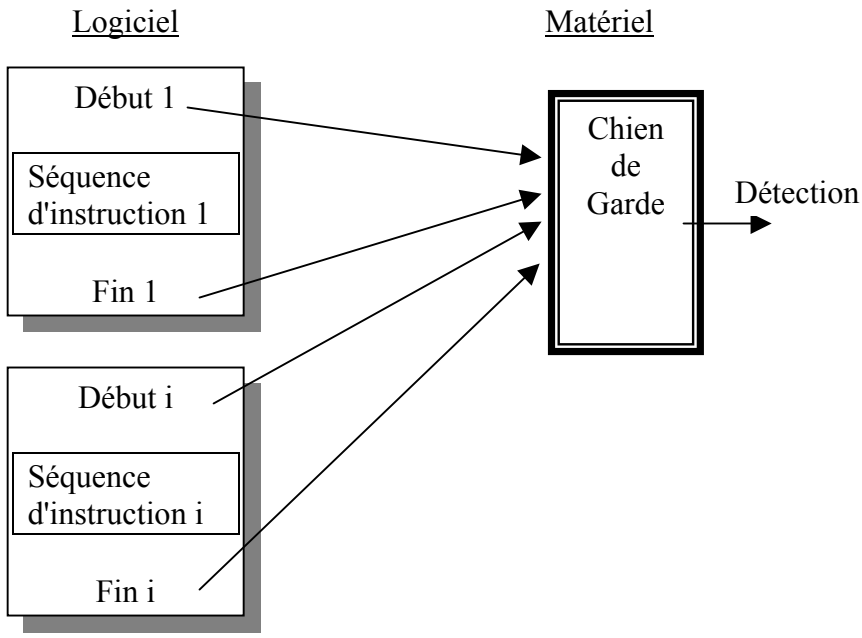



**Figure 2-5 : Principe de fonctionnement d'un chien de garde**

Les fautes détectées par le chien de garde seront celles qui ont provoqué soit la disparition de l'émission par le logiciel d'une ou plusieurs informations de début ou de fin de séquence, soit la variation de l'intervalle de temps séparant ces informations.

On peut envisager de détecter :

- des allongements de la durée d'exécution d'une séquence d'instruction, en vérifiant que le temps mesuré est inférieur au temps de référence ;
- des allongements et des raccourcissements de la durée d'exécution d'une séquence d'instruction, en vérifiant que le temps mesuré se situe dans une fourchette de temps centrée sur un temps de référence.

#### *2.1.2.2. Règles qualitatives de conception*

De la mise en pratique de ces deux méthodes de détection, il en résulte les règles suivantes [Charpentier, 1993] :

- Le chien de garde doit être indépendant du ou des composants qu'il surveille, ce qui évite que les fautes communes aux modules surveillant / surveillé ne soient pas détectées.
- Le chien de garde doit impérativement être relancé par un mécanisme matériel à chaque réinitialisation pour détecter les fautes se traduisant au démarrage par un blocage complet du fonctionnement du microprocesseur ( sans impulsions de relance).

- Le chien de garde ne doit pas être relancé par une procédure d'interruption, pour ne pas continuer à émettre des impulsions de relance alors que l'exécution du programme est altérée.

- Il est nécessaire de tester régulièrement le chien de garde pour s'assurer, le plus en aval de la chaîne de détection, que les circuits qui le composent ne sont pas défaillants.

- La meilleure précision sera obtenue lorsque la durée séparant deux impulsions de relance est la plus proche possible de la durée $T_{det}$ de déclenchement du chien de garde. Le choix de cette durée résulte d'un compromis :

    - Une faible durée nécessite d'insérer au sein du logiciel un nombre élevé de points de relance, augmentant la probabilité de réarmer le chien de garde suite à une rupture de séquence. De plus, si le logiciel comporte des séquences d'instructions de durées très grandes devant $T_{det}$, il est nécessaire de diviser ces séquences pour insérer les points de réarmement, ce qui peut rapidement devenir fastidieux.

    - Une durée élevée (par exemple celle de la boucle de programme la plus longue) peut conduire à réarmer le chien de garde bien avant l'expiration de sa durée de déclenchement, donc à en diminuer l'efficacité. Dans le cas d'un chien de garde à allongement et raccourcissement, il pourra être nécessaire d'introduire des boucles d'attente qui retarderont d'autant le fonctionnement de l'application. Enfin, une valeur élevée de $T_{det}$ pourra dans certains cas retarder la détection de fautes, ce qui peut être préjudiciable pour certaines applications.

#### *2.1.2.3. Évaluation des performances*

Les performances d'un chien de garde à "allongement" et à "allongement et raccourcissement" ont été évaluées par injection de fautes (collages d'un bit à 0 ou à 1) au sein de la mémoire programme du microprocesseur. Cette injection fournit un taux de couverture $\mathbf{T_c}$, rapport du nombre des fautes détectées (**Nb Det**) par le chien de garde sur le nombre des fautes injectées. L'exécution de deux modules logiciel différents a été contrôlée : logiciel 'cellule' pour gestion d'une cellule photoélectrique et logiciel 'calcul' pour le calcul du CRC et du checksum d'une zone mémoire programme. La période d'émission des impulsions de relance du chien de garde a été prise égale à 10 ms. 2552 défauts ont été injectés sur le logiciel 'cellule' et 2994 sur le logiciel de 'calcul'.

Les tableaux suivants regroupent les résultats obtenus.

<u>Proportions allongements / raccourcissements</u>

La fenêtre retenue est de 1 ms, centrée autour d'une valeur de 10 ms.

| | Logiciel 'cellule' | | Logiciel de 'calcul' | |
|---|---|---|---|---|
| | Nb Det | Tc en % | Nb Det | Tc en % |
| Nombre d'allongements | 1081 | 42,3 | 564 | 18,8 |
| Nombre de | 304 | 11,8 | 82 | 2,7 |

**Tableau 2-1 : Proportion allongements / raccourcissements**

Variation de la détection en fonction de la largeur de la fenêtre de détection

La borne maximale étant maintenue constante (10,3 ms), nous avons fait varier la borne minimale d'un chien de garde à allongements et raccourcissements pour le logiciel 'cellule'.

| $T_{min}$ en ms | Logiciel 'cellule' | |
|---|---|---|
| | Nb Det | Tc en % |
| 9,8 | 1371 | 53,7 |
| 8,4 | 1358 | 53,2 |
| 6,6 | 1338 | 52,4 |
| 4,1 | 1301 | 50,9 |
| 2,6 | 1298 | 50,8 |
| 0,98 | 1253 | 49,1 |
| 0,47 | 1251 | 49,0 |
| 0,13 | 1219 | 47,8 |

**Tableau 2-2 : Influence de la largeur de la fenêtre – borne minimale**

Variation de la détection en fonction de la largeur de la fenêtre de détection

Nous avons fait varier la borne maximale d'un chien de garde à allongements, la borne minimale étant fixée à 10 ms.

| | Logiciel 'cellule' | | Logiciel de 'calcul' | |
|---|---|---|---|---|
| $T_{min}$ en ms | Nb Det | Tc en % | Nb Det | Tc en % |
| 10,5 | 984 | 38,6 | 541 | 18,1 |
| 11,1 | 911 | 35,7 | 510 | 17,0 |
| 12,7 | 852 | 33,4 | 504 | 16,8 |
| 15,5 | 832 | 32,6 | 490 | 16,4 |
| 19,0 | 799 | 31,3 | 477 | 15,9 |
| 22,2 | 627 | 24,6 | 437 | 14,6 |
| 24,3 | 618 | 24,2 | 427 | 14,3 |
| 27,5 | 602 | 23,6 | 421 | 14,1 |
| 30,5 | 553 | 21,7 | 404 | 13,5 |
| 32,5 | 547 | 21,4 | 360 | 12,0 |

**Tableau 2-3 : Influence de la largeur de la fenêtre – borne maximale**

En conclusion

La variation importante des nombres de fautes détectées (donc des taux de couverture) d'une application à l'autre ne permet pas de généraliser les résultats obtenus. Cependant, on constate :

- que le nombre des fautes se traduisant par des raccourcissements est inférieur à celui obtenu pour les allongements, mais que ce nombre est suffisant pour recommander la détection des raccourcissements ;
- que la durée de la fenêtre de détection a une influence sur le nombre des fautes détectées. Il faut toutefois noter que des fenêtres de faibles durées sont très difficiles à mettre en œuvre ;
- qu'il est préférable de fixer la durée de déclenchement d'un chien de garde à allongements le plus proche possible de la période des impulsions de réarmement.

## *2.2. ÉVITEMENT DES FAUTES*

L'élimination des fautes est définie par le LAAS comme étant l'application de méthodes et techniques destinées à réduire la présence des fautes, en nombre et en sévérité. Ces techniques sont essentiellement relatives au test, du matériel comme du logiciel. A la différence de la tolérance aux fautes, l'élimination des fautes touche principalement les aspects organisationnels et tests liés au développement des systèmes.

Les méthodes d'élimination des fautes sont destinées à réduire la probabilité globale de défaillance systématique à un niveau que les experts de la norme IEC 61508 jugent équivalent à la probabilité de défaillances aléatoires du matériel.

Les fautes systématiques affectant essentiellement le logiciel, nos travaux, en coopération avec les sociétés VERIDATAS et SURLOG, ont été menés sur les moyens d'évitement par l'application d'un ensemble de prescriptions en terme de qualité et de sûreté de fonctionnement du logiciel, ainsi que par l'élimination des fautes grâce au test du logiciel [Charpentier, 1999]. Les logiciels concernés sont les logiciels embarqués des systèmes à microprocesseurs.

### 2.2.1. Prévention des fautes

Un ensemble optimal de prescriptions pour la construction de logiciel assurant des fonctions de sécurité a été défini. Il a servi de base à la rédaction d'une annexe à la norme CEI 62061 relative au logiciel embarqué. L'objectif principal est, par l'application de ces prescriptions, de prévenir les fautes logicielles ainsi que les comportements qui pourraient conduire à des dangers pour l'application globale [Charpentier, 1997].

Pour cela, trois groupes de prescriptions ont été définis [Charpentier et al., 2000b], [Charpentier et al., 2000c] :

- Prescriptions relatives au produit logiciel, qui décrivent les principales caractéristiques qu'un produit logiciel doit satisfaire pour garantir sa qualité et sa sûreté ;

- Prescriptions relatives au processus de développement, qui jalonnent les activités techniques liées au développement d'un logiciel ;

- Prescriptions relatives à la vérification du logiciel, qui servent de référence pour l'évaluation du logiciel.

Les prescriptions données (de l'ordre de 60) sont en partie issues de la norme CEI 61508-3. Elles ne concernent que les produits logiciel inclus dans des systèmes de commande devant assurer des fonctions de sécurité. Elles peuvent être utilisées pour tous types de machines, sachant que le niveau de sécurité à atteindre n'est pas aussi élevé que celui demandé par exemple dans les industries nucléaires ou aéronautiques.

Le Tableau 2-4 donne l'exemple des prescriptions relatives au codage du logiciel. Deux niveaux ont été définis, qui sont en lien avec le niveau de criticité du logiciel à développer. Le niveau 1 est le niveau d'exigence le moins élevé. Il s'applique à des logiciels qui ont une faible influence sur le niveau de sécurité global de l'application (par exemple parce qu'une redondance diversitaire a été mise en œuvre). La lettre O indique une prescription 'Obligatoire' et la lettre R correspond à une prescription 'Recommandée'. Les indications en caractères italiques correspondent à des commentaires susceptibles d'aider le développeur à comprendre et respecter la prescription.

La suite de ce chapitre donne, pour chaque classe de prescriptions, le but visé par la ou les prescriptions. L'ensemble des prescriptions est disponible dans la référence [Charpentier et al., 2000b].

| No. | Codage | Niveau | |
|---|---|---|---|
| | | 1 | 2 |
| 1.14 | Le code source doit :<br>1. pouvoir être lu, compris et soumis à essais,<br>2. répondre aux spécifications de conception du module logiciel,<br>3. répondre aux prescriptions du manuel de codage. | O | O |
| 1.15 | Les règles de codage applicables au logiciel doivent être détaillées dans un manuel de codage et utilisés pour le développement du logiciel. Le manuel de codage doit :<br>- indiquer les principes de programmation à appliquer et interdire les caractéristiques peu sûres du langage.<br>- décrire les règles de présentation et de documentation du code source<br>- inclure les conventions de dénomination des composants, sous-programmes, variables et constantes.<br>*Par exemple, présentation décalée des différents blocs d'instructions (indentation), l'utilisation des lignes blanches, le contenu de l'entête de fichier source (nom de l'auteur, données d'entrées et de sortie, données globales modifiées, ...). Ces conventions concourent à une meilleure lisibilité et maintenabilité du logiciel.* | R | O |

**Tableau 2-4 : Prescriptions relatives au codage**

#### *2.2.1.1. Prescriptions relatives au produit logiciel*

Ces prescriptions sont destinées à obtenir un logiciel d'une qualité telle que le nombre de fautes qu'il contient soit minimisé. Pour cela, il est nécessaire d'établir le plus tôt possible dans le cycle de développement un certain nombre d'activités, de mettre en place une organisation et de donner des principes de conception.

- Interface avec l'architecture du système

L'objectif est de définir et de documenter l'ensemble des contraintes imposées par l'architecture matérielle sur le logiciel, ainsi que les conséquences des interactions entre le matériel et le logiciel. C'est donc une approche "système", grâce à laquelle le concepteur peut anticiper d'éventuels problèmes d'interfaçage difficiles à détecter puis à corriger en fin de cycle de vie.

- Spécification du logiciel

Ces prescriptions amènent le concepteur à spécifier les caractéristiques du logiciel à développer : fonctions de sécurité avec leurs caractéristiques d'intégrité de sécurité, de temps de réponse, de contrôles périodiques, …

- Logiciel paramétrable par l'utilisateur

Ces prescriptions concernent les produits logiciels conçus pour être paramétrés par l'utilisateur (cas courant et recommandé en sécurité des machines). Le paramétrage peut avoir différents degrés de complexité, de la table de messages à des options du système. Comme pour le logiciel, ces paramètres doivent être spécifiés précisément et leur programmation ne doit pas influer sur le logiciel système. De plus, le logiciel système doit vérifier la validité des paramètres.

- Logiciel déjà existant

Le terme de logiciel 'préexistant' fait référence à des modules sources qui n'ont pas été développés spécifiquement pour le système dans lequel ils vont être intégrés. Sont notamment inclus les logiciels développés pour d'autres projets ainsi que les logiciels commerciaux, par exemple des modules de calculs scientifiques.

Dans ce dernier cas, le concepteur a rarement accès aux éléments nécessaires pour s'assurer du respect des prescriptions formulées. Il faut alors s'assurer que l'introduction de tels modules ne dégradera pas le niveau de sécurité du système, soit en réalisant les mêmes vérifications que celles opérées pour les autres parties du logiciel, soit en tenant compte du retour d'expérience obtenu suite à l'utilisation de ces modules dans des applications similaires.

- Conception du logiciel

L'objectif est de s'assurer que le concepteur a décrit un ensemble minimal de caractéristiques (architecture, entrées / sorties, structure des données, …) pour concevoir un logiciel exempt de fautes. Le moyen proposé consiste d'une part à imposer une programmation modulaire (un module / une fonction) et d'autre part à réaliser une interface simple et claire avec l'environnement.

- Langages de développement

L'objectif est de s'assurer que le langage de programmation et les outils de développement associés sont adaptés au programme à développer et qu'ils n'introduiront pas de fautes dans le code exécutable. Les limites éventuelles devront être clairement définies et, le cas échéant, être palliées par le respect de règles de programmation.

Note : Les langages de programmation couramment utilisés pour programmer le logiciel embarqué des systèmes de commande de machines sont des langages assembleur ou des langages évolués de type C.

- Codage

Le codage est la dernière étape de la partie descendante (conception) du cycle de vie d'un logiciel. L'objectif essentiel est de développer un code lisible, cohérent avec les spécifications et qui respecte les règles de codage associées au langage utilisé.

#### *2.2.1.2. Prescriptions relatives au processus de développement*

L'objectif principal des prescriptions relatives au processus de développement du logiciel est d'orienter le développement de façon telle qu'il satisfasse les spécifications fonctionnelles prescrites pour le système et qu'il donne la confiance nécessaire pour satisfaire les contraintes de sécurité. Le but recherché est d'éviter les fautes, en faisant l'hypothèse qu'un logiciel développé en suivant un processus formellement défini contiendra moins de fautes que le même logiciel développé de façon 'anarchique' ou désordonnée.

- Cycle de vie

Le cycle de vie du logiciel est utilisé pour définir et coordonner les activités liées au développement du logiciel (de la spécification à la validation) ainsi que les activités connexes telles que la documentation ou la gestion des modifications et de la configuration. La maîtrise du développement d'un logiciel passe par une spécification claire du cycle de vie et des différentes phases qui le composent.

- Organisation

La première des actions à mener pour organiser le processus de développement d'un logiciel consiste à mettre en place un système d'assurance qualité adapté. On s'assure ainsi que le produit logiciel, sa documentation et les activités du cycle de vie sont conformes aux prescriptions édictées et que les éventuelles divergences seront détectées, évaluées, gérées et résolues.

La 'supervision' de la sécurité et la gestion des tâches sont utilisées pour gérer la mise en place des mesures de sécurité lors du développement du projet et pour s'assurer que toutes les activités attendues sont effectivement mises en place. Elle nécessite notamment de contrôler la disponibilité des ressources, de surveiller les points de rendez-vous critiques du projet et le cas échéant, de suivre les sous-contractants.

- Documentation

L'objectif à atteindre n'est pas de prescrire un ensemble prédéfini de documents mais d'établir une base commune de référence pour le développeur (au sens large du terme) et l'analyste chargé d'évaluer le logiciel. On demandera donc de lister les documents produits lors du développement et de leur donner une forme définie facilitant l'exploitation par les deux parties. Le repérage par activités du cycle de vie sera facilité par l'utilisation de matrices de traçabilité reliant l'activité à un ou plusieurs documents.

- Gestion de la configuration et des modifications

Un rapport d'analyse n'étant valide que pour une configuration bien identifiée, la gestion de la configuration, donc des versions, est indispensable à un développement devant être approuvé. Elle inclut les activités d'identification de la configuration, de gestion des modifications, d'archivage du logiciel et des données associées. La gestion de configuration et des modifications produit :

- une configuration définie, maîtrisée et archivée du logiciel pouvant être utilisée pour reproduire un code exécutable cohérent ;

- une base de référence reconnue pour l'évaluation et la gestion des modifications ;
- un moyen de contrôle garantissant que les problèmes ont été convenablement traités et que les modifications éventuelles ont été effectuées.

- Outils de développement

L'objectif est de ne pas introduire de fautes logicielles via les outils de développement et de programmation utilisés. Pour cela, il faut veiller :

- à ne pas utiliser les possibilités d'optimisations de code permises par le compilateur ;
- à s'assurer que les outils utilisés sont correctement référencés et qu'un changement de ces outils est sans influence sur le code généré, ou que les modifications sont sans conséquences sur la sécurité du système.

#### *2.2.1.3. Prescriptions relatives à la vérification du logiciel*

Les activités de vérification ont pour but de démontrer que les produits logiciels issus d'une phase du cycle de développement sont conformes aux spécifications établies lors des phases précédentes ainsi qu'aux règles ou normes applicables. Elles ont également pour but de détecter et de rendre compte des erreurs introduites en cours de développement du logiciel.

La vérification du logiciel n'est pas uniquement composée de tests, bien que pour les logiciels de faible taille considérés en sécurité des machines il s'agisse de la partie prédominante. Les activités de revues et d'analyse sont également des activités de vérification. Elles peuvent dans certains cas remplacer certains tests (exemple : un test impossible à réaliser sans détérioration d'un composant matériel).

Il est important de noter la rapide augmentation du coût de correction d'une erreur en fonction de la phase où elle est découverte. Le coût optimal correspond à une correction le plus tôt possible dans le cycle de vie. Par exemple, la correction d'une erreur découverte dès la phase de spécification du logiciel par une vérification de cohérence entre spécification du logiciel et spécification du système est bien moins coûteuse que si elle est découverte en fin de cycle de développement (en validation du logiciel ou du système). Une découverte tardive nécessite la reprise de toutes les phases de développement impactées et déjà achevées.

Les prescriptions de test (unitaire, intégration, validation) sont présentées en correspondance de chaque étape de réalisation du logiciel (spécification, conception préliminaire et détaillée, codage) en complément de prescriptions générales sur la vérification du logiciel.

- Prescriptions générales de validation et de vérification

Ces prescriptions doivent donner à l'analyste les moyens d'effectuer les vérifications dans des conditions optimales : travail en amont, si possible dès la phase de spécification ; évaluation à chaque phase du cycle de vie ; mise à disposition de la documentation correspondant aux vérifications à effectuer.

- Prescriptions pour la vérification

Les activités de vérification doivent faire l'objet de rapports pour notamment identifier les non-conformités afin d'être capable d'en analyser l'influence (et la criticité) sur la sécurité du système.

Les revues sont une des composantes importantes de la vérification. Elles doivent au moins être menées pour les phases :

- de spécification, pour s'assurer que les besoins réels sont bien intégrés dans les spécifications, que les risques ont été identifiés et réduits par l'utilisation de mesures adéquates, et que les spécifications du logiciel sont cohérentes avec celles du système ;
- de validation, pour déterminer si le produit satisfait ou non les spécifications.

La vérification du code et des données complète les revues. C'est une vérification statique basée sur des lectures croisées ou des inspections. C'est un préalable à la vérification dynamique (c'est à dire aux tests).

- Prescriptions pour le test du logiciel

Une stratégie de test doit être établie, qui indique l'approche adoptée, les objectifs établis en termes de couverture de test, de critère de succès, d'environnement et de techniques utilisées. Ces objectifs doivent être adaptés au type de logiciel développé ainsi qu'à son niveau d'intégrité. On peut ensuite déterminer la nature des tests à effectuer : tests fonctionnels, tests aux limites, tests de performance, tests de charge, … ainsi que les items concernés par les tests : fonctions de sécurité, tous les éléments de la spécification, …

Les tests du logiciel peuvent être réalisés à différentes phases du cycle de vie :

- **des tests unitaires** (vérification de la conception détaillée). Ils démontrent que chaque module effectue toute la fonction prévue et seulement la fonction prévue. On distingue : les tests de logique (recherche d'erreur, vérification de l'enchaînement correct des branches parcourues, recherche de comportement anormaux) et les tests de calcul (vérification des résultats des calculs, des performances, de l'exactitude des algorithmes).

Typiquement, les essais de calcul comprennent les tests de données dans les limites des spécifications, en dehors des limites de spécification (état anormal), aux limites spécifiées, aux singularités algorithmiques. Les tests de comportement anormaux (hors limites, singularités algorithmiques, erreurs) sont généralement appelés tests de robustesse.

Cette activité, indispensable pour des logiciels complexes et de taille importante, est seulement conseillée pour les logiciels de petite taille. Cette conformité peut également être démontrée par des techniques statiques (relecture de code par exemple). Cette phase de vérification révèle des erreurs du type :

- inaptitude d'un algorithme à satisfaire une spécification du logiciel,
- opérations de boucle incorrecte,
- décision logique incorrecte,
- inaptitude à traiter correctement des combinaisons valides de données d'entrées,
- réponses incorrectes à des données d'entrées manquantes ou altérées,
- violation de limites de tableaux,
- séquence de calcul incorrecte,
- précision, justesse ou performances inadéquates d'un algorithme.

  - **des tests d'intégration** (vérification de la conception). Ils démontrent le bon fonctionnement d'unités fonctionnelles constituées d'un assemblage de modules. Ils portent principalement sur la vérification des enchaînements de modules, la circulation des données, les aspects dynamiques et les séquences d'événements prévus. Typiquement, ils comportent des tests de couplages entre modules, des aspects dynamiques, des séquences d'événements prévus, des reprises en cas d'interruption.

Les tests d'intégration révèlent des erreurs du type :

- initialisation incorrecte des variables et des constantes,
- erreurs dans le passage de paramètres,
- altération de données, en particulier de données globales,
- résolution numérique de bout en bout inadéquate,
- séquencement incorrect d'événements et d'opérations.

  - **des tests de validation** (vérification des spécifications). Ils vérifient que le logiciel implanté dans le matériel répond aux spécifications fonctionnelles en vérifiant plus particulièrement les interfaces matériel/logiciel, les performances, le fonctionnement temps réel, les fonctions générales, l'utilisation et l'allocation des ressources.

Les erreurs relevées par ces tests sont du type :

- mécanisme incorrect de traitement des interruptions,
- non respect des prescriptions de temps d'exécution,

- réponse incorrecte du logiciel en fonctionnement transitoire (mise en fonctionnement, flux d'entrées, basculement en fonctionnement dégradé, ...),

- conflits d'accès à des ressources ou problèmes d'organisation de la mémoire,

- inaptitude des tests intégrés à détecter des pannes,

- erreurs d'interface logiciel/matériel,

- débordement de piles.

### 2.2.2. Élimination des fautes

Nous avons vu précédemment que le test est la composante principale de la vérification du logiciel. Il permet de s'assurer de la conformité d'une entité vis-à-vis de ses spécifications, l'entité pouvant être une fonction, un module logiciel ou encore un chemin particulier. Du fait de la complexité et de la taille des logiciels, il est inapproprié de se référer à une quelconque exhaustivité, qui signifierait que tous les chemins (entités de base d'un logiciel) seraient parcourus dans toutes les configurations possibles. On recherchera donc plutôt la suffisance des tests, laquelle donnera confiance à l'analyste dans le futur comportement en opérationnel. Les travaux ont consisté à proposer une démarche de construction des tests du logiciel pour en assurer la suffisance [Charpentier et al., 2000a], [Charpentier et al., 2000d].

#### *2.2.2.1. Généralités*

Le test du logiciel [Myers, 1979], [Beizer, 1992], [Xanthakis et al., 1994] est une approche dynamique de la vérification, destinée à s'assurer que ce logiciel possède effectivement les caractéristiques requises pour son contexte d'utilisation. La première action à entreprendre est donc de décrire avec précision ce contexte, en particulier les fonctionnalités attendues, les contraintes d'environnement ou encore les situations dangereuses à considérer.

Le test a pour objectifs :

1. de détecter d'éventuels écarts entre le comportement attendu et le comportement observé au cours des tests, ce qui élimine un grand nombre de fautes présentes dans le logiciel.

2. d'obtenir la confiance nécessaire avant l'utilisation opérationnelle. Il faut cependant noter que le nombre de fautes détectées ne peut pas être considéré comme un critère de réussite des tests. Le retour d'expérience montre en effet qu'à complexité technique et industrielle constante, un grand nombre d'erreurs détectées par rapport à d'autres projets "de référence" indique seulement un logiciel contenant un très grand nombre de fautes et non comme l'atteinte d'un bon taux de détection des fautes présentes. En conséquence, il est très difficile d'avoir confiance en un logiciel ayant un grand nombre de fautes détectées par le test.

**Remarques :**

- L'observation d'un logiciel sous test, l'analyse de l'architecture du logiciel et du codage, l'évaluation du processus de développement sont autant d'éléments pour répondre à la question : *Est-ce que le logiciel est bien fait?*

  Mais, pour traiter complètement la sûreté de fonctionnement d'un système, il faut être également en mesure de répondre à la question : *Est-ce que le logiciel réalisé correspond bien l'application à traiter?* Seule l'analyse du système pourrait apporter des réponses à cette question, ce qui dépasse l'étude des tests du logiciel.

- Il est illusoire de qualifier un système complexe à la seule observation de son comportement sur un nombre limité de jeux de tests. Les tests sont essentiels pour détecter des erreurs et améliorer la confiance dans l'aptitude du système à accomplir sa mission, mais insuffisants pour "qualifier" le comportement d'un système.

- Il existe un risque pour le concepteur de faire définir par une même entité (individu, équipe), la spécification, la conception, la stratégie de tests et les cas de tests. Deux raisons au moins justifient le recours à un tiers pour tester un logiciel [Myers, 1979] :

  - le but des tests est d'exécuter un programme avec l'intention de trouver ses erreurs [Myers, 1979], ce qui constitue un processus mental non naturel difficile à mener par une même entité,
  - le programme peut contenir des erreurs dues à la non compréhension de l'implémentation ou des spécifications par le développeur [Myers, 1979]. Dans ce cas, il est probable que celui-ci aura ces mêmes non compréhensions quand il testera son propre programme (mode commun de défaillance).

- La correction des fautes logicielles peut :

  - injecter de nouvelles fautes et perturber des parties correctes déjà testées ;
  - rendre active une partie du logiciel jusqu'alors inaccessible et donc révéler un grand nombre de nouvelles fautes, qui ne pouvaient être constatées avant cette correction.

La suite de ce paragraphe donne la démarche établie à l'INRS pour le test des logiciels implantés dans les systèmes de sécurité de machines. Elle est destinée à faciliter le respect des prescriptions évoquées au chapitre précédent et éviter ainsi au maximum l'introduction de fautes au sein d'un logiciel. Les activités liées aux trois phases de tests sont identiques.

La démarche de test (unitaire, d'intégration et de validation), pour qu'elle soit de qualité, doit suivre les mêmes principes que tous les autres processus de réalisation ou de création à savoir :

- définition au préalable des tests à réaliser, qui doit être préparée dès la phase de spécification / conception correspondante ;
- exécution selon des procédures prévues à chaque étape de la partie ascendante du cycle de vie ;
- vérification selon des modalités fixées en fin de test.

#### *2.2.2.2. Définition préalable des tests à réaliser*

La définition des tests fait l'objet des plans de tests (de validation, d'intégration, unitaires) et nécessite la rédaction de procédures associées.

- Objectifs de tests

Définition des objectifs

Dans l'absolu, l'effort de test d'un logiciel peut être aussi important que l'on veut. La combinatoire des données d'entrée, des modes d'utilisation, des modes de fonctionnement, des paramètres d'exploitation, la variété des aspects vérifiables (conformités aux spécifications, aux manuels d'utilisation et d'exploitation, aux prescriptions de robustesse, de performance, d'ergonomie, comportement nominal et en cas de défaillance, etc.) sont telles qu'une infinité de tests peut être menée sans qu'on soit certain de la conformité totale aux prescriptions.

En pratique, l'effort de test doit donc être adapté aux enjeux. Il nécessite de se fixer des objectifs préalablement à toute définition d'une stratégie, de méthodes et techniques, etc... et finalement des tests eux-mêmes.

Définir ces objectifs permet de réaliser les tests avec un maximum d'efficacité en garantissant la couverture des objectifs définis, donc de focaliser les efforts sur les aspects essentiels.

Les **objectifs** de chaque phase de tests seront identifiés et définis en considérant :

- les prescriptions fonctionnelles et de performance (sans omettre les contraintes particulières de l'environnement final d'utilisation du logiciel), exprimées dans les documents de spécification / conception du produit logiciel ;
- les prescriptions en matière de qualité et de sûreté de fonctionnement (robustesse, maintenabilité, disponibilité, sécurité, …) ;
- toutes les contraintes identifiées d'implémentation (utilisation d'une base de données, utilisation des nombres réels, langage de programmation, logiciel temps réel…) ;

- les exigences légales et réglementaires ;
- les contraintes de coût et de temps de test.

Il en résulte la définition des **types de tests** à réaliser (chemins structurels, chemins fonctionnels, données d'entrées / sorties aux limites, hors limites, de robustesse, de performance ...) et la justification de ces choix.

Taux de couverture

Le taux de couverture des tests du logiciel est défini par rapport aux objectifs de test. C'est le ratio entre les objectifs de tests effectivement réalisés et les objectifs fixés initialement pour le test concerné. Pour chaque type de tests défini, **la prescription sur le taux de couverture des tests** à atteindre est donc en toute rigueur de 100%.

**Exemple :** Si le choix de tester les chemins fonctionnels est retenu, un taux de couverture de 100% signifie qu'après réalisation des tests, chacun des chemins fonctionnels du logiciel doit avoir été parcouru au moins une fois. Il est bien sûr nécessaire d'avoir les moyens de mesurer le taux de couverture réel obtenu pour le comparer au taux de couverture souhaité (ici de 100%).

Mais ce taux peut être réduit par des hypothèses à justifier :

- Il se peut, par exemple, lors des tests de "validation site", que le logiciel soit dans un environnement particulier empêchant le test de certaines fonctionnalités prévues pour un environnement différent. Le taux de couverture ne pourra donc pas atteindre 100% des fonctionnalités décrites dans les spécifications. La justification de la diminution du taux de couverture devra être consignée dans le dossier de test.
- De même, certains tests portant sur les aspects "temps réel" ne sont pas toujours réalisables en tests unitaires. Une justification adaptée limitera le taux de couverture des aspects temps réel durant la phase de tests unitaires.

**Attention :**

Fixer a priori un objectif en terme de taux de couverture peut conduire le concepteur à dégrader son code pour atteindre plus facilement les objectifs. Une valeur de 100% pour un taux de couverture en branches ou en appels de fonctions peut être obtenue en diminuant "artificiellement" le nombre d'appels de procédures par duplication du code ou allongement de la taille moyenne des procédures.

Toute démarche d'objectif de taux de couverture doit donc être accompagnée de règles de programmation évitant ce type d'effet pervers.

- Stratégie de test

Les tests doivent être conçus et réalisés sur la base d'une stratégie. Elle constitue un préalable nécessaire à l'élaboration des tests et à la conception des

jeux d'essai. La stratégie de test témoigne d'une réflexion sur les tests. Son absence indique un risque d'improvisation dans l'élaboration des tests.

La stratégie doit définir comment, globalement, les tests vont être menés pour satisfaire aux objectifs préalablement définis, en relation avec les environnements que le concepteur souhaite mettre en œuvre. Elle doit organiser les tests en une structure lisible d'étapes, ayant des objectifs clairs ou répondant à des contraintes d'environnement précises (par exemple relatives à la disponibilité d'un outil).

On définit donc les **méthodes** et les **moyens de tests** pour y parvenir, puis l'enchaînement - ou **planification** - des tâches de réalisation des tests. L'ensemble des moyens de tests et de la planification constitue la **stratégie de test** mise en œuvre.

La définition des tests aboutit à la réalisation des **fiches de tests**.

Définition des procédures de test

Une stratégie de test correcte implique que les procédures de test soient réalisées le plus en amont possible dans le cycle de développement.

Cette remarque est particulièrement importante en l'évaluation. En effet, le valideur comme le concepteur auront toujours intérêt à ce que ces procédures soient évaluées avant leur déroulement plutôt qu'après. On évitera ainsi de remettre en cause la phase correspondante de test et l'évaluateur n'aura pas à concevoir de tests complémentaires lors de l'évaluation finale.

Il est nécessaire que ces procédures contiennent toutes les informations nécessaires :

- à leur identification et leur traçabilité vis-à-vis des objectifs et / ou de la stratégie ;
- à l'exécution du test pendant la phase (activités de préparation, de déroulement des tests, d'enregistrement des anomalies et le critère d'arrêt des tests).

Méthodes et techniques de test : Techniques de base

Les techniques de base sont réparties en deux catégories suivant qu'elles autorisent ou non l'observation du comportement interne du composant logiciel sous tests :

- boîte noire ou test fonctionnel : pas de visibilité du composant sous test, les entrées et les résultats attendus sont exprimés en terme de comportement du composant logiciel d'un point de vue externe, sans se soucier de la structure interne du composant ;
- boîte blanche ou test structurel : visibilité du composant sous test, les jeux de test sont produits en analysant le code source.

Moyens de test

La définition de la stratégie et des méthodes de test doit s'accompagner de la définition des moyens nécessaires pour les mettre en œuvre. L'absence de cette information diminue fortement la crédibilité de la stratégie et des méthodes envisagées.

A titre d'exemple, la liste ci-dessous fournit des moyens à définir au préalable :

- outils : simulateurs, comparateurs, outils de mesure des signaux, outils de test, etc.
- environnements de test : configurations différentes du système, en usine, sur site, éventuellement variable en fonction du paramétrage, etc.
- jeux d'essai préenregistrés, tables pré-définies de paramètres, etc.

En particulier, pour les tests unitaires, la définition d'une stratégie au plus tôt impose l'utilisation d'outils adaptés à la taille du logiciel. Ces outils automatisent l'exécution (et la réexécution éventuelle) des tests et la mesure des taux de couverture.

Produits issus de la définition des tests

Les produits d'une phase de test (ou livrables) sont des documents ou des résultats sous forme informatique (fichier texte, objets d'un atelier de génie logiciel) observables comme preuve d'une activité du processus de test.

Outre les variations de cycle de vie entre les différents projets, il peut y avoir des différences plus ou moins importantes sur le nom et la répartition des informations entre ces livrables. On vérifiera la présence des informations utiles et leur pertinence plutôt que leur répartition dans une arborescence documentaire « théorique ».

#### *2.2.2.3. Exécution des tests*

L'exécution des tests doit se conformer aux procédures et aux conditions définies préalablement dans les plans de test et les fiches de test (activités de préparation, de déroulement des tests, d'enregistrement des anomalies). Tout non-respect de celles-ci doit être argumenté pour justifier ces divergences lors de la vérification des tests.

- Table de traçabilité

Il est souhaitable d'établir une table de traçabilité lors de l'exécution des tests, pour démontrer :

- que les tests référencés par la table sont bien documentés par une procédure ;
- qu'il n'existe pas de tests qui ne soient pas référencés par la table.

Les tables de traçabilité fournissent une correspondance entre les tests réalisés et les objectifs définis pour ces tests. Le tableau ci-dessous présente un format possible de table de traçabilité.

| Objectif élémentaire à vérifier | type de tests | | | |
|---|---|---|---|---|
| | TN | THL | TR | … |
| Objectif 1 | 1,2 | | 4,5 | |
| Objectif 2 | 6 | 3 | | |
| Objectif 3 | 7,8 | 9 | | |
| Objectif 4 | … | | | |
| Objectif … | | | | |
| Objectif n | m | m+1 | | |

n° du test

**Tableau 2-5 : Table de traçabilité Objectif / Type de test / Numéro de test**

TN : Test Nominal, THL : Test Hors Limite, TR : Test de Robustesse

- Critère d'arrêt des tests

Le critère d'arrêt des tests traduit la politique relative à la poursuite des tests sur détection d'anomalie. Il doit être défini au préalable pour éviter de réaliser des tests inutiles, qui devront être rejoués. Tout événement bloquant pour la suite correcte du processus de test doit être considéré dans la définition de ce critère. Cela peut être la découverte d'un bug ou encore d'une non-conformité majeure. La détection d'un événement constituant le critère d'arrêt des tests va entraîner l'arrêt du groupe de tests en cours avant son terme et le passage au groupe de tests suivant. Le groupe de tests arrêté devra de nouveau être réalisé après correction.

Remarques :

- La durée des tests et les efforts engagés ne sont, en aucun cas, des critères d'arrêt des tests : aucune garantie sur le logiciel ne peut être établie sur ces seules observations.
- La diminution du nombre d'erreurs constatées par unité de temps n'est pas non plus un critère d'arrêt. Elle résulte souvent de la difficulté pour l'équipe de tests de trouver de nouveaux tests pour mettre en évidence des erreurs logicielles. La modification de l'équipe de tests ou la mise en place de nouveaux outils augmente souvent le nombre d'erreurs détectées. Seule la non diminution est un élément bloquant pour l'évaluation.
- Les mesures de taux de couverture[6] pourraient être un critère d'arrêt s'il était possible d'évaluer la probabilité de présence d'erreurs non détectées. Elles seront cependant prises en considération comme un des éléments de l'évaluation.

[6] Taux de couverture: Pourcentage de cas de tests réalisés par rapport aux cas de tests envisageables dans une classe de tests.

Conclusion

En pratique, l'arrêt des tests se fait suite à l'analyse des événements redoutés du système et des aspects qui ne sont couverts par aucune des étapes de tests. De cette analyse on déduit si le risque lié à cette "non couverture" est acceptable et conforme aux prescriptions applicables.

#### *2.2.2.4. Vérification des tests*

La vérification des tests doit démontrer que les objectifs de test sont atteints. Elle consiste à analyser les produits de la phase à vérifier pour :

- Établir la traçabilité des tests réalisés pour chacun des types de tests définis dans les objectifs. Il est nécessaire d'établir la traçabilité au fur et à mesure de la réalisation des tests afin d'optimiser la phase de vérification des tests pour laquelle cette traçabilité est nécessaire.

Si la matrice de traçabilité n'existe pas, le contenu de chaque procédure de tests doit être analysé pour établir la matrice. Cette évaluation doit être affinée en examinant sur le fond les procédures pour s'assurer que :

- les tests sont valides, pertinents et reproductibles ;
- les tests couvrant un objectif donné sont réellement probants pour l'objectif ;
- les méthodes de test spécifiées lors de la planification des tests sont bien utilisées.

  - Évaluer le taux de couverture atteint par analyse des résultats de tests.
  - Établir l'acceptation ou le refus des résultats de la phase en prenant en compte les justifications de divergences établies lors de l'exécution des tests.

## *2.3. PRÉVISION DES FAUTES*

La prévision des fautes est un des aspects à couvrir pour construire la Sûreté de Fonctionnement d'un système. Elle consiste à évaluer le comportement du système par rapport à l'apparition des fautes et se base sur un ensemble des méthodes et techniques pour estimer **la présence, la création** et **les conséquences** des fautes.

L'évaluation du comportement d'un système par rapport à l'apparition de fautes accidentelles, nécessaire à la prévision des fautes, peut être qualitative ou quantitative [Laprie et al., 1995] :

- ***Évaluation qualitative***

L'évaluation qualitative est destinée à identifier, classer et ordonner les défaillances ou les méthodes mises en œuvre pour les éviter. Elle est à rapprocher des prescriptions déterministes contenues dans la norme EN 954 [EN954, 1996], qui imposent à

l'analyste de vérifier le comportement d'un système en présence de fautes prédéterminées. Elle pose le problème de la définition des fautes à considérer pour l'évaluation, que celle-ci se fasse sur le système considéré ou sur un modèle. Ce problème est d'autant plus aigu que les composants utilisés sont complexes.

En effet, si les fautes sont connues de façon satisfaisante pour les composants électroniques discrets ou faiblement intégrés (court-circuit, circuit ouvert collage aux états logiques '0' ou '1'), il n'en est pas de même pour les composants complexes de type microprocesseur, et plus encore pour le logiciel. Il faut effectuer les analyses après avoir formulé des hypothèses sur les modes de défaillances de ces composants. De telles hypothèses trouvant rapidement leurs limites, la solution généralement adoptée consiste à se placer à un niveau supérieur et à considérer les défaillances au niveau des fonctions.

- ***Évaluation quantitative***

L'évaluation 'déterministe' permet d'appréhender le niveau de sécurité d'un dispositif utilisant des composants électromécaniques, discrets ou faiblement intégrés, pour lesquels les modes de défaillances sont connus de façon satisfaisante. Par contre, ses capacités sont plus limitées lorsque des composants fortement intégrés sont employés, puisqu'il est impossible de connaître avec certitude et de façon exhaustive les modes de défaillances de ces composants.

D'autres approches doivent être envisagées, comme l'évaluation quantitative (par des calculs probabilistes) telle que préconisée dans le projet de norme CEI 62061. Plusieurs résultats peuvent être obtenus grâce à ces calculs : fiabilité, disponibilité, sécurité. Étant donné le domaine d'application visé par ce document - sécurité des machines - seuls les calculs nécessaires à la détermination de la probabilité de défaillance dangereuse seront développés dans le chapitre 4.

### 2.3.1. Injection de fautes via la mémoire programme des microprocesseurs

L'injection de fautes est une des techniques qualitatives utilisable pour appréhender le comportement d'un système électronique en présence de fautes de ses composants [Laprie et al., 1995], [Charpentier et al., 1994]. La démarche naturelle consiste à recenser les fautes pouvant affecter les différents composants d'un système, pour les injecter soit sur un modèle (simulation informatique), soit directement sur le système à valider. L'observation du comportement du système est destinée à s'assurer que la faute a été détectée. On détermine ainsi le rapport du nombre des fautes détectées sur le nombre des fautes injectées, qui peut être identifié à un taux de couverture des mécanismes de détection de fautes du système. Elle permet aussi de vérifier que le comportement du système est sûr en déterminant un taux de défaillances dangereuses, rapport du nombre des défaillances dangereuses observées sur le nombre de fautes injectées. Ces deux taux ont des valeurs relatives aux fautes injectées et n'ont pas de signification absolue.

Cette technique est aisément applicable aux technologies électriques ou électroniques peu complexes pour lesquelles les fautes susceptibles d'affecter les composants sont connues, notamment par le retour d'expérience. C'est le

cas par exemple des technologies électromécaniques, pour lesquelles il est admis que les fautes à considérer sont des collages ou des courts-circuits des contacts de sortie. Ce type de fautes peut aussi être appliqué aux semi-conducteurs discrets (transistors) ou faiblement intégrés (portes ET, NAND, …), les collages et courts-circuits étant injectés sur les broches d'entrés / sorties des composants. L'injection physique de ces fautes ne pose pas de problèmes rédhibitoires puisqu'il suffit soit de reproduire une tension égale aux valeurs hautes ou basses de la tension d'alimentation, soit de déconnecter une broche d'un composant, soit de relier deux pistes d'un circuit imprimé. Cette technique est encore appliquée à l'INRS pour la validation de logiques de commande de presses lorsque des composants électriques peu complexes sont utilisés.

Par contre, l'injection de fautes pose des problèmes (comme les techniques analytiques de type Analyse des Modes de Défaillances et de leurs Effets ou Arbre des Fautes) lorsqu'il s'agit de composants électroniques fortement intégrés (par exemple les microprocesseurs) :

- Problème du modèle des fautes à injecter. Compte tenu de la méconnaissance des modes de défaillances de ces composants, l'injection de collages et courts-circuits sur les broches est une vue très partielle de la réalité.
- Problème du moyen physique d'injection. La très forte intégration et l'encapsulation des chips rend impossible une intervention directe sur les transistors élémentaires.
- Problème de l'instant d'injection. Le caractère séquentiel des systèmes utilisant des composants électroniques de type microprocesseur ajouté à la présence d'autotests de périodicité variable suivant les applications fait que les taux de couverture et de défaillances dangereuses peuvent varier en fonction des instants d'injection des fautes et d'observation des conséquences. Ces instants ne pourront être définis que suite à une analyse temporelle du fonctionnement du système.

D'autres solutions doivent être envisagées :

- Injection de collages et courts-circuits sur tous les transistors (> à 100 000) du circuit intégré. Cette solution est irréaliste compte tenu d'une part du nombre très élevé des combinaisons potentielles de fautes de ces transistors et d'autre part de la difficulté pratique à atteindre physiquement les transistors.

- Recherche de nouveaux modèles / techniques adaptés aux technologies électroniques complexes en exploitant le retour d'expérience ou d'éventuels développements théoriques.

Cette deuxième voie a été explorée à l'INRS, via le développement d'une technique d'injection de fautes dans la mémoire programme d'un système à microprocesseur [Gérardin et al., 1991]. Plusieurs modèles de fautes ont été définis : fautes de la mémoire programme (modification de l'état d'un ou plusieurs bits d'un octet mémoire) pour reproduire une défaillance de la

mémoire ; faute du microprocesseur (modification de l'état de tous les octets mémoires identiques) pour modéliser une défaillance du décodeur d'instruction du microprocesseur, fautes des bus de données ou d'adresses, pour représenter des collages aux niveaux logiques 0 ou 1 de ces bus. Les instants d'apparition des fautes ainsi que la nature des fautes injectées (permanentes ou fugitives) sont programmables par l'analyste en fonction du système à analyser.

Cet outil a été utilisé sur différents systèmes [Gérardin et al., 1991] : Télécommandes industrielles, systèmes de détection de travailleurs isolé, automates programmables industriels, contrôleurs de brûleur de chaudière à gaz, contrôle / commande de centrales nucléaires, … Les taux de couverture obtenus sont représentatifs de l'ensemble des mécanismes de détection de fautes implantés sur ces systèmes. Ils varient de 10% pour un des automates programmables (non dédié à la sécurité) à 99,95% pour le contrôle / commande de centrales nucléaires. Pour ces mêmes systèmes, les taux de défaillances dangereuses obtenus varient de 37% à 0%. Dans le cas des automates programmables, la détection de fautes était uniquement réalisée par un chien de garde à allongements alors que le taux de couverture de 99,95 % correspond à un système redondant spécifiquement conçu pour assurer des fonctions de sécurité hautement critiques. Les résultats obtenus sont bien en rapport avec les dispositions prises à la conception pour assurer la sécurité.

Ce type d'injection de fautes est un outil intéressant pour évaluer l'efficacité des mécanismes de détection de fautes implantés sur un système. Par contre, les modèles de fautes utilisés étant limités et insuffisamment représentatifs de la réalité, les taux de défaillances dangereuses obtenus n'ont pas de valeurs absolues.

L'augmentation de la taille des mémoires utilisées pour mémoriser le programme d'un microprocesseur a été à l'origine de l'arrêt du développement de l'outil DEF.Injector associé à la méthode.

### 2.3.2. Bases de données pour l'évaluation de la probabilité de défaillances dangereuses d'un système électronique

Les calculs de fiabilité et de probabilité de défaillance dangereuse sont effectués à l'aide de bases de données contenant les taux de défaillances des composants. Ces bases peuvent être créées par un constructeur et sont adaptées à son propre domaine d'application. D'autres bases, plus universelles, existent et servent à de nombreux industriels. Un point bibliographique [Brandt & Charpentier, 1999] a été fait à l'INRS sur les bases de données les plus diffusées [Bot et al., 1997] :

- MIL-HDBK-217 du Department of Defense (DoD, Etats-Unis),
- Bellcore de Bell Communication Research (Etats-Unis),
- HRD5 de British-Telecom,
- RDF93 (3) du Centre National d'Études des Télécommunications (CNET, France).

Les bases RDF 93 et MIL-HDBK-217 ont été plus particulièrement analysées.

Présentation des bases de données de composants électroniques

Les bases de données de composants électroniques rassemblent des modèles de taux de défaillance empiriques développés à partir de données de défaillances de retours d'exploitation. Par la nature même du retour d'expérience, celles-ci sont souvent limitées à un environnement d'exploitation et données en terme de nombre de défaillances et de causes de défaillances.

Les modèles sont valides pour des composants électroniques qui suivent la distribution exponentielle. Deux approches sont possibles pour élaborer un modèle :

- "Parts Count", utilisé au début de la phase de développement lorsque les informations connues sur la conception sont faibles, de sorte que les contraintes électrique et thermique ne sont pas disponibles.
- "Parts Stress", utilisé plus tard dans le cycle de vie, lorsque les composants à utiliser sont déterminés, qu'il y a suffisamment d'informations disponibles sur la conception et que les contraintes électriques et thermiques sont définies. C'est l'unique modèle proposé dans RDF93.

Obtention des données de fiabilité

Les données de fiabilité constituant les bases génériques citées précédemment sont classiquement obtenues en combinant les moyens suivants :

- Collecte de données à partir du retour d'expérience. Elle est fondée sur des situations de panne à partir desquelles, pour chaque famille de composants, on cherche à déterminer les taux de défaillance dans un fonctionnement continu.
- Fiabilité prévisionnelle. Cette démarche est illustrée par le CNET [Monfort, 1998]. Lorsqu'un produit nouveau est créé ou amélioré, ou lorsque ses conditions d'exploitation changent, il n'a pas encore été mis en exploitation et aucune défaillance en condition réelle de fonctionnement n'a pu être observée à ce stade. La fiabilité évaluée est appelée fiabilité prévisionnelle. Différentes sources de données sont identifiées (dires d'experts, essais, données fournisseurs…) puis évaluées par la qualité et la quantité de l'information qu'elles apportent. Plusieurs familles de traitement de données sont envisagées (théorie des probabilités ou théorie des possibilités) pour fusionner les données, à la base des calculs de fiabilité.

Un utilisateur peut aussi développer sa propre base de données de fiabilité. Des référence normatives sont disponibles pour en faciliter le développement : CEI 61709 "Composants électroniques - Fiabilité - Conditions de référence pour les taux de défaillance et modèles d'influence des contraintes pour la conversion", CEI 60319 pour la présentation des données de fiabilité à partir d'essais en

laboratoire et CEI 60300-3-2 pour l'acquisition des données de fiabilité à partir de résultats d'exploitation.

Description sommaire de la base RDF93 du CNET [CNET, 1993]

Les données de fiabilité sont principalement issues du retour d'expérience :

- résultats de fiabilité de matériels en exploitation, notamment des matériels de télécommunication (résultats fournis par les constructeurs Alcatel-CIT, SAT, CROUZET et par France Télécom), des matériels ferroviaires (GEC-ALSTHOM) et des matériels informatiques (BULL),
- résultats d'essais de composants (constructeurs, fabricants, CNET),
- expérience acquise par un certain nombre d'experts à la suite d'analyses de défaillances et d'analyses de construction de composants (constructeurs, fabricants, CNET).

Cette base a été constituée en retenant les hypothèses suivantes :

- Les défaillances considérées sont des défaillances intrinsèques, c'est-à-dire propres à la nature de chaque composant.
- Les taux de défaillance sont supposés constants.
- La période de défaillances précoces n'a été prise en compte que pour quelques familles de composants. Cette hypothèse très réaliste est confirmée par le retour d'expérience provenant de l'exploitation d'équipements dont la conception a été soignée, dont les composants ont été bien choisis (compatibilité avec l'utilisation) et qui ont une production correctement maîtrisée.
- Pour la très grande majorité de composants, la période d'usure - au cours de laquelle les défaillances prennent un caractère systématique – n'a pas été considérée car très loin des périodes d'utilisation (celles-ci étant de 3 à 20 ans). Pour les autres, la compatibilité de la durée de vie donnée par le recueil avec l'utilisation doit être vérifiée.

Remarques :

- Le taux de défaillance d'un composant dépend d'un certain nombre de contraintes de fonctionnement et d'environnement. C'est pourquoi, pour chaque famille de composants le recueil donne une valeur de base d'un taux de défaillance (en général une valeur correspondant à la température interne la plus courante, prise comme référence) pondérée par plusieurs facteurs d'influence : température, contraintes particulières d'utilisation, tension appliquée, technologie, environnement, qualité.
- L'expérience montre que les composants s'améliorent progressivement grâce au retour d'expérience et aux améliorations

des moyens de production. Selon une idée déjà en pratique dans le recueil britannique HRD, le recueil intègre l'année de production des circuits intégrés.

Description sommaire de la base MIL-HDBK-217

Cette base fait partie des premiers travaux de collecte de données de fiabilité des composants électroniques [MIL-HDBK, 1991]. Les taux de défaillance des composants sont généralement déterminés à partir d'un taux de défaillance de base, fonction de la complexité et du type de composant. Les taux de défaillance de base sont multipliés par des facteurs pour déterminer le taux de défaillance de composants spécifiques.

La MIL-HDBK-217 concerne en premier lieu le développement du matériel militaire. Elle est destinée à fournir une base de données destinée à la prédiction de fiabilité lorsqu'il n'existe pas d'expérience solide de fiabilité pour un équipement particulier, pour [Morris, 1990] :

- Fournir une première évaluation de la faisabilité d'un équipement,
- Comparer des conceptions concurrentes,
- Identifier des problèmes potentiels de fiabilité,
- Constituer une donnée d'entrée pour d'autres tâches de fiabilité/maintenabilité.

Des critiques [Stadermann et al., 1995] ont été formulées à son sujet. Elles reprochent à son modèle de prédiction d'être inapproprié : l'approche est jugée trop pessimiste de telle sorte que le produit évalué est en théorie moins fiable qu'il ne l'est en réalité en phase d'exploitation.

Sur le fond, elles soulignent que la méthodologie de prédiction du taux de défaillance du composant ne donne pas au concepteur, ni au constructeur, de contrôle des causes réelles de défaillance car les relations causes à effets ayant un impact sur la fiabilité ne sont pas identifiées.

Conclusion

Les bases de données de composants électroniques constituent le point de départ des calculs de fiabilité. Les moyens utilisés pour leur construction (en particulier le retour d'expérience) font que leur contenu est relatif au domaine d'application et aux conditions environnementales. De ce fait, elles ne reflètent pas exactement la réalité de l'exploitation, mais elles fournissent un moyen d'avoir quelques repères et donnent une échelle pour la comparaison de produits.

La sécurité des machines n'ayant à ce jour jamais utilisé explicitement l'approche probabiliste pour évaluer le niveau de sécurité d'un système de sécurité, il n'existe pas de base de données générique adaptées à ce domaine spécifique. Plusieurs solutions peuvent être envisagées pour pallier cette absence :

- Utiliser la base de données établie par un constructeur pour les équipements considérés (Automates Programmables Industriels) utilisés dans des conditions d'exploitation différentes du domaine manufacturier (en l'occurrence le domaine du process). La société SIEMENS dispose d'une telle base mais ne l'a pas diffusé.

- Constituer une base de données propre au secteur de la sécurité des machines. Compte tenu des difficultés inhérentes à la constitution d'une base (temps, coût, maintenance et évolution), cette solution est difficile à envisager. Deux voies sont possibles : chaque société constitue sa propre base de données à partir de ses retours d'expérience, ou les résultats sont mis en commun. La première solution nécessite un nombre d'installations suffisamment élevé pour être représentatif, alors que la seconde impose une uniformisation de la collecte pour ne pas inclure des données "incompatibles".

- Utiliser une base de données existante, en relativisant les résultats obtenus. Si une telle solution était retenue, les résultats quantitatifs obtenus ne pourraient être que relatifs.

Le choix entre ces différentes solutions n'a pas encore été fait. Il ne pourra résulter que d'un consensus (certainement normatif) entre les partenaires impliqués dans la sécurité des machines. Le contexte devrait imposer un critère de choix d'ordre économique.

Les évaluations quantitatives effectuées dans la suite de ce document le seront à partir de données fournies par les constructeurs d'automates.

## *2.4. HARMONISATION DES MÉTHODES DE VALIDATION DES SYSTÈMES ÉLECTRONIQUES DE SÉCURITÉ*

Même si d'autres référentiels sont en émergence, la norme EN 954-1 est la référence normative couramment utilisée en sécurité des machines. Ce référentiel donne des objectifs de résultats, laissant libres les concepteurs et les organismes de validation du choix des méthodes utilisées pour satisfaire les prescriptions. Ces méthodes sont connues et font l'objet d'un consensus pour les technologies électromécaniques ou électroniques faiblement intégrées. Par contre, les paragraphes précédents ont montré sur quelques exemples que les moyens pour construire puis valider le niveau de sécurité d'un système à composants électroniques complexes sont nombreux. Conscient de ce problème, la communauté européenne a financé le projet de recherche STSARCES[7] pour harmoniser les méthodes de validation des systèmes électroniques à composants complexes utilisés en sécurité des machines [STSARCES, 2000], [Charpentier et al., 2002]. Le projet, mené suite à un appel dédié émis dans le cadre du programme “ Normes, mesures et essais ”, portait sur un programme de recherche en appui à la normalisation européenne. Ce projet, coordonné par l'INERIS, a mobilisé 11 partenaires de 6 pays de l’UE : organismes de recherche (dont l'INRS), organismes notifiés et fabricants de composants de sécurité.

[7] *STSARCES 'STandards for SAfety Related Complex Electronic Systems' (DG XII / Contract SMT 4CT97-2191)*

Il s'est attaché à étudier la validation des parties des systèmes de commande relatives à la sécurité utilisant des systèmes électroniques complexes, pour harmoniser les pratiques des différents laboratoires européens concernés par ce sujet. La question a été abordée par une série de sujets connexes, incluant à la fois les aspects matériel et logiciel, dans la perspective initiale de contribuer à l'élaboration du prEN 954 partie 2 " validation ". Le cadre général retenu est basé sur le concept de cycle de vie de sécurité, concept relativement nouveau dans le secteur des machines mais nécessaire pour couvrir globalement les problèmes posés.

Les aspects logiciels ont été traités au niveau des spécifications mais aussi en définissant des prescriptions qualité et sûreté de fonctionnement dans le but d'éviter les fautes lors du développement du logiciel système (cf. paragraphe 2.2 "Évitement des fautes").

L'approche markovienne, appliquée à des architectures dédiées et validées conformément au concept de catégorie de l'EN 954, a été utilisée pour proposer des relations plausibles et compréhensibles, mais non absolues, entre les catégories de l'EN 954 et les niveaux d'intégrité de sécurité (SIL : Safety Integrity Level) de la CEI 61508 (cf tableau 2-6). Ces relations sont utiles pour la conception et le développement des circuits de commande des machines utilisant à la fois des composants basés sur le concept de catégorie (composants mécaniques, hydrauliques, pneumatiques et électromécaniques) et des composants électroniques complexes, mieux caractérisés par le concept de SIL.

Enfin, une analyse des divergences et complémentarités de la norme EN 954 par rapport à la norme CEI 61508 a été menée, dans le but notamment de faire coexister ces deux référentiels.

| SIL | System Architecture | Mean Time to dangerous Failure $MTTF_d$ (years) | *CCF* $\beta$ (%) | Diagnostic Coverage (each Channel) (%) | Cat. |
|---|---|---|---|---|---|
| | | In/Processing/Out | | In/Processing/Out | |
| - | Single PE, Single I/O | 15/15/30 | - | 0/0/0 | **B** |
| | Single PE, Single I, Ext. WD(u/t) | 15/15/30 | - | 0/60/0 | **B** |
| | Dual PE, Dual I/O, 1oo2 | 15/15/30 | 5 | 0/0/0 | **?** |
| 1 | Single PE, Single I, Ext. WD(u/t) | 15/15/30 | - | 100/60/100 | **2** |
| | Single PE, Single I, Ext. WD(u/t) | 7.5/15/10 | - | 100/60/100 | **2** |
| | Dual PE, IPC, Dual I/O, 1oo2 | 15/15/30 | 5 | 100/60/100 | **3** |
| | Dual PE, IPC, Dual I/O, 1oo2 | 15/15/30 | 10 | 100/90/100 | **3** |
| | Dual PE, IPC, Dual I/O, 1oo2 | 45/15/60 | 10 | 100/90/100 | **3** |
| | Triple PE, IPC, Triple I/O, 1oo3 | 15/15/30 | 5 | 100/60/100 | **3** |
| | Triple PE, IPC, Triple I/O, 1oo3 | 15/15/30 | 10 | 100/90/100 | **4** |
| 2 | Single PE, Single I, Ext. WD(t) | 15/15/30 | - | 100/90*/100 | **2** |
| | Dual PE, IPC, Dual I/O, 1oo2 | 15/15/30 | 1 | 100/90/100 | **3** |
| | Dual PE, IPC, Dual I/O, 1oo2 | 30/30/60 | 5 | 100/90/100 | **3** |
| | Dual PE, IPC, Dual I/O, 1oo2 | 7.5/15/10 | 1 | 100/99/100 | **4** |
| | Mixed Dual Processing, Dual O, 1oo2 | ∞/(15/100)/(15/100) | - | 0/(30/100)/(100/100) | **3** |
| | Triple PE, IPC, Triple I/O, 1oo3 | 15/15/30 | 1 | 100/60/100 | **3** |
| | Triple PE, IPC, Triple I/O, 1oo3 | 100/100/200 | 10 | 100/90/100 | **4** |
| 3 | Single PE, Single I, Ext. WD(t) | 30/30/60 | - | 100/99*/100 | **2** |
| | Dual PE, IPC, Dual I/O, 1oo2 | 45/45/90 | 1 | 100/99/100 | **4** |
| | Triple PE, IPC, Triple I/O, 1oo3 | 100/100/200 | 1 | 100/90/100 | **4** |

**Conditions for single channel systems:**

All test rates: 1/(15 min)
Demand rate: 1/(24 h)
Repair rate: 1/(8h)
Mission time (life time): 10 years

$MTTF_d$ of watchdog: 100 years
$MTTF_d$ of switch-off path for watchdog: equal to normal switch-off path
WD(u/t): Watchdog and pertinent switch-off path untested or tested
WD(t): Watchdog and pertinent switch-off path tested

(* not achievable by simple watchdog)

**Conditions for dual or triple channel systems:**

All test rates: 1/(24h)
Demand rate: 10/h
Repair rate: 1/(8h)
Mission time (life time): 10 years

$MTTF_d$ of output sensor of mixed system: 15 years (output sensor not tested)

IPC: Inter-processor communication

**Tableau 2-6 : Liens Catégorie / SIL**

(Source BIA, Rapport STSARCES)

Méthodologie de validation des systèmes électroniques de sécurité à composants complexes

Les travaux ont mis en évidence que, pour traiter les problèmes de sécurité de manière correcte, il est désormais nécessaire d'avoir une vue d'ensemble des différentes étapes qui jalonnent la " vie " d'une application, de la définition de ses limites jusqu'à la validation de la sécurité (les questions de maintenance et de déclassement final sont

souvent difficiles à traiter dans le cas des machines). Ces étapes sont généralement regroupées dans un "cycle de vie de sécurité" couvrant l'application dans son ensemble (cf. CEI 61508-1). Il est défini comme l'ensemble des activités nécessaires à la mise en œuvre des différents moyens destinés à réduire le risque : systèmes électroniques de sécurité, systèmes de sécurité basés sur d'autres technologies, dispositifs externes de réduction du risque le cas échéant.

La validation suivra les étapes suivantes (les étapes en italiques ne sont pas spécifiques à la validation, mais leurs produits doivent être validés) :

- ✓ Définir les exigences de sécurité pour les différents moyens de réduction du risque. Mettre à jour la définition du plan de sécurité en fonction des besoins apparus au cours du développement des CES[8] relatifs à la sécurité (SRCES[9]).
- ✓ Déterminer les prescriptions relatives aux SRCES, y compris les prescriptions concernant l'intégrité de la sécurité, pour chacune des fonctions de sécurité. Définir des prescriptions pour le logiciel.
- ✓ Démarrer la définition du plan de validation des SRCES.
- ✓ Établir l'architecture du système logique des SRCES, des capteurs et des éléments finaux.
- ✓ *Développer un modèle pour l'architecture matérielle dédiée aux systèmes de sécurité. Développer ce modèle en examinant séparément chaque fonction de sécurité et déterminer le sous-système (composant) qui sera utilisé pour assurer cette fonction. Une analyse déterministe et une analyse probabiliste sont complémentaires.*

  *Pour l'approche déterministe, deux méthodes analytiques sont souvent employées :*

  - *La méthode de l'Arbre des Fautes : cette méthode déductive part d'une défaillance dangereuse du système, déterminée par exemple par une analyse de risque, et recherche les combinaisons d'évènements qui ont pu conduire à cette défaillance. Elle met en évidence les défaillances aléatoires, les défaillances systématiques et les défaillances de mode commun. Toutes les branches logiques de l'arbre doivent être développées jusqu'aux évènements de base. En pratique, l'arbre est développé de manière à analyser l'effet des défaillances des entrées, de la logique et des sorties.*
  - *L'Analyse des Modes de Défaillance et de leurs Effets (AMDE) : il s'agit d'une méthode inductive qui part des défaillances des fonctions ou des composants du système à analyser afin de déterminer les défaillances dangereuses susceptibles d'affecter le système. Elle peut aussi mettre en évidence les défaillances de mode commun qui affectent le logiciel ou le matériel* [Thireau, 1986], [Letrung et al., 1995].

  *Pour l'approche probabiliste, on utilise généralement les graphes de Markov et les arbres de fautes. Ces techniques sont retenues du fait de leur capacité à traiter un grand nombre des caractéristiques techniques des dispositifs de sécurité actuels. Les graphes de Markov, notamment, permettent une modélisation relativement commode des évènements périodiques tels que les tests en ligne.*

[8] CES : Complex Electronic System – Système Électronique Complexe
[9] SRECS : Safety-Related Complex Electronic System – Système Électronique Complexe Relatif à la Sécurité

- ✓ Établir les paramètres du système pour chacun des composants utilisés dans les systèmes électroniques complexes relatifs à la sécurité et créer un modèle de fiabilité pour chacune des fonctions de sécurité que le SRCES doit assurer. Pour chacun des composants, déterminer :
  - le temps moyen de remise en service ;
  - la couverture du diagnostic ;
  - la probabilité de défaillance.
- ✓ Avec le fournisseur / le développeur du logiciel vérifier, en itérant si nécessaire, l'architecture matérielle et logicielle ainsi que les conséquences sur la sécurité des compromis entre matériel et logiciel.
- ✓ *Concevoir le SRCES. Choisir des mesures et des techniques pour contrôler les défaillances systématiques du matériel, les défaillances causées par l'influence de l'environnement et les défaillances opérationnelles.*
- ✓ Valider le logiciel. La méthodologie pour la validation d'un logiciel comporte six niveaux (cf. paragraphe 2.2.2) :
  - S'assurer que les spécifications relatives au logiciel sont conformes aux besoins de l'utilisateur et aux exigences de sécurité. On s'assurera que toutes les informations spécifiant les différentes fonctions sont complètes, précises, explicites, cohérentes et correctes.
  - Passer des spécifications au code exécutable, via la conception du logiciel ; le code exécutable doit être conforme aux spécifications du logiciel.
  - Vérifier le comportement du logiciel en exécutant le code.
  - S'assurer que le code final est conforme aux besoins de l'utilisateur. Pour des raisons inhérentes à la nature des besoins de l'utilisateur, cette conformité est difficile à démontrer.
  - Passer des spécifications au comportement du logiciel, en vérifiant que le comportement du logiciel est conforme à ce qui est décrit dans les spécifications.
  - Valider le logiciel.
- ✓ *Intégrer le logiciel vérifié au matériel cible et, parallèlement, développer les procédures que les utilisateurs et le personnel de maintenance devront suivre pour faire fonctionner le système.*
- ✓ En collaboration avec le développeur du logiciel, valider le SRCES. La validation de la sécurité vise à vérifier que toutes les parties du système relatives à la sécurité satisfont aux spécifications de sécurité. Elle est effectuée conformément au plan de validation. Elle permet de conclure que le système relatif à la sécurité satisfait aux exigences de sécurité puisque celles-ci sont toutes validées. Lorsque des écarts se produisent entre les résultats attendus et les résultats réels, il faut décider s'il est nécessaire de modifier le système ou les spécifications et les champs d'application correspondants. Il faut en outre décider soit de continuer et de procéder ultérieurement aux modifications, soit de procéder immédiatement aux modifications et de redémarrer le processus de validation à une phase antérieure.

- ✓ Enfin, le SRCES doit satisfaire également à des prescriptions génériques :
  - Sécurité électrique.
  - Compatibilité électromagnétique : la directive CEM européenne fixe des prescriptions concernant la susceptibilité et l'émission.
  - Compatibilité environnementale.
  - Contrainte climatique et contrainte mécanique.
  - Gestion de la qualité de la production, des essais et du traitement des révisions. Ceci est particulièrement important pour les systèmes à base de logiciels.

Considérations relatives aux systèmes de commande à composants complexes utilisés en sécurité des machines

Les travaux de recherche entrepris à l'occasion du projet STSARCES ont mis en évidence certaines limites inhérentes à la nature des problèmes rencontrés :

- ✓ La CEI 61508 alloue un niveau d'intégrité de sécurité (SIL) aux fonctions de sécurité alors que la EN 954-1 utilise le concept de performance de sécurité pour répartir les systèmes en 5 catégories. Il n'a pas été possible d'établir de relations univoques entre les SIL et les catégories. Ceci est dû en premier lieu aux définitions des catégories dans l'EN 954-1, qui ne fixent aucune prescription quantifiable concernant les probabilités de défaillances dangereuses des fonctions de sécurité.
- ✓ La norme CEI 61508 couvre toutes les phases de la vie de l'équipement, depuis la conception jusqu'au déclassement. Dans le secteur des machines, il est très rare que la responsabilité d'un intervenant soit engagée sur tout le cycle de vie d'un équipement, celui-ci n'en fournissant qu'un élément. On considère donc qu'il est nécessaire de délimiter l'étendue des responsabilités. C'est le cas par exemple des fabricants de composants de sécurité destinés à être utilisés dans une grande variété d'applications : une analyse complète des dangers et des risques, de même que l'identification des fonctions de sécurité adéquates pour toutes les applications, n'est généralement pas réalisable par le fabricant à un stade précoce du cycle de vie de sécurité. Le fabricant doit alors impérativement fournir des informations suffisantes et adéquates (incluant le SIL) pour que les utilisateurs puissent prendre correctement en considération les performances des équipements dans l'application finale.
- ✓ Il existe une grande différence entre les pratiques de développement des logiciels et les travaux théoriques traitant du sujet. Il est difficile d'élaborer une méthodologie de sûreté de fonctionnement pour les PME. Des contraintes organisationnelles et le manque de culture de sûreté, compliquent grandement l'élaboration d'une méthodologie. Il n'existe pas d'outils logiciels particuliers pour spécifier la sécurité d'applications de taille réduite.
- ✓ Le degré de complexité des composants électroniques actuels est si élevé qu'il est impossible d'en prévoir tous les modes de défaillance. Cette méconnaissance est généralement palliée par le recours à la redondance (diversitaire ou non) et / ou à la surveillance. Ceci signifie que l'architecture du système de commande a une importance majeure pour minimiser les risques causés par les systèmes à composants complexes.

- ✓ La complexité croissante des nouveaux systèmes (fonctions intégrées, vitesses de traitement, miniaturisation des composants et des techniques d'assemblage, etc…) complique largement l'exécution des tests réalisés à un stade tardif de la conception (au stade de la validation par exemple). On doit alors appliquer des techniques de conception qui facilitent les tests ultérieurs des circuits et des programmes (dites techniques de testabilité) et à passer de tests directs physiques à la simulation.
- ✓ En matière d'injection de défauts, l'injection physique sur les broches (cf. paragraphe "Injection de fautes") et l'injection par le logiciel s'avèrent être des techniques intéressantes dans le cas des prototypes de systèmes électroniques programmables. Toutes deux injectent un sous-ensemble de tous les défauts potentiels, de sorte que l'analyste devra choisir la technique et prévoir le test en fonction de l'ensemble des fautes à émuler. Toutefois, il est difficile de dire dans quelle mesure une technique couvre l'autre parce que la capacité de ces techniques à émuler les fautes dépend largement de la nature du système.
- ✓ À ce jour, les SRECS destinés aux machines n'ont le plus souvent pas été conçus sur la base de la norme CEI 61508 ; il est par conséquent probable qu'on ne dispose pas de la documentation requise pour vérifier les différentes étapes du cycle de vie de sécurité. La documentation, sous une forme convenable pour l'évaluation, ne sera disponible que lorsque des normes telle que la CEI 62061 (basée sur la norme CEI 61508) seront disponibles. D'ici là, il sera difficile de réaliser des évaluations, notamment des analyses quantitatives, des SRECS équipant les machines. Le projet vise à encourager les fabricants à combler ce fossé dans un futur proche et à éditer la documentation nécessaire au fur et à mesure du développement de nouveaux produits de sécurité.

Remarque : les considérations concernant la CEI 61508 sont aussi applicables à la CEI 62061.

# 3. SDF DES SYSTÈMES ÉLECTRONIQUES PROGRAMMABLES ET DÉFAILLANCES DÉPENDANTES

Nous montrerons dans le premier paragraphe de ce chapitre que les défaillances dépendantes sont inhérentes à la redondance, technique constructive de tolérance permettant à un système de remplir la mission pour laquelle il a été conçu, en dépit des fautes [Charpentier, 2000], [Dhillon et Anude, 1994].

Les paragraphes suivants décriront les moyens et techniques de sûreté de fonctionnement destinés à s'affranchir de ces défaillances. L'évitement sera essentiellement abordé sous les aspects diversité et check-listes. La prévision sera quant à elle abordée en considérant les défaillances matérielles aléatoires.

Remarque : Par son caractère constructif, la diversité pourrait être envisagée comme une technique de tolérance aux fautes. Nous avons choisi de la présenter comme un moyen d'évitement des défaillances de mode commun introduites suite à la mise en œuvre d'une structure redondante.

## 3.1. INTRODUCTION

### 3.1.1. Définitions

Définir les défaillances de mode commun permet d'éviter toute confusion ou incompréhension relative aux problèmes posés et à leurs solutions éventuelles. Beaucoup utilisent indifféremment les termes fautes de mode commun, défaillances de mode commun, défaillances de cause commune lorsqu'il s'agit de nommer des phénomènes communs affectant plusieurs entités distinctes. Ce paragraphe est destiné à clarifier la notion de défaillance de mode commun[10] en expliquant l'enchaînement des phénomènes mis en jeu pour conduire à ces défaillances.

Pour plus de précision, l'enchaînement faute / erreur / défaillance proposé par le LAAS sera utilisé pour décrire les phénomènes.

***Mode de défaillance / Mécanisme de défaillance***

Le mode de défaillance ne doit pas être confondu avec le mécanisme de défaillance, qui est le processus physique (ex : corrosion) ayant entraîné la défaillance [Pecht & Ramappan, 1992]. Aux différences de forme près, [Pecht & Ramappan, 1992], [MDCI, 1994] et [Laprie et al., 1995] définissent le mode de défaillance comme étant la manifestation observable de la défaillance (ex : court-circuit, collage de la sortie d'un transistor, coupure d'une piste).

***Défaillance de Mode Commun***

Cette clarification étant faite, la définition des défaillances de mode commun[11] donnée par [Taylor, 1975] synthétise bien les différentes formulations trouvées dans la bibliographie [Villemeur, 1988], [Laprie et al., 1995], [CEI61508, 2000]. Les défaillances de mode commun sont des défaillances simultanées, affectant de multiples entités et dépendantes d'une seule cause initiale. Cette définition a pour avantage de s'appliquer à tout type d'architecture, sans que la nature de la redondance n'intervienne.

La simultanéité est un point important à noter lorsqu'on évoque des défaillances de mode commun. [Edwards & Watson, 1979] précisent que les défaillances peuvent avoir lieu à des instants identiques ou distincts mais, qu'à un moment donné, les états défaillants coïncident. La taille de l'intervalle de temps est primordiale. Elle discrimine les défaillances corrélées des défaillances multiples indépendantes [Villemeur, 1988].

Remarques :

- Lyu introduit pour le logiciel la notion de défaillance coïncidente [LYU, 1995]. Les défaillances coïncidentes affectent alors deux (ou plus) composants logiciels, fonctionnellement équivalents, soumis à une même entrée.

[10] CMF en anglais: Common Mode Failure

[11] On trouve aussi dans la littérature les termes de défaillances corrélées ou défaillances couplées pour désigner les phénomènes induits par une même cause.

- La norme EN 292 [EN292, 1997] distingue les défaillances de cause commune des défaillances de mode commun. Les premières sont des défaillances d'entités différentes résultant d'un même événement, qui n'ont pas de liens entre elles. Les secondes caractérisent des défaillances d'entités suivant le même mode. Humphrey quant à lui définit mathématiquement les défaillances de cause commune comme les défaillances d'un ensemble d'événements, dont la probabilité ne peut être exprimée comme un simple produit des probabilités inconditionnelles de défaillance des événements individuels [Humphrey, 1989].

***Mécanisme d'apparition d'une défaillance de mode commun***

La définition donnée par le LAAS montre bien l'enchaînement des phénomènes conduisant à la défaillance de mode commun. Au départ se trouve une cause commune, capable de générer des fautes corrélées. Ces fautes sont à distinguer des fautes indépendantes, qui sont attribuées à des causes différentes. Ces fautes corrélées vont être à l'origine d'erreurs similaires qui, lorsqu'elles seront activées ou révélées, provoqueront une défaillance de mode commun. On a donc le schéma suivant :

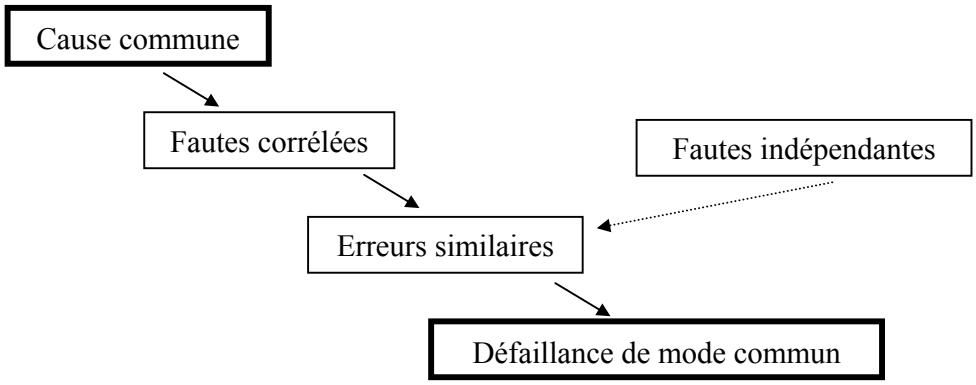



**Figure 3-1 : Mécanisme d'apparition d'une défaillance de mode commun**

Le chemin classique de 'production' des défaillances de mode commun est celui qui part d'une cause commune. Dans certains cas, il se peut que des fautes indépendantes conduisent à des erreurs similaires [Laprie et al., 1995]. Du fait de leur faible probabilité d'apparition, ces fautes seront négligées dans la suite de ce document.

### 3.1.2. Identification des défaillances de mode commun

De façon très générale, [Taylor, 1975] indique que la cause d'une défaillance de mode commun peut être une propriété commune, un processus commun, un environnement commun, ou un événement externe commun. Ces influences communes [EWICS, 1988] peuvent affecter le système avant sa mise en

service opérationnelle : conception, fabrication, installation. Dans ce cas, les manifestations ne sont pas immédiates et le système est 'potentiellement' défaillant. Elles peuvent aussi affecter le système au cours de son fonctionnement, par exemple dans le cas d'une conception erronée ou d'agressions de l'environnement.

En pratique, les défaillances de mode commun n'apparaissent que s'il existe :

**UN INITIATEUR SOURCE**

C'est la cause commune capable de provoquer des fautes corrélées, c'est à dire un ou plusieurs événements internes ou externes provoquant de multiples défaillances de systèmes élémentaires. Ce premier point est confirmé par [Buchner, 1994] qui évoque un facteur d'activation pour déclencher des défaillances de mode commun ainsi que l'accumulation de fautes systématiques au sein du système . Ce dernier point peut d'ailleurs être exclu des analyses si des contrôles suffisamment fréquents sont implantés pour détecter toute accumulation. En reprenant la classification proposée par le LAAS (cf. 1.2.5), il est possible de distinguer plusieurs types d'initiateurs :

**Dégradation des composants électroniques**

Les fautes provoquées sont des fautes physiques accidentelles internes, permanentes ou temporaires. Il est cependant difficile de distinguer les fautes apparaissant en cours de développement (composants mal dimensionnés) et en opérationnel (défaillances aléatoires de plusieurs composants).

**Perturbations externes**

L'intégration croissante des composants électroniques fait qu'ils sont très sensibles aux fautes physiques accidentelles opérationnelles externes, comme par exemple les perturbations d'origine électromagnétique.

**Fautes humaines introduites dans les différentes étapes du cycle de vie du produit**

Ce sont des fautes de conception accidentelles internes liées au développement des systèmes. Du fait des fautes systématiques qu'elles génèrent, les fautes humaines sont très certainement la première source de défaillances de mode commun. Les fautes logicielles constituent un exemple typique de fautes humaines.

**ET**

**UNE CORRÉLATION ENTRE SYSTÈMES ÉLÉMENTAIRES OU COMPOSANTS**

Il ne peut y avoir de défaillance de mode commun que si une corrélation existe entre plusieurs 'composants' ou 'sous-systèmes' matériel ou logiciel soumis à un initiateur. Elle peut être liée à des équipements communs, à des interactions physiques ou à des actions humaines. [Edwards & Watson, 1979] distingue deux types de corrélation :

Type 1 :

Elle correspond à la défaillance d'un sous-système élémentaire ou d'un composant commun à tous les canaux d'un système redondant.

Dans ce cas, la corrélation est au maximum puisqu'il s'agit d'une partie commune, dont la défaillance provoquera l'injection d'une défaillance simultanée dans les deux voies. Ce type de corrélation entre bien dans le cadre de la définition des défaillances de mode commun.

Type 2 :

Les corrélations peuvent exister entre systèmes élémentaires ou entre composants, la différence étant dans le niveau de description choisi [Villemeur, 1988]. Elles correspondent à la coïncidence des défaillances de deux ou plus de deux systèmes élémentaires ou composants identiques de canaux séparés d'un système redondant, due à une cause commune ;

Avec ces critères d'identification, la Figure 3-1 devient pour le Type 1 :

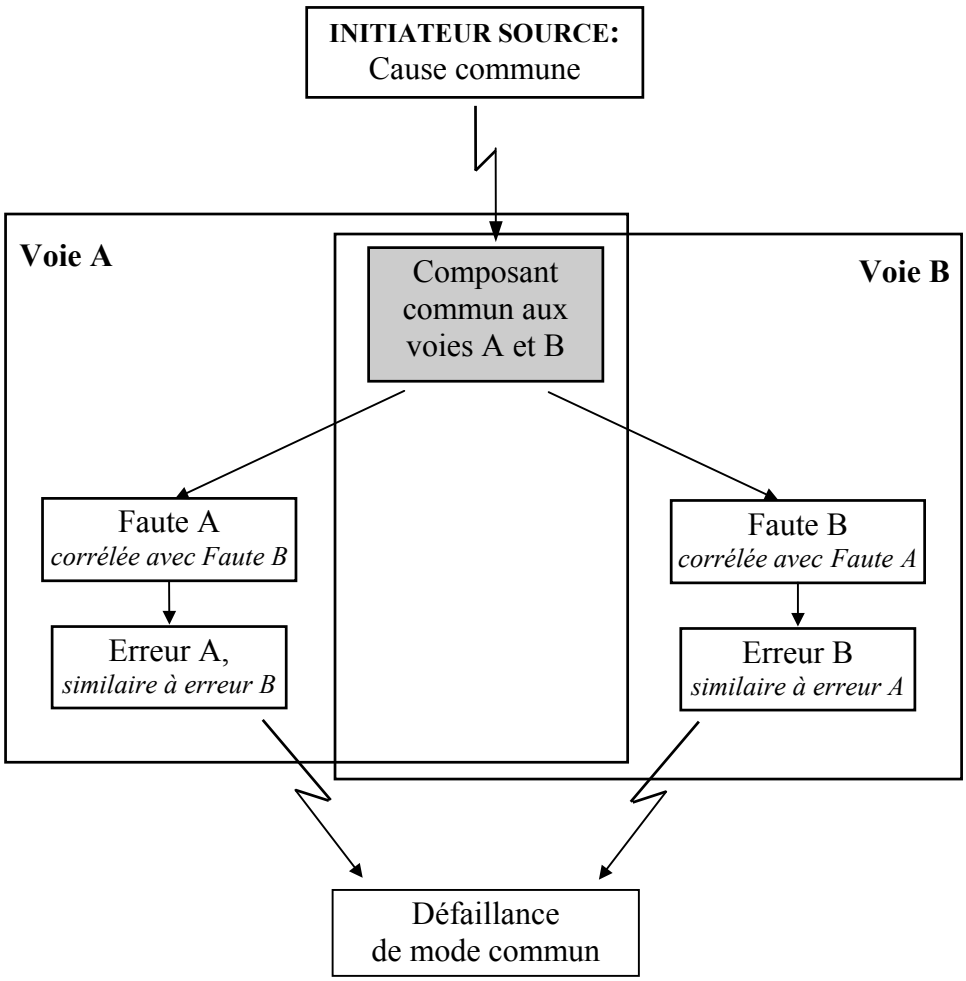



**Figure 3-2 : Corrélation de type 1**

Pour le type 2, elle devient :

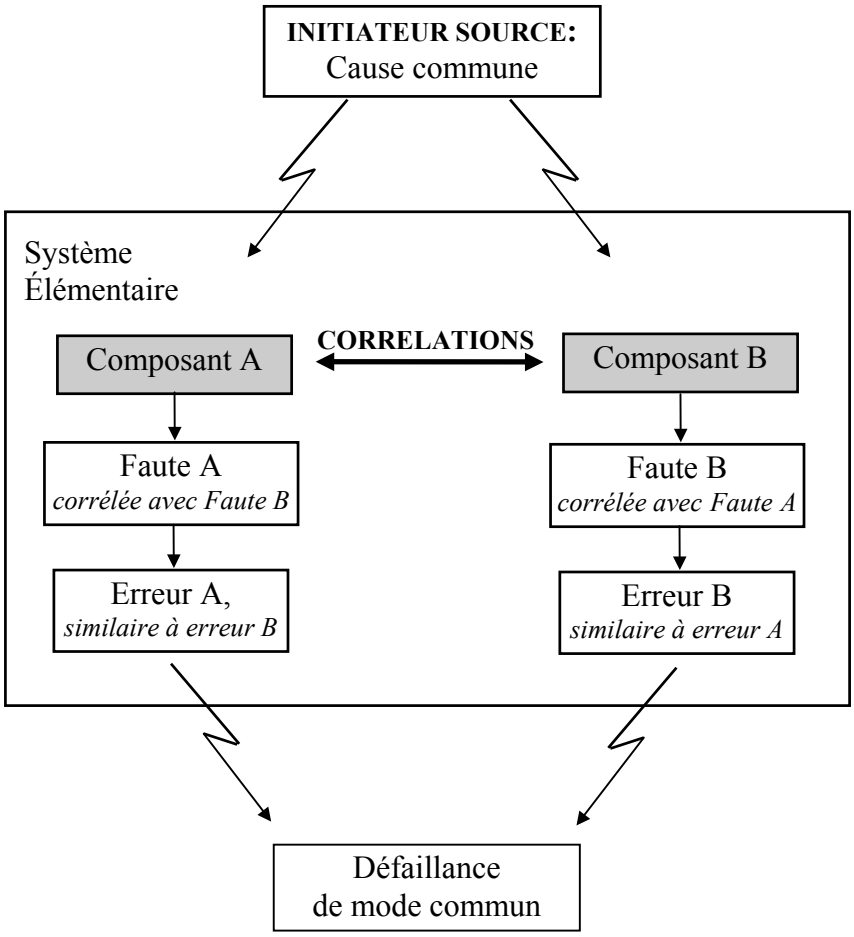



**Figure 3-3 : Corrélation de type 2**

***Exemple de fautes corrélées : fautes de compilation***

On considère deux structures redondantes homogènes - programme source et matériel identiques -, la première étant développée à l'aide d'un seul compilateur C1, la seconde avec deux compilateurs différents C1 et C2. L'ensemble des fautes corrélées générées par la compilation dans les programmes exécutables de chaque microprocesseur peut être représenté par la figure suivante :

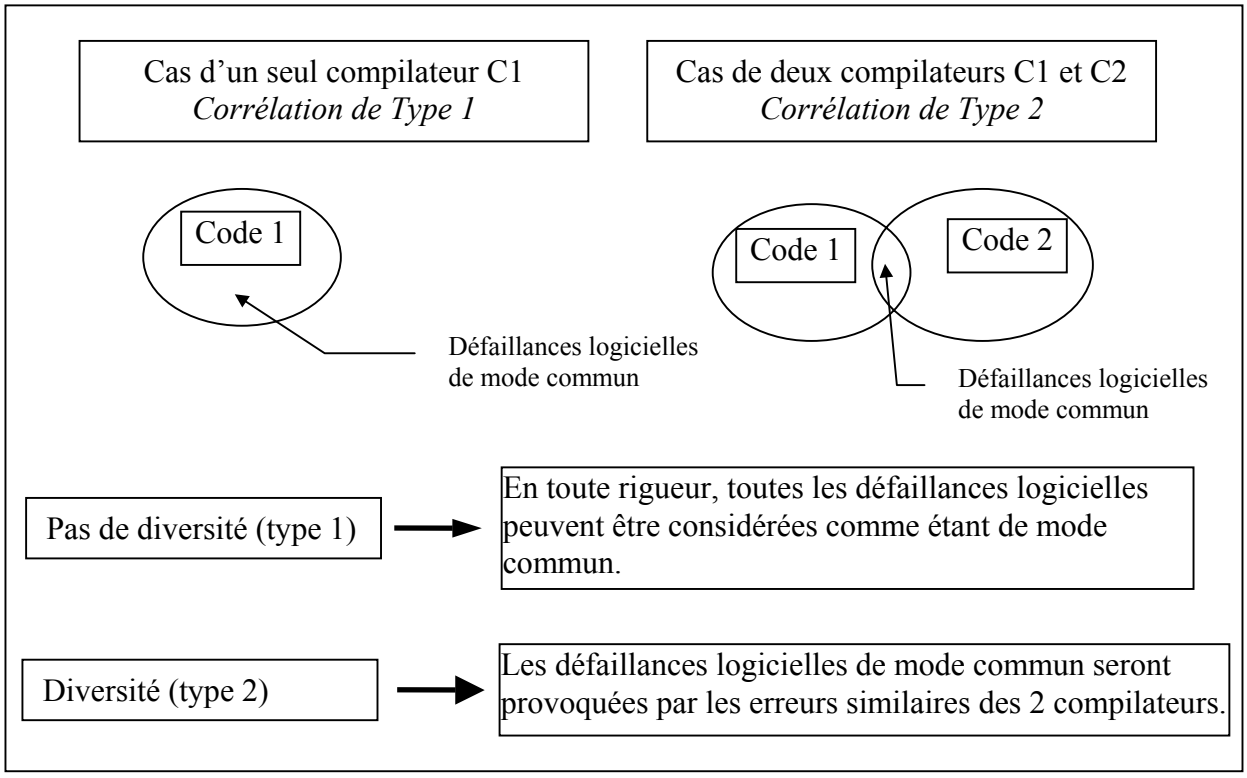



**Figure 3-4 : Exemple des fautes corrélées dues à la compilation**

L'initiateur est l'homme, qui a introduit des fautes au sein des deux compilateurs. Dans le premier cas, la corrélation entre les programmes exécutables est totale. Dans le second, elle n'est que l'intersection des défaillances introduites par les erreurs similaires des deux compilateurs.

Dans le cas où deux compilateurs distincts sont utilisés, l'analyse des défaillances de mode commun nécessite de déterminer toutes les erreurs relatives à chacune des fautes du compilateur C1. Il faut ensuite mener la même démarche pour le compilateur C2 et vérifier qu'aucune erreur similaire n'existe.

### 3.1.3. Défaillances de mode commun : Le retour d'expérience

La première voie à envisager pour appréhender un phénomène quelconque est de tenter d'en connaître les causes possibles. Il faut donc dans un premier temps rechercher les fautes susceptibles de provoquer des défaillances au sein de dispositifs à microprocesseur.

Lorsqu'ils abordent les causes des défaillances de mode commun, les auteurs de la bibliographie consultée relatent systématiquement les retours d'expérience, seul moyen apparent pour obtenir des informations sur ces phénomènes.

*Remarque préliminaire :* Les articles relatifs aux causes et aux conséquences des défaillances de mode commun sont peu nombreux, les constructeurs hésitant certainement à publier leurs résultats et les utilisateurs ne faisant que

rarement la démarche de comprendre la cause précise des défaillances de leurs équipements. Les retours d'expérience relatés dans ces articles sont spécifiques au domaine d'application duquel ils sont issus et font état d'équipements donnés travaillant dans des conditions opérationnelles bien définies. De plus, les seuls éléments disponibles concernent essentiellement des systèmes dont la complexité n'a aucune commune mesure avec celle des systèmes de commande de machines [Tokmachev, 1997], [Rutledge & Mosleh, 1995]. Les résultats qui en découlent sont donc très 'pointus' et peuvent difficilement être étendus à tout type d'application.

Ce constat sur la difficulté à identifier les causes possibles de défaillances de mode commun était déjà fait en 1979 par Edwards et Watson [Edwards & Watson, 1979]. Il restait vrai 15 ans plus tard lors de l'état de l'art sur les CMF fait par Dhillon et Anude [Dhillon & Anude, 1994].

L'intérêt des informations de ce paragraphe est donc plus dans la typologie des fautes rencontrées que dans les chiffres donnés, qui le plus souvent ne peuvent même pas être considérés comme des ordres de grandeur.

#### *3.1.3.1. Généralités*

Les diverses causes de défaillances données par [Miller et al., 1981] pour des retours d'expérience issus du nucléaire sont très générales. 14 causes ont été recensées dont : personnel d'exploitation, de maintenance et de test, erreur de conception, erreur de fabrication, de construction ou de contrôle qualité, procédures défaillantes, environnement extrême, dysfonctionnement électrique ou mécanique, défaillance ou dérive de composants.

Dans le même domaine d'activité, l'analyse des retours d'expérience donnée dans [Taylor, 1975] conduit à la répartition suivante :

- 36% dues à la conception,
- 19% dues à des défaillances de composants,
- 12% dues à des erreurs de maintenance ou d'installation,
- 11% dues à des erreurs opérationnelles,
- 10% dues à des erreurs administratives ou de procédures,
- 12% d'origines inconnues.

Dans un article traitant de données issues des retours d'expérience sur 20 ans de l'industrie avionique, [Pecht & Ramappan, 1992] a pu répartir les défaillances dans les classes suivantes :

- composants,
- interconnexions,
- conception mécanique et électrique du système,
- non reproductibles ou disparition des problèmes après réinitialisation (Could Not Duplicate),

- agressions de l'environnement,
- utilisation.

Les résultats relatifs aux composants sont difficiles à interpréter du fait de l'imprécision dans la définition de ces composants. On peut noter cependant que les fautes affectant l'unité centrale sont en nombre très faible (21 sur 2486) et qu'elles sont principalement dues à des dysfonctionnements électriques ou à des défaillances de composants. Il apparaît aussi que les fautes affectent principalement les composants d'entrées/sorties ou d'interface.

#### *3.1.3.2. Fautes spécifiques aux défaillances de mode commun*

Les données fournies par [Villemeur, 1988] proviennent en grande partie de systèmes complexes multi-technologies (mécanique, hydraulique, électrique). Elles sont essentiellement qualitatives et ne sont pas obligatoirement propres au mode commun.

Agression de l'environnement : Ces causes sont externes au système.

Erreurs de conception : Ces erreurs sont difficiles à prévoir car associées aux limites du savoir-faire. Elles concernent notamment :

- l'inadaptation d'un composant ou d'un système élémentaire à assurer sa mission. L'impossibilité de réaliser des tests exhaustifs ou de simuler les conditions réelles (pour des raisons de coût ou de délais) peut empêcher leur détection lors des essais (cas du logiciel) ;
- tests périodiques inadéquats, comme par exemple une mauvaise conception des tests périodiques susceptible de générer des défauts de cause commune ;
- système élémentaire difficile à exploiter ; cette source sera minimisée au niveau du logiciel en écrivant des programmes simples, structurés et modulaires ;
- système élémentaire difficile à entretenir ; c'est le cas de certains logiciels mal conçus ou mal documentés qui posent des problèmes en phase de modification ;
- mauvaise optimisation en ce qui concerne les défaillances de cause commune ; la protection contre une défaillance de cause commune peut en faire resurgir ou en favoriser une autre ; c'est le cas de certaines structures diversifiées, qui semblent prémunir contre les défaillances de mode commun tout en favorisant le développement d'autres défaillances de ce type par relâchement ;
- oubli lors des études de conception, dont la parade peut être le contrôle qualité ; c'est le principal problème rencontré avec le logiciel.

Erreurs de fabrication : C'est le plus souvent des non-conformités aux spécifications techniques de fabrication ; elles peuvent être rencontrées sur le logiciel si les conditions de sa 'fabrication' (compilateur ou autre) changent. Dans le cas du matériel, elles peuvent apparaître si les technologies utilisées pour les composants ne sont pas matures.

Erreurs d'exploitation : C'est une source conséquente d'erreur lorsque l'on considère les aspects 'système'.

Pour information, on citera les résultats de [Miller et al., 1981], indiquant que les fautes à l'origine des défaillances de mode commun sont provoquées par le personnel (à 57%), la maintenance ou les procédures. Une très faible proportion de ces fautes (<5%) est due à un environnement extrême. Aucune précision n'est donnée quant à ces environnements et à leur mode d'action.

Enfin, dans le domaine de l'avionique, [Taylor, 1975] a relevé 83 défaillances couplées (ou CMF) qui se répartissent en 3 types principaux :

- 16% de défaillances de conception, révélées sous certaines conditions opérationnelles peu usitées,
- 77% de défaillances résultant de l'augmentation des taux de défaillance des composants pris isolément, qui augmente la probabilité d'avoir plusieurs composants en panne,
- 7% de défaillances dues à des erreurs de maintenance sur un groupe de composants.

#### *3.1.3.3. Fautes logicielles*

Quelques expérimentations universitaires donnent une idée des fautes logicielles à l'origine des défaillances de mode commun [Knight & Leveson, 1986a], [Knight & Leveson, 1986b], [Eckhardt & Lee, 1985], [Brilliant et al., 1990]. Elles ont toutes comme cadre des recherches sur la programmation en N-versions qui consistent, à partir d'une même spécification, à développer de façon indépendante N programmes. Elles ont été réalisées sur des traitements algorithmiques de données, pour des applications du domaine aérospatial.

Les auteurs de ces expérimentations ont analysé les origines des défaillances communes à au moins deux versions. Les fautes révélées sont très liées à l'application et concernent le plus souvent des incompréhensions des spécifications. Les fautes suivantes ont été relevées :

- les programmeurs ont fait l'hypothèse d'une équivalence entre la comparaison des cosinus et des angles correspondants, cette faute ne pouvant être attribuée aux spécifications [Knight & Leveson, 1986a],
- le calcul de la valeur d'un angle est erroné, essentiellement à cause des faiblesses en géométrie des programmeurs [Knight & Leveson, 1986a],

- le calcul de l'angle formé par 3 points alignés est erroné : certains programmeurs n'ont pas fait de distinction entre 0 et $\pi$, d'autres n'ont pas considéré l'ordre des points. C'est l'exemple typique de fautes en relation logique, ayant provoqué des défaillances corrélées [Brilliant et al., 1990],
- incompréhension commune des différentes propriétés des systèmes à coordonnées non orthogonales ; les fautes n'étaient pas tout à fait identiques mais ont conduit à des défaillances coïncidentes [Eckhardt & Lee, 1985],
- fautes de logiques [Eckhardt & Lee, 1985],
- calcul de seuil [Eckhardt & Lee, 1985],
- variable mal initialisée [Eckhardt & Lee, 1985].

On constate qu'aucune des fautes recensées ne provient d'une erreur de spécification, qu'elles sont étroitement liées à l'application et qu'elles sont le plus souvent dues à des incompréhensions des spécifications. D'autres fautes ont été recensées, qui n'ont pas provoqué de défaillances de mode commun, par exemple l'omission par le programmeur d'assigner une valeur à une fonction ou l'utilisation d'une expression erronée pour indexer un tableau.

#### *3.1.3.4. En conclusion sur le retour d'expérience*

Suite à l'exploitation de données issues de l'avionique, [Pecht & Ramappan, 1992] fait un certain nombre de remarques :

- ce qui est une cause majeure de défaillance pour une étude ne l'est pas pour une autre ;
- aucune précision ne peut être donnée sur les chiffres, qui varient dans certains cas dans une fourchette allant de 25 à 75% du nombre de défaillances trouvées ;
- une étude portant sur l'avionique montre que seulement 3 à 9% des défaillances d'un système sont dues aux composants, le reste étant dû à des CND (Could Not Duplicate), à l'homme, à une application erronée, .... , ce qui met en évidence la fiabilité élevée et toujours croissante des composants ;
- les causes de défaillances varient dans le temps, certaines étant réduites par les progrès de la technologie ou des procédés de fabrication ;
- les types de défaillances varient eux aussi dans le temps. La grande difficulté à comprendre exactement les mécanismes de défaillances conduit souvent à une typologie de type défaillances 'inconnues', non vérifiées ou autres ;
- la localisation et les mécanismes de défaillances doivent être examinés pour des cas spécifiques.

Ces remarques résument les problèmes rencontrés lorsque l'on veut constituer un retour d'expérience puis en exploiter avec rigueur les résultats. Une telle exploitation est extrêmement délicate et peut conduire à des généralisations hâtives. Ajoutées au problème de confidentialité, ces restrictions émises sur les retours d'expérience expliquent certainement le peu de données bibliographiques disponibles sur ce sujet.

Cette lacune est confirmée par les travaux relatifs aux modes de défaillances des circuits intégrés réalisés par le groupe MDCI de l'ISDF[12] [MDCI, 1994]. Ces travaux concluent à la très grande difficulté, voire à l'impossibilité de connaître précisément les défaillances pouvant affecter les circuits intégrés.

Le problème des défaillances de mode commun doit donc être étudié sans avoir une connaissance parfaite des phénomènes mis en jeu. Il faut se baser sur des 'suspicions' issues de certains retours d'expérience que le concepteur ou l'analyste jugera significatives.

C'est la raison qui conduit à appliquer de façon sélective [Lala & Harper, 1994] des techniques et méthodes pour tolérer, éliminer et prévoir les défaillances de mode commun.

## *3.2. TOLÉRANCE AUX FAUTES*

La conception fail-safe est un moyen adapté pour tout type de défaillances, qu'elles soient dépendantes ou indépendantes. Cependant, c'est la seule véritable technique de tolérance citée par les auteurs d'articles sur les défaillances de mode commun [Bourne, 1981], [Edwards & Watson, 1979], [EWICS, 1988].

Un système est fail-safe si une défaillance interne du système de sécurité se traduit par une mise en repli du processus ou de la machine, les réparations étant effectuées dans un état non opérationnel. Un système fail-safe peut être redondant, mais il n'y a généralement pas une redondance suffisante pour qu'il soit considéré comme tolérant aux fautes.

Toutes les défaillances doivent, dans une mesure acceptable, être des défaillances bénignes [Laprie et al., 1995]. Cela signifie que leurs conséquences sont du même ordre de grandeur que le bénéfice procuré par le service délivré en l'absence de défaillances.

Cette dernière définition laisse une part non négligeable à une appréciation subjective : "dans une mesure acceptable", "du même ordre de grandeur". On lui préférera la définition donnée par la norme EN292-1 [EN292, 1997] qui considère la notion de Fail-Safe comme une condition théorique qui serait atteinte si une fonction de sécurité restait inchangée en cas de défaillance de l'alimentation ou d'un composant contribuant à la réalisation de cette fonction. En pratique, cette condition sera d'autant mieux respectée que l'effet des défaillances sur la fonction de sécurité sera réduit.

Pour concevoir un système Fail-Safe [Bourne, 1981], [Edwards & Watson, 1979], [EWICS, 1988], on devra donc au moins s'assurer que les courts-circuits, les circuits

[12] ISDF : Institut de Sûreté de Fonctionnement

ouverts et les collages sur les câbles, pistes, composants discrets et faiblement intégrés n'altèrent pas la fonction de sécurité. En outre, on préférera des signaux dynamiques à des signaux statiques.

## *3.3. ÉVITEMENT DES FAUTES*

### 3.3.1. Séparation

La séparation [Bourne, 1981], [Edwards & Watson, 1979], [Mosleh, 1991], [Smith, 1997] est une technique pour ne pas propager les défaillances d'une fonction à l'autre, donc de limiter les analyses aux points sensibles. Deux types de propagation sont envisageables [Preckshot, 1994].

**Propagation physique**

Elle concerne le matériel d'un système et permet, à partir d'une même cause, de ne pas générer de fautes corrélées à l'origine de CMFs. Les moyens de lutte contre cette propagation sont :

- séparation physique,
- isolation électrique des différents canaux,
- séparation des alimentations,
- mise en place de blindages électriques,
- affectation des équipements d'urgence à cette seule fonction (pas de partage de ressources avec d'autres parties du système),
- non utilisation de composants communs à plusieurs chaînes,
- éviter de regrouper un trop grand nombre de câbles ou de sous-ensembles,
- ....

**Propagation logique**

Elle concerne essentiellement le logiciel d'un système. Les moyens de lutte contre cette propagation sont :

- isolation physique des modules logiciel exécutés par deux microprocesseurs différents,
- pas d'interactions par l'intermédiaire de mémoires partagées,
- pas de liaisons bidirectionnelles entre systèmes,
- mise en place de protocoles de sécurisation des transferts de données par réseaux,
- ....

Remarque :- La séparation sera appliquée à chaque canal.

- Dans la mesure du possible, la séparation sera appliquée à toutes les phases du cycle de vie du produit : installation, fabrication, ... .

### 3.3.2. Une technique particulière pour les fautes corrélées : La diversité

La diversité est une technique qui consiste à créer n versions d'une entité (matérielle ou logicielle), en introduisant une ou plusieurs modifications dans chaque entité ou de son processus de développement, pour éviter les défaillances de mode commun.

La diversité fonctionnelle est souvent citée pour pallier les défaillances de mode commun [Bourne, 1981], [Edwards & Watson, 1979], [EWICS, 1988], [Mitra et al., 2000], [Avizienis et Laprie, 1986]. Elle consiste à acquérir différents paramètres, à utiliser des technologies différentes, des logiques ou des algorithmes différents, ou des moyens d'activation différents des sorties pour fournir plusieurs voies de détection [Preckshot, 1994].

Il est difficile de justifier la diversité tout au long de la chaîne de traitement, hormis pour les applications à très haute sécurité. Les bénéfices les plus conséquents de la diversité sont au niveau de l'Unité Centrale, de la mémoire, des interfaces, du programme et du format des données [EWICS, 1988].

La diversité est vue comme une parade aux défaillances de mode commun que l'on ne peut prévoir. Elle est complémentaire aux notions d'indépendance et de séparation.

Remarques : - L'apport de la diversité, quel qu'en soit le type, est difficile à appréhender.

- Le recours à la diversité ne doit pas constituer un argument pour diminuer le niveau de qualité d'un système.

Six types de diversité peuvent être distingués [Preckshot, 1994].

#### *3.3.2.1. Diversité humaine*

Les effets de l'homme à tous les stades du cycle de vie d'un système sont très variables et sont à l'origine de beaucoup de défaillances. Managée correctement, la diversité humaine est donc un atout pour la sécurité d'un système.

Exemple : Des concepteurs différents concevant des systèmes de sécurité fonctionnellement différents ont peu de chances de commettre les mêmes erreurs de conception.

Les facteurs augmentant la diversité humaine sont, dans l'ordre décroissant d'efficacité :

- différentes organisations pour la conception (2 sociétés différentes),

- différentes équipes de la même société pour gérer le projet,
- différents concepteurs ou programmeurs,
- différents testeurs, installateurs et personnels de certification.

#### *3.3.2.2. Diversité de conception*

C'est l'utilisation de différentes approches (matérielles et logicielles) pour résoudre un problème identique ou similaire. On fait l'hypothèse que des conceptions différentes ne seront pas affectées par des influences communes.

Les facteurs augmentant la diversité de conception sont, dans l'ordre décroissant d'efficacité :

- différentes technologies (analogique / digitale), pour obtenir des réponses différentes à une même cause de faute.
- différentes approches, à technologie identique (conversion Analogique / Digitale ou inverse ),
- différentes architectures (assemblage et connexions des composants).

Remarque : Cet ordre (proposé par [Preckshot, 1994]) peut être discuté. La diversité d'architecture peut dans certains cas être un moyen efficace d'éviter les fautes corrélées d'origine humaine, du fait de la nécessité de mener en amont une réflexion distincte pour chaque architecture.

#### *3.3.2.3. Diversité logicielle*

C'est en fait un sous-ensemble de la diversité de conception, qui est isolé de par son importance [Avizienis, 1985], [Kelly et al., 1986], [Hourtolle, 1994]. C'est le développement puis l'exécution de versions (ou variantes) différentes mais fonctionnellement équivalentes, dans le but de détecter d'éventuelles erreurs par comparaison en temps réel des résultats obtenus. Suite à l'identification d'un état d'erreur, on procède à un recouvrement d'erreurs par :

- recouvrement arrière, dans lequel on retourne à l'état du système précédemment sauvegardé ;
- recouvrement avant, dans lequel on branche dans un nouvel état (généralement un mode dégradé) dans lequel le système pourra opérer ;
- compensation d'erreurs, basé sur un algorithme utilisant la redondance construite dans le système pour fournir la bonne réponse.

La détection d'erreurs par vérification des résultats issus des différentes variantes peut se faire par :

- test (interne) d'acceptation, pour contrôler les résultats de l'exécution d'un programme. Le calcul du checksum est un exemple typique de test d'acceptation.
- cohérence externe, dans lequel on contrôle les résultats grâce à une intervention externe.
- vérification automatique des résultats numériques. C'est la vérification de la précision numérique des résultats algorithmiques, par exemple pour les calculs en flottant pour lesquels une faible erreur sur un résultat peut se propager en s'amplifiant.

Le plus souvent, les versions sont conçues par des équipes différentes. Comme pour la diversité de conception, on part de l'hypothèse que des concepteurs différents ne feront pas les mêmes erreurs. **Ni la bibliographie, ni le retour d'expérience ne permettent de connaître les conditions d'une diversité logicielle optimale** (il est peut-être suffisant de concevoir deux programmes différents à partir d'une même spécification) [Lyu, 1995]. En fait, le logiciel doit être suffisamment diversifié en paramètres dynamique et logique pour être considéré comme diversifié.

[Preckshot, 1994] propose de classer les facteurs augmentant la diversité logicielle dans l'ordre décroissant d'efficacité suivant :

- différents algorithmes, logique et architecture de programme,
- différents séquencements et ordre d'exécution,
- différents langages de programmation,
- différents environnements de programmation.

Deux techniques de base sont utilisées pour la tolérance aux fautes [Lyu, 1995] :

##### 3.3.2.3.1. Les blocs de recouvrement

Plusieurs blocs fonctionnellement équivalents (M1, M2, M3, …) sont crées et sont exécutés séquentiellement tant qu'aucune erreur n'est détectée par des modules réalisant des tests d'acceptation (A1, A2, A3, …) affectés à chaque bloc Mi.

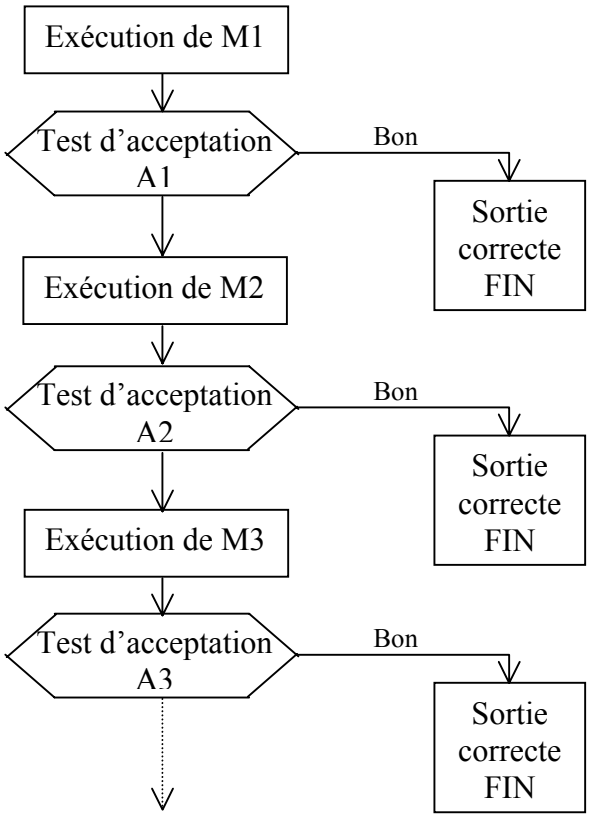



**Figure 3-5 : Comparaison de données dans les blocs de recouvrement**

En toute rigueur, les tests d'acceptation (A1, A2, A3, …) doivent être distincts, mais en pratique, un seul test commun à tous les blocs est souvent développé. Un cas extrême consiste à mettre en œuvre un test d'acceptation similaire aux blocs et à comparer la sortie du bloc contrôlé avec celle du test d'acceptation [Lyu, 1995].

Un des problèmes posé par cette méthode dans un environnement monoprocesseur est le caractère séquentiel de l'exécution des versions [Lyu, 1995].

#### 3.3.2.3.2. La programmation N-versions

La programmation N-versions a fait l'objet d'expérimentations universitaires destinées en particulier à en déterminer les limites, en terme d'efficacité vis-à-vis des défaillances de mode commun [Lyu, 1995], [Knight & Leveson, 1986], [Brilliant et al., 1990], [Eckhardt & Lee, 1985], [Eckhardt et al., 1991]. Cette technique consiste à exécuter en parallèle de multiples versions (N) d'un logiciel et à effectuer un vote majoritaire pour déterminer le résultat final. Le nombre de versions dépend du nombre de fautes que l'on veut tolérer (3 versions seront capables de tolérer 1 faute), mais il est difficile de connaître exactement l'apport du nombre de versions sur le nombre de fautes corrélées restant.

L'hypothèse sur laquelle son efficacité repose est fondée sur le schéma suivant :

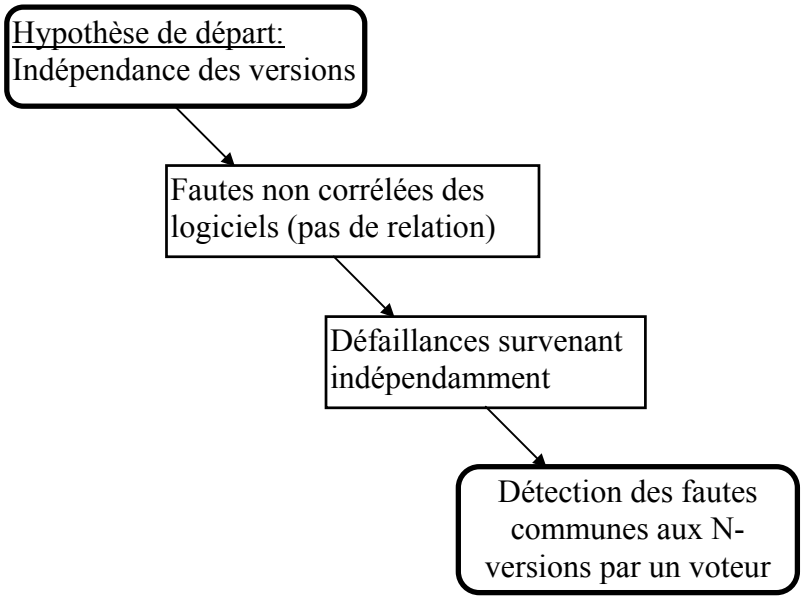



**Figure 3-6 : Détection des fautes dans une structure N-versions**

Pour être réellement efficace, cette technique devra être mise en œuvre en suivant les règles suivantes [Lyu, 1995] :

- les exigences doivent être spécifiées et analysées par des méthodes formelles ;
- les documents de spécifications doivent être débugués et stabilisés avant le développement des composants, par exemple en développant des prototypes du code final ;
- un protocole doit exister pour rapporter et résoudre les problèmes. Ce protocole devra contenir des dispositions pour s'assurer de l'indépendance du développement et éviter d'introduire des fautes corrélées, comme par exemple des erreurs de communication, des manques communs de connaissance ou des échanges d'informations erronés parmi les équipes de développement ;
- la vérification, la validation et le test doivent être formalisés et mettre en évidence l'absence de fautes corrélées ;
- les spécifications, la conception et le code doivent être testés.

Remarque : Ces règles concernent essentiellement l'évitement des fautes et sont applicables quelle que soit la structure logicielle implantée. Certaines, comme l'utilisation de méthodes formelles, sont pour l'instant difficiles à

envisager en sécurité des machines car complexes à mettre en œuvre et nécessitant un personnel spécialement formé.

Les avantages de la programmation N-versions sont :

- Simplification du test puisqu'il suffit d'exécuter les N programmes avec les mêmes entrées et de comparer les sorties obtenues.
- La fiabilité de chaque version peut être moindre, l'apport au niveau global étant apporté par la comparaison. Elle doit cependant être suffisante pour ne pas dégrader le niveau de fiabilité de l'ensemble des N-versions.
- Le surplus en coût de développement peut être compensé par une réduction des coûts de validation, ces avantages étant liés à l'hypothèse de non corrélation des N versions.

Ces avantages sont minimisés par les conclusions des diverses expérimentations :

- Les gains en fiabilité apportés par la programmation N-versions sont fonction de la non corrélation des défaillances des différentes versions [Brilliant et al., 1990], [Knight & Leveson, 1986]. L'expérience montre qu'il n'y a pas de véritable indépendance entre les différentes versions développées [Knight & Leveson, 1986], [Eckhardt et al., 1991]. Des développements (rigoureusement) indépendants ne suffisent donc pas à garantir des gains conséquents en terme de fiabilité [Brilliant et al., 1990].
- Même si, en moyenne, des gains substantiels sont possibles en utilisant la programmation N-versions, ces gains sont tellement variables qu'il est toujours possible de combiner plusieurs versions pour obtenir un mauvais résultat [Brilliant et al., 1990].
- Les résultats sont relatifs aux expérimentations effectuées. Une extension à tout type d'application est donc difficile [Knight & Leveson, 1986].
- L'utilisation de différents langages pour réaliser différentes versions d'un logiciel n'a pas un impact majeur dans la réduction des causes des défaillances corrélées [Brilliant et al., 1990].

Outres ces conclusions, divers enseignements peuvent être tirés des expérimentations réalisées :

- Les différentes expérimentations ont été menées sur des modules relativement simples et de taille réduite. [Knight & Leveson, 1986] conseille, lorsqu'il s'agit de programmes importants composés de nombreux modules interconnectés, d'identifier et de séparer les parties critiques et de n'appliquer la programmation N-versions qu'à ces parties.

- Une règle générale s'applique pour le développement de programmes diversifiés : plus tôt les équipes de développement ont des contacts, plus les chances d'introduire des fautes de mode commun sont élevées [Eckhardt & Lee, 1985].

- Au départ du cycle de vie d'un logiciel, il y a obligatoirement une spécification commune. Mais on doit avoir au minimum des processus de conception différents, pour éviter que des erreurs à ce niveau ne se propagent tout au long du cycle de vie du logiciel [Knight & Leveson, 1986].

- La diversité crée un dilemme sur lequel il est très difficile de se prononcer : ne pas préciser d'algorithme dans une spécification favorise la diversité mais peut générer des fautes dues par exemple aux niveaux de compréhension des programmeurs [Knight & Leveson, 1986].

- Littlewood et Miller ont montré que plus les méthodes de développement sont diversifiées, plus les comportements seront diversifiés, diminuant ainsi la probabilité d'avoir des défaillances coïncidentes [Littlewood & Miller, 1989].

- Pour les fautes de conception, Popov conclut que deux versions du même logiciel, développées indépendamment, seront en moyenne plus fréquemment défaillantes que ce que l'hypothèse d'indépendance impliquerait. Il y a corrélation positive entre les défaillances. L'indépendance des développements n'entraîne donc pas l'indépendance des défaillances [Popov et al., 1998b]

Des travaux pour tenter d'évaluer l'apport de la diversité logicielle sont en cours [Partridge, 1997], [Popov et al., 1999], qui passent notamment par la recherche de modèles adaptés [Littlewood et al., 2000].

#### *3.3.2.4. Diversité fonctionnelle*

La diversité fonctionnelle consiste à réaliser des fonctions physiques différentes ayant des actions similaires sur la sécurité. Dans le cas d'un système à microprocesseur, cette diversité pourra par exemple conduire à travailler sur un signal et son complément, ou encore à tester un résultat et son contraire. Le concepteur devra s'interroger sur l'efficacité de cette technique, pour ne pas créer de diversités artificielles ne répondant à aucun problème réel.

Les facteurs augmentant la diversité fonctionnelle sont, dans l'ordre décroissant d'efficacité :

- différence des mécanismes sous-jacents,

- différence de fonction, de logique de contrôle ou de moyens d'activation,

- différence dans les temps de réponse.

#### *3.3.2.5. Diversité des signaux*

C'est l'utilisation de signaux en provenance de capteurs différents, lesquels doivent être capables indépendamment de signaler des conditions anormales, même si les autres senseurs sont défaillants.

Les facteurs augmentant la diversité des signaux sont, dans l'ordre décroissant d'efficacité :

- différences des effets physiques mesurés,
- différences des paramètres mesurés, avec le même principe mis en jeu,
- redondance de capteurs identiques.

#### *3.3.2.6. Diversité des équipements*

Elle consiste à utiliser des équipements différents pour réaliser des fonctions de sécurité similaires, les différences devant être suffisantes pour diminuer la vulnérabilité aux CMFs. Il faut faire attention aux fausses diversités (composants vendus par deux sources différentes). La diversité des unités centrales peut avoir un effet bénéfique sur la diversité logicielle.

Les facteurs augmentant la diversité des équipements sont, dans l'ordre décroissant d'efficacité :

- différents fournisseurs de produits fondamentalement différents,
- mêmes fournisseurs de produits fondamentalement différents,
- différents fournisseurs de produits similaires,
- différentes versions d'un même produit.

En outre, à un niveau très technique :

- différentes architectures pour les microprocesseurs, cette différence induit des différences dans les compilateurs, éditeurs de liens et autres,
- différentes versions d'un microprocesseur,
- différents circuits imprimés,
- différentes structures de bus.

### 3.3.3. Check listes

Les concepteurs sont à la recherche de méthodes pour considérer les CMF dès le début de la conception d'un système (et pas à la fin comme pour les analyses quantitatives). D'où le développement de check-listes pour l'évaluation de la conception d'un système.

Pour être efficace, les check-listes doivent s'appliquer à toutes les phases du cycle de vie d'un produit. Elles doivent être établies par des personnes expérimentées et être périodiquement évaluées. Elles sont qualitatives par nature et ne fournissent pas un classement des questions posées.

L'annexe 5 présente une synthèse des check listes proposées par [HSE, 1987], [Summers & Raney, 1999], [EWICS, 1988].

## *3.4. PRÉVISION DES FAUTES*

Remarque préliminaire :

La prévision des défaillances logicielles de mode commun ne sera pas abordée dans ce document. L'extrait suivant de l'article de Littlewood [Littlewood, 1996] en résume la problématique :

> *" .... Il est clair qu'il reste de grandes lacunes dans la compréhension des problèmes de base. On sait que l'on aimerait avoir une indépendance dans le comportement en présence de fautes de composants dans un système redondant ou diversifié, ce qui permettrait d'effectuer des calculs simples pour déterminer la fiabilité d'un système. Lorsqu'une indépendance ne peut être envisagée, il est nécessaire d'estimer le degré de dépendance du système examiné pour en calculer la fiabilité. Malheureusement, les informations manquent sur la conception des systèmes pour estimer ces dépendances, et les preuves empiriques sont, par nature, extrêmement éparses. ....*
>
> *.... Avant de développer des théories et des modèles pour prédire la fiabilité d'un système en présence de défaillances de cause commune, il faut avoir une connaissance de base de ces phénomènes et de leurs relations. Il faut répondre aux questions telles que : qu'est-ce que la diversité?, Y a-t-il des conceptions plus diversifiées que d'autres?, dans quelle mesure deux conceptions sont diversifiées ?, quelle diversité peut-on attendre en laissant une complète liberté aux concepteurs, et en leur interdisant toute communication? ...."*.

Cette problématique reste actuelle. Les travaux en cours s'attachent plus à déterminer l'efficacité de la diversité qu'à rechercher à introduire les fautes logicielles dans les calculs probabilistes [Littlewood et al., 2000a], [Popov et al., 1998a], [Popov et al., 1998a]. La recherche de solution pour valider la fiabilité obtenue par la diversification se fait par des approches Bayésiennes [Littlewood et al., 2000b].

### 3.4.1. Les différents modèles de défaillances de mode commun du matériel

Les défaillances de mode commun peuvent être introduites dans les calculs de probabilité de défaillance de façon directe (modèle paramétrique de base) [Fleming, 1989]. On évalue les paramètres de calcul à partir de données issues du retour d'expérience. Étant donné la difficulté à obtenir de telles données, des modèles paramétriques ont été développés. Ce sont les premières méthodes de

modélisation et de quantification des défaillances de mode commun. Ils font les hypothèses nécessaires pour combler l'incomplétude des données pouvant être obtenues par le retour d'expérience.

Plusieurs modèles sont envisageables, que l'on peut classer en fonction de la façon dont les défaillances multiples apparaissent :

- ✓ "Shock models". Ils estiment la fréquence des défaillances multiples des composants en faisant l'hypothèse que le système est soumis à des "chocs" de cause commune à un certain taux, utilisé pour estimer la probabilité conditionnelle de défaillance des composants. La fréquence de défaillance de mode commun est le produit du taux de "chocs" par la probabilité conditionnelle de défaillance. Cette classe peut de nouveau être divisée en deux parties : modèle à un seul paramètre (le facteur β) et modèles multiparamètres (lettres grecques multiples et facteur α).
- ✓ "Nonshock models". Le modèle "taux de défaillance binomial" est représentatif de cette approche [Fleming, 1989]. Ce modèle complexe est essentiellement théorique et il n'existe pas de données pour estimer les valeurs à affecter à ses paramètres. Il n'est donc pas utilisé en pratique et ne sera pas développé dans ce document.

L'objet de ces modèles est multiple :

1. Modéliser l'impact des événements de cause commune sur les performances du système.
2. Quantifier le degré d'indépendance atteint dans des systèmes redondants à l'aide de paramètres.
3. Créer une boucle de retour entre l'expérience de terrain sur les défaillances de mode commun et les modèles de fiabilité des systèmes.
4. Donner des bases pour l'évaluation des caractéristiques fiabilistes d'un système redondant.
5. Donner des moyens d'identification et d'évaluation des mesures défensives à prendre, pour contribuer aux performances de fiabilité.

Remarque : un certain nombre d'autres modèles ont été étudiés comme le Common Load Method [Dewailly et al., 2002]. Du fait de leurs spécificités (adaptation à des systèmes hautement redondants ou mécaniques), ils ne seront pas développés dans ce document.

#### *3.4.1.1. Généralités*

Les modèles sont généralement introduits en considérant l'influence des dépendances entre les 3 composants A, B, C d'un système 2/3 [Mosleh & Fleming, 1988], [Fleming, 1989]. La défaillance d'un composant A du système est donnée par :

$A_t = A_I + C_{AB} + C_{AC} + C_{ABC}$

En considérant que $A_I$ (respectivement $B_I$ et $C_I$ représente l'ensemble des défaillances indépendantes du composant A (respectivement B et C), que $C_{AB}$

($C_{AC}$ et $C_{BC}$) représente l'ensemble des défaillances du composant A communes à B (à C) et que $C_{ABC}$ représente l'ensemble des défaillances communes aux 3 composants A, B et C. La représentation Booléenne réduite de la défaillance du système (sachant que le système a une structure 2/3) est :

$S=A_I.B_I + A_I.C_I + B_I.C_I + C_{AB} + C_{BC} + C_{AC} + C_{ABC}$

Ce qui donne en terme de probabilités :

$P(S)=P(A_I).P(B_I) + P(A_I).P(C_I) + P(B_I).P(C_I) + P(C_{AB}) + P(C_{BC}) + P(C_{AC}) + P(C_{ABC})$

Une pratique courante en analyse de fiabilité est de formuler l'hypothèse suivante : les probabilités d'événements similaires impliquant des types de composants similaires sont les mêmes. Cette hypothèse permet de réduire le nombre de paramètres à quantifier. Cette hypothèse faite par Flemming, à l'origine du modèle β, se traduit par :

$P(A_I) = P(B_I) = P(C_I) = Q_1 = Q_i$ (Q indépendant)

$P(C_{AB}) = P(C_{BC}) = P(C_{AC}) = Q_2$

$P(C_{ABC}) = Q_3 = Q_{CCF}$

Les probabilités de défaillances des trois composants (qu'elles soient indépendantes ou de mode commun) sont donc considérées comme identiques. La probabilité de défaillance pour un seul composant s'écrit :

$P(A_t) = Q_t = Q_i + 2.Q_2 + Q_{CCF}$ (probabilité totale de défaillance du composant A)

Et pour le système :

$P(S) = Q_S = 3.Q_1^2 + 3.Q_2 + Q_3 = 3.Q_i^2 + 3.Q_2 + Q_{CCF}$

Cette égalité se généralise pour un composant appartenant à un groupe de M composants [Mosleh & Fleming, 1988] :

$$Q_t = \sum_{k=1}^{m} \binom{m-1}{k-1} \cdot Q_k \; avec \binom{m-1}{k-1} = \left( \frac{(m-1)!}{(m-k)!\,(k-1)!} \right)$$

#### *3.4.1.2. Le modèle β facteur*

Le modèle β représente le modèle mono-paramètre le plus répandu pour introduire les défaillances de mode commun dans les analyses de fiabilité et calculer la part de ces défaillances sur la probabilité de défaillance d'un système [Fleming, 1975]. Il associe **une fraction β du taux de défaillance** d'un composant à un événement de cause commune (caractérisé par $\lambda_{CCF}$) partagé par les autres composants du groupe. Ce modèle fait l'hypothèse que, lorsqu'un événement de cause commune apparaît, tous les composants du même groupe de composants de cause commune sont défaillants.

Si $\lambda$ est le taux de défaillance d'un des composants d'un groupe de cause commune, les taux de défaillances à considérer seront :

Pour les défaillances de mode commun → $\lambda_{CCF} = \beta.\lambda$

Pour les défaillances indépendantes → $\lambda_i = (1-\beta).\lambda$

avec $\lambda = \lambda_{CCF} + \lambda_i$

Le facteur β est défini par : $\beta = \frac{\lambda_{CCF}}{\lambda} = \frac{\lambda_{CCF}}{\lambda_i + \lambda_{CCF}}$

Pour une loi exponentielle et puisque les défaillances de mode commun et les défaillances indépendantes sont des événements indépendants, la probabilité de défaillance totale $Q_t(t)$ d'un composant s'écrit :

$$Q_t(t) = 1 - e^{-\lambda.t} = 1 - e^{-(\lambda_i + \lambda_{CCF}).t}$$

Cette égalité se simplifie si on considère que $\lambda.t << 1$ :

$$Q_t(t) = (\lambda_i + \lambda_{CCF}).t = Q_{CCF}(t) + Q_i(t)$$

Avec :

$Q_{CCF}(t)$ : probabilité de défaillance d'un composant due à des causes communes,

$$Q_{CCF}(t) = 1 - e^{-\beta.\lambda.t} \approx \beta.\lambda.t \approx \beta.Q_t(t)$$

$Q_i(t)$ : probabilité de défaillance indépendante d'un composant (non-due à des causes communes),

$$Q_i(t) = 1 - e^{-(1-\beta).\lambda.t} \approx (1-\beta).\lambda.t \approx (1-\beta).Q_t(t)$$

Dans le cas d'un système basé sur un système 2/3 à 3 composants identiques et étant donné l'hypothèse de départ (tous les 3 composants sont affectés),on a :

$$Q_1(t) = Q_i(t) = (1-\beta).Q_t(t) \neq 0$$

$$Q_2(t) = 0$$

$$Q_3(t) = Q_{CCF}(t) = \beta.Q_t(t) \neq 0$$

Et la probabilité de défaillance du système devient :

$$Q_S(t) = 3.(1-\beta)^2.Q_t^2(t) + \beta.Q_t(t)$$

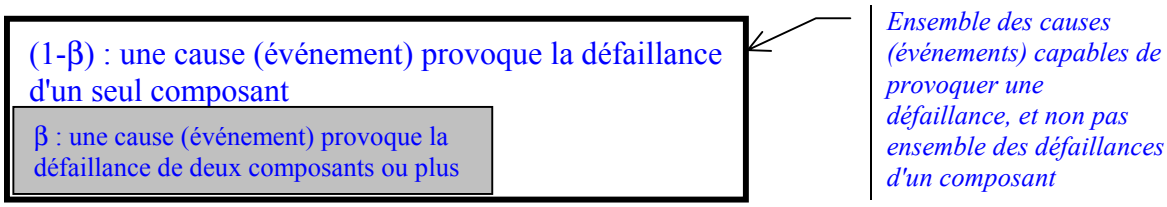
Ce modèle est donc basé sur une estimation du taux de défaillance total du composant (par les bases de données classiques) et du facteur β. Il fournit des résultats corrects et plutôt plus mauvais que la réalité pour des niveaux de redondance inférieurs ou égaux à 4 [Fleming 1989].

Remarques :

i. On considérera dans ce paragraphe que les composants d'un groupe de cause commune sont quelconques et on montrera dans le paragraphe 3.4.2 que le

modèle β n'est en fait applicable qu'à des groupes de cause commune composés de composants identiques.

ii. W.M. Goble (ISA) modélise les causes communes par un facteur β, proportion des causes capables de provoquer la défaillance de plus d'un composant d'un groupe de composants sensibles à une cause commune. Les causes sont : les perturbations environnementales ou les erreurs humaines ou de conception.




Il y a donc une différence par rapport à la définition précédente du facteur β. Il s'agit dans ce cas d'une proportion d'événements capables de provoquer une cause commune au lieu d'une proportion d'un taux de défaillance d'un composant. La cause est soit sous une limite telle qu'elle ne provoque la défaillance que d'un seul composant, soit au-dessus d'une limite telle qu'elle provoque la défaillance de deux ou plus de deux composants.

Malgré cette différence d'approche, Gobble donne exactement les mêmes égalités (portant sur les taux de défaillances) que précédemment pour obtenir $\lambda_{CCF}$ et $\lambda_i$ :

$$\lambda_i = (1-\beta).\lambda \text{ et } \lambda_{CCF} = \beta.\lambda$$

#### *3.4.1.3. Le modèle des lettres grecques multiples (MGL)*

C'est une extension du facteur β. Des paramètres supplémentaires sont ajoutés au facteur β pour traiter des niveaux élevés de redondance [Fleming 1989], [Modarres et al., 1998]. On introduit dans la probabilité totale de défaillance l'effet de toutes les contributions indépendantes et de cause commune de ces composants, ainsi que les probabilités conditionnelles des défaillances de mode commun qu'un composant peut partager avec les composants d'un groupe de cause commune. Pour un système de m composants redondants, on définit donc (m-1) paramètres différents qui, pour un système à 4 composants par exemple, sont :

- β : probabilité conditionnelle que la cause commune de la défaillance d'un composant soit partagée par un composant distinct du premier,
- γ : probabilité conditionnelle que la cause commune de la défaillance d'un composant soit partagée par au plus 2 composants distincts du premier,

- δ : probabilité conditionnelle que la cause commune de la défaillance d'un composant soit partagée par au plus 3 composants distincts du premier.

Ces paramètres sont introduits dans les calculs des différentes probabilités de défaillances à l'aide des équations suivantes :

$$Q_k = \frac{1}{\binom{m-1}{k-1}} \cdot (1-\rho_{k+1}) \cdot \left( \prod_{i=1}^{k} \rho_i \right) Q_t$$

$k = 1, \ldots, m$

$\rho_1 = 1, \ \rho_2 = \beta, \ \rho_3 = \gamma, \ \ldots$

Dans le cas d'un système à 3 composants, le facteur δ est nul. Les autres facteurs multiplicatifs de la probabilité totale $Q_t$ sont obtenus à l'aide des équations précédentes :

$Q_1 = Q_i = (1/2).(1-\beta).Q_t$

$Q_2 = (1/2).\beta.(1-\gamma).Q_t$

$Q_{CCF} = \beta.\gamma.Q_t$

et $Q_S = 3 \cdot (1-\beta)^2 \cdot Q_t^2 + \frac{3}{2} \cdot \beta \cdot (1-\gamma) \cdot Q_t + \beta \cdot \gamma \cdot Q_t$

On constate donc que le facteur β est un cas particulier du modèle MGL en prenant γ = 1.

Le tableau suivant issu de [Modarres et al.,1998] donne une estimation couramment utilisée pour les paramètres du modèle MGL. Il est donné à titre indicatif, les auteurs n'ayant précisé ni l'application concernée, ni la technologie considérée.

| Nombre m de composants | β | γ | δ |
|---|---|---|---|
| 2 | 0.1 | --- | --- |
| 3 | 0.1 | 0.27 | --- |
| 4 | 0.11 | 0.42 | 0.4 |

**Tableau 3-1 : Valeurs des paramètres du modèle MGL**

#### *3.4.1.4. Le modèle α facteur*

L'estimation des paramètres des deux modèles décrits précédemment est assez difficile à obtenir malgré l'existence et l'utilisation de méthodes d'approximation. L'estimation rigoureuse de ces paramètres peut être obtenue en introduisant un paramètre intermédiaire basé sur les événements (a priori les

causes de défaillance), qui est plus facile à estimer à partir des données observées plutôt qu'à partir des défaillances du système. Cette estimation est à la base du modèle α facteur [Mosleh & Siu, 1987]. Les paramètres du modèle α facteur sont les suivants :

- $\alpha_k$ : fraction impliquant la défaillance de k composants parmi un groupe de m composants, due à une cause commune, avec $\alpha_1 + \alpha_2 + \alpha_3 + \ldots + \alpha_m = 1$.

A la différence des modèles précédents, les paramètres du modèle α facteur représentent une fraction des événements qui surviennent dans le système plutôt qu'une fraction des taux de défaillances des composants.

Ces paramètres sont introduits dans les calculs des différentes probabilités de défaillances à l'aide des équations suivantes [Mosleh & Siu, 1987] :

$$Q_k = \frac{k}{\binom{m-1}{k-1}} \cdot \frac{\alpha_k}{\alpha_t} \cdot Q_t \ , \ \ pour\ k = 1, \ldots, m$$

$$avec\ \ \alpha_t = \sum_{k=1}^{m} k \cdot \alpha_k$$

D'une façon générale, les paramètres $\alpha_k$ sont définis par l'égalité suivante :

$$\alpha_k = \frac{\binom{m}{k} \cdot Q_k}{\sum_{k=1}^{m} \binom{m}{k} \cdot Q_k}$$

où $\binom{k}{m} \cdot Q_k$ est la fréquence des événements entraînant la défaillance de k composants dans un groupe de cause commune de m composants. Les paramètres $\alpha_k$ sont donc le rapport de la probabilité des événements (causes) de défaillances impliquant k composants sur la probabilité totale de tous les événements (causes) des défaillances d'un groupe de m composants.

Pour un système à trois composants, on a :

$Q_1 = \frac{\alpha_1}{\alpha_t} Q_t$ (1 composant défaillant)

$Q_2 = \frac{\alpha_2}{\alpha_t} Q_t$ (2 composants défaillants)

$Q_3 = 3 \cdot \frac{\alpha_3}{\alpha_t} Q_t$ (3 composants défaillants)

où $\alpha_t = \alpha_1 + 2.\alpha_2 + 3.\alpha_3$ est un facteur de normalisation

et $Q_S = 3 \cdot \left(\frac{\alpha_1}{\alpha_t}\right)^2 \cdot Q_t^2 + 3 \cdot \left(\frac{\alpha_2}{\alpha_t}\right) \cdot Q_t + 3 \cdot \left(\frac{\alpha_3}{\alpha_t}\right) \cdot Q_t$

Les auteurs de ce modèle donnent un exemple de détermination des valeurs des paramètres $\alpha_k$ pour un système de 3 pompes identiques dont on a pu collecter un retour d'expérience significatif : 85 défaillances simples, 3 CMFs impliquant 2 pompes et 1 CMF impliquant les 3 pompes [Mosleh et Siu, 1987]. Les valeurs numériques obtenues sont :

| | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ |
|---|---|---|---|
| 3 composants | 0.95 | 0.04 | 0.01 |

**Tableau 3-2 : Exemple de valeurs des paramètres du modèle α facteur**

#### *3.4.1.5. Comparaison des modèles β facteur et MGL*

Il est possible de comparer les résultats obtenus d'un système à 2 ou 3 composants pour les modèles β facteur et MGL.

**Système à deux composants**

La défaillance d'un composant A du système est donnée par :

$A_t = A_I + C_{AB}$

En considérant que $A_I$ représente l'ensemble des défaillances indépendantes du composant A et que $C_{AB}$ représente l'ensemble des défaillances du composant A communes à B. La représentation Booléenne réduite de la défaillance du système s'écrit :

$S=A_I.B_I + C_{AB}$

Ce qui donne en terme de probabilités :

$P(S)=P(A_I).P(B_I)+ P(C_{AB})$

En faisant les mêmes hypothèses que pour la structure 2/3, on a :

$P(A_I) = P(B_I) = Q_1 = Q_i$ (Q indépendant)

$P(C_{AB}) = P(C_{BC}) = P(C_{AC}) = Q_2 = Q_{CCF}$

Les probabilités de défaillances des deux composants (qu'elles soient indépendantes ou de mode commun) sont donc considérées comme identiques. La probabilité de défaillance du système s'écrit :

$P(S) = Q_S = Q_1^2 + Q_2 = Q_i^2 + Q_{CCF} = (1-\beta)^2.Q_t^2 + \beta.Q_t$

Exemple numérique : On peut considérer que la probabilité totale de défaillance (dépendante et indépendante) $Q_t$ de chaque composant est égale à $8.10^{-3}$.

***β facteur :***

Par exemple β = 0.1

$$Q_S = (1\text{-}0.1)^2.(\ 8.10^{-3})^2 + 0.1.(8.10^{-3})$$

$$Q_S = 6.46.10^{-6} + 8.10^{-4} = 8.06.10^{-4}$$

***MGL :***

Un seul facteur intervient pour une structure à 2 composants. Du tableau, on tire β = 0.1

La valeur est identique à celle obtenue avec le facteur β.

Pour un système à 2 composants, les résultats sont identiques pour les modèles β facteur et MGL.

**Système à 3 composants**

On peut considérer que la probabilité totale de défaillance (dépendante et indépendante) $Q_t$ de chaque composant est égale à $8.10^{-3}$.

***β facteur :***

Par exemple β = 0.1

$$Q_S = 3.(1\text{-}\beta)^2.Q_t^2 + \beta.Q_t$$

$$Q_S = 3.(1\text{-}0.1)^2.(8.10^{-3}) + 0.1.(8.10^{-3})$$

$$Q_S = 9.6.10^{-4}$$

***MGL :***

Du tableau, on tire β = 0.1 et γ = 0.27.

$$Q_S = 3 \cdot (1-\beta)^2 \cdot Q_t^2 + \frac{3}{2} \cdot \beta \cdot (1-\gamma) \cdot Q_t + \beta \cdot \gamma \cdot Q_t$$

$$Q_S = 3 \cdot (1-0.1)^2 \cdot \left(8.10^{-3}\right)^2 + \frac{3}{2} \cdot 0.1 \cdot (1-0.27)^2 \cdot 8.10^{-3} + 0.1 \cdot 0.27 \cdot 8.10^{-3}$$

$$Q_S = 1.1.10^{-3}$$

Pour un système 2/3, on ne constate donc pas de différences sensibles entre les résultats fournis par les deux possibilités envisagées. Il apparaît que le facteur β conduit à des résultats minorés par rapport aux deux autres modèles.

Ces résultats rejoignent ceux décrits dans [Fleming et al., 1986] pour une technologie mécanique.

### 3.4.2. Le facteur β

Compte tenu de la problématique, les niveaux de redondance utilisés en sécurité des machines sont généralement égaux à 2. La modélisation des défaillances de mode commun par le facteur β est donc la plus appropriée, les

autres modèles étant destinés à des niveaux de redondance plus élevés. Allié à sa simplicité, c'est certainement la raison principale de l'utilisation courante du facteur β pour modéliser les défaillances de mode commun dans les calculs de probabilités. Dans ce paragraphe, nous allons décrire les travaux que nous avons menés pour :

- Analyser une méthode disponible pour évaluer ce paramètre.
- Rechercher les possibilités de modéliser les défaillances de mode commun de deux composants totalement hétérogènes.
- Étudier la façon rigoureuse d'introduire les défaillances de mode commun dans les arbres de fautes (modélisation d'un système couramment utilisée en sécurité des machine).

#### *3.4.2.1. Estimation du facteur β – CEI 61508*

Dans le cadre de la détermination de la probabilité de défaillance dangereuse d'un système, la CEI 61508 propose une méthode pour évaluer le facteur β, qui reliela probabilité de défaillance de mode commun à la probabilité de défaillance aléatoire du matériel.

***Éléments pour la détermination du facteur β***

La détermination du facteur β considère les moyens utilisés pour diminuer :

- la zone de chevauchement (ou l'intersection) entre les ensembles de défaillances des canaux d'une structure redondante. On peut agir, d'une part, en réduisant le nombre global de défaillances systématiques et de défaillances aléatoires du matériel de chaque canal et, d'autre part, en assurant une indépendance maximale entre les canaux. Cette réduction est notamment obtenue en appliquant un certain nombre de techniques spécifiées dans la norme CEI 61508. Des techniques d'évitement sont données aux paragraphes 2.2 et 3.3 de ce document.
- les possibilités de coïncidence entre les défaillances des canaux. On cherche alors à détecter sur au moins un canal les fautes potentiellement génératrices de défaillances de mode commun avant que l'autre canal ne soit affecté par la même faute. Une faute identique peut exister à l'état latent sur les deux canaux d'une structure redondante, il suffit de détecter la faute sur un canal pour passer en sécurité et ne pas avoir de défaillance de mode commun. Cette réduction peut par exemple être réalisée par les tests de diagnostics. La détection rapide sur un canal de 99% des fautes potentiellement à l'origine de défaillances de mode commun minimisera la probabilité de défaillance de mode commun du système.

Le second point différencie la méthode de détermination de la CEI 61508 de celle proposée par Humphrey [Humphrey, 87].

***Détermination du facteur β***

Le facteur β tel que défini par la norme CEI 61508 est une fraction du taux des défaillances matérielles aléatoires dangereuses d'un des canaux de l'architecture analysée alors que les théories exposées précédemment appliquaient ce facteur à toutes les défaillances matérielles aléatoires des composants d'un groupe de cause commune, qu'elles soient sûres ou dangereuses. Si β modélise les défaillances de mode commun en l'absence de tests de diagnostics, on a :

$\lambda_{CCF} = \lambda_D.\beta$

La prise en compte des capacités de détection des tests de diagnostic se traduit par :

$\lambda_{CCF} = \lambda_{DU}.\beta + \lambda_{DD}.\beta_D$

avec :

$\lambda_{DU}$ : probabilité de défaillance dangereuse non détectée d'un canal (ou d'un composant),

$\lambda_{DD}$ : probabilité de défaillance dangereuse détectée d'un canal,

β : facteur modélisant les défaillances de cause commune pour des fautes dangereuses non détectables, égal au facteur applicable en l'absence de tests de diagnostic,

$\beta_D$ : facteur modélisant les défaillances de cause commune, pour des fautes dangereuses détectées par les tests de diagnostic. Plus la fréquence des tests de diagnostic est grande, plus la valeur de $\beta_D$ descend en dessous de β.

Les paramètres à considérer pour évaluer le facteur β sont multiples et quelquefois difficiles à déterminer avec précision, d'où le choix subjectif de l'analyste.

L'évaluation proposée ne considère que les défaillances matérielles aléatoires, excluant les défaillances logicielles. Elle utilise une procédure identique pour l'ensemble des éléments de la chaîne de traitement. Cependant, pour intégrer les différences de conditions environnementales et de capacités à se tester, les valeurs numériques affectées aux capteurs et actionneurs diffèrent de celles affectées aux éléments de traitements.

Pour chaque technique d'évitement des défaillances de mode commun, un tableau donne deux valeurs, en distinguant celles dont la contribution est améliorée par l'utilisation des tests de diagnostic (colonne Y) de celles dont le contribution n'est pas améliorée par ces mêmes tests (colonne X). Une pondération Xi et Yi est affecté à chaque technique. Les valeurs à utiliser pour déterminer les grandeurs X et Y d'une structure redondante sont :

$$X = \sum Xi \ , \ Y = \sum Yi$$

Ces deux grandeurs sont utilisées pour déterminer les valeurs à considérer pour β et $\beta_D$ , à l'aide des fonctions suivantes :

$$\beta = f(X+Y) \text{ et } \beta_D = f((Z+1) \bullet X + Y)$$

La valeur de Z est donnée par un tableau. Elle est fonction de la couverture des diagnostics et du temps entre deux tests de diagnostic. Dès que ce temps est supérieur à 5 minutes, la valeur de Z pour l'élément de traitement est nulle, ce qui donne $\beta = \beta_D$. La valeur de Z est nulle pour les capteurs et actionneurs dès que le temps entre deux tests est supérieur à la semaine.

***En conclusion***

- ✓ La méthode proposée par la CEI 61508-6 tente d'objectiver au mieux la détermination du facteur β. C'est à ce jour la seule méthode de détermination du facteur β pour des applications en sécurité des machines. Elle fait intervenir les capacités de diagnostics des systèmes électroniques programmables pour éviter la simultanéité d'apparition de fautes dues à une cause commune, donc éviter les défaillances de mode commun.
- ✓ Les valeurs contenues dans les tableaux résultent de dires d'experts. Du fait de son impossibilité à être étalonnée, cette méthode reste théorique.
- ✓ Des problème d'évaluation pourront se poser lorsqu'une (ou de façon plus aiguë lorsque plusieurs) technique utilisée pour lutter contre les défaillances de mode commun ne sera pas listée dans le tableau (quelle sera son influence sur le facteur β ?).
- ✓ L'application de la méthode proposée pour la détermination de β et $\beta_D$ peut être discutée. Elle conduit par exemple au constat suivant :

    - Pour une structure redondante homogène exécutant un test de diagnostic satisfaisant, c'est à dire un test exécuté au moins une fois par minute avec une couverture de 99%, on obtient :

$$\lambda^1_{D/CMF} = 0{,}02 \bullet \lambda_{DU} + 0{,}01 \bullet \lambda_{DD}$$

    - Pour une structure redondante diversifiée exécutant un test de diagnostic médiocre, c'est à dire un test exécuté moins d'une fois par minute avec une couverture de 60%, on obtient :

$$\lambda^2_{D/CMF} = 0{,}02 \bullet \lambda_{DU} + 0{,}02 \bullet \lambda_{DD}$$

A $\lambda_{DU}$ et $\lambda_{DD}$ identiques, de telles égalités signifient que, vis-à-vis de fautes aléatoires matérielles de mode commun, une redondance diversifiée peut être moins performante qu'une redondance homogène. En pratique, un tel constat semble très improbable, en particulier si tous les composants d'un canal diffèrent de ceux de l'autre canal.

### *3.4.2.2. Applicabilité du facteur β*

#### 3.4.2.2.1. Composants homogènes - Facteur β unique

Soient 2 composants A et B différents. Si on considère que le facteur β est le même pour tous les composants d'un groupe de composants soumis à une

même cause commune, on a, en reprenant les équations du paragraphe précédent :

$$Q_{i_A}(t) = (1-\beta).Q_{t_A}(t) \qquad Q_{i_B}(t) = (1-\beta).Q_{t_B}(t) \quad (1)$$

et $$Q_{CCF_A}(t) = \beta.Q_{t_A}(t) \qquad Q_{CCF_B}(t) = \beta.Q_{t_B}(t) \quad (2)$$

le facteur β étant défini par les égalités suivantes :

$$\beta = \frac{\lambda_{A_{CCF}}}{\lambda_A} = \frac{\lambda_{A_{CCF}}}{\lambda_{A_i} + \lambda_{A_{CCF}}} = \frac{\lambda_{B_{CCF}}}{\lambda_B} = \frac{\lambda_{B_{CCF}}}{\lambda_{B_i} + \lambda_{B_{CCF}}} \quad (3)$$

La défaillance étant commune aux deux composants, les probabilités correspondantes (de mode commun) sont identiques, d'où :

$$Q_{CCF_A}(t) = Q_{CCF_B}(t) \quad (4)$$

Si le facteur β est le même pour tous les composants, on déduit de (2) et de (4) :

$$Q_{t_A}(t) = Q_{t_B}(t) \quad (5)$$

Ce qui signifie que l'utilisation d'un seul facteur β pour modéliser les défaillances de mode commun impose de **considérer des événements qui possèdent la même fonction de répartition** $Q_t(t)=P(T<t)$.

De (5) on déduit que, pour une loi de fiabilité exponentielle et en considérant λ.t petit :

$$\lambda_A \cdot t = \lambda_B \cdot t \quad d'où \quad \lambda_A = \lambda_B$$

ce qui revient à considérer que **les deux composants ont des taux de défaillances identiques**. Ils ont de ce fait une forte probabilité d'être eux-mêmes identiques, des composants hétérogènes ou partiellement hétérogènes n'ayant aucune chance de posséder des caractéristiques probabilistes communes.

**En pratique, la modélisation par un facteur β unique impose donc de considérer que les composants d'un groupe de cause commune sont identiques. Ceci n'est explicitement indiqué dans aucune des références décrivant la modélisation par un facteur β.**

⇨ En toute rigueur, un facteur β unique tel qu'il a été défini n'est donc pas adapté à la modélisation des défaillances de mode commun de deux composants différents.

⇨ La solution consistant à postuler que l'utilisation de deux composants hétérogènes élimine les défaillances de mode commun et à ne considérer que les défaillances indépendantes n'est pas satisfaisante. Il est difficile en effet de postuler que les défaillances de mode commun n'existent pas pour deux composants hétérogènes utilisant par exemple des technologies identiques.

#### 3.4.2.2.2. Composants hétérogènes : Facteurs β multiples

L'objet de ce paragraphe est de rechercher les conditions pour introduire les défaillances de mode commun entre deux composants ou canaux hétérogènes A et B en continuant d'utiliser une modélisation de type facteur β. En terme de probabilité, si on décompose les défaillances en défaillances indépendantes et de mode commun on a, pour 2 composants A et B :

$$Q_{t_A}(t) = Q_{i_A}(t) + Q_{CCF_A}(t)$$

$$Q_{t_B}(t) = Q_{i_B}(t) + Q_{CCF_B}(t)$$

Les deux composants A et B étant différents, les taux de défaillances sont différents :

$$\lambda_A \neq \lambda_B$$

et les probabilités de défaillances correspondantes (tous types de défaillances comprises) sont aussi différentes :

$$Q_{t_A}(t) \neq Q_{t_B}(t)$$

Par contre, les probabilités de défaillances de mode commun de deux composants C1 et C2 sont identiques (puisqu'il s'agit de deux événements identiques), ce qui donne :

$$Q_{CCF_A}(t) = Q_{CCF_B}(t)$$

On considère toujours que le facteur β représente la proportion des défaillances d'un composant due à une cause commune. Étant donné la loi de fiabilité retenue (exponentielle) et de la différence des composants (donc de $Q_{t_A}(t)\ et\ Q_{t_B}(t)$), l'égalité précédente ne peut être vraie que si les facteurs β appliqués aux CMF des deux composants sont différents. En première approximation, si on prend une loi de fiabilité exponentielle et que λt est suffisamment petit, on a :

$$Q_{CCF_A}(t) = \beta_A \cdot Q_{t_A}(t) = \beta_A \cdot \lambda_A \cdot t$$

$$Q_{CCF_B}(t) = \beta_B \cdot Q_{t_B}(t) = \beta_B \cdot \lambda_B \cdot t$$

$$Q_{i_A}(t) = (1-\beta_A).Q_{t_A}(t) = (1-\beta_A) \cdot \lambda_A t$$

$$Q_{i_B}(t) = (1-\beta_B).Q_{t_B}(t) = (1-\beta_B) \cdot \lambda_B t$$

D'où :

$$\boxed{\beta_A \cdot \lambda_A = \beta_B \cdot \lambda_B}$$

Le facteur β est toujours considéré comme une proportion de taux de défaillance mais, comme les taux de défaillances des composants sont différents, les proportions de ces taux dues aux modes communs sont aussi différentes. Si un composant A a un taux de défaillance $\lambda_A$, alors la proportion

de défaillances de cause commune (donc dues à un stress commun à d'autres composants) sera $\beta_A.\lambda_A$. Si un composant B (du même groupe de composants de cause commune) a un taux de défaillance $\lambda_B$, et qu'il est soumis au même stress que le composant A, alors la proportion de défaillances de cause commune (stress commun identique à A) sera $\beta_B.\lambda_B$.

Cette approche pose le problème de la détermination des facteurs $\beta_A$ et $\beta_B$ . Jusqu'à présent, aucune étude n'a été faite en ce sens.

#### 3.4.2.2.3. Composants hétérogènes : Décomposition en éléments homogènes

En théorie, l'hétérogénéité existe mais on peut montrer rapidement qu'en pratique, elle est très difficile à atteindre. Il existe toujours des dépendances qui font que l'hétérogénéité n'est pas parfaite. Il est donc raisonnable de considérer des défaillances de causes communes pour des composants a priori hétérogènes.

Par contre, la modélisation par un facteur β unique présupposant l'identité des composants, celle-ci n'est en toute rigueur pas adaptée à des canaux hétérogènes. La solution que nous avons étudié consiste à isoler les parties communes ou dépendantes du reste de l'architecture (qui sera considérée comme totalement hétérogène) [Blanchet, 2000]. On se ramène alors dans le cadre d'utilisation du modèle "β" : un paramètre $\beta_i$ est introduit pour chacun des composants élémentaires dépendants : $\mathbf{C_i}$ et $\mathbf{C'_i}$.

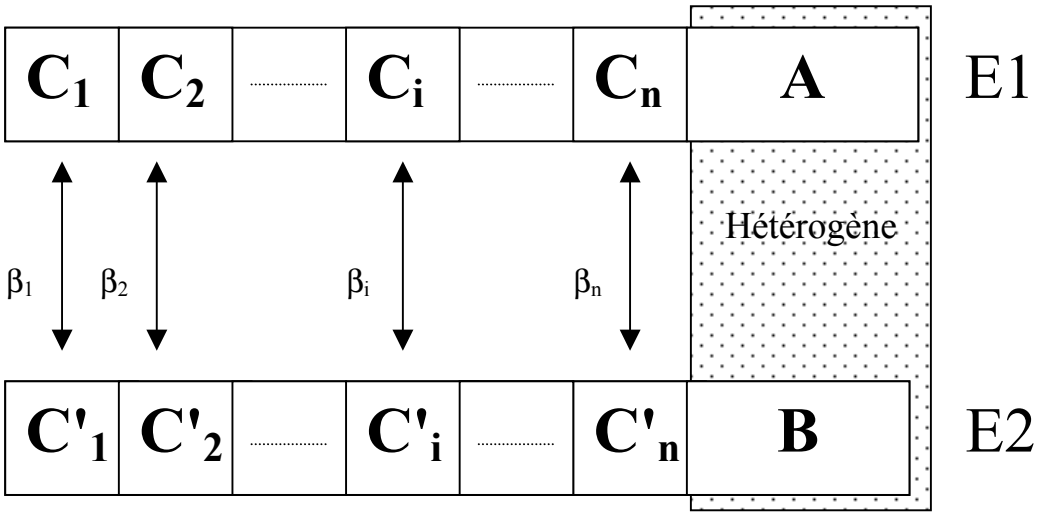



**Figure 3-7 : Facteurs β multiples**

**A** et **B** représentent les parties considérées comme strictement hétérogènes des 2 architectures. On fera l'hypothèse que ces parties ne sont pas affectées de défaillances de mode commun et que, de ce fait, le facteur β correspondant est nul

Nous avons étudié les cas n = 1, n = 2, pour en déduire une généralisation au cas n quelconque.

**Cas n = 1 :**

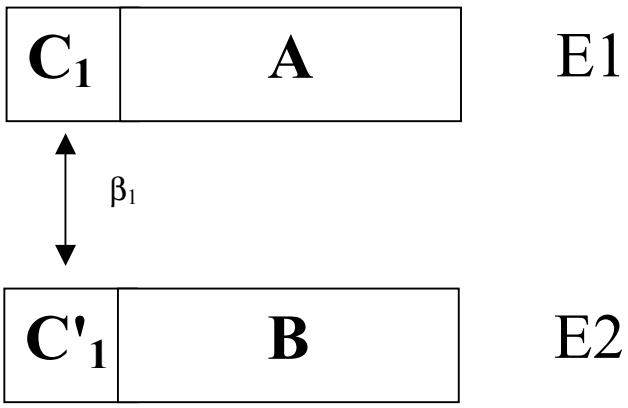



**Figure 3-8 : Cas d'un seul facteur β**

N.B. :

Pour alléger l'écriture, on notera P(CI) la probabilité de défaillance du composant CI.

Les composants C1 et C'1 étant considérés comme homogènes, leurs probabilités de défaillances sont telles que P(C1) = P(C'1)

Les ensembles de défaillances de C1 et A (respectivement C2 et B) sont disjoints.

P(E1∩E2) (respectivement P(E1∪E2)) représente la probabilité de défaillance dans le cas où la sortie est défaillante si E1 et E2 (respectivement E1 ou E2) sont défaillants.

Probabilité de défaillance des architectures E1 et E2 avec prise en compte des modes communs :

P(E1∩E2)

= P[(C1∪A)∩(C'1∪B)]

= P[(C1∩C'1)∪(C1∩B)∪(A∩C'1)∪(A∩B)]

= P[(C1∩C'1)∪(C1∩B)] + P[(A∩C'1)∪(A∩B)]
- P[((C1∩C'1)∪(C1∩B))∩((A∩C'1)∪(A∩B))]

= P(C1∩C'1) + P(C1∩B) - P(C1∩C'1∩B) + P(A∩C'1) + P(A∩B) -
P(A∩C'1∩B) - P[(C1∩C'1∩A)∪((C1∩C'1∩A∩B)∪(C1∩A∩B))]

= P(C1∩C'1).[1 - P(B)] + P(C1).P(B) + P(A).P(C'1) + P(A).P(B) -
P(A).P(C'1).P(B) - P(C1∩C'1∩A) - P[(C1∩C'1∩A∩B)∪(C1∩A∩B)] +
P(C1∩C'1∩A∩B)

$= P(C1 \cap C'1).[1 - P(A) - P(B) + P(A).P(B)] + P(C1).[P(A) + P(B) - P(A).P(B)] + P(A).P(B) - P(C1 \cap C'1 \cap A \cap B) - P(C1 \cap A \cap B) + P(C1 \cap C'1 \cap A \cap B)$

$= P(C1 \cap C'1).[1 - P(A) - P(B) + P(A).P(B)] + P(C1).[P(A) + P(B) - 2.P(A).P(B)] + P(A).P(B)$

Or, $P(C1 \cap C'1) = \beta_1.P(C1) + (1 - \beta_1).P(C1)^2$ (c.f. 3.4.1.5)

D'où,

$P(E1 \cap E2) = [(\beta_1.P(C1) + (1 - \beta_1).P(C1)^2)].[1 - P(A) - P(B) + P(A).P(B)]$

$+ P(C1).[P(A) + P(B) - 2.P(A).P(B)] + P(A).P(B)$

<u>Probabilité de défaillance des architectures E1 et E2 en les supposant strictement hétérogènes :</u>

$P(E1 \cap E2)$

$= P(E1).P(E2)$

$= P(C1 \cup A).P(C'1 \cup B)$

$= [P(C1) + P(A) - P(C1).P(A)].[P(C'1) + P(B) - P(C'1).P(B)]$

$= P(C1)^2 + P(C1).P(B) - P(C1)^2.P(B) + P(A).P(C1) + P(A).P(B) - 2.P(C1).P(A).P(B) - P(C1)^2.P(A) + P(C1)^2.P(A).P(B)$

$= P(C1)^2.[1 - P(A) - P(B) + P(A).P(B)] + P(C1).[P(A) + P(B) - 2.P(A).P(B)] + P(A).P(B)$

Si $\Delta P(E1 \cap E2)$ représente la différence apportée par les dépendances sur la probabilité de défaillances, on a :

$\Delta P(E1 \cap E2) = \beta_1.P(C1).[1 - P(C1)].[1 - P(A) - P(B) + P(A).P(B)]$

$\Delta P(E1 \cap E2) \approx \beta_1.P(C1)$ si les probabilités sont suffisamment petites par rapport à 1 pour négliger les termes du deuxième et troisième ordre.

De plus,

$\Delta P(E1 \cup E2) = \Delta(P(E1) + P(E2) - P(E1 \cap E2))$

$= \Delta P(E1) + \Delta P(E2) - \Delta P(E1 \cap E2)$

$= - \Delta P(E1 \cap E2)$

D'où

$\Delta P(E1 \cup E2) = - \beta_1.P(C1).[1 - P(C1)].[1 - P(A) - P(B) + P(A).P(B)]$

$\Delta P(E1 \cup E2) \approx - \beta_1.P(C1)$ si les probabilités sont suffisamment petites par rapport à 1.

**Cas n = 2 :**

On affecte des facteurs β différents en fonction du degré d'homogénéité des groupes de composants élémentaires sujets aux défaillances de mode commun.

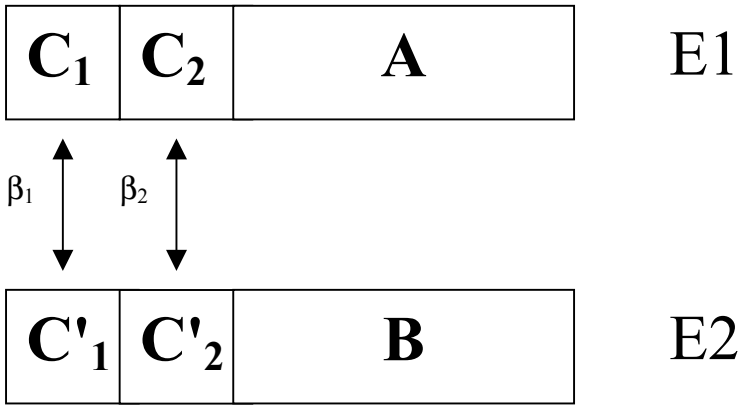



**Figure 3-9 : Cas de deux facteurs β**

Probabilité de défaillance des architectures E1 et E2 avec prise en compte des modes communs :

P(E1∩E2)

= P[(C1∪C2∪A)∩(C'1∪C'2∪B)]

= P[(C1∩C'1)∪(C1∩C'2)∪(C1∩B)∪(C2∩C'1)∪(C2∩C'2)∪(C2∩B)

∪(A∩C'1)∪(A∩C'2)∪(A∩B)]

Or d'après le théorème de Poincaré, si les $\mathbf{C_i}$ sont des événements (quelconques),

$$P(C_1 \cup ... \cup C_n) = \sum_{i=1}^{n} P(C_i) - \sum_{j=2}^{n} \sum_{i=1}^{j-1} P(C_i \cap C_j) + \sum_{k=3}^{n} \sum_{j=2}^{k-1} \sum_{i=1}^{j-1} P(C_i \cap C_j \cap C_k) - ... + (-1)^n . P(C_1 \cap ... \cap C_n)$$

ce qui donne par approximation au 1er ordre :

$$P(C_1 \cup ... \cup C_n) \approx \sum_{i=1}^{n} P(C_i)$$

Ainsi,

P(E1∩E2)

≈ P(C1∩C'1) + P(C1∩C'2) + P(C1∩B) + P(C2∩C'1) + P(C2∩C'2) + P(C2∩B) + P(A∩C'1) + P(A∩C'2) + P(A∩B)

≈ P(C1∩C'1) + P(C1).(P(A) + P(B)) + P(C2∩C'2) + P(C2).(P(A) + P(B)) + 2.P(C1).P(C2) + P(A).P(B)

≈ $\beta_1$.P(C1) + (1 - $\beta_1$).P(C1)² + P(C1).(P(A) + P(B)) + $\beta_2$.P(C2) + (1 - $\beta_2$).P(C2)² + P(C2).(P(A) + P(B)) + 2.P(C1).P(C2) + P(A).P(B)

Probabilité de défaillance des architectures E1 et E2 en les supposant strictement hétérogènes :

$P(E1 \cap E2)$

$= P(E1).P(E2)$

$= P(C1 \cup C2 \cup A).P(C'1 \cup C'2 \cup B)$

$\approx [P(C1) + P(C2) + P(A)].[P(C1) + P(C2) + P(B)]$

$\approx P(C1)^2 + 2.P(C1).P(C2) + P(C2)^2 + P(C1).P(B) + P(C2).P(B) + P(A).P(C1) + P(A).P(C2) + P(A).P(B)$

$\approx P(C1)^2 + P(C1).[P(A) + P(B)] + P(C2)^2 + P(C2).[P(A) + P(B)] + 2.P(C1).P(C2) + P(A).P(B)$

D'où,

$\Delta P(E1 \cap E2) \approx \beta_1.P(C1).[1 - P(C1)] + \beta_2.P(C2).[1 - P(C2)]$

$\Delta P(E1 \cap E2) \approx \beta_1.P(C1) + \beta_2.P(C2)$,

et $\Delta P(E1 \cup E2) \approx - \beta_1.P(C1) - \beta_2.P(C2)$

**Généralisation au cas n quelconque :**

Probabilité de défaillance des architectures E1 et E2 avec prise en compte des modes communs :

$P(E1 \cap E2) = P[ (C1 \cup C2 \cup \ldots \cup Cn \cup A) \cap ( C'1 \cup C'2 \cup \ldots \cup C'n \cup B) ]$

$= P[ (C1 \cap C'1) \cup (C1 \cap C'2) \cup \ldots \cup (C1 \cap C'n) \cup (C1 \cap B) \cup$
$(C2 \cap C'1) \cup (C2 \cap C'2) \cup \ldots \cup (C2 \cap C'n) \cup (C2 \cap B) \cup$
$\ldots.$
$(Cn \cap C'1) \cup (Cn \cap C'2) \cup \ldots \cup (Cn \cap C'n) \cup (Cn \cap B) \cup$
$(A \cap C'1) \cup (A \cap C'2) \cup \ldots \cup (A \cap C'n) \cup (A \cap B) ]$

Or d'après le théorème de Poincaré, si les $\mathbf{C_i}$ sont des événements (quelconques), on a par approximation au 1er ordre :

$$P(C_1 \cup \ldots \cup C_n) \approx \sum_{i=1}^{n} P(C_i)$$

On a donc :

$P(E1 \cap E2) = P(C1 \cap C'1) + P(C1 \cap C'2) + \ldots + P(C1 \cap C'n) + P(C1 \cap B) +$
$P(C2 \cap C'1) + P(C2 \cap C'2) + \ldots + P(C2 \cap C'n) + P(C2 \cap B) +$
$\ldots.$
$P(Cn \cap C'1) + P(Cn \cap C'2) + \ldots + P(Cn \cap C'n) + P(Cn \cap B) +$
$P(A \cap C'1) + P(A \cap C'2) + \ldots + P(A \cap C'n) + P(A \cap B)$

Compte tenu des hypothèses d'indépendances, on a :

P(E1∩E2) = P(C1∩C'1) + P(C1).P(C'2) + … + P(C1).P(C'n) + P(C1).P(B) +
P(C2).P(C'1) + P(C2∩C'2) + … + P(C2).P(C'n) + P(C2).P(B) +
….
P(Cn).P(C'1) + P(Cn).P(C'2) + … + P(Cn∩C'n) + P(Cn).P(B) +
P(A).P(C'1) + P(A).P(C'2) + … + P(A).P(C'n) + P(A).P(B)

$$P(E1 \cap E2) = \sum_{i=1}^{n} P(Ci \cap C'i) + 2.P(C1).\sum_{i=2}^{n} P(C'i)$$
$$+ 2.P(C2).\sum_{i=3}^{n} P(C'i) + ... + 2.P(Cn-1).P(C'n)$$
$$+ \left[P(A) + P(B)\right].\sum_{i=1}^{n} P(Ci) + P(A).P(B)$$

<u>Probabilité de défaillance des architectures E1 et E2 en les supposant strictement hétérogènes :</u>

P(E1∩E2) = P(E1).P(E2)

P(E1∩E2) = P[ (C1∪C2∪ … ∪Cn∪A) ] . P[ ( C'1∪C'2∪ … ∪C'n∪B) ]

P(E1∩E2) = [P(C1).P(C2). … .P(Cn).P(A)] . [P(C'1).P(C'2). … .P(C'n).P(A)]

$$P(E1 \cap E2) = \sum_{i=1}^{n} P^2(Ci) + 2.P(C1).\sum_{i=2}^{n} P(C'i)$$
$$+ 2.P(C2).\sum_{i=3}^{n} P(C'i) + ... + 2.P(Cn-1).P(C'n)$$
$$+ \left[P(A) + P(B)\right].\sum_{i=1}^{n} P(Ci) + P(A).P(B)$$

D'où:

$$\Delta P(E1 \cap E2) = \sum_{i=1}^{n} P(Ci \cap C'i) - \sum_{i=1}^{n} P^2(Ci)$$
$$= \sum_{i=1}^{n} \left[\beta_i.P(Ci) + (1-\beta_i).P^2(Ci)\right] - \sum_{i=1}^{n} P^2(Ci)$$
$$= \sum_{i=1}^{n} \left[\beta_i.P(Ci).(1-P(Ci))\right]$$
$$\Delta P(E1 \cap E2) = \sum_{i=1}^{n} \left[\beta_i.P(Ci)\right]$$

et

$$\Delta P(E1 \cup E2) = -\sum_{i=1}^{n} \left[\beta_i.P(Ci)\right]$$

Les résultats que nous avons obtenus montrent donc qu'il est possible de modéliser les défaillances de mode commun de deux canaux E1 et E2 hétérogènes en utilisant le facteur β. Il suffit pour cela d'identifier les composants élémentaires homogènes qui composent chaque canal et d'appliquer un facteur β à chaque groupe de composants. Si les probabilités sont suffisamment faibles pour négliger les termes au carré, les dépendances se traduisent par une variation positive ou négative de la probabilité de défaillances du système d'une valeur βi.P(Ci), le sens étant fonction de l'architecture des deux canaux.

#### *3.4.2.3. Mise en œuvre du modèle β dans les arbres de défaillances*

On considère deux composants identiques A et B (mêmes lois de probabilité). Soient $A_t$, $B_t$ les événements représentant l'ensemble des défaillances de ces composants. On a :

$P(A_t) = P(B_t) = Q_t(t)$, probabilité totale de défaillance des composants A et B, tout type de défaillances confondues ;

$P(A_I) = P(B_I) = (1-\beta).Q_t(t)$, probabilité des défaillances indépendantes des composants A et B ;

$P(CCF) = Q_{CCF}(t) = \beta.Q_t(t)$, probabilité totale des défaillances de mode commun des composants A et B.

Les événements $A_I$, $B_I$ et CCF sont considérés comme indépendants.

Les arbres de défaillances sont constitués d'un ensemble de portes ET et OU [ISAdTR84, 1998]. L'introduction rigoureuse du facteur β dans ces arbres se fait de la façon suivante :

##### 3.4.2.3.1. Cas "A ∩ B"

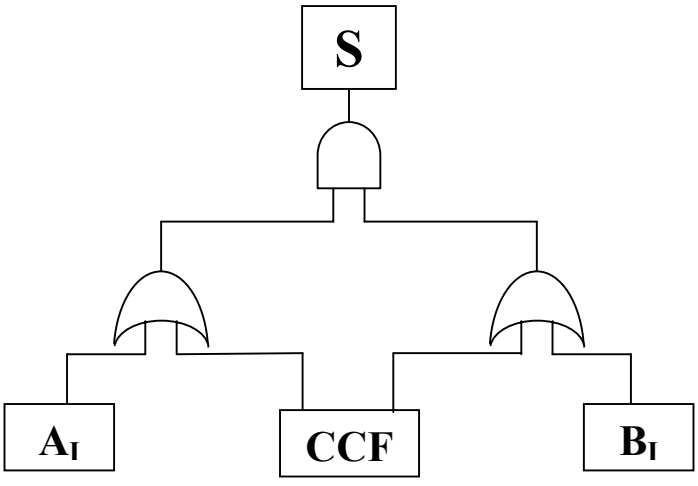



**Figure 3-10 : Arbre des fautes A ∩ B**

$P(S) = P((A_I \cup CCF) \cap (B_I \cup CCF))$

$= P((A_I \cap B_I) \cup (A_I \cap CCF) \cup (B_I \cap CCF) \cup CCF)$

$= P((A_I \cap B_I) \cup CCF))$

$= P(A_I \cap B_I) + P(CCF) - P(A_I \cap B_I \cap CCF)$

$= P(A_I).P(B_I) + P(CCF) - P(A_I).P(B_I).P(CCF)$

$P(S) = (1 - \beta)^2 . Q_t^2(t) + \beta . Q_t(t) - \beta . (1 - \beta)^2 . Q_t^3(t)$

#### 3.4.2.3.2. Cas "A ∪ B"

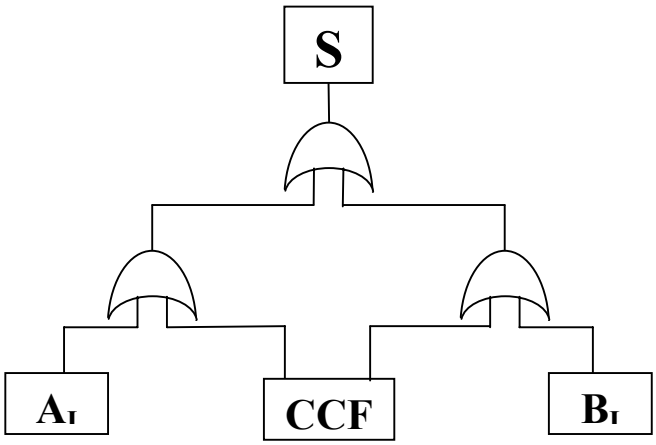



**Figure 3-11 : Arbre des fautes A ∪ B**

$P(S) = P((A_I \cup CCF) \cup (B_I \cup CCF))$

$= P(A_I \cup B_I \cup CCF)$

$= P(A_I \cup B_I) + P(CCF) - P((A_I \cup B_I) \cap CCF)$

$= P(A_I \cup B_I) + P(CCF) - P(A_I \cup B_I).P(CCF)$

$= P(A_I \cup B_I).(1 - \beta . Q_t(t)) + \beta . Q_t(t)$

$= [P(A_I) + P(B_I) - P(A_I).P(B_I)].[1 - \beta . Q_t(t)] + \beta . Q_t(t)$

$= [2.(1 - \beta) . Q_t(t) - (1 - \beta)^2 . Q_t^2(t)].[1 - \beta . Q_t(t)] + \beta . Q_t(t)$

$= \beta . Q_t(t) + 2.(1 - \beta) . Q_t(t) - (1 - \beta)^2 . Q_t^2(t) - 2.\beta.(1 - \beta) . Q_t^2(t) + \beta.(1 - \beta)^2 . Q_t^3(t)$

$= \beta . Q_t(t) + 2.(1 - \beta) . Q_t(t) - (1 - \beta^2) . Q_t^2(t) + \beta.(1 - \beta)^2 . Q_t^3(t)$

$P(S) = (2 - \beta) . Q_t(t) - (1 - \beta^2) . Q_t^2(t) + \beta.(1 - \beta)^2 . Q_t^3(t)$

#### *3.4.2.4. Discussion*

En ignorant les modes communs (β=0), on obtient :

- $P(A \cap B) = Q_t^2$ (t) qui correspond bien à la probabilité de l'intersection de deux ensembles indépendants ($P(A \cap B) = P(A).P(B)$)
- $P(A \cup B) = 2.Q_t(t) - Q_t^2$ (t) qui correspond bien à la probabilité de la réunion de deux ensembles indépendants (P(A ou B) = P(A) + P(B) – P(A et B))

Les formules approchées couramment utilisées ne font généralement pas intervenir le terme $\beta.(1 - \beta)^2. Q_t^3$ (t). De ce fait, les valeurs obtenues pour la probabilité de défaillances dangereuses sont des approximations soit par excès dans le cas d'une structure parallèle, soit par défaut dans le cas d'une structure série.

Si on modélise les taux de défaillances par une loi exponentielle, la formule peut être considérée comme exacte dès que $\lambda$ est suffisamment petit et que les temps d'observation ne sont pas trop élevés, ce qui est généralement le cas des taux de défaillances des composants électroniques.

En ignorant les termes en $Q_t^3$ (t), le fait de considérer des modes communs en utilisant le modèle $\beta$ entre les composants A et B fait varier la probabilité de :

$$\Delta P(ET) = \beta.Q_t(t).[(2 + \beta). Q_t(t) + 1]$$

$$\Delta P(OU) = \beta.Q_t(t).[\beta - 1]$$

Or, dans le cadre d'utilisation de cette théorie, à savoir, la sécurité des machines, l'ordre de grandeur des taux de défaillances est de $10^{-6}$, et la durée de vie maximum d'un composant est de l'ordre de 10 ans :

$$\Rightarrow \; Q_t(t) < p(10 \text{ ans}) = (1 - e^{-\lambda.t_0}) = (1 - e^{-10^{-6} \times 10 \times 365{,}25 \times 24}) \approx 8{,}3.10^{-2} << 1$$

D'où, les probabilités varient approximativement de la façon suivante :

$$\Delta P(ET) \approx \beta. Q_t(t)$$

$$\Delta P(OU) \approx - \beta. Q_t(t)$$

Il est à noter que l'introduction de défaillances de causes communes dans le cas du "OU" fait diminuer la probabilité de défaillance globale : cette constatation peut paraître surprenante dans un premier temps, considérant que de nouvelles sources de défaillances ont été envisagées. En fait, l'introduction de défaillances de causes communes ne consiste pas à ajouter des probabilités à celles déjà existantes, mais à partitionner les défaillances existantes.

# 4. ANALYSE DE SÉCURITÉ D'ARCHITECTURES TYPES

L'objectif de ce chapitre est de rechercher de façon méthodique l'architecture redondante la mieux adaptée au problème posé, en fonction notamment des objectifs de sécurité, du domaine d'application et de la capacité à prendre correctement en compte les défaillances de mode commun [Charpentier & Aubry, 2001], [Charpentier & Aubry, 2002]. Pour cela, on analysera les différentes architectures de sûreté recensées par W.M. Goble [Goble, 1998a], [Goble, 1998b] et la norme CEI 61508 [CEI61508, 2000] pour :

- Déterminer leurs principales caractéristiques et en déduire l'adaptation à la sécurité des machines.
- Déterminer leur probabilité de défaillance dangereuse avec et sans la prise en compte des modes communs, les architectures pouvant être réalisées avec deux canaux identiques ou différents.

En plus de son intérêt pour le choix d'une architecture adaptée à la sécurité, cette analyse donnera des points de repère pour rechercher parmi les équipements industriels existants [EXERA, 2000] celui répondant le mieux à une application donnée, en termes de sécurité et de disponibilité.

## *4.1. CONDITIONS POUR L'ÉVALUATION QUANTITATIVE*

Grandeurs évaluées

L'évaluation quantitative de la sécurité d'une architecture pour des défaillances matérielles aléatoires se fait en déterminant :

- **La probabilité de défaillance à la demande PFD(t) sur l'intervalle [0,t]**, souvent appelée probabilité de défaillance dangereuse (PFD : Probability of Failure on Demand) [Olf, 2001]. Cette probabilité est calculée pour des fortes sollicitations ou des sollicitations en continu de la fonction de sécurité (par exemple une logique de commande d'une presse). On détermine généralement cette probabilité sur un intervalle de temps [0,Ti]. Ti correspond au temps de mission du système, que l'on prend égal au temps entre deux proof tests pour des raisons de simplification des calculs.

  Remarque : PFD (Ti) est une probabilité de défaillance sur un intervalle de temps [0,Ti]. Les prescriptions en terme de SIL pour une sollicitation en continu s'expriment en probabilité de défaillance dangereuse par heure. On passera de PFD(Ti) à la valeur demandée par la CEI 61508 par une division par Ti, pour obtenir une valeur en défaillance par heure.

- **La probabilité moyenne de défaillance à la demande PFDavg(t) sur l'intervalle [0,t]**, qui est calculée lorsque la

fonction de sécurité est faiblement sollicitée (par exemple l'arrêt d'urgence d'une machine). Elle est caractéristique d'un fonctionnement à faible demande, pour lequel la défaillance par unité de temps n'a pas de sens. Elle est généralement évaluée sur l'intervalle de temps [0,Ti].

La probabilité moyenne de défaillance à la demande sur l'intervalle de temps [0,Ti] peut être obtenue à partir de la formule suivante :

$$PFD_{avg}(Ti) = \frac{1}{Ti} \cdot \int_{0}^{Ti} PFD(t) \cdot dt$$

Étant donné les évolutions de la norme CEI 62061 [CEI62061, 2002], nous nous placerons dans des conditions de fortes sollicitations, ce qui nous conduira à évaluer PFD(Ti).

Les évaluations seront faites dans les trois cas suivants :

- Deux canaux homogènes, sans tenir compte des CMFs.
- Deux canaux homogènes, en modélisant les CMF par le facteur β.
- Deux canaux strictement hétérogènes, sans tenir compte des CMFs.

Note : En toute rigueur, les évaluations faites dans le premier cas ont peu de sens : ne pas considérer les CMFs n'a de sens que si les deux canaux sont parfaitement hétérogènes. Ces égalités peuvent tout de même être utilisées pour comparer les PFD sans et avec les CMF et conclure à la nécessité de considérer ce type de défaillance lors des évaluations quantitatives.

Notation

La convention IooJ sera utilisée pour caractériser l'architecture d'un système pour laquelle il est nécessaire que I canaux (sur les J que compte le système) fonctionnent correctement pour que la fonction de sécurité soit exécutée.

Dans les architectures indicées D, les tests de diagnostics, lorsqu'ils sont capables d'identifier le canal défaillant, agissent directement sur la sortie.

Exemples :

1oo2 caractérisera une architecture redondante d'ordre 2 pour laquelle la fonction de sécurité (lorsqu'elle est sollicitée) est exécutée dès qu'un canal est capable de fonctionner correctement. La défaillance d'un canal n'empêche donc pas l'exécution de la fonction de sécurité. Il faut que les deux canaux soient défaillants pour que le système soit défaillant.

2oo2 caractérisera une structure redondante d'ordre 2 pour laquelle il est nécessaire que les deux canaux traitent correctement la fonction de sécurité pour que celle-ci soit exécutée. Dans cette configuration, la défaillance dangereuse d'un seul canal entraînera la défaillance dangereuse du système.

Modélisation de l'architecture

Les équations donnant PFD seront obtenues à partir de l'arbre des fautes modélisant le comportement du système en présence de fautes.

Une approche plus fine consisterait à modéliser les états intermédiaires (système hors service pendant la réparation, système aux potentialités dégradées par la réparation) par des graphes de Markov [Brandt, 1998], [Bukowski & Goble, 1995]. Les graphes obtenus deviennent rapidement très complexes dès que le nombre de composants est important et que les états de défaillances considérés sont nombreux (sûr détecté, sûr non détecté, dangereux détecté, dangereux non détecté). La mise en œuvre de cette technique peut être simplifiée en utilisant une modélisation par micro-Markov, qui combine les avantages des graphes de Markov et la simplicité des Blocs Diagramme de Fiabilité [Knetgtering & Brombacher, 1999]. Cette solution consiste à modéliser dans un premier temps le système par des Blocs Diagramme. Des micro-modèles de Markov peu complexes peuvent alors être crées pour chaque bloc. La fiabilité du système complet est évaluée en combinant ces micro-modèles. Les résultats obtenus apparaissent plus pénalisant (dons plus sûrs) qu'avec des graphes de Markov classiques.

D'autres voies sont envisageables, qui consistent par exemple à introduire les défaillances de mode commun dans un modèle dynamique du fonctionnement du système [Kaufman et al., 2000] ou à réaliser une simulation dynamique lorsque les CMF affectent des éléments dynamiques [Berg et al., 2002].

Détection des fautes

Les tests peuvent être classés en fonction de leur mode de sollicitation (en ligne ou hors ligne) :

*Les Tests de diagnostics (Diagnostic tests ou on-line diagnostics en anglais)*. Ce sont des tests en ligne (exemple tests RAM, PROM, …) qui détectent essentiellement les défaillances aléatoires (dues au matériel) d'un composant, module ou système. Ils sont le plus souvent exécutés à la mise sous tension, puis périodiquement. Ils sont caractérisés par un taux de couverture C, défini comme étant la probabilité qu'une défaillance soit détectée si elle survient.

*Les tests périodiques ou tests d'inspection (Proof tests en anglais)*. Les tests périodiques exécutés hors ligne doivent être différenciés des tests de diagnostics. Ils sont destinés à détecter les défaillances d'un système non détectées en fonctionnement, de telle sorte que le système puisse être rétabli dans une condition 'comme neuf' ou aussi proche que possible de celle-ci. Cette condition suppose d'une part que le taux de couverture des tests périodiques est de 100% et d'autre part, que la réparation est parfaite.

Des tests de diagnostics sont présents dans toutes les architectures analysées dans ce chapitre. Deux comportements sont envisagés suite à la détection d'une défaillance par ces tests :

- La défaillance est signalée pour être réparée suite à un temps de réparation Tr. Il n'y a aucune action pour passer automatiquement en position de repli ou de sécurité. Dans ce cas, une défaillance peut donc être détectée tout en étant potentiellement dangereuse durant l'intervalle de temps Tr compris entre l'instant de sa détection et celui de sa réparation. Dans le cas d'une structure redondante, le danger ne surviendra que lors d'une accumulation de fautes (structures 1oo1, 1oo2, 2oo2) .
- La détection de la défaillance provoque automatiquement le passage dans une position de sécurité ou dans un mode dégradé dans le but de continuer à assurer la fonction jusqu'à la réparation. Dans ce cas, en faisant l'hypothèse que ce passage est instantané, une défaillance détectée n'est pas dangereuse (structures 1oo1D, 1oo2D, 2oo2D) .

Taux de couverture des mécanismes de détection.

Le taux de couverture intervient dans la détermination des taux de défaillance Dangereux Non détecté $\lambda^{DU}$, Dangereux Détecté $\lambda^{DD}$, Sûr Non détecté $\lambda^{SU}$ et Sûr Détecté $\lambda^{SD}$ .

La méthode la plus précise pour déterminer le taux de couverture d'un diagnostic consiste à effectuer une AMDE (Analyse des Modes de Défaillances et de leurs Effets) au niveau des différents composants d'un système [Goble et al., 1998], [Goble & Brombacher, 1999]. Elle nécessite de lister l'ensemble des fautes pouvant affecter les composants et d'analyser si au moins un test de diagnostic est capable de détecter la faute. Cette méthode est une vue théorique lorsqu'il s'agit de composants électroniques complexes pour lesquels il est impossible de répertorier de façon satisfaisante (sans même évoquer l'exhaustivité) cet ensemble de fautes.

Une autre approche consiste à calculer un taux de couverture théorique pour des hypothèses de fautes déterminées (cf. paragraphe 2.1). Comme précédemment, les résultats dépendent de la modélisation des fautes qui aura été faite. Un modèle de fautes uniques équiprobables pourra être retenu si aucune autre donnée n'est disponible (par exemple issues du retour d'expérience).

En toute rigueur, on doit définir un taux de couverture pour chaque composant d'un système. En pratique et en fonction du niveau d'analyse, on considère souvent un taux de couverture global pour un module complet (ou un canal) d'une architecture. Une approximation pourra être obtenue à partir des tables contenues dans la norme CEI 61508-7.

Temps de réparation

Dans le cas où des mécanismes de détection de fautes sont implantés, on appellera Tr le temps nécessaire à la réparation. Ce temps comprend le temps de détection de la faute et le temps pour réparer effectivement la faute.

On considérera pour les calculs que :

- les défaillances sont réparées en ligne sans que le système soit mis hors ligne ;
- la réparation est parfaite et remet le système dans un état comme neuf.

Ces hypothèses ne sont par irréalistes et simplifient les arbres de fautes utilisés pour la modélisation du système.

Pour les architectures indicées D, on fait l'hypothèse que le temps de détection des tests de diagnostics est très inférieur au temps de réparation du système (l'ISA donne des temps de détection de l'ordre de 1 à 5 secondes).

Approximations

Une approximation au premier ordre est généralement faite pour simplifier les évaluations probabilistes. Elle consiste à considérer que :

$$1 - e^{-\lambda t} \approx \lambda.t$$

Cette approximation n'est valable que si λT est très petit devant 1, ce qui en pratique limite la validité des équations obtenues.

Le comportement de l'architecture est observé sur une durée Ti égale à la périodicité des tests d'inspection (proof tests). Pour simplifier les calculs, la période de ces tests sera prise égale au temps de mission.

Les applications numériques sont souvent faites en prenant une durée de mission Ti égale à 1 an (8760 h).

| λ | $10^{-4}$ | $10^{-5}$ | $10^{-6}$ | $10^{-7}$ |
|---|---|---|---|---|
| λ.Ti | $8{,}76.10^{-1}$ | $8{,}76.10^{-2}$ | $8{,}76.10^{-3}$ | $8{,}76.10^{-4}$ |
| $1\text{-}e^{-\lambda.Ti}$ | 0,58 | $8{,}39.10^{-2}$ | $8{,}72.10^{-3}$ | $8{,}75.10^{-4}$ |
| erreur | 0,29 | $3{,}7.10^{-3}$ | $3{,}8.10^{-5}$ | $3{,}8.10^{-7}$ |

**Tableau 4-1 : Erreur d'approximation des calculs de PFD**

Pour une valeur de Ti égale à 1 an, les formules approchées obtenues suite aux approximations ne seront donc valables que pour des valeurs de λ inférieures à $10^{-5}$.

Les termes au cube des équations obtenues au paragraphe 3.4.2.3 ne seront pas considérés.

Modélisation des défaillances matérielles aléatoires

Les défaillances matérielles aléatoires de chaque canal seront modélisées par les taux de défaillances λ. Si $\lambda^D_{canal}$ est le taux de défaillances dangereuses d'un canal et $\lambda^S_{canal}$ le taux de défaillances sûres, les relations entre les différents taux de défaillances sont les suivantes :

$$\lambda_{canal} = \lambda^D_{canal} + \lambda^S_{canal}$$

L'objectif étant de déterminer la probabilité de défaillance dangereuse, on ne considérera que les taux de défaillances dangereuses ($\lambda^D$) des modules composant les architectures étudiées. Ce taux sera décomposé en défaillances dangereuses détectées par les tests de diagnostics $\lambda^{DD}$ et non détectées $\lambda^{DU}$.

$$\lambda^D_{canal} = \lambda^{DU} + \lambda^{DD}$$

avec $\lambda^{DU} = (1\text{-}C).\lambda^D_{canal}$ et $\lambda^{DD} = C.\lambda^D_{canal}$, si des tests de diagnostics de taux de couverture C sont exécutés.

Modélisation des défaillances de mode commun

Seuls les modes communs relatifs aux défaillances matérielles aléatoires seront considérés. Pour deux canaux identiques dont les défaillances de mode commun sont modélisées par un facteur β, les défaillances dangereuses seront réparties en défaillances de mode commun $\lambda^{DCCF}$ et défaillances indépendantes $\lambda^{DI}$.

$$\lambda^{DCCF} = \beta.\lambda^D_{canal},$$

$$\lambda^{DI} = (1\text{-}\beta).\lambda^D_{canal}$$

et si des tests de diagnostics de taux de couverture C sont exécutés :

$$\lambda^{DUCCF} = (1\text{-}C).\beta.\lambda^D_{canal} \text{ et } \lambda^{DDCCF} = C.\beta.\lambda^D_{canal},$$

$$\lambda^{DUI} = (1\text{-}C).(1\text{-}\beta).\lambda^D_{canal} \text{ et } \lambda^{DDI} = C.(1\text{-}\beta).\lambda^D_{canal}$$

## *4.2. CONVENTIONS POUR LA MISE EN ŒUVRE MATÉRIELLE*

De-energised to trip (référence 61508-7)

Traduit en français par "*Principe du courant au repos*". Les systèmes basés sur ce principe sont appelés systèmes normalement alimentés [Goble, 1998a] et sont conçus de façon à supprimer l'énergie suite à une erreur (par exemple une défaillance de l'alimentation). La description donnée par la norme CEI 61508 est la suivante : "*La fonction de sécurité est assurée si les contacts (des relais de sorties de la carte automate et des préactionneurs/contacteurs de puissance) sont ouverts et que le courant ne passe pas."*

Cette technique est basée sur le principe suivant : un grand nombre de fautes sont des coupures de fils (plutôt que des courts-circuits) → pour obtenir un comportement sûr en présence de défaillance, il est préférable d'activer la fonction de sécurité sur une ouverture du contact du relais.

Dans le cas d'une défaillance de l'alimentation, le système est placé dans un état similaire à celui obtenu suite à l'exécution de la fonction de sécurité. Les défaillances dangereuses se traduisent par un maintien de l'alimentation de la charge. Si l'énergie est supprimée sans la présence d'une condition potentiellement dangereuse, on parle de 'false trip'.

La CEI 61508 donne l'exemple suivant : *"Dans le cas où des freins sont utilisés pour arrêter un mouvement dangereux d'un moteur, les freins sont desserrés en fermant les contacts dans le système relatif à la sécurité et serrés en ouvrant les contacts dans le*

*système relatif à la sécurité."* Un tel comportement peut être obtenu grâce à un frein maintenu desserré par un électroaimant. La coupure de la tension aux bornes de l'électroaimant provoquera un serrage des mâchoires du frein, garanti par exemple par un dispositif mécanique à base de ressort.

Cette configuration est adaptée aux applications en sécurité des machines pour lesquelles ce type de comportement suite à la disparition de l'énergie fait partie des principes de conception éprouvés [EN 954-1, 1996]. Elle sera utilisée dans la suite de ce chapitre pour les schémas de réalisation proposés.

Représentation des contacts des relais

Par convention, les relais sont toujours représentés à l'état repos, c'est-à-dire sans alimentation de la bobine. Pour une interprétation correcte des schémas qui seront donnés dans la suite de ce chapitre, on considère qu'à la mise sous tension, une commande issue du 'canal' ou du module de diagnostic amène les contacts dans un état inverse de l'état de repos. En sécurité des machines, l'activation de la fonction de sécurité ou une défaillance du circuit de commande doit amener les contacts des relais dans le même état que l'état repos (cf. annexe 7).

Remarque générale :

Dans toutes les structures non indicées, [Goble, 1998a] choisit de ne pas représenter le module de diagnostic. Ce choix peut être justifié par le fait que ce module n'intervient pas sur la sortie de sécurité. Les diagnostics intervenant dans les évaluations probabilistes, les schémas du paragraphe 4.3 seront donnés en représentant ces modules.

## *4.3. ANALYSE D'ARCHITECTURES TYPES*

### 4.3.1. 1oo1

Cette architecture consiste en un seul canal, pour lequel toute faute dangereuse conduira à la défaillance de la fonction de sécurité en cas de demande. Les diagnostics ne sont présents que pour assurer une détection de fautes en vue de réparer le système. Ils contribuent donc à la disponibilité de l'ensemble, mais pas à sa sécurité.

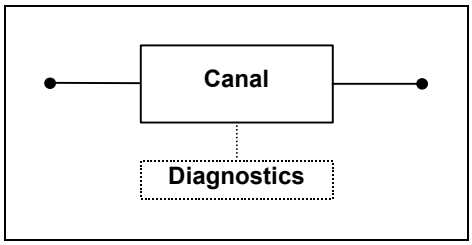



**Figure 4-1 : Schéma de principe d'une architecture 1oo1**

C'est une architecture minimale qui ne tolére pas les fautes et qui, de ce fait ne peut être utilisée dans les applications de sécurité. Il n'existe pas de réalisations concrètes en sécurité basées sur cette architecture.

**Calcul de la probabilité de défaillance dangereuse sans tenir compte des CMF**

Cette architecture ayant recours à un seul canal, il n'y a pas lieu de tenir compte des défaillances de mode commun. La probabilité PDF(Ti) s'écrit :

$$PFD(Ti) = \lambda^{DU} \cdot Ti + \lambda^{DD} \cdot Tr$$

Remarque : En règle générale, on recherche l'influence du temps de mission sur la probabilité de défaillance dangereuse. C'est la raison pour laquelle on écrit PFD(Ti). Le temps de réparation Tr est généralement un invariant, au même titre que les taux de défaillances. Cette remarque s'applique à toutes les égalités données dans la suite de ce chapitre.

### 4.3.2. 1oo1D

C'est une architecture mono-canale dans laquelle les tests de diagnostics sont capables de couper l'énergie des sorties en cas de détection de fautes. Les signaux en provenance du canal de traitement et ceux issus du système de diagnostic sont capables de supprimer l'alimentation de la charge, ce qui conduit à deux possibilités de passer dans un état sûr.

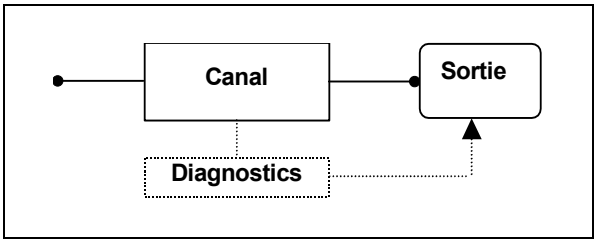



**Figure 4-2 : Schéma de principe d'une architecture 1oo1D**

C'est une amélioration de la structure 1oo1 puisque la détection d'une défaillance dangereuse conduit à une position de repli sûre. La couverture et la fréquence des tests de diagnostics ont une influence sur le niveau de sécurité d'une telle architecture. Les fautes dangereuses (qui entraînent la perte de la fonction de sécurité) sont les fautes non détectées par les tests.

La sortie peut être modélisée par deux relais câblés en série, ouverts au repos. À la mise sous tension, le canal ainsi que le circuit de diagnostic ferment les contacts des relais correspondant. On passe en état sûr (relais ouvert) si la fonction de sécurité est activée ou si les diagnostics ont détecté une défaillance.

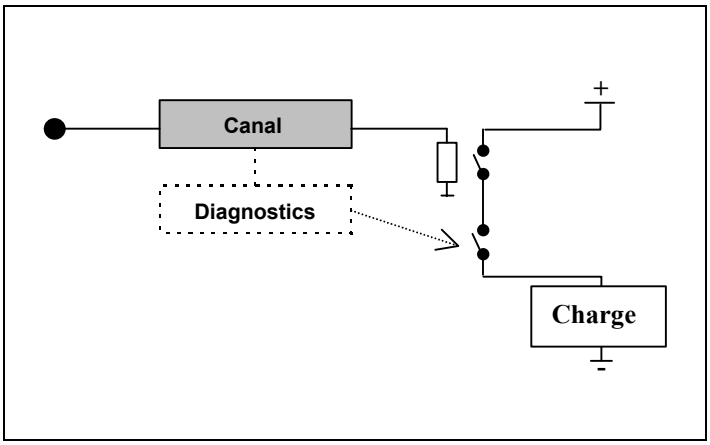



**Figure 4-3 : Schéma électrique de principe d'une architecture 1oo1D**

**Calcul de la probabilité de défaillance dangereuse**

Cette architecture ayant recours à un seul canal, il n'y a pas lieu de tenir compte des défaillances de mode commun. La probabilité PDF(Ti) s'écrit :

$$PFD(Ti) = \lambda^{DU} \cdot Ti$$

### 4.3.3. 1oo2

Cette architecture consiste en deux canaux réalisant chacun la fonction de sécurité. **La fonction de sécurité est exécutée dès qu'un canal en fait la demande**, ce qui correspond à une fonction OU entre les deux canaux. **Il faut donc une défaillance dangereuse des deux canaux pour conduire à la défaillance de la fonction de sécurité sur demande.**

Dans cette configuration, les tests de diagnostic signalent les fautes mais ne changent pas l'état de la sortie. Aucune action n'a donc lieu en temps réel pour passer en sécurité suite à la détection d'une faute. Cette détection a pour but de remettre le système à l'état initial suite à réparation. En faisant l'hypothèse d'une réparation parfaite, on évite ainsi l'accumulation de fautes.

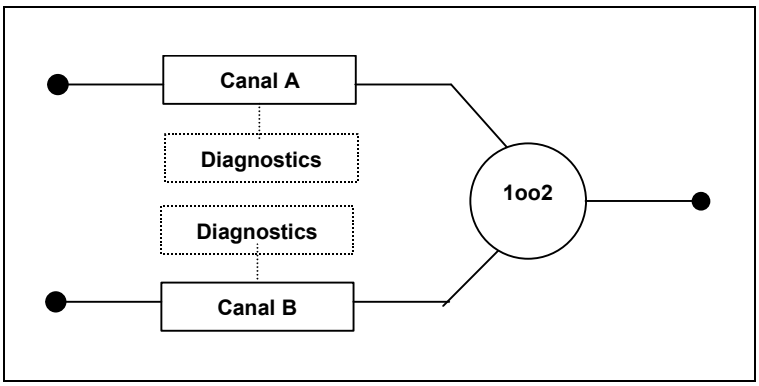



**Figure 4-4 : Schéma de principe d'une architecture 1oo2**

Une telle architecture a généralement une faible probabilité de défaillance à la demande. La capacité à passer en position de repli a comme contrepartie une augmentation du taux de fausses alarmes.

**Schéma électrique de principe**

Un montage de type 'de-energized to trip' se traduit par un schéma série pour lequel une coupure d'énergie provoque un passage en position de sécurité. L'activation de la fonction de sécurité se traduit par l'ouverture des relais de sortie. Si une des voies est défaillante avec sa sortie active (relais fermé), l'autre voie peut désactiver la charge (ouvrir le relais) et assurer la fonction de sécurité.

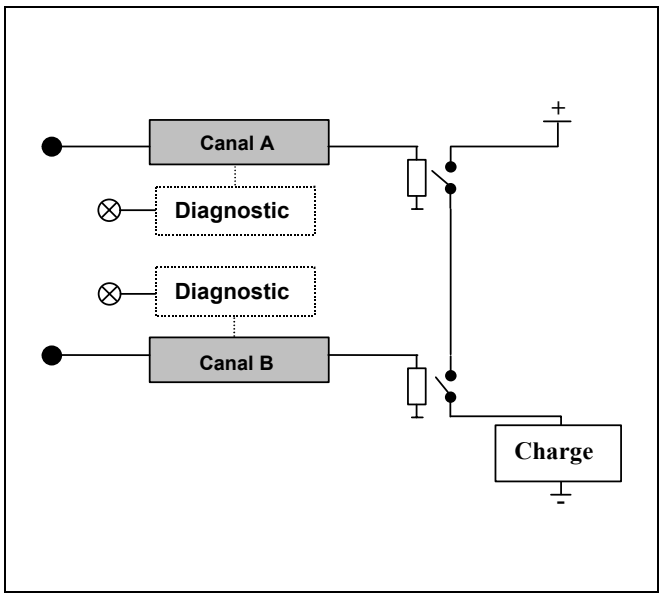



**Figure 4-5 : Schéma électrique de principe d'une architecture 1oo2**

Les différents états possibles sont :

- *État repos* : sans alimentation, les contacts des relais sont ouverts et la sortie n'est pas alimentée.
- *État sous tension, sans appel de la fonction de sécurité* : les deux contacts des relais sont fermés
- *Défaillance non dangereuse* : un au moins des contacts est ouvert. La sortie est coupée (non alimentée) en permanence. Le système est sûr mais n'est plus disponible.
- *Défaillance dangereuse* : les deux contacts sont bloqués à l'état fermé. Il est impossible d'ouvrir les relais, donc de couper la sortie suite à la sollicitation de la fonction de sécurité.

Exemple : On considère la fonction de sécurité suivante : Passage dans le faisceau d'un barrage immatériel → coupure de l'énergie par ouverture des contacts de relais. Chaque canal exécute l'algorithme de détection et l'énergie est coupée dès qu'un canal détecte une présence. Le relais d'alimentation est toujours fermé sauf s'il y a un passage dans le faisceau. Il faut donc une

défaillance dangereuse des deux canaux pour conduire à la défaillance de la fonction sur demande. Si les tests de diagnostics n'ont pas d'influence sur la sortie, la modélisation adéquate est bien de type 1oo2.

| | | | |
|---|---|---|---|
| **Fonctionnement normal** | | | |
| Fonction de sécurité non activée | Les contacts des relais de sortie des deux canaux sont fermés. | | |
| Fonction de sécurité activée | Les contacts des relais de sortie de chaque canal sont ouverts (ce qui coupe l'alimentation du moteur). | | |
| **Canal i défaillant** | | | |
| Contact canal i bloqué à l'état fermé | Le canal non défaillant continue à assurer la mission (ouvrir son relais de sortie si la fonction est activée). | Sûr | Disp. |
| Contact canal i bloqué à l'état ouvert | La sortie est toujours à l'état non alimenté. | Sûr | ~~Disp~~. |
| **Deux canaux défaillants** | | | |
| Contacts des relais de sortie des deux canaux fermés | Impossibilité d'actionner la sortie, qui reste bloquée à l'état alimenté. | ~~Sûr~~ | ~~Disp~~. |
| Au moins un des contacts est ouvert | La sortie est toujours à l'état non alimenté. | Sûr | ~~Disp~~. |

**Tableau 4-3 : Principe de fonctionnement d'une architecture 1oo2**

**Calcul de la probabilité de défaillance dangereuse sans tenir compte des CMF**

- Cas de deux canaux homogènes

Si des tests de diagnostics sont exécutés :.

$$PFD(Ti) = (\lambda^{DU} \cdot Ti + \lambda^{DD} \cdot Tr)^2$$

Si aucun test de diagnostic n'est exécuté :

$C=0, \lambda^{DU} = \lambda^{D}, \lambda^{DD} = 0$

et $PFD(Ti) = (\lambda^{D} \cdot Ti)^2$

- Cas de deux canaux strictement hétérogènes

Si des tests de diagnostics sont exécutés :

$$PFD(Ti) = (\lambda_A^{DU} \cdot Ti + \lambda_A^{DD} \cdot Tr) \cdot (\lambda_B^{DU} \cdot Ti + \lambda_B^{DD} \cdot Tr)$$

Si aucun diagnostic n'est exécuté durant Ti, c'est à dire si le taux de couverture C des tests en ligne est nul :

$$PFD(Ti) = (\lambda_A^D \cdot Ti) \cdot (\lambda_B^D \cdot Ti)$$

**Calcul de la probabilité de défaillance dangereuse en tenant compte des CMFs**

Si des tests de diagnostics sont exécutés :

$$PFD(Ti) = (\lambda^{DUI} \cdot Ti + \lambda^{DDI} \cdot Tr)^2 + \lambda^{DUCCF} \cdot Ti + \lambda^{DDCCF} \cdot Tr \ .$$

Si aucun diagnostic n'est exécuté durant Ti :

$$PFD(Ti) = (\lambda^{DI} \cdot Ti)^2 + \lambda^{DCCF} \cdot Ti$$

### 4.3.4. 2oo2

Cette architecture consiste en deux canaux connectés en parallèle, de sorte que **les deux canaux doivent demander la fonction de sécurité pour que celle-ci soit activée**. L'opération logique correspond à un ET entre les sorties des deux canaux. Le système a un comportement dangereux **dès qu'une défaillance dangereuse survient dans un des deux canaux**.

Remarque : Cette architecture est similaire à l'architecture 1oo2D analysée en 4.3.6, mais il n'y a pas possibilité d'inhiber une voie pour continuer à travailler sur l'autre ; en cas de détection de fautes, les diagnostics n'agissent pas sur la sortie.

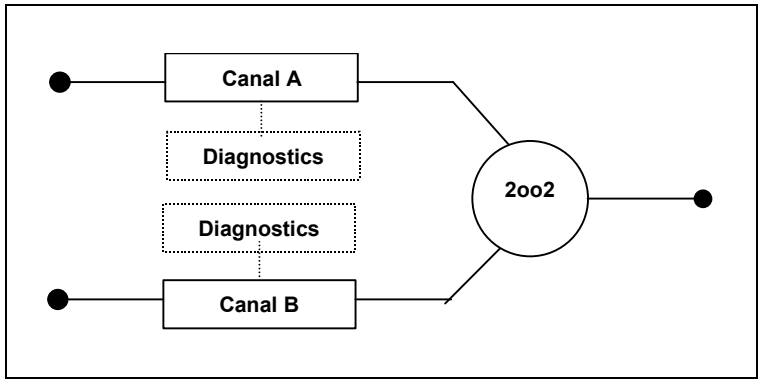



**Figure 4-6 : Schéma de principe d'une architecture 2oo2**

**Schéma électrique de principe**

Un montage de type 'de-energized to trip' se traduit par un schéma parallèle pour lequel une coupure d'énergie provoque un passage en position de sécurité. La sortie est modélisée par deux relais câblés en parallèle, ouverts au repos. A la mise sous tension, chaque canal commande la fermeture du contact de son relais de sortie. En fonctionnement normal, l'activation de la fonction de sécurité ouvre les relais correspondant à chaque canal. Les 2 canaux doivent demander la fonction de sécurité pour que celle-ci soit exécutée.

Le système a un comportement dangereux (sortie alimentée alors que la fonction de sécurité est sollicitée) dès qu'une défaillance dangereuse (collage du relais de sortie) survient dans un des deux canaux. A moins que chaque canal ait une conception fail-safe, cette structure n'est pas adaptée aux applications en sécurité des machines.

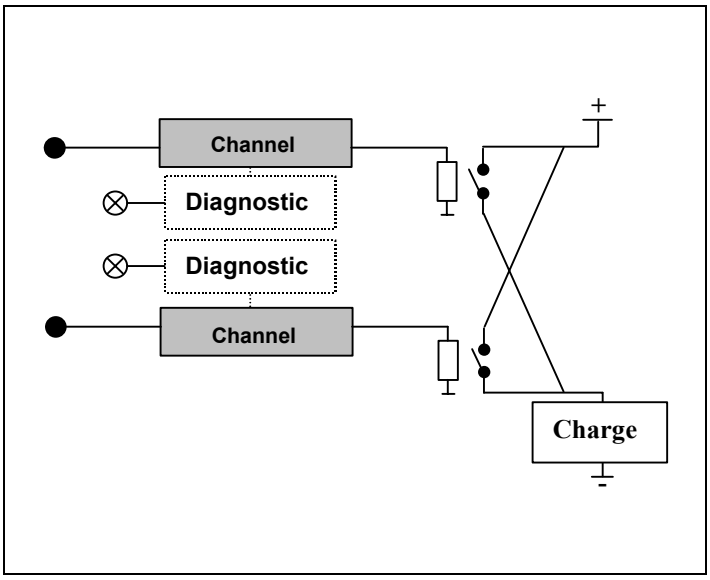



**Figure 4-7 : Schéma électrique de principe d'une architecture 2oo2**

| | | | |
|---|---|---|---|
| **Fonctionnement normal** | | | |
| Fonction de sécurité non activée | Les contacts des relais de sortie de chaque canal sont fermés. | | |
| Fonction de sécurité activée | Les contacts des relais de sortie des deux canaux sont ouverts. | | |
| **Un canal défaillant** | | | |
| Contact canal i bloqué à l'état ouvert | Le canal non défaillant continue à assurer la mission (ouvrir son relais de sortie si la fonction est activée). | Sûr | Disp. |
| Contact canal i bloqué à l'état fermé | La sortie est toujours alimentée. | ~~Sûr~~ | ~~Disp~~. |
| **Deux canaux défaillants** | | | |
| Contacts des relais de sortie des deux canaux fermés | Impossibilité d'actionner la sortie, qui reste bloquée à l'état alimenté. | ~~Sûr~~ | ~~Disp~~. |
| Un des contacts est ouvert, l'autre est fermé | Impossibilité d'actionner la sortie, qui reste bloquée à l'état alimenté. | ~~Sûr~~ | ~~Disp~~. |
| Les 2 contacts sont ouverts | La sortie est toujours à l'état non alimenté. | Sûr | ~~Disp~~. |

**Tableau 4-4 : Principe de fonctionnement d'une architecture 2oo2**

**Calcul de la probabilité de défaillance dangereuse sans tenir compte des CMF**

- Cas de deux canaux homogènes

Si des tests de diagnostics sont exécutés :

$$PFD(Ti) = 2 \cdot \left[ \lambda^{DD} \cdot Tr + \lambda^{DU} \cdot Ti \right]$$

Si aucun diagnostic n'est exécuté durant Ti, c'est à dire si le taux de couverture C des tests en ligne est nul :

$$PFD(Ti) = 2 \cdot \left[ \lambda^{D} \cdot Ti \right]$$

- Cas de deux canaux strictement hétérogènes

Si des tests de diagnostics sont exécutés :

$$PFD(Ti) = (\lambda_A^{DU} \cdot Ti + \lambda_A^{DD} \cdot Tr) + (\lambda_B^{DU} \cdot Ti + \lambda_B^{DD} \cdot Tr)$$

Si aucun diagnostic n'est exécuté durant Ti, c'est à dire si le taux de couverture C des tests en ligne est nul :

$$PFD(Ti) = (\lambda_A^{D} \cdot Ti) + (\lambda_B^{D} \cdot Ti)$$

**Calcul de la probabilité de défaillance dangereuse en tenant compte des CMFs**

On répartit les défaillances entre défaillances de mode commun $\lambda^{DCCF}$ et défaillances indépendantes $\lambda^{DI}$.

Si des tests de diagnostics sont exécutés :

$$PFD(Ti) = \lambda^{DDCCF} \cdot Tr + \lambda^{DUCCF} \cdot Ti + 2 \cdot \lambda^{DDI} \cdot Tr + 2 \cdot \lambda^{DUI} \cdot Ti$$

Si aucun diagnostic n'est exécuté durant Ti :

$$PFD(Ti) = 2 \cdot \lambda^{DI} \cdot Ti + \lambda^{DCCF} \cdot Ti$$

Exemple : Dans le cas de la double commande, deux fonctions sont réalisées :

Fonction 1 : Appui simultané sur les deux B.P. → démarrage du cycle

Fonction 2 : Pas d'appui simultané → pas de démarrage du cycle. C'est en fait la fonction de sécurité associée à la double commande.

Les deux canaux doivent demander la Fonction 1 pour que le relais de sortie soit fermé, ce qui correspond bien à une architecture de type 2oo2, sans la notion de défaillance dangereuse. Dans ce cas, une défaillance de cette fonction ne conduit à aucun danger, mais uniquement à une indisponibilité de la fonction de démarrage du cycle.

Pour la Fonction 2, il suffit qu'un canal détecte la non simultanéité pour que la fonction soit réalisée. C'est donc un OU entre les deux canaux, modélisé par l'architecture 1oo2. La défaillance dangereuse correspond à l'activation du

relais de sortie alors qu'il n'y a pas d'appui simultané (c'est la boucle de comptage de l'intervalle de temps séparant les appuis sur les deux boutons poussoirs).

### 4.3.5. 2oo2D

Cette architecture consiste en deux canaux de type 1oo1D connectés en parallèle, réalisant chacun la fonction de sécurité. **En fonctionnement normal, les deux canaux doivent demander la fonction de sécurité pour que celle-ci soit activée**, ce qui peut être modélisé par un ET logique entre les deux canaux.

Dans cette configuration, la détection d'une défaillance par les tests de diagnostic fait passer la sortie du canal défaillant en position de repli. La fonction de sécurité peut encore être traitée en mode dégradé par le canal non défaillant. Par contre, l'existence d'une défaillance dangereuse non détectée sur une des voies conduira à une défaillance dangereuse du système en cas d'activation de la fonction de sécurité. **La fonction de sécurité ne sera donc pas assurée dès qu'une défaillance dangereuse non détectée est présente dans un des deux canaux.**

Remarque : le module de diagnostic détecte les défaillances du canal lui correspondant, sans effectuer par exemple de comparaison avec les informations de l'autre canal.

Cette architecture a une **disponibilité accrue** par rapport à l'architecture 2oo2 puisqu'il est possible de continuer à assurer la mission (en mode dégradé) en présence d'une défaillance détectée potentiellement dangereuse d'une des deux voies.

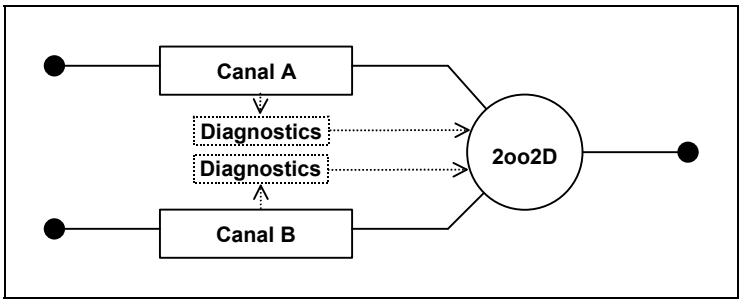



**Figure 4-8 : Schéma de principe d'une architecture 2oo2D**

**Schéma électrique de principe**

Un montage de type 'de-energized to trip' se traduit par un schéma parallèle pour lequel une coupure d'énergie provoque un passage en position de sécurité. La sortie est modélisée par deux relais câblés en parallèle, ouverts au repos.

Pour être capables de passiver les sorties de chaque canal en cas de détection d'une défaillance, les relais correspondant aux modules de diagnostics sont câblés en série avec les relais de chaque canal. Leurs contacts doivent s'ouvrir dès qu'une faute a été détectée.

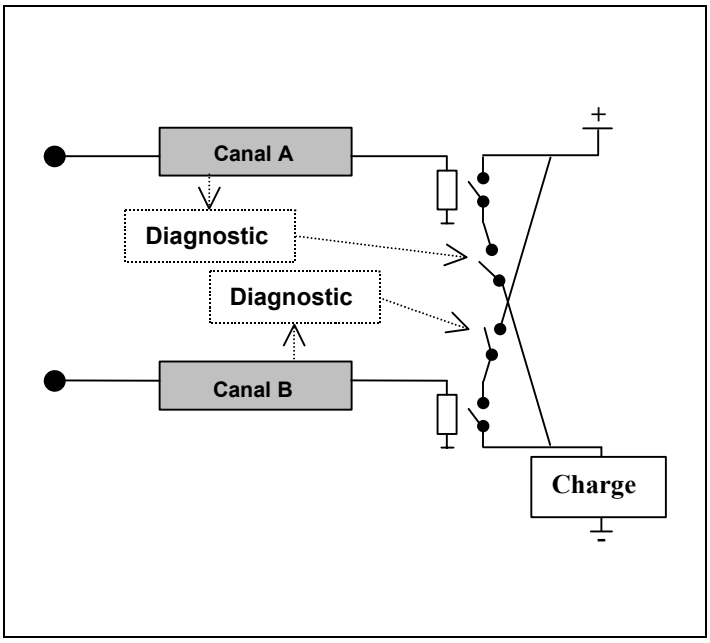



**Figure 4-9 : Schéma électrique de principe d'une architecture 2oo2D**

Le tableau suivant synthétise les différents comportements en présence de fautes.

| | | | |
|---|---|---|---|
| **Fonctionnement normal** | | | |
| Fonction de sécurité non activée | Les contacts des relais de sortie des deux canaux sont fermés, de même pour les contacts des relais de diagnostics. | | |
| Fonction de sécurité activée | Les contacts des relais de sortie des deux canaux sont ouverts. Les contacts des relais correspondants aux modules de diagnostics sont maintenus fermés puisque aucune faute n'est présente. | | |
| **Un canal défaillant** | | | |
| Contact canal i bloqué à l'état ouvert | Quelle que soit la détection par les diagnostics, on assure la mission avec l'autre canal (ouvrir son relais de sortie si la fonction est activée) | Sûr | Disp. |
| Contact canal i bloqué à l'état fermé | Défaillance détectée : Le relais de diagnostic ouvre la branche correspondante. L'autre canal peut assurer la mission. | Sûr | Disp. |
| | Défaillance non détectée : Impossibilité de couper l'alimentation de la sortie. | ~~Sûr~~ | ~~Disp~~. |
| **Deux canaux défaillants** | | | |
| Contacts des deux canaux ouverts | On est toujours dans un état de sécurité (sortie non alimentée). | Sûr | ~~Disp~~. |
| Un des contacts est fermé | Défaillance détectée : Le relais de diagnostic ouvre la branche correspondante. | Sûr | ~~Disp~~. |
| | Défaillance non détectée : Impossibilité de couper l'alimentation de la sortie. | ~~Sûr~~ | ~~Disp~~. |

| | | | |
|---|---|---|---|
| Les deux contacts sont fermés | Défaillances détectées : Le relais de diagnostic ouvrent les deux branches. | Sûr | ~~Disp~~ |
| | Défaillances non détectées : impossibilité de couper l'alimentation de la sortie. | ~~Sûr~~ | ~~Disp~~ |

**Tableau 4-5 : Principe de fonctionnement d'une architecture 2oo2D**

**Calcul de la probabilité de défaillance dangereuse sans tenir compte des CMF**

Dans ce cas, on ne considère plus que les défaillances non détectées.

- Cas de deux canaux homogènes :

$$PFD(Ti) = 2 \cdot \left[\lambda^{DU} \cdot Ti\right]$$

- Cas de deux canaux strictement hétérogènes :

$$PFD(Ti) = (\lambda_A^{DU} \cdot Ti) + (\lambda_B^{DU} \cdot Ti)$$

**Calcul de la probabilité de défaillance dangereuse en tenant compte des CMFs**

On répartit les défaillances dangereuses non détectées entre défaillances dangereuses non détectées de mode commun $\lambda^{DUCCF}$ et défaillances dangereuses non détectées indépendantes $\lambda^{DUI}$.

$$PFD(Ti) = 2 \cdot \lambda^{DUI} \cdot Ti + \lambda^{DUCCF} \cdot Ti$$

### 4.3.6. 1oo2D

**La fonction de sécurité est exécutée dès qu'un canal en fait la demande**, ce qui correspond à une fonction logique OU entre les sorties des deux canaux. Il faut donc une défaillance dangereuse des deux canaux pour conduire à la défaillance de la fonction de sécurité sur demande. Les relais de sortie étant câblés en parallèle, cette architecture est similaire à l'architecture 2oo2D, mais une ligne de contrôle supplémentaire lui donne des capacités fonctionnelles de sécurité d'une architecture de type 1oo2. **En fonctionnement normal, cette ligne permet à un canal de désactiver l'autre voie par l'intermédiaire de son relais de diagnostic suite à l'activation de la fonction de sécurité**, donc d'activer la fonction de sécurité dès qu'un canal en fait la demande.

Dans cette configuration, le relais de diagnostic affecté à un canal peut être désactivé par :

- le module de diagnostic lui-même, en cas de détection d'une défaillance du canal qu'il surveille,
- le canal opposé, si la fonction de sécurité est activée et que ce canal n'est pas défaillant.

Cette potentialité est utile dans le cas où une défaillance potentiellement dangereuse survient dans un canal et n'est pas détectée par les diagnostics implantés sur ce canal. On a alors la possibilité de passer en sécurité en passivant la voie défaillante par l'intermédiaire d'une action de la voie opposée sur le relais de diagnostic.

Remarque : Le schéma de principe donné par [Goble, 1998a] laisse penser que le relais de diagnostic peut aussi être désactivé par le canal lui-même. Cette troisième possibilité ne sera pas envisagée dans ce document.

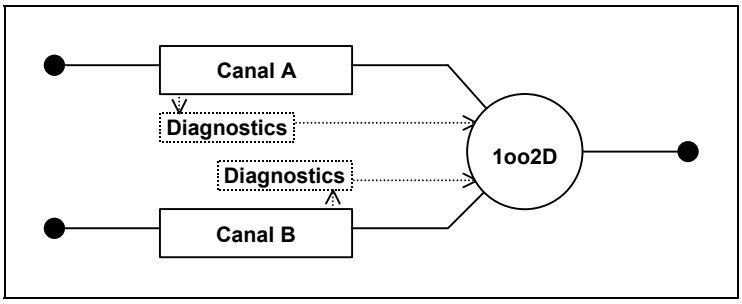



**Figure 4-10 : Schéma de principe d'une architecture 1oo2D**

**Schéma électrique de principe**

Un montage de type 'de-energized to trip' se traduit par un schéma parallèle pour lequel une coupure d'énergie provoque un passage en position de sécurité. La sortie est modélisée par deux relais câblés en parallèle, ouverts au repos.

Pour être capables de passiver les sorties de chaque canal, les relais correspondant aux modules de diagnostics sont câblés en série avec les relais de chaque canal. Leurs contacts s'ouvrent dès qu'une faute a été détectée par le module de diagnostic ou que la fonction de sécurité a été activée et traitée par la voie opposée.

A la mise sous tension, les canaux et les modules de diagnostics commandent la fermeture des contacts des relais correspondants. L'activation de la fonction de sécurité doit ouvrir les contacts des relais correspondant à chaque canal.

Le système a un comportement dangereux (sortie alimentée en cas de sollicitation de la fonction de sécurité) si une défaillance dangereuse non détectée survient dans les deux canaux. Par contre, le comportement reste sûr si un seul canal est défaillant, la voie défaillante ayant été passivée par la voie encore en fonctionnement. C'est donc bien une architecture de type 1oo2.

**Cette configuration n'est utilisable en sécurité des machines que si on est certain que les modules de diagnostics détectent toutes les fautes de chaque canal.** La disponibilité est améliorée par rapport à une structure 1oo2.

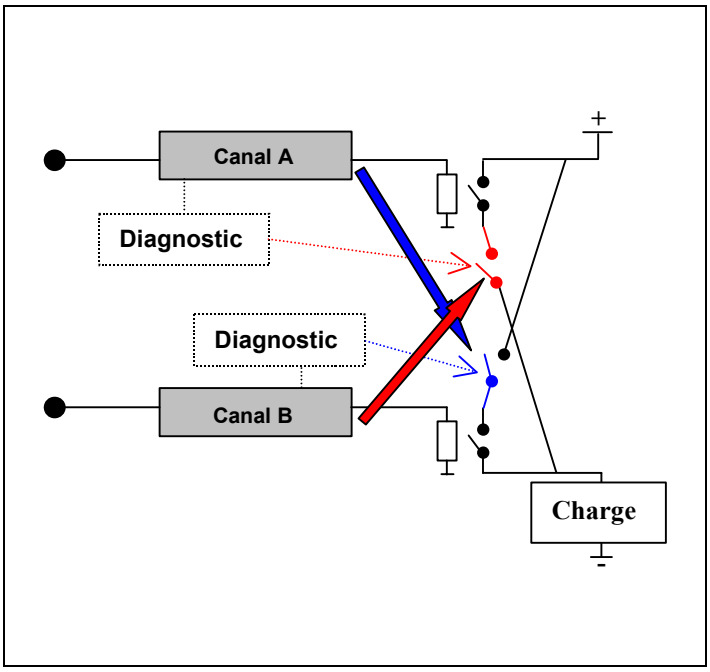



**Figure 4-11 : Schéma électrique de principe d'une architecture 1oo2D**

| | | | |
|---|---|---|---|
| **Fonctionnement normal** | | | |
| Fonction de sécurité non activée | Les contacts des relais de sortie des deux canaux sont fermés. De même pour les contacts des relais de diagnostics. | | |
| Fonction de sécurité activée | Les contacts des relais de sortie des deux canaux sont ouverts. Les contacts des relais correspondants aux modules de diagnostics sont ouverts par les canaux opposés . | | |
| **Un canal défaillant** | | | |
| Contact canal i bloqué à l'état ouvert | Défaillance détectée : Le relais de diagnostic du canal défaillant est ouvert par le module de diagnostic de ce canal (ainsi que par le canal opposé non défaillant). | Sûr | Disp. |
| | Défaillance non détectée : Le relais de sortie du canal défaillant est ouvert, son relais de diagnostic est ouvert par le canal opposé non défaillant, qui continue à accomplir la mission. | Sûr | Disp |
| Contact canal i bloqué à l'état fermé | Défaillance détectée : Le relais de diagnostic du canal défaillant est ouvert par le module de diagnostic de ce canal (ainsi que par le canal opposé non défaillant). | Sûr | Disp. |
| | Défaillance non détectée : Le relais de sortie du canal défaillant est à l'état fermé. Son relais de diagnostic est ouvert par le canal opposé non défaillant, qui continue à accomplir la mission. | Sûr | Disp. |
| **Deux canaux défaillants** | | | |
| Contacts des deux canaux ouverts | On est toujours dans un état de sécurité (sortie non alimentée). | Sûr | ~~Disp~~. |

| | | | |
|---|---|---|---|
| Un des canaux a son contact bloqué à l'état fermé | Défaillance détectée : Le relais de diagnostic du canal défaillant est ouvert par le module de diagnostic. L'autre canal a son relais de sortie ouvert. La charge n'est pas alimentée, donc le système est dans un état sûr. | Sûr | ~~Disp~~. |
| | Défaillance non détectée : Le contact du relais de sortie est fermé et le relais de diagnostic du canal défaillant l'est aussi. La charge est toujours alimentée. | ~~Sûr~~ | ~~Disp~~. |
| Les deux contacts de sortie sont fermés. | Défaillances détectées : Les relais de diagnostic des deux canaux sont ouverts par les modules de diagnostics. La charge n'est pas alimentée. | Sûr | ~~Disp~~. |
| | Au moins une défaillance non détectée : Pour au moins un canal, le contact du relais de sortie est fermé et son relais de diagnostic l'est aussi. La charge est toujours alimentée. | ~~Sûr~~ | ~~Disp~~. |

**Tableau 4-6 : Principe de fonctionnement d'une architecture 1oo2D**

**Calcul de la probabilité de défaillance dangereuse sans tenir compte des CMF**

L'hypothèse que le temps de détection des tests de diagnostics est très inférieur au temps de réparation du système (de l'ordre de 1 à 5 secondes) a pour effet de supprimer les termes en Tr des équations donnant la PFD.

- Cas de deux canaux homogènes :

$$PFD(Ti) = (\lambda^{DU} \cdot Ti)^2$$

Remarque : La PFD est meilleure que celle d'une structure 1oo2 puisque les défaillances non détectées ne sont plus considérées (par hypothèse, les défaillances détectées le sont instantanément et de ce fait, ne créent aucun danger potentiel).

- Cas de deux canaux strictement hétérogènes :

$$PFD(Ti) = (\lambda_A^{DU} \cdot Ti) \cdot (\lambda_B^{DU} \cdot Ti)$$

**Calcul de la probabilité de défaillance dangereuse en tenant compte des CMFs**

Dans ce cas, on répartit les défaillances entre défaillances de mode commun $\lambda^{DCCF}$ et défaillances indépendantes $\lambda^{DI}$.

Si des tests de diagnostics sont exécutés :

$$PFD(Ti) = \lambda^{DUCCF} \cdot Ti + (\lambda^{DUI} \cdot Ti)^2$$

### 4.3.7. 1oo2D avec comparaison croisée

La base de cette structure est identique à celle de la structure 1oo2D. La fonction de sécurité est exécutée dès qu'un canal en fait la demande. La seule modification consiste à comparer périodiquement les données issues des deux canaux de traitement. Deux moyens de détection des défaillances existent donc : les tests de diagnostic (comme pour la structure 1oo2D) et la comparaison.

L'avantage de la comparaison de données est double : elle améliore le taux de détection des fautes et elle permet de passer en position de repli avant même que la faute ait une conséquence sur la sortie.

- Cas de deux canaux homogènes

$$PFD(Ti) = (\lambda^{DU} \cdot Ti)^2$$

L'égalité reste inchangée, mais le taux des défaillances dangereuses non détectées diminue puisqu'il y a un niveau de détection supplémentaire. Le coefficient C devient $C_{comp}$ et doit tenir compte des deux niveaux de détection.

$$\lambda^{DU} = (1- C_{comp}).\lambda^{D}_{canal}$$

**Important** : Le problème consiste à évaluer l'apport de la comparaison de données sur ce coefficient.

- Cas de deux canaux strictement hétérogènes

$$PFD(Ti) = (\lambda_A^{DU} \cdot Ti) \cdot (\lambda_B^{DU} \cdot Ti)$$

**Calcul de la probabilité de défaillance dangereuse en tenant compte des CMFs**

Dans ce cas, la détection par la comparaison n'influe que sur les défaillances indépendantes. On a toujours :

$$PFD(Ti) = \lambda^{DUCCF} \cdot Ti + (\lambda^{DUI} \cdot Ti)^2$$

mais avec :

$$\lambda^{DUCCF} = (1- C) \, . \, \beta \, . \, \lambda^{D}$$

$$\lambda^{DUI} = (1- C_{comp}) \, . \, (1-\beta) \, . \, \lambda^{D}$$

## 4.4. SYNTHÈSE DES PFD EN FONCTION DE L'ARCHITECTURE

### 4.4.1. Choix d'une architecture pour des applications de sécurité

| PFD (Ti) | Canaux homogènes | Canaux strictement hétérogènes | Canaux homogènes avec facteur β |
|---|---|---|---|
| 1oo1 | $\lambda^{DU} \cdot Ti + \lambda^{DD} \cdot Tr$ | | |
| 1oo1D | $\lambda^{DU} \cdot Ti$ | | |
| 1oo2 | $(\lambda^{DU} \cdot Ti + \lambda^{DD} \cdot Tr)^2$ | $(\lambda_A^{DU} \cdot Ti + \lambda_A^{DD} \cdot Tr) \cdot (\lambda_B^{DU} \cdot Ti + \lambda_B^{DD} \cdot Tr)$ | $(\lambda^{DUI} \cdot Ti + \lambda^{DDI} \cdot Tr)^2 + \lambda^{DUCCF} \cdot Ti + \lambda^{DDCCF} \cdot Tr$ |
| 1oo2D | $(\lambda^{DU} \cdot Ti)^2$ | $(\lambda_A^{DU} \cdot Ti) \cdot (\lambda_B^{DU} \cdot Ti)$ | $\lambda^{DUCCF} \cdot Ti + (\lambda^{DUI} \cdot Ti)^2$ |
| 2oo2 | $2 \cdot \left[\lambda^{DD} \cdot Tr + \lambda^{DU} \cdot Ti\right]$ | $(\lambda_A^{DU} \cdot Ti + \lambda_A^{DD} \cdot Tr) + (\lambda_B^{DU} \cdot Ti + \lambda_B^{DD} \cdot Tr)$ | $\lambda^{DDCCF} \cdot Tr + \lambda^{DUCCF} \cdot Ti + 2 \cdot \lambda^{DDI} \cdot Tr + 2 \cdot \lambda^{DUI} \cdot Ti$ |
| 2oo2D | $2 \cdot \left[\lambda^{DU} \cdot Ti\right]$ | $(\lambda_A^{DU} \cdot Ti) + (\lambda_B^{DU} \cdot Ti)$ | $2 \cdot \lambda^{DUI} \cdot Ti + \lambda^{DUCCF} \cdot Ti$ |

**Tableau 4-7 : Synthèse des PFD en fonction de l'architecture**

De façon générique, les probabilités de défaillances dangereuses des structures 1oo2 (avec ou sans l'indice D) sont de la forme $A^2$ alors que les probabilités de défaillances des structures 2oo2 (avec ou sans l'indice D) sont de la forme 2.A. Ce constat montre que, pour des valeurs réalistes des paramètres de calculs (Ti, taux de défaillances, taux de couverture, facteur β), **les structures 1oo2 et 1oo2D seront les structures à retenir lorsque la sécurité est le paramètre de Sûreté de Fonctionnement à privilégier.**

On constate que les stuctures de type 2oo2 (avec ou sans indice D) ont les moins bons résultats. Ceci est dû au fait que la disponibilité a été privilégiée au détriment d'un comportement sûr en présence de fautes.

### 4.4.2. Application numérique

Les tableaux suivants donnent, pour deux temps de mission différents, donc deux périodicités de "Proof tests" différentes (Ti=10 ans et Ti=1 an), les valeurs de PFD pour les différentes architectures étudiées précédemment.

On peut considérer que la durée de 10 ans correspond à la durée de vie d'un système pour lequel on n'exécuterait aucun "Proof test". Une durée de mission de 1 an correspond à un système pour lequel on effectuera des "Proof tests" annuels.

Le temps de réparation Tr sera pris égal à 8 heures.

Sauf indication contraire, le taux de couverture des tests de diagnostic est de 50%.

Les taux de défaillances considérés pour les canaux homogènes sont de $4.10^{-6}$,. Cette valeur a été déduite des données fournies par la société SIEMENS pour l'automate S7-216 (Voir annexe 8). La valeur de $1.10^{-6}$ pour le second canal hétérogène est arbitraire.

Les probabilités de défaillances dangereuses seront données en défaillances par heures pour être plus aisément comparables aux valeurs de SIL requises par la norme CEI 61508.

| PFD(Ti) Ti=10 ans | Homogène | Hétérogène β=0 | Homogène β=0,02 |
|---|---|---|---|
| | $\lambda=4.10^{-6}$ | $\lambda_A=4.10^{-6}$ $\lambda_B=1.10^{-6}$ | $\lambda=4.10^{-6}$ |
| 1oo1 | 1,00E-06 | | |
| 1oo1D | 1,00E-06 | | |
| 1oo2 | 8,76E-08 | 2,19E-08 | 1,04E-07 |
| 1oo2D | 8,76E-08 | 2,19E-08 | 1,04E-07 |
| 1oo2D Comp (C=0,90) | 3,50E-09 | 8,76E-10 | 7,37E-09 |
| 2oo2 | 2,00E-06 | 1,25E-06 | 1,98E-06 |
| 2oo2D | 2,00E-06 | 1,25E-06 | 1,98E-06 |

**Tableau 4-8 : PFD (architecture) pour Ti=10 ans**

| PFD(Ti) Ti=1 an | Homogène | Hétérogène β=0 | Homogène β=0,02 |
|---|---|---|---|
| | $\lambda=4.10^{-6}$ | $\lambda_A=4.10^{-6}$ $\lambda_B=1.10^{-6}$ | $\lambda=4.10^{-6}$ |
| 1oo1 | 1,00E-06 | | |
| 1oo1D | 1,00E-06 | | |
| 1oo2 | 8,78E-09 | 2,19E-09 | 2,84E-08 |
| 1oo2D | 8,76E-09 | 2,19E-09 | 2,84E-08 |
| 1oo2D Comp (C=0,90) | 3,50E-10 | 8,76E-11 | 4,34E-09 |
| 2oo2 | 2,00E-06 | 1,25E-06 | 1,98E-06 |
| 2oo2D | 2,00E-06 | 1,25E-06 | 1,98E-06 |

**Tableau 4-9 : PFD (architecture) pour Ti=1 an**

Étant donné la valeur considérée pour Tr (8 heures) par rapport à la valeur de Ti (8760 ou 87600 heures), les termes en $\lambda^{DU} \cdot Ti$ sont prédominants par rapport aux termes en $\lambda^{DD} \cdot Tr$, quelle que soit la structure choisie. Ce constat justifie :

- La différence insensible entre les structures avec et sans indice D, c'est à dire avec ou sans action du signal de détection de fautes sur la mise en repli du système. Malgré cette faible différence, on privilégiera les structures indicées D dans le but de passer en sécurité dès la détection d'une défaillance.

- Le rapport de la forme $(\lambda^{DU} \cdot Ti)/2$ pour les valeurs des probabilités de défaillances dangereuses des structures 1oo2 et 2oo2 (avec ou sans indice D).

Les tableaux montrent l'influence des défaillances de mode commun sur la probabilité de défaillance dangereuse. Les résultats obtenus varient dans un rapport allant de 1.2 à plus de 10. L'influence est d'autant plus sensible que la valeur de Ti est faible.

Les résultats obtenus en considérant deux canaux totalement hétérogènes (β=0) à taux de défaillances différents sont trop influencés par la valeur du taux de défaillance ($1.10^{-6}$) retenue pour le deuxième canal pour être vraiment significatifs.

### 4.4.3. Étude du compromis Taux de couverture / facteur β

Ce paragraphe a pour objet d'étudier, pour la structure 1oo2D (optimale pour la sécurité), l'influence des paramètres "Taux de couverture des tests" et "facteur β" sur la probabilité de défaillance dangereuse (exprimée dans les tableaux en défaillances par heure).

| Ti=8760 h | λ=4.10$^{-6}$ | Architecture 1oo2D | | |
|---|---|---|---|---|
| | PFD | PFD | PFD | PFD |
| Beta | C =0,1 | C =0,2 | C =0,5 | C =0,9 |
| 0,01 | 4,58E-08 | 3,80E-08 | 1,86E-08 | 2,34E-09 |
| 0,02 | 6,33E-08 | 4,75E-08 | 2,84E-08 | 4,34E-09 |
| 0,03 | 8,07E-08 | 6,91E-08 | 3,82E-08 | 6,33E-09 |
| 0,04 | 9,82E-08 | 8,47E-08 | 4,81E-08 | 8,32E-09 |
| 0,05 | 1,16E-07 | 1,00E-07 | 5,79E-08 | 1,03E-08 |
| 0,06 | 1,33E-07 | 1,16E-07 | 6,77E-08 | 1,23E-08 |
| 0,07 | 1,51E-07 | 1,31E-07 | 7,76E-08 | 1,43E-08 |
| 0,08 | 1,68E-07 | 1,47E-07 | 8,74E-08 | 1,63E-08 |
| 0,09 | 1,86E-07 | 1,63E-07 | 9,73E-08 | 1,83E-08 |
| 0,1 | 2,03E-07 | 1,78E-07 | 1,07E-07 | 2,03E-08 |

**Tableau 4-10 : PFD(β) pour différents taux de couverture**

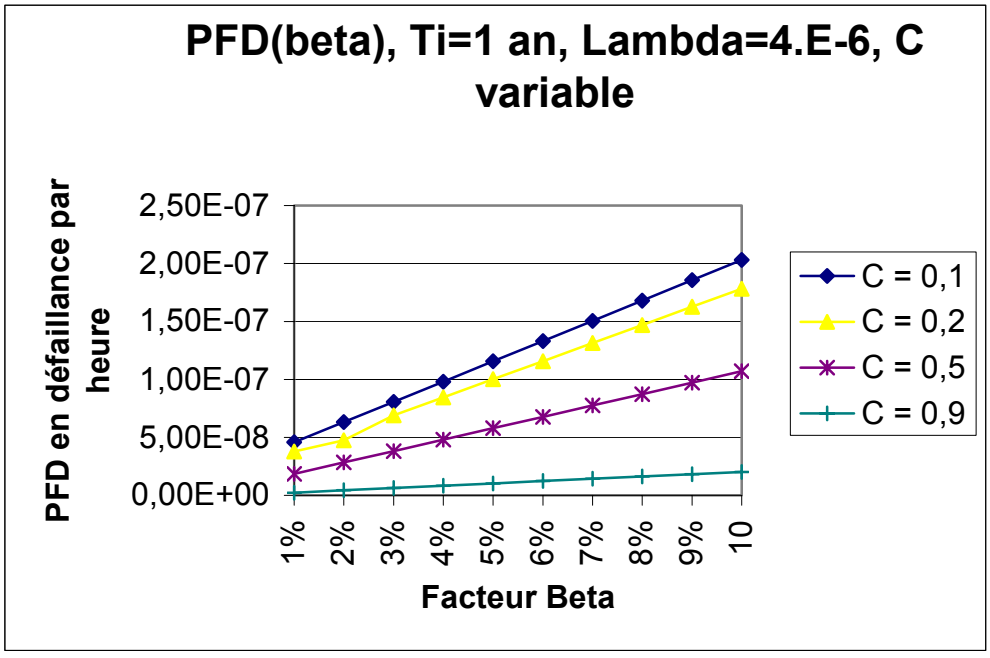



**Figure 4-12 : Courbes PFD(β) pour différents taux de couverture**

| | |
|---|---|
| | Les prescriptions en terme de PFD sont satisfaites pour le SIL 2, c'est à dire ≥ $10^{-7}$ et < $10^{-6}$ |
| | Les prescriptions en terme de PFD sont satisfaites pour le SIL 3, c'est à dire ≥ $10^{-8}$ et < $10^{-7}$ |
| | Les prescriptions en terme de PFD sont satisfaites pour le SIL 4, c'est à dire ≥ $10^{-9}$ et < $10^{-8}$ |

| Ti=8760 h | λ=4.10$^{-6}$ | Architecture 1oo2D | | |
|---|---|---|---|---|
| | PFD | PFD | PFD | PFD |
| C | Beta=0,01 | Beta=0,02 | Beta=0,05 | Beta=0,1 |
| 0 | 5,43E-08 | 7,37E-08 | 1,32E-07 | 2,28E-07 |
| 0,1 | 4,58E-08 | 6,33E-08 | 1,16E-07 | 2,03E-07 |
| 0,2 | 3,80E-08 | 5,35E-08 | 1,00E-07 | 1,78E-07 |
| 0,3 | 3,08E-08 | 4,45E-08 | 8,55E-08 | 1,54E-07 |
| 0,4 | 2,44E-08 | 3,61E-08 | 7,14E-08 | 1,30E-07 |
| 0,5 | 1,86E-08 | 2,84E-08 | 5,79E-08 | 1,07E-07 |
| 0,6 | 1,35E-08 | 2,14E-08 | 4,51E-08 | 8,45E-08 |
| 0,7 | 9,09E-09 | 1,50E-08 | 3,28E-08 | 6,26E-08 |
| 0,8 | 5,37E-09 | 9,35E-09 | 2,13E-08 | 4,11E-08 |
| 0,9 | 2,08E-09 | 4,34E-09 | 1,03E-08 | 2,03E-08 |
| 0,99 | 2,03E-10 | 4,03E-10 | 1,00E-09 | 2,00E-09 |

**Tableau 4-11 : PFD(C) pour différents β**

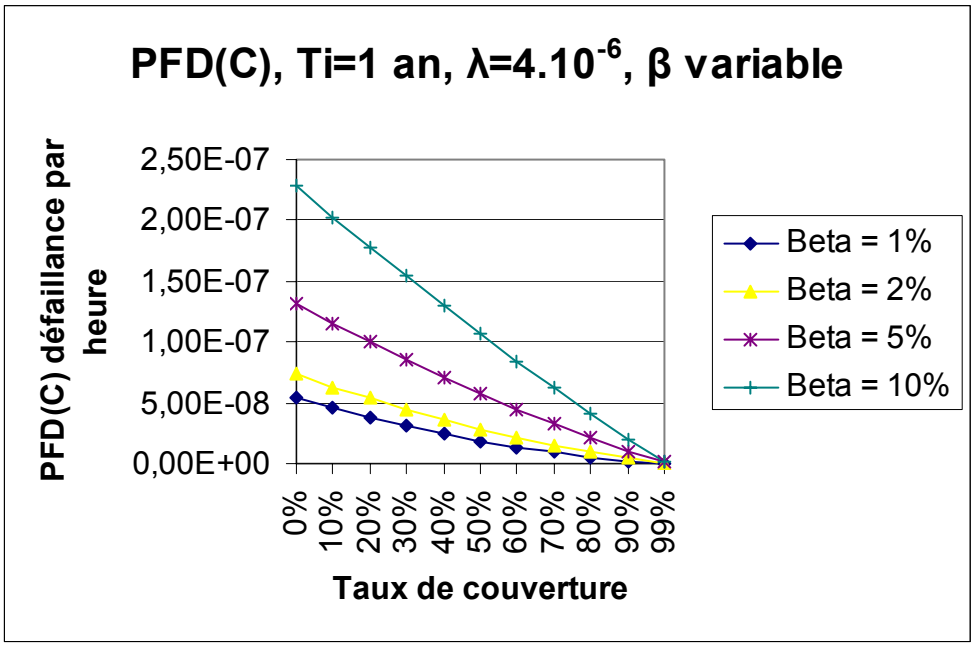



**Figure 4-13 : Courbes PFD(C) pour différents β**

Plusieurs possibilités existent pour obtenir une probabilité de défaillance dangereuse satisfaisant à un SIL (niveau d'intégrité) donné.

Exemple du SIL 2 : A Ti et Taux de défaillance donnés pour un canal, les couples suivants (au pire) peuvent convenir :

| Beta | 5 % | | | 6 % | | 7 % | | 8 % | | 9 % | | 10 % | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C en % | 0 | 10 | 20 | 0 | 10 | 0 | 10 | 0 | 10 | 0 | 10 | 0 | 10 | 20 | 30 | 40 | 50 |

Ces tableaux montrent que :

- Pour une structure 1oo2D avec Ti=1 an, Tr=8 heures et $\lambda = 4.10^{-6}$, il est possible de respecter les prescriptions du SIL 2 en terme de probabilité de défaillances dangereuses avec deux canaux identiques fortement corrélés ($\beta$ =10%) sans aucun mécanisme de détection de fautes (C = 0%).
- Des tests de diagnostic à taux de couverture élevé peuvent augmenter le niveau d'intégrité de sécurité de deux valeurs (SIL 2 à SIL 4). Ceci ne pourra être obtenu qu'en utilisant des canaux développés spécifiquement pour la sécurité, dans lesquels des mécanismes de détection performants auront été implantés.
- L'influence du couple (taux de couverture / facteur $\beta$) sur la probabilité de défaillance dangereuse d'un système est certaine. Le véritable problème consiste alors à déterminer / évaluer les valeurs de ces deux paramètres de la façon la plus précise (la moins imprécise?) possible [Goble et al., 1998], [Goble, 2000].

### 4.4.4. Conclusions

- Le paragraphe 1.1.3 a montré que le niveau de sécurité d'une fonction (en terme de SIL par exemple) est déduit d'une analyse de risque. On recherche ensuite une architecture capable de respecter les prescriptions du SIL en terme de tolérance aux fautes. On calcule ensuite la PFD pour l'architecture, en fonction du lambda de chaque canal, des taux de couverture des autotests, des temps de réparation et de la fréquence des proof tests. On peut ajuster ces différents paramètres pour obtenir le SIL à atteindre.
- Ce constat (il est facile d'obtenir un SIL fixé) montre que les contraintes en terme d'architecture (tolérance aux fautes) ont une grande importance (voir note). Sans cela, on pourrait très bien obtenir une architecture 1oo1 de SIL 4 en exécutant des proof tests suffisamment fréquemment. Une architecture de sécurité devra donc impérativement être analysée pour s'assurer que sa capacité à tolérer les fautes correspond bien aux exigences fixées par le SIL.
- Les résultats du paragraphe "Étude du compromis taux de couverture / facteur $\beta$" relativisent les conclusions pouvant être tirées de l'évaluation quantitative par le calcul de la probabilité de défaillance dangereuse. Ils

confirment la pertinence de la norme CEI 61508 lorsqu'elle propose une démarche globale pour atteindre un niveau d'intégrité de sécurité : management de la sécurité, prescriptions en termes de probabilité de défaillance dangereuse et de tolérance aux fautes, mesures pour l'évitement et la détection des fautes.

- Comme prévu lors de l'analyse qualitative (cf. tableaux), on constate que les structures 1oo2 et 1oo2D sont les mieux adaptées au traitement de signaux de sécurité. Les valeurs obtenues pour les structures 2oo2 et 2oo2D, pour lesquelles la disponibilité est privilégiée, sont telles qu'elles ne pourront pas être employées pour traiter la sécurité.
- La comparaison des données entre les processeurs, par son influence sur le taux de couverture, devrait améliorer de façon sensible la probabilité de défaillance dangereuse de la structure 1oo2D.

Note : Contraintes en terme d'architecture et de taux de couverture des tests

*En plus de la détermination de la probabilité de défaillance dangereuse, le projet de norme CEI 62061 [CEI62061, 2002] impose aux sous-systèmes utilisés pour exécuter une fonction de sécurité de respecter des contraintes en terme d'architecture et de taux de couverture des tests [Olf, 2001]. Celles-ci ont généralement pour effet de diminuer le niveau d'intégrité de sécurité (SIL) de la fonction par rapport à celui obtenu suite aux calculs de probabilité de défaillance. Le tableau suivant synthétise ces contraintes en fonction du SIL à atteindre.*

| *Proportion des défaillances en sécurité* | *Tolérance aux fautes matérielles : **0*** | *Tolérance aux fautes matérielles : **1*** | *Tolérance aux fautes matérielles : **2*** |
|---|---|---|---|
| | *Architecture de type **1oo1*** | *Architecture de type **1oo2 / 2oo2*** | *Architecture de type **1oo3*** |
| *< 60 %* | *Non autorisé* | *SIL1* | *SIL2* |
| *60 % - < 90 %* | *SIL1* | *SIL2* | *SIL3* |
| *90 % - < 99 %* | *SIL2* | *SIL3* | *SIL3*[2] |
| *≥ 99 %* | *SIL3* | *SIL3* | *SIL3*[2] |
| *Une tolérance aux fautes du matériel de N signifie que N+1fautes peuvent provoquer la perte de la fonction de sécurité.* | | | |

***Tableau 4-12 : Contraintes sur le SIL d'un système***

*La proportion de défaillances en sécurité est donnée par la formule suivante :*

$$DéfSécu = \frac{\sum \lambda^{S} + \sum \lambda^{DD}}{\sum \lambda^{D} + \sum \lambda^{DD} + \sum \lambda^{DU}}$$

*Ce rapport est destiné à prendre en compte les taux de couverture de l'ensemble des mécanismes de détection associés à tous les composants ou sous-systèmes de l'architecture considérée. Comme pour le taux de couverture C, sa détermination nécessite en toute rigueur de mener une AMDE au niveau de l'ensemble des composants du système. En cas de difficulté à déterminer la proportion des défaillances sûres et dangereuses d'un composant (par exemple le microprocesseur ou une mémoire), la norme propose de considérer une répartition égale de ces deux modes de défaillances (50% / 50%).*

# 5. MISE EN ŒUVRE D'UNE ARCHITECTURE DIVERSITAIRE

Le chapitre précédent a montré que les architectures 1oo2 étaient les mieux adaptées pour traiter la sécurité d'un système de commande devant passer en position de repli suite à l'apparition d'une défaillance. Nous avons aussi constaté que les probabilités de défaillances dangereuses de ces architectures sont d'autant meilleures que le degré de diversité est grand.

Ces constats nous conduisent à rechercher les conditions pour mettre en œuvre une logique de commande basée sur ce type de structure en utilisant deux canaux hétérogènes. Deux solutions sont envisageables :

- ✓ Concevoir deux canaux en assurant un maximum de diversité. Cette solution a comme avantage principal de maîtriser l'implantation des tests de diagnostics capables de détecter des défaillances avec un taux de couverture supérieur à 90%. Par contre, elle pose le problème de l'indépendance des deux canaux, qui ne pourrait être assurée qu'en sous-traitant la conception et le développement à deux "entités" distinctes.
- ✓ Concevoir une structure à partir de deux canaux standards disponibles sur le marché. Cette solution a les avantages / inconvénients inverses de la solution précédente. L'indépendance est assurée puisque les sociétés ayant développé les deux canaux n'ont eu aucune relation. Par contre, il n'est pas possible d'influencer la conception de chaque canal, en particulier en ce qui concerne le choix des tests de diagnostics (cf. annexe 10).

Ces critères de choix (indépendance / tests de diagnostics) sont à rapprocher de l'influence du taux de couverture C et du facteur β sur la probabilité de défaillance dangereuse en fonction. Le même niveau d'intégrité de sécurité peut en effet être atteint en utilisant des couples (C, facteur β) différents.

Un dernier critère de choix vient du contexte dans lequel les travaux sont effectués : études d'application menées dans le cadre des travaux de l'INRS sur la sécurité des systèmes de commande de machines. Il n'est pas rare de rencontrer des équipements dans lesquels le concepteur a mis à profit la présence de deux automates programmables industriels standards assurant le fonctionnel de l'application pour traiter une ou plusieurs fonctions de sécurité .

Ce dernier critère nous fait retenir la seconde possibilité, qui minimise les défaillances de mode commun, mais qui ne permet aucune action sur le taux de couverture des tests de diagnostic en vue d'améliorer la probabilité de défaillance dangereuse, donc le SIL de l'architecture.

**Cahier des charges :**

L'architecture redondante sera de type 1oo2. Les traitements seront réalisés par deux Automates Programmables Industriels (API) standards [EXERA, 1998] développés et commercialisés par deux sociétés différentes.

Elle devra être capable de traiter les signaux de sécurité d'une machine de type presse à métaux. Ses potentialités capacités devront être similaires à celles de l'automate PSS 3056 commercialisé par la société PILZ : traitement d'au moins 32 entrées et 8 sorties, avec un temps de réponse de l'ordre de 20ms (cf. annexe 9).

Ces capacités seront atteintes grâce à l'utilisation des automates S7-216 de SIEMENS et TSX MICRO DE SCHNEIDER.

Deux problèmes seront principalement abordés : synchronisation des deux canaux et détection des fautes par comparaison des signaux de sortie issus de ces canaux [Blanchet, 2000].

## 5.1. ÉTUDE DE LA SYNCHRONISATION D'UNE STRUCTURE 1OO2 HÉTÉROGÈNE

**Temps de réponse d'un automate programmable**

Il est nécessaire de synchroniser deux automates aux comportements temporels différents. Le temps de réponse d'un automate se décompose de la façon suivante :

Échantillonnage

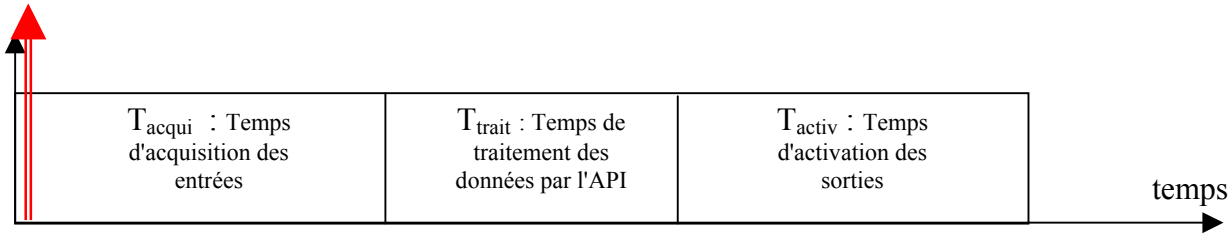



**Figure 5-1 : Séquence de traitement du signal par un automate**

- ✓ $T_{acqui}$ (temps d'acquisition des entrées) : temps entre l'acquisition du phénomène par le capteur et sa lecture effective par l'automate (suite à échantillonnage). Des instants d'échantillonnage différents par les deux automates peuvent engendrer un désaccord entre les valeurs acquises par les automates si le signal a évolué entre temps.
- ✓ $T_{trait}$ (temps de traitement des données par les API) : ce temps correspond au temps de cycle des automates. Les architectures de chaque automate étant complètement différentes, les temps de traitement des données seront différents.
- ✓ $T_{activ}$ (temps d'activation des sorties) : temps entre le chargement de la mémoire image des sorties et l'activation effective de la sortie.

Avec $T_R = T_{acqui} + T_{trait} + T_{activ}$

Remarque :

Certains automates peuvent gérer des tâches en parallèle (acquisition des entrées parallèle, ou traitements parallèles, ou activation des sorties parallèles). De telles architectures contribuent à diminuer le temps réponse, dont l'évaluation exposée représente une majoration.

**Temps de désynchronisation d'une structure hétérogène 1oo2 :**

On suppose les deux automates soumis à un même signal d'entrée de la forme échelon. L'instant d'échantillonnage du signal par l'automate sera représenté par une flèche verticale. Les temps T1 et T2 correspondent aux temps de réponse des automates 1 et 2 (égaux au temps entre deux échantillonnages d'un même automate).

On se place dans le cas critique, où un échelon de tension est observé à l'entrée de chaque automate. Le désynchronisme le plus élevé a donc lieu lorsque l'automate ayant le plus grand temps de réponse échantillonne le signal juste avant l'échelon, et le plus lent, juste après.

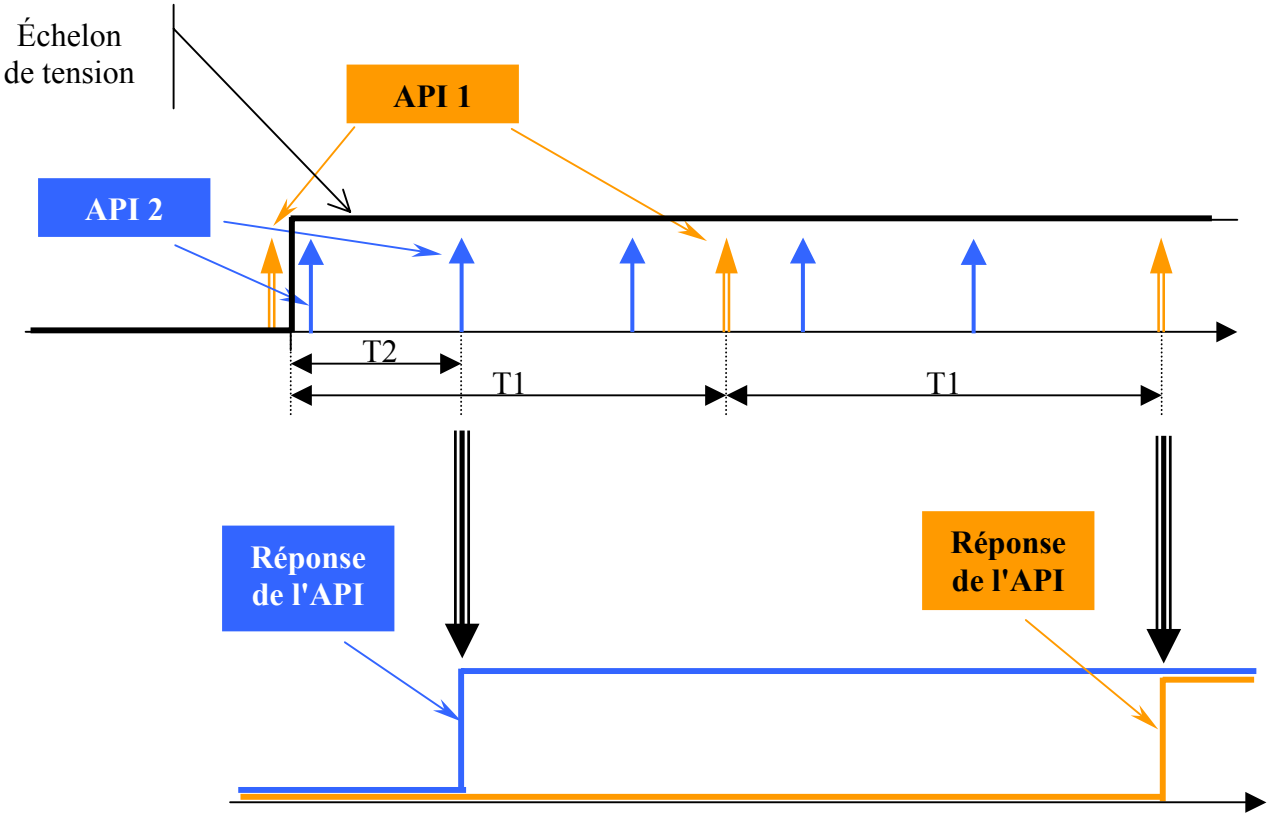



**Figure 5-2 : Prise en compte d'une entrée par 2 automates hétérogènes : Pire cas**

Le temps de réponse de la structure est donc au plus de "2.T1", soit le double du temps de réponse de l'automate le plus lent. De plus, le temps de désynchronisme maximum est de "2.T1-T2".

L'objectif à atteindre étant un temps de réponse de 20ms, chaque automate doit donc avoir un temps de cycle d'au plus 10ms. Dans ce cas, la désynchronisation maximale des 2 automates sera au plus de 20ms.

La désynchronisation inhérente à cette structure hétérogène peut être traitée à différents niveaux :

- ✓ entrée : on s'assure que la valeur du signal qui sera acquise par chaque unité de traitement est identique, et qu'elle ne sera traitée qu'à partir du moment où l'unité de traitement de l'automate le plus lent devient disponible.
- ✓ traitement : des points de rendez-vous sont fixés entre l'exécution des 2 programmes afin d'attendre à chaque étape le plus lent. Ceci peut être réalisé à l'aide d'un bus de terrain (AS-interface, CAN, Profibus DP, …) ou d'une liaison série de type RS 232. Cette solution suppose que les 2 automates choisis possèdent un des moyens de communication en commun.

- ✓ sortie : chaque automate traite les signaux à son rythme. Une fonction "ET" acceptant les desynchronismes est réalisée entre les 2 sorties.

Dans les 2 premiers cas, le désynchronisme entre les sorties des automates est certes diminué, mais non éliminé ; de plus, il faudra intégrer dans le temps de réponse global le temps nécessaire à la communication pour établir le synchronisme.

Le paragraphe suivant étudie un comparateur pour synchroniser le signal de sortie issu des deux voies de traitement.

## *5.2. DISPOSITIF DE COMPARAISON*

**Spécification du comparateur**

Pour chaque information de sortie, le comparateur devra répondre aux exigences fonctionnelles suivantes :

- ✓ Détection sûre d'un désaccord sur les signaux de sortie des API.
- ✓ Passage et maintien dans un état "sécurité".
- ✓ Tolérance à des désynchronismes faibles.

La détection d'un désynchronisme / désaccord peut se faire de deux façons différentes :

- ✓ Temps $t_1$ s'écoulant entre l'occurrence de 2 changements d'états (montants ou descendants) des signaux émis par les deux canaux. On considère qu'un désynchronisme est présent si le temps $t_1$ est supérieur à une valeur de référence égale au temps de désynchronisation maximal tolérable. Dans ce cas, un parasite sur une des voies sera assimilé à une désynchronisation, le dispositif de comparaison ne détectant pas la présence de ce parasite sur l'autre voie. Ce type de détection peut altérer la disponibilité de la structure.
- ✓ Temps $t_2$ pendant lequel les signaux sont en désaccord. On mesure en temps réel la durée de désaccord des signaux de sortie des 2 voies. Une valeur supérieure à une référence prédéterminée sera caractéristique d'un désynchronisme. Cette détection n'est opérationnelle que si les signaux observés sont suffisamment longs pour qu'un fonctionnement normal des automates engendre un chevauchement des signaux de sortie de ceux-ci. Ceci impose donc que la durée du signal d'entrée des API doit être plus grande que le désynchronisme maximal, à savoir "2.T1-T2". C'est ce mode de détection qui sera retenu.

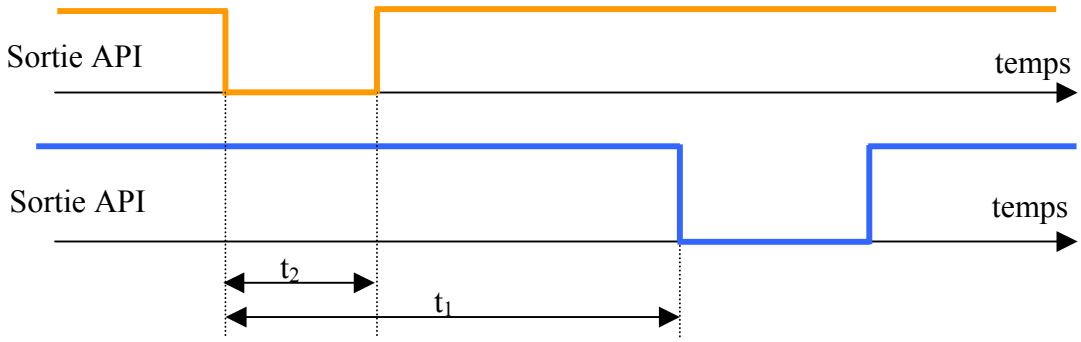



**Figure 5-3 : Chronogrammes des désynchronismes**

**Analyse fonctionnelle du comparateur :**

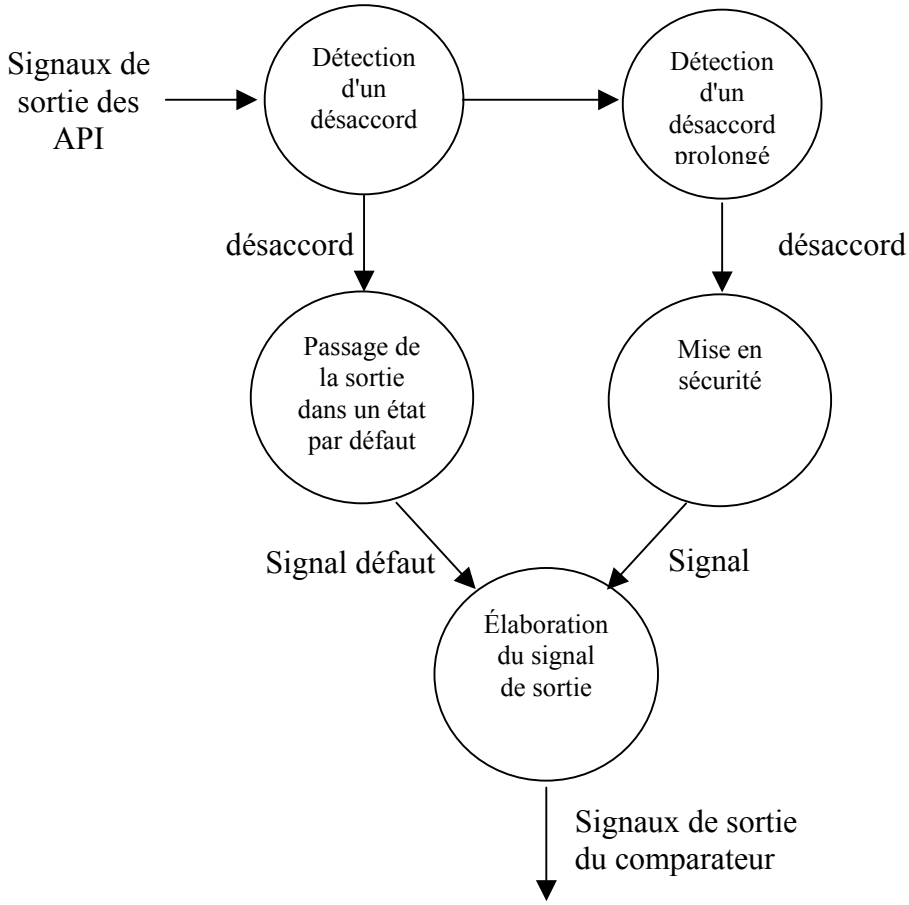


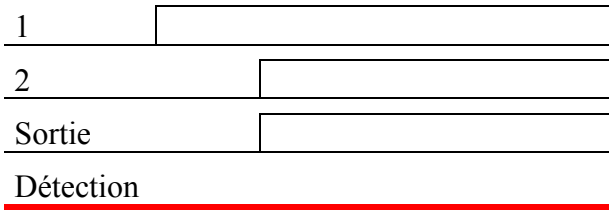
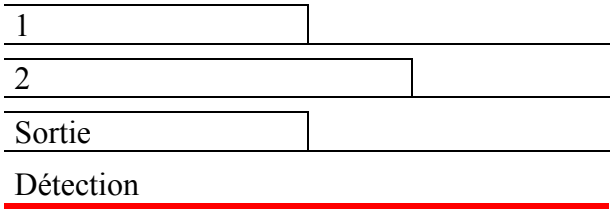
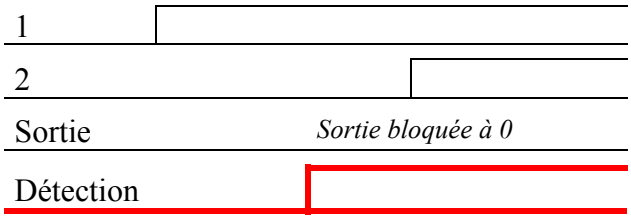
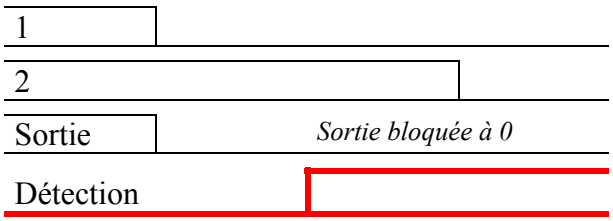
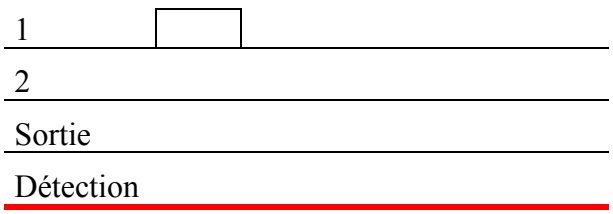

**Figure 5-4 : Représentation schématique du fonctionnement du comparateur**

L'état de la sortie par défaut suite à la détection d'un désaccord tolérable peut être :

- ✓ l'état des sorties qui précède le désaccord. Cette solution nécessite une mémorisation de l'état précédent le désaccord.

- ✓ un état fixé au préalable ("1" si l'on souhaite privilégier la disponibilité, ou "0" si l'on souhaite privilégier la sécurité). Compte tenu du domaine d'application, c'est cette solution qui sera retenue (avec blocage de la sortie à '0').

**Spécifications temporelles du comparateur**

| Caractéristiques temporelles des entrées (1,2) et de la sortie S | Description |
|---|---|
|  | Désynchronisation faible à la montée<br>***FONCTIONNEMENT NORMAL*** |
|  | Désynchronisation faible à la descente<br>***FONCTIONNEMENT NORMAL*** |
|  | Désynchronisation importante à la montée<br>***DÉTECTION D'UNE DÉFAILLANCE*** |
|  | Désynchronisation importante à la descente<br>***DÉTECTION D'UNE DÉFAILLANCE*** |
|  | Créneau à "1" d'une durée plus faible que le désynchronisme maximal autorisé<br>***FONCTIONNEMENT NORMAL*** |

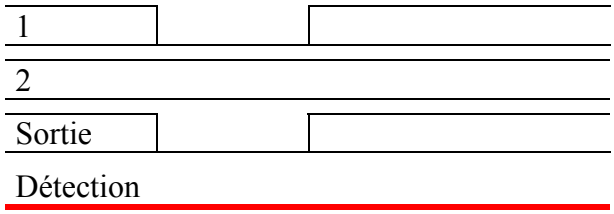
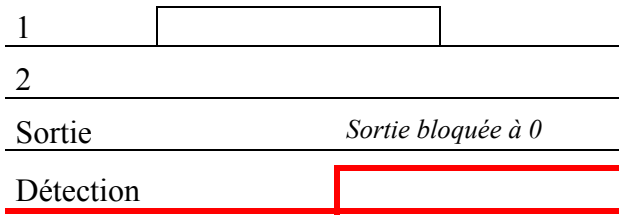
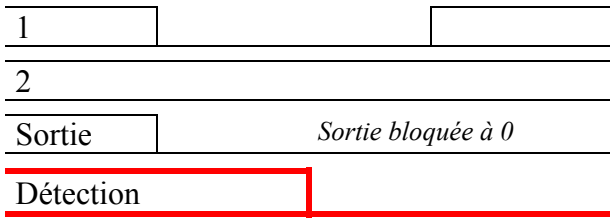
| | |
|---|---|
|  | Créneau à "0" d'une durée plus faible que le désynchronisme maximal autorisé<br><br>***FONCTIONNEMENT NORMAL*** |
|  | Créneau à "1" d'une durée plus grande que le désynchronisme maximal autorisé<br><br>***DÉTECTION D'UNE DÉFAILLANCE*** |
|  | Créneau à "0" d'une durée plus grande que le désynchronisme maximal autorisé<br><br>***DÉTECTION D'UNE DÉFAILLANCE*** |

**Tableau 5-1 : Analyse temporelle du comparateur**

**Choix d'une solution technique**

Compte tenu du contexte (sécurité des machines), plusieurs solutions sont envisageables pour réaliser le comparateur :

- ✓ *Logique câblée uniquement :*

Détection de désaccord à l'aide d'une fonction logique "OU EXCLUSIF". Le maintien en sécurité devra être prévu même si la faute qui a entraîné la mise en sécurité disparaît. Seule une action volontaire de l'opérateur pourra réinitialiser le comparateur.

- ✓ *Logique câblée + automate :*

Cette solution propose de répartir les fonctions du comparateur entre de la logique câblée simple, et la programmation des automates. Par exemple, la détection de désaccord peut être réalisée par un automate, et la mémorisation du passage par un état de défaillance grâce à la logique câblée.

Pour minimiser les coûts, l'utilisation des automates de l'architecture de base s'impose par rapport à l'emploi d'un automate supplémentaire. Cependant, les fonctionnalités de « détection » et de « mémorisation » se rapportant à des défaillances d'un des deux API, il n'est pas envisageable de les traiter par un seul des 2 automates. Ce problème pourrait être résolu en utilisant la redondance déjà existante pour dupliquer ces fonctionnalités. Par

contre, cette solution est coûteuse en câblage, et engendre une programmation supplémentaire des automates.

✓ *Bloc logique standard + logique câblée :*

Les fonctions de sécurité d'une machine (en particulier le contrôle de la discordance de deux signaux) sont souvent réalisées par des blocs logiques certifiés. On peut donc envisager d'utiliser un de ces blocs pour réaliser la comparaison des signaux issus des deux voies.

- Bloc d'arrêt d'urgence [Mougeot & Fauconnet, 1993] : Ceux-ci ne présentent que des tolérances au désynchronisme dans des situations d'ouverture de circuit, alors que pour le module de comparaison envisagé, des caractéristiques de désynchronisme sont souhaitées aussi bien à l'ouverture qu'à la fermeture. De plus, aucune garantie n'est donnée quant à la validité de cette caractéristique de désynchronisme, aucune contrainte n'étant spécifiée à ce sujet. Le réarmement de ces blocs présente également des problèmes ; celui-ci peut être choisi « manuel » ou « automatique ». Dans le premier cas, un passage par l'état ouvert suffit à désarmer le dispositif, même si aucun désynchronisme excessif n'a été constaté ; dans le deuxième cas, une situation de désynchronisme excessif à l'ouverture ne sera pas mémorisée car une demande de fermeture remettra le dispositif à l'état fermé automatiquement.

- Module de sécurité pour surveillance des vannes hydrauliques sur presses linéaires : (module de sécurité PREVENTA XPS-PVT). Ce module a l'avantage de considérer des désynchronismes à l'ouverture comme à la fermeture. Par contre, il nécessite l'utilisation d'un module supplémentaire de commande bi-manuelle pour chaque sortie. Cette contrainte fait perdre l'avantage à utiliser des automates pour réduire le coût de la structure proposée ainsi que la complexité et la densité du câblage.

Cette analyse rapide nous conduit à retenir la solution basée exclusivement sur la logique câblée.

**<u>Réalisation du comparateur</u>**

Les bases nécessaires à la mise en œuvre des technologies à relais sont rappelées en annexe.

*Fonction détection de désaccord :*

La détection d'un désaccord en logique correspond à une fonction « OU EXCLUSIF », dont la mise en œuvre à l'aide de relais est représentée à la figure suivante (seuls les contacts de relais sont représentés ici) :

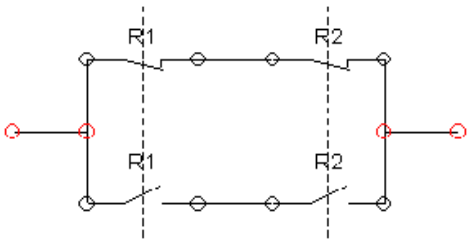



**Figure 5-5 : Fonction "OU exclusif" réalisée à l'aide de 2 relais à contacts guidés R1 et R2**

Si les bobines R1 et R2 sont dans le même état d'excitation, le circuit équivalent est un circuit fermé. Par contre, si les bobines R1 et R2 sont dans des états d'excitation différents, le circuit équivalent est un circuit ouvert.

*Fonction tolérance au désynchronisme :*

La tolérance au désynchronisme consiste à retarder la détection du désaccord. Cette fonction sera réalisée en introduisant un condensateur en parallèle avec les bobines d'excitation des relais de détection des désaccords.

*Fonction de maintien en sécurité :*

Cette fonction assure le maintien dans un état de sécurité dès qu'un désynchronisme excessif est rencontré, même si d'autres défaillances apparaissent. L'état " circuit ouvert " sera choisi comme l'état de sécurité.

Une démarche de sécurité consiste à choisir un état instable pour le fonctionnement normal, et un état stable pour un passage en mode sécurité. Le principe de l'auto-alimentation sera retenu. Il consiste dans un premier temps à alimenter provisoirement (ou amorcer) une bobine de relais qui va commander la fermeture d'un de ses contacts, lui-même intégré dans un circuit d'alimentation de la bobine. L'alimentation provisoire initiale n'est donc plus nécessaire et la bobine peut rester excitée. Par contre, si pour une raison quelconque, l'excitation de la bobine vient à être interrompue, le relais passe dans un état stable : le contact qui avait permis l'auto-alimentation de la bobine s'ouvre. Ainsi, même si ce qui a engendré la perte d'excitation vient à disparaître, le relais restera dans son état désexcité.

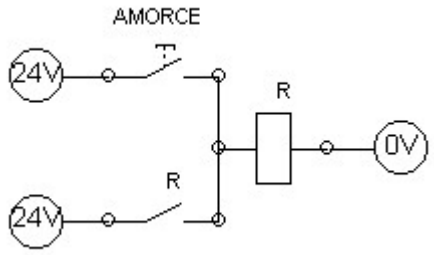



**Figure 5-6 : Maintien en sécurité**

*Sortie du dispositif*

La sortie du dispositif sera réalisée par la mise en série :

- Des contacts des relais de sortie commandés par les deux voies de traitement ;
- Des contacts des relais effectuant la détection des désaccords avec maintien en position de sécurité.

*Contrôle du dispositif :*

L'objectif est de détecter les fautes des deux canaux de la structure de façon sûre. Pour cela, il est nécessaire de mettre en place des contrôles destinés à détecter les collages des contacts des relais utilisés. Ces relais possédant des contacts aux états contraires, la soudure d'un contact pourra être contrôlée à l'aide d'un contact d'état contraire du même relais. Par exemple, dans le cas précédent de l'auto-alimentation, le contact R utilisé peut être contrôlé en assurant qu'un amorçage ne sera possible que si ce contact est initialement ouvert.

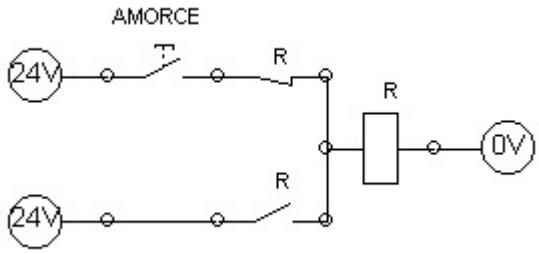



**Figure 5-7 : Contrôle du collage d'un contact de relais**

## 5.3. SCHÉMA D'UNE ARCHITECTURE 1oo2 À BASE DE DEUX API STANDARDS

La figure suivante donne le schéma complet d'une architecture 1oo2 de base utilisant deux API standards.

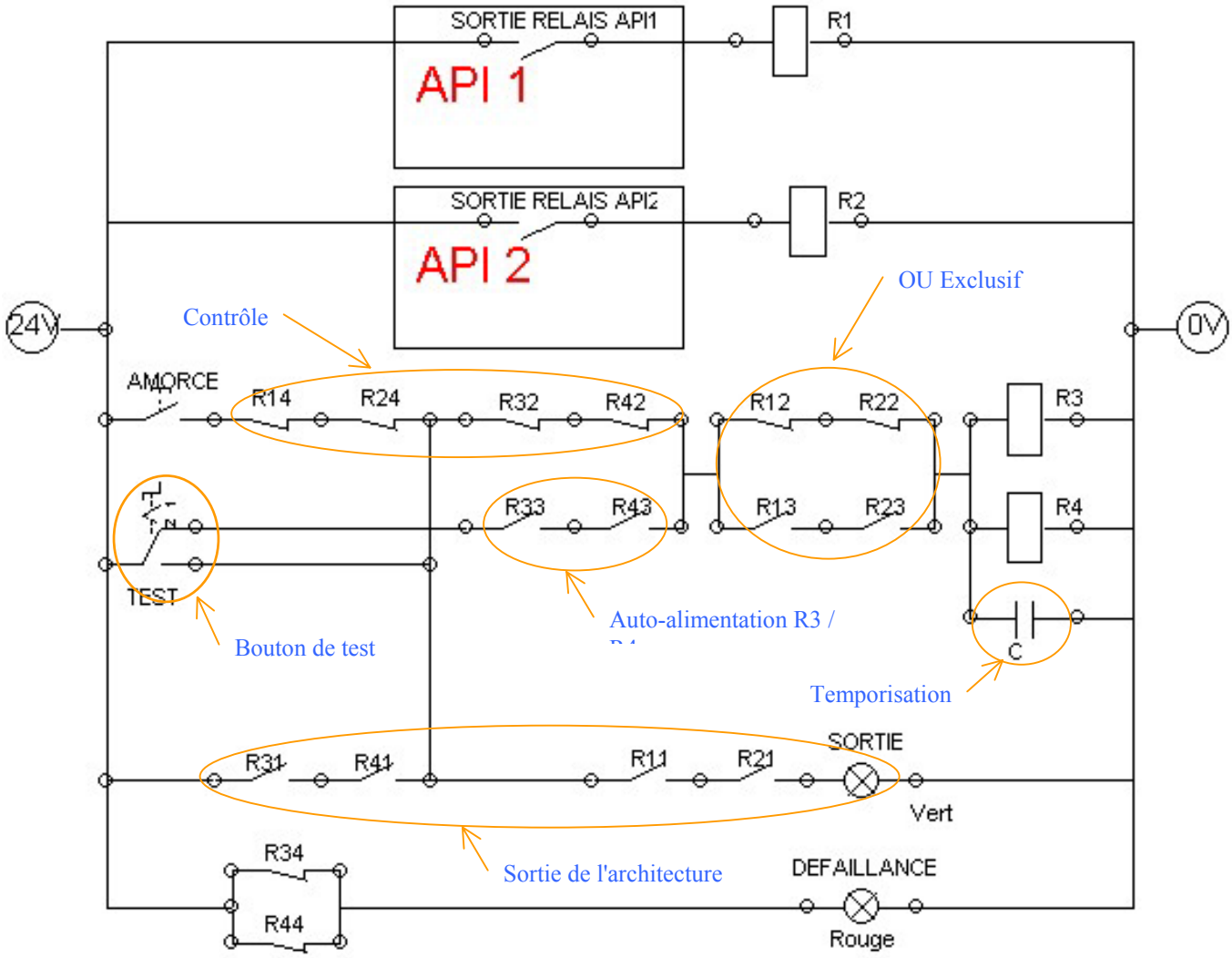



**Figure 5-8 : Schématisation de l'architecture finale**

Remarques :

- Ce montage permet de ne pas passer en position de sécurité suite à l'apparition d'une impulsion parasite sur une des 2 voies de traitement. Par contre, cette impulsion sera transmise en sortie.
- Le schéma proposé est un intermédiaire entre les structures 1oo2 et 1oo2D analysées précédemment : la détection des fautes de chaque canal agit sur le signal de sortie mais il n'y a pas d'action croisée d'un canal sur l'autre.
- Dans le cas de plusieurs couples de sorties d'automates, cette architecture peut être répétée avec comme parties communes le bouton d'amorce et le bouton de test

**Analyse en présence de défaillances :**

Le comportement de l'architecture de la figure précédente a été analysé en présence de défaillances de composants à l'aide de l'outil de simulation informatique Automation Studio.

Les défaillances considérées pour cette architecture sont les soudures individuelles de chacun des contacts de relais, les blocages de l'armature d'un relais, la coupure d'une bobine de relais, la coupure d'une connexion à un contact de relais, les pannes d'alimentation, et enfin la défaillance d'un des deux automates (fermeture intempestive d'un contact de sortie). Par contre, les défaillances des boutons poussoir de test et d'amorçage ne sont pas envisagées, ainsi que deux défaillances quelconques simultanées. La notion de simultanéité considérée correspond dans ce cas à un temps inférieur au temps de désynchronisme maximum autorisé. La fréquence des tests est supposée suffisante pour qu'au moins un test soit réalisé avant l'occurrence d'une seconde soudure de contact.

La simulation des fautes listées précédemment n'a mis aucun comportement dangereux en évidence. L'architecture envisagée satisfait donc la première des prescriptions des catégories 3 et 4 de la norme EN 954 relative au comportement sûr en présence d'une faute unique. Il est à noter que cette conclusion partielle n'a de sens qu'après avoir précisé les défaillances effectivement considérées, et celles qui sont ignorées. Ainsi, la conclusion devrait plutôt être : aucune des défaillances simulées ne s'est avérée dangereuse.

## *5.4. DISCUSSION*

La maquette de faisabilité montre qu'une structure de type 1oo2 à base de deux Automates Programmables Industriels standards est réaliste. Plusieurs points mériteraient d'être approfondis.

### 5.4.1. Évaluation réelle des taux de couverture des tests de diagnostic de chaque API

L'annexe recense les différents moyens de détection de fautes implantés sur chacun des deux APIs constituant la maquette. Ces moyens ont été fournis par les constructeurs. Aucune précision n'a été fournie quant aux fautes effectivement détectées, à la criticité de ces fautes, à leur traitement (par le système ou par

l'utilisateur) et à l'action qui est entreprise suite à ce traitement (signalisation, blocage).

On constate une grande variété des mécanismes associés à chaque API. D'après les indications fournies par les constructeurs, il semblerait que les tests de diagnostic soient plus complets pour l'automate S7-200 que pour le Micro.

Aucun des constructeurs n'a fourni d'informations sur la périodicité de ces tests ainsi que sur les taux de couverture associés. Cette absence de réponse s'explique par le fait que les automates en question sont des produits standards développés en suivant des considérations minimales de sécurité. Les tests de diagnostic implantés sont essentiellement destinés au diagnostic fonctionnel et à la maintenance.

Les informations fournies ne permettent pas d'évaluer correctement le taux de couverture de chacun des moyens de détection. Une approche au niveau global de l'automate pourrait être obtenue :

- ✓ Par analyse de type Analyse des Modes de Défaillances et de leurs Effets.
- ✓ Par injection de fautes physiques au sein de l'automate (collages et courts-circuits sur les broches de composants, injection via la mémoire programme).

Pour obtenir des résultats significatifs, ces techniques ne peuvent être utilisées qu'en partenariat étroit avec concepteurs des automates. De plus, ils posent la même question : quelles sont les fautes à considérer pour l'analyse ou l'injection de fautes?

### 5.4.2. Échange de données entre les API

La détection des fautes de la structure étudiée au paragraphe précédent est réalisée par la comparaison des données de sortie fournies par chaque automate. C'est donc une détection peu précise puisqu'elle s'attache à comparer l'instant d'apparition (au désynchronisme toléré près) et l'état de deux signaux logiques. Ne seront donc détectées que les fautes qui ont une influence sur ces informations. De plus, aucun diagnostic n'est possible quant à la faute à l'origine de la défaillance.

Un des moyens pour améliorer la détection des fautes en temps réel consisterait à implanter un échange de données entre les deux automates. La comparaison périodique des données intermédiaires issues des deux voies de traitement permettrait une détection amont plus fine des fautes susceptibles d'affecter chaque automate, en particulier les fautes qui n'ont pas d'influence sur la sortie des automates. Dans la structure actuelle, le cumul de ces fautes (à l'état latent dans le système) pourrait provoquer des dysfonctionnement dangereux du système.

Plusieurs points sont à étudier pour mettre en œuvre un tel échange de données :

- ✓ Quel est le support physique à utiliser : bus standard disponible sur les deux automates, liaison série, …?
- ✓ Quelles sont les informations à échanger pour optimiser la détection des fautes : Tableaux des entrées, des sorties, résultats de calculs ou d'opérations logiques intermédiaires, …?
- ✓ Quelle est la fréquence des comparaisons à effectuer : cycle automate, plusieurs fois par cycle, …?

- ✓ Comment être certain que de nouveaux modes communs ne sont pas introduits? La présence d'un lien entre les deux voies de traitement ne doit pas créer de défaillances de mode commun.
- ✓ Comment évaluer l'apport de l'échange de données en terme de taux de couverture? C'est certainement la question la plus importante, qui pose les mêmes problèmes que ceux évoqués au paragraphe précédent, en particulier en ce qui concerne les fautes à considérer pour l'analyse.

### 5.4.3. Introduction des CMF dans la probabilité de défaillance dangereuse d'une structure hétérogène

Nous avons vu précédemment que la technique d'évaluation de l'effet des défaillances de mode commun sur un système nécessitait la modélisation de ces défaillances. En toute rigueur, aucun des modèles envisagés précédemment n'est utilisable pour une structure hétérogène. Deux voies sont envisageables :

- ✓ Formuler l'hypothèse que la structure proposée est telle que les défaillances de mode commun ont été supprimées (ou suffisamment réduites pour être négligées).
- ✓ Utiliser une des modélisations existante (par exemple la modélisation par facteur $\beta$ pour sa simplicité) et effectuer les calculs prévisionnels dans le pire cas, c'est à dire en prenant le taux de défaillance de l'automate le moins fiable. Cette solution approximative pose néanmoins le problème de l'évaluation correcte du facteur $\beta$.
- ✓ Développer une modélisation adaptée aux structures hétérogènes.

### 5.4.4. Prise en compte des défaillances systématiques

Il n'a été question dans les développements précédents que des défaillances matérielles aléatoires. La prise en compte correcte des défaillances de mode commun nécessite de traiter les défaillances systématiques (matérielles et logicielles) en utilisant des méthodes adaptées (cf. chapitre 3).

Des mesures devront dont être prises pour éviter l'introduction de fautes dans les logiciels applicatifs de chaque automate. La diversification des deux voies évitera les fautes dans les étapes du cycle de vie spécifiques à chaque voie : De la phase de conception de l'architecture jusqu'aux tests d'intégration. Par contre, des mesures devront être prises lors des spécifications et des tests de validation pour éviter les fautes. Ces deux étapes sont en effet communes aux deux développements et peuvent donc se trouver affectées d'erreurs similaires à l'origines de défaillances de mode commun.

# 6. CONCLUSION ET DISCUSSION

La sécurité d'une machine résulte d'une démarche globale de construction, qui doit impérativement débuter par une analyse des risques et se terminer par la phase d'intégration et le cas échéant de modification (le démantèlement tel qu'il est compris dans l'industrie nucléaire par exemple ne s'applique pas pour une machine).

- ✓ ***Analyse des risques***

La première étape de la construction de la sécurité d'une machine consiste à réaliser une analyse en vue de déterminer les mesures adéquates de réduction des risques. Actuellement, seules deux normes [EN292, 1997], [EN1050, 1991] donnent des informations générales pour guider l'analyste dans sa démarche. Elles s'avèrent insuffisantes, en particulier dans le cas où des objectifs quantifiés en terme de niveau d'intégrité de sécurité doivent être fixés.

Suite aux travaux du HSL[13] [Worsell, 1997], le projet de norme CEI 62061 comprend maintenant une annexe décrivant une procédure pour, suite à une analyse globale des risques d'une machine, fixer de tels objectifs quantifiés. Pour être effectivement applicable et être consensuelle au niveau européen, des études doivent être entreprises pour d'une part l'améliorer et prendre en compte la diversité des applications en sécurité des machines, et d'autre part l'éprouver dans diverses situations représentatives du contexte de la sécurité des machines. Cette tâche devrait être réalisée dans le cadre du projet européen RAMSEM [RAMSEM, 2002].

Dans le cas où la réduction des risques de la machine nécessite une action sur le système de commande de la machine, plusieurs points doivent être traités, tout d'abord au niveau des sous-systèmes utilisés, puis à celui de l'intégration de ces différents sous-systèmes pour réaliser une application industrielle de sécurité.

- ✓ ***Conception des sous-systèmes***

Le chapitre 2 de ce document expose quelques méthodes qui contribuent à la sûreté de fonctionnement d'un sous-système utilisant des composants électroniques complexes : détection des fautes d'une mémoire programme par Cyclic Rendundancy Check et checksum, détection des modifications des durées d'exécution des instructions exécutées par un microprocesseur à l'aide d'un chien de garde ou par analyse de signature, prévention des fautes logicielles par le respect de prescriptions relatives à la qualité et à la sûreté de fonctionnement, élimination des fautes logicielles par le test, prévision des fautes par injection de fautes ou évaluation probabiliste. D'autres méthodes existent, qui sont recensées notamment dans la norme CEI 61508 [CEI61508, 2000]. Pour le domaine de la sécurité des machines, il apparaît que les méthodes actuelles sont suffisantes et donnent satisfaction aux concepteurs.

Taux de couverture

Le paragraphe 4.4 a montré l'influence du taux de couverture des tests sur la valeur de la probabilité de défaillance dangereuse d'un système. Ce dernier est le plus souvent donné de façon empirique. Des valeurs minimales et maximales sont fournies [CEI61508, 2000] sans donner d'informations, ni sur les modèles de fautes utilisés pour établir ces valeurs, ni sur les

[13] *Health & Safety Laboratory (GB)*

conditions de mise en œuvre de ces méthodes. Sans études précises pour évaluer plus précisément l'efficacité de ces méthodes, les résultats quantifiés obtenus seront toujours critiquables (et critiqués). Les études (théoriques ou expérimentales) telles que décrites dans les paragraphes 2.1.1 et 2.1.2 devraient être menées pour l'ensemble des mécanismes de détection de fautes. Ces études devraient aussi considérer que le mécanisme de détection de fautes est généralement implanté sur tout ou partie de l'élément qu'il doit surveiller. Ceci introduit un système bouclé qui commence seulement à faire l'objet d'études [Kanoun & Ortalo-Borrel, 2000].

Sans être aussi critique que la détermination du taux de couverture, la complémentarité entre les différents moyens de détection des fautes implantés sur un système mériterait aussi d'être étudiée. Elle permettrait aux concepteurs de choisir l'ensemble optimal de mesures à appliquer pour assurer la sûreté de leurs applications.

Base de données de fiabilité

Les résultats quantifiés ne pourront être crédibles que si toutes les données le sont. Hormis la base "constructeur" constituée par SIEMENS, les bases de données actuellement utilisées pour les calculs de la probabilité de défaillance dangereuse d'un système de commande sont mal adaptées aux machines. Elles posent des problèmes au niveau du domaine d'application (souvent le nucléaire ou l'aéronautique, dont les conditions environnementales sont différentes de celles de l'industrie manufacturière) ou encore de la technologie (les technologies électroniques sont sans cesse en évolution, entraînant une difficulté à prévoir la fiabilité des composants nouveaux). La constitution d'une base de données spécifique améliorerait la crédibilité des résultats obtenus. Elle ne pourra se faire que par un partenariat européen, dans lequel fabricants de composants, concepteurs et utilisateurs de systèmes seraient impliqués.

- ✓ ***Intégration des sous-systèmes***

L'intégration de sous-systèmes (capteurs, actionneurs, automates programmables industriels) doit se faire de manière à ne pas dégrader le niveau de sécurité intrinsèque de chaque élément. Pour cela, le concepteur doit rechercher l'architecture répondant le mieux à son application.

Architecture

Les problèmes rencontrés au niveau des sous-systèmes lorsqu'ils sont réalisés avec des composants électroniques complexes sont dus à la méconnaissance des phénomènes de base que sont les fautes (du matériel comme du logiciel). Cette méconnaissance amène les concepteurs à rechercher des solutions en terme d'architecture redondante pour en particulier s'affranchir des conséquences des fautes simples indépendantes. Les architectures les plus couramment utilisées en sécurité des machines ont été analysées au chapitre 4. Les analyses qualitatives et les résultats quantifiés obtenus objectivent les prescriptions des normes EN 954 [EN954-1, 1996] et EN 292 [EN292, 1997] en terme d'architecture redondante. Il reste à se préoccuper des défaillances de mode commun.

Les chapitres 3 et 4 ont montré que la principale solution, applicable au matériel comme au logiciel, consiste à recourir à la diversité. Le véritable problème consiste alors à assurer un maximum d'indépendance entre les développements des différents canaux. La solution idéale étudiée au chapitre 5 pour le matériel consiste à utiliser deux sous-systèmes développés par deux sociétés différentes. L'indépendance tout au long du cycle de vie des sous-systèmes est totale. Par contre, trois problèmes au moins se posent :

Le principal problème rencontré est l'impossibilité de maîtriser les tests de diagnostic implantés sur chaque sous-système. Si les produits ont été développés pour satisfaire à des exigences essentiellement fonctionnelles, les tests de diagnostic seront insuffisants pour garantir la détection de toutes les fautes (cf. annexe). Les exigences de la norme EN 954 pour la catégorie 4 ne pourront donc pas être satisfaites. La structure développée pourra au plus prétendre à la catégorie 3, en fonction des contraintes de détection fixées par l'application.

La solution envisageable consisterait à compenser l'absence de tests par d'autres moyens de détection. La mise en œuvre d'un échange entre les deux voies est un moyen de détection des conséquences des fautes (erreurs ou défaillances), sans intrusion dans les sous-systèmes. Elle nécessite d'une part la synchronisation des deux canaux, dont les fonctionnements temporels sont différents, et d'autre part un média commun permettant l'échange : bus de données par exemple. Ces deux problèmes techniques étant résolus, il restera à déterminer les informations pertinentes à comparer, puis à évaluer les capacités de détection de l'échange. Ce dernier problème est de même nature que celui rencontré pour déterminer les taux de couverture des tests des sous-systèmes. La différence portera sur le fait que la détection s'applique essentiellement aux conséquences des fautes, et aux fautes elles-mêmes.

Modélisation des CMF

Il faut ensuite évaluer l'efficacité des mesures prises pour assurer la diversité. Le chapitre 3 a recensé divers modèles représentant ces défaillances pour le matériel, généralement issus du domaine de la sûreté nucléaire, pour des technologies mécaniques ou électrotechniques. Deux problèmes restent en suspend concernant la modélisation par le facteur β, simple et couramment utilisée en sécurité des machines [EN954-1, 1996], [CEI61508, 2000].

- *Modélisation des CMF du matériel pour des structures hétérogènes*

Le modèle β s'applique difficilement aux structures hétérogènes (cf. paragraphe 3.4.2.2). Des travaux pourraient être entrepris pour rechercher une modélisation adéquate, en gardant à l'esprit la difficulté à constituer un retour d'expérience significatif permettant d'appréhender de façon satisfaisante les phénomènes à caractériser.

- *Évaluation du facteur β*

La valeur de ce paramètre est difficile à évaluer. Comme pour les taux de couverture, cette difficulté est pénalisante pour la crédibilité des résultats quantifiés. La démarche proposée par la norme CEI 61508 est intéressante pour objectiver l'analyse. Elle mériterait d'être précisée pour dresser une liste plus complète des mesures destinées à réduire les défaillances de mode commun et à en déterminer l'influence sur le facteur β. Comme pour le point précédent, la difficulté d'une telle action réside dans l'obtention d'un échantillon suffisamment représentatif de défaillances de mode commun ayant affecté des systèmes à composants électroniques complexes.

On constate donc qu'avant d'être en mesure de proposer un guide méthodologique complet pour les concepteurs de systèmes de sécurité à base d'électronique programmable, il est nécessaire de mener des investigations complémentaires. A ces problèmes viennent s'ajouter les aspects liés aux transferts d'informations de sécurité par des réseaux de terrain. Cette problématique est en cours d'instruction pour déterminer les actions de recherche à entreprendre conjointement entre l'INRS et le CRAN, comme l'évaluation des performances exactes de protocoles tels que ceux employés par les réseaux Asi Safety at Work ou Profi safe.

# 7. BIBLIOGRAPHIE

***Publications Ph. CHARPENTIER***


[Charpentier et al., 1988] **(Non référencé)**

P. Charpentier, B. Kopka, J.P. Gérardin - *Étude et validation d'un système à microprocesseur à haut niveau de sécurité basé sur une redondance à hétérogénéité temporelle,*
6ième colloque international de fiabilité et de maintenabilité, Strasbourg, octobre 1988, pp 24-29.
Électroniques, Techniques et Industries, n° 73, novembre 1989, pp 54-60.

[Charpentier et al., 1991]

P. Charpentier, J.P. Gérardin, A. Emond - *Contrôle d'intégrité du contenu d'une mémoire programme : étude comparative du CRC et du Checksum*, L'onde électrique, mars-avril 1991, vol. 71, n° 2, pp 52-57.

[Charpentier, 1991]

P. Charpentier - *Autotest d'une mémoire programme : deux solutions*, Électronique, janvier 1991, n° 4, pp 52-56.

[Gérardin et al., 1991]

J.P. Gerardin, P. Charpentier, C. Vigneron - *Some answers to the problem of safety in microprocessor-based devices*, Conference SAFECOMP'91, Trondheim, November 1991, pp 63-68.

[Charpentier, 1992]

P. Charpentier - *Conception d'un dispositif à microprocesseur sûr de fonctionnement – Règles et méthodes*, Cahiers de Notes Documentaires, 1ier trimestre 1992, n° 146,pp 5-14.

[Charpentier, 1993]

P. Charpentier - *Le chien de garde appliqué au microprocesseur*, Électronique, janvier 1993, n° 24, pp 51-59.

[Charpentier et al., 1994]

P. Charpentier, C. Vigneron, P. Kacprzak - *Analyse des performances d'un mécanisme de détection de fautes*, Industronique, juin 1994, n° 26, pp 23-25 & oct. 1994, n° 27, pp 31-34.

[Kacprzak et al., 1994a]

P. Kacprzak, J. Bremont, M. Lamotte, P. Charpentier - *Optimization of data compression methods*, Microelectronic and reliability, 1994, vol. 34, n° 6, pp 1013-1020.

[Kacprzak et al., 1994b]

P. Kacprzak, P. Charpentier, J. Bremont, M. Lamotte - *Un module matériel d'analyse de signature pour le test en ligne*, 9ième colloque international de fiabilité et de maintenabilité, La Baule, juin 1994, pp 947-953.

[Vautrin et al., 1996]
J.P. Vautrin, P. Charpentier, C. Vigneron - *La sécurité en machinerie : Introduction au concept de catégorie*, Cahiers de Notes Documentaires, 2ième trim. 1996, n° 163, pp 255-262.

[Charpentier, 1997]
P. Charpentier - *L'évitement des fautes logicielles par la qualité*, Cahiers de Notes Documentaires, 2ième trim. 1997, n° 167, pp 249-259.

[Brandt & Charpentier, 1999]
C. Brandt, P. Charpentier - *Validation quantitative des systèmes électroniques du point de vue de la sécurité*, 1ière conférence Internationale Sécurité des Systèmes Industriels Automatisés, Montréal, octobre 1999, pp 34-38.

[Charpentier, 1999]
P. Charpentier - *Évitement des fautes logicielles par la qualité et le test*, 1ière conférence Internationale Sécurité des Systèmes Industriels Automatisés, Montréal, octobre 1999, pp 39-43.

[Charpentier et al., 2000a]
P. Charpentier, N. Diette, F. Escaffre - *Comment construire les tests d'un logiciel*, Cahiers de Notes Documentaires, 4ième trim. 2000, n° 181, pp 65-77.

[Charpentier et al., 2000b]
P. Charpentier, N. Diette, F. Escaffre - *Tools for software fault avoidance- Software fault avoidance through quality*, European Project STSARCES, SMT 4CT97-2191, March 2000, Annexe 2 au rapport final, 37p.

[Charpentier et al., 2000c]
P. Charpentier, N. Diette, F. Escaffre - *Tools for software fault avoidance- Guide to evaluating software quality and safety requirements*, European Project STSARCES, SMT 4CT97-2191, March 2000, Annexe 3 au rapport final, 81p.

[Charpentier et al., 2000d]
P. Charpentier, N. Diette, F. Escaffre - *Tools for software fault avoidance- Guide for the construction of software tests*, European Project STSARCES, SMT 4CT97-2191, March 2000, Annexe 4 au rapport final, 34p.

[Charpentier, 2000]
P. Charpentier - *Tools for software fault avoidance- Common mode faults in safety systems*, European Project STSARCES, SMT 4CT97-2191, March 2000, Annexe 5 au rapport final, 54p.

[Charpentier & Aubry, 2001]
P. Charpentier, J.F. Aubry – *Architecture of safety automatic control systems and common mode failures, 2ième* International conference Safety of Industrial Automated Systems, Bonn, November 2001, pp 161-168.

[Charpentier, 2002]
P. Charpentier – *Sécurité des systèmes de commande de machines : le point sur la normalisation*, Revue de l'Électricité et de l'Électronique, février 2002, n° 2, pp 82-85.

[Charpentier et al., 2002]
Charpentier D., J.L. Durka, P. Villeneuve de Janti, P. Charpentier – *Validation des équipements électroniques de sécurité : Bilan du projet européen STSARCES*, Revue de l'Électricité et de l'Électronique, avril 2002, n° 4, pp 14-16.

[Charpentier & Aubry, 2002]
P. Charpentier, J.F. Aubry – *Consideration of common mode failures in the design of the architecture of safety dedicated programmable controller,* 8th International conference Quality, Reliability, Maintenance CCF 2002, Sinaia (Roumanie), september 2002, 8p., A paraître.

***Contexte***

[98/37/CE, 1998]
*Directive 98/37/CE du 22 juin 1998 concernant le rapprochement des législations des états membres relatives aux machines*, J.O.C.E. n° L207 du 23 juillet 1998, CE, Bruxelles, pp 1-46.

[EN1050, 1991]
*EN 1050 "Sécurité des machines – Principes pour l'appréciation du risque",* décembre 1991, 34p.

[EN292, 1997]
*EN 292 "Sécurité des machines - Concepts de base, principes généraux de conception"*, janvier 1997, 26p.

[EN954-1, 1996]
*Sécurité des machines - Parties des systèmes de commandes relatives à la sécurité Partie 1 : Principes généraux de conception*, décembre 1996, 32p.

[CEI61508, 2000]
*Sécurité fonctionnelle des systèmes électriques / électroniques / électroniques programmables relatifs à la sûreté.*
*Partie 1 : Prescriptions générales,* avril 2000, UTE C 46-061, 59p.
*Partie 2 : Exigences pour les systèmes électriques / électroniques / électroniques programmables,* avril 2000, UTE C 46-062, 71p.
*Partie 3 : Prescriptions concernant les logiciels,* avril 2000, UTE C 46-063, 49p.
*Partie 4 : Définitions et abréviations,* avril 2000, UTE C 46-064, 26p.
*Partie 5 : Exemples de méthodes pour la détermination des niveaux d'intégrité de sécurité,* avril 2000, UTE C 46-065, 28p.
*Partie 6 : Guide pour l'application des parties 2 et 3,* avril 2000, UTE C 46-066, 75p.
*Partie 7 : Bibliographie des techniques et des mesures,* avril 2000, UTE C 46-067, 115p.

[CEI62061, 2002]
*CEI 62061"Sécurité des machines – Sécurité fonctionnelle des systèmes Électriques / Électroniques / Électroniques Programmables"*, Version n° 44/380/CD, May 2002, 90p.

[Avizienis, 1967]
A. Avizienis - *Design of fault-tolerant computers*, Proceedings Fall Joint Computer Conference, Montvale N.J., 1967, pp 733-743.

[Geoffry & Motet, 1998]
J.C. Geoffry, G. Motet - *Sûreté de fonctionnement des systèmes informatiques*, Édition InterEditions, Paris, 1998, 349p, ISBN 2-225-83417-2.

[Jouffroy, 1999]
D. Jouffroy – *Vers une démarche d'intégration de la sécurité à la conception des machines à bois semi-automatisées*, Thèse de doctorat de l'université H. Poincaré de Nancy 1, mars 1999, Les Notes Scientifiques et Techniques de l'INRS, NST n° 177, 182p.

[Lacore, 1993]
J.P. Lacore – *Normes et normalisation européennes de santé et de sécurité dans le cadre de la "nouvelle approche". Aspects généraux,* Cahiers de Notes Documentaires, 1ier Trimestre 1993, n° 150, pp 79-86.

[Lacore, 1998]
J.P. Lacore – *Rôle joué par la normalisation : Bilan et perspectives*, Actes de la journée INRS SRS 98, Systèmes Relatifs à la Sécurité, Paris, mars 1998, 69p.

[Laprie et al., 1995]
J. Arlat, J.P. Blanquart, A. costes, Y. Crouzet, Y. Deswarte, J.C. Fabre, H. Guillermin, M. Kaâniche, K. Kanoun, C. Mazet, D. Powell, C. Rabéjac, P. Thevenot, sous la direction de J.C. Laprie - *Guide de la sûreté de fonctionnement*, CEPADUES Editions, Toulouse, mai 1995, 324p.

[Paques et al., 1999]
J.J. Paques, J.L. Durka, R. Bourbonnière – *Practical use of IEC 61508 and EN 954 for the safety evaluation of an automatic minig track*, Reliability, engineering & system safety, 1999, vol. 66, pp 127-133.

[Sourisse et Boudillon, 1996a
C. Sourisse, L. Boudillon - *La sécurité des machines automatisées*, J'automatise, mars-avril 1996, pp 21-25.

[Sourisse et Boudillon, 1996b]
C. Sourisse, L. Boudillon - *La sécurité des machines automatisées. Tome 1 : Notions de base, réglementation*, normes, techniques de prévention, Institut Schneider Formation Éditions, Grenoble, 1996, ISBN 2-907314-29-7, 236p.

[Sourisse et Boudillon, 1997]
C. Sourisse, L. Boudillon - *La sécurité des machines automatisées. Tome 2 : Techniques et moyens de prévention opératifs* - Systèmes de commande - Utilisation des machines, Institut Schneider Formation Éditions, Grenoble, 1997, ISBN 2-907314-1, 301p.

[Villemeur, 1988]
A. Villemeur - *Sûreté de fonctionnement des systèmes industriels*, Éditeur Eyrolles, Paris, 1988, pp 371-410.

[Chevance, 1999]
R.J. Chevance – *Systèmes à haute disponibilité – Concepts*, Techniques de l'Ingénieur, Août 1999, H2-508, 12p.

[Kanoun & Ortalo-Borrel, 2000]
K. Kanoun, M. Ortalo-Borrel – *Fault-tolerant system dependability – Explicit modelling of hardware and software component-interactions*, IEEE Transactions on reliability, December 2000, vol. 49, n° 4, pp 363-373.

[Mortureux, 2001]
Y. Mortureux – *La sûreté de fonctionnement : Méthodes pour maîtriser les risques,* Techniques de l'Ingénieur, 2001, AG 4 670, 17p.

[Zwingelstein, 1999]
G. Zwingelstein – *Sûreté de fonctionnement des systèmes industriels complexes,* Techniques de l'Ingénieur, 1999, S 8 250, 32p.

Évitement des fautes

[Beizer, 1992]
B. Beizer - *Software testing techniques*, Ed International Thomson Computer press, Boston, 2ième édition, 1992, 549p.

[Myers, 1979]
G.J. Myers - *The art of software testing*, Ed John Wiley & Sons, New York, 1979, 177p.

[Xanthakis et al., 1994]
S. Xanthakis, M. Maurice, A. De Amescua, O. Houri, L. Griffet - *Test et contrôle des logiciels,* Éditeur EC2, Paris, 1994, 341p, ISBN 2-910085-05-8.

Bases de données

[Bot et al., 1997]
Y. Bot, Y. Herrer et al. - *Reliability of electronic components failure rates prediction methods, Advances in Safety and Reliability*, ESREL '97, Lisbonne, 1997, pp 1075-1081.

[CNET, 1993]
CNET France Télécom - *Recueil de données de fiabilité des composants électroniques RDF93,* Éditeur CNET, Centre de Recherche et de Développement de France Télécom, Lannion, 1993.

[HDBK-MIL, 1991]
MIL-HDBK-217F – *Reliability prediction of electronic equipment*, Department of Defense, USA, December 1991.

[Monfort, 1998]
M.L. Monfort - *Estimation de la fiabilité d'un produit (nouveau ou existant) à améliorer à partir de retour d'expérience multiples ou d'expertises*, 11ième colloque international de fiabilité et de maintenabilité, Arcachon, 1998, pp xxx-xxx.

[Morris, 1990]
S.F. Morris - *MIL-HDBK-217 : Use and Application,* Reliability Review, 1990, n° 10, pp 10-13.

[Stadermann et al., 1995]
T.J. Stadermann et al. - *The transition from statistical-field failure based models to physics-of-failure based models for reliability assessment of electronic packages*, Advances in Electronic Packaging ASME 2, Hawaii, 1995, vol. 10, pp 619-625.

Autres

[Letrung et al., 1995]
B. Le Trung, M.C. Monégier du Sorbier, B. Soubiès, O. Elsensohn - *Logiciels de sécurité : nouvelle approche pour optimiser la vérification et la validation*
Le magazine d'ASPROM, 1995, n° 10, pp 25-28.
10ième colloque international de fiabilité et de maintenabilité, St Malo, octobre 1996, pp 388-396.

[STSARCES, 2000]
STSARCES : *Standards for Safety Related Complex Electronic Systems* – European Project SMT 4CT97-2191, march 2000, Final report, 138p.

[Thireau, 1986]
P. Thireau - *Méthodologie d'analyse des effets des erreurs logicielles (AEEL) appliquée à l'étude d'un logiciel de haute sécurité*, 5ième colloque international de fiabilité et de maintenabilité, Biarritz, 1986, pp 111-117.

***Sûreté de Fonctionnement des systèmes électroniques et défaillances dépendantes***

[Buchner, 1994]
H. Buchner - *Occurrence of common mode failure*, Reliability engineering and system Safety, 1994, vol. 45, n° 1-2, pp 201-204.

[Dhillon et Anude, 1994]
B.S. Dhillon, O.C. Anude - *Common-cause failures in engineering systems : a review*, International Journal of Reliability, Quality and Safety Engineering, 1994, vol. 1, n° 1, pp 103-129.

[Edwards & Watson, 1979]
Edwards G.T., Watson I.A. - *A study of common mode failures*, UK Atomic Energy Authority Report SRD-R146 – 1979, 168p.

[EWICS, 1988]
EWICS/TC7 – *Dependability of critical computers systems*, Ed. F.J. Redmill, London, 1988, tome 1, 292p., tome 2, 286p.

[Humphrey, 1989]
P. Humphrey – *Analysis procedure for identification of Multiple Related Failures*, Advanced seminar on Common Cause Failure analysis in probabilistic safety assessment, Amendola Editor, Kluwer academic publishers, Dordrecht (Pays-Bas), 1989, pp 113-129.

[Lyu, 1995]
M.R. Lyu – *Handbook of software reliability engineering*, Computing Mac Graw-Hill / IEEE Computer Society Press, New York, 1995, 850p.

[MDCI, 1994]
Groupe de travail Mécanismes de Défaillances des Circuits Intégrés de l'Institut de Sûreté de Fonctionnement - *Constat des problèmes posés*, Éditeur ISdF, Paris, 1994, 15p.

[Pecht & Ramappan, 1992]
M. Pecht, V. Ramappan – *Are components still the major problem : a review of electronic system and device field failure returns*, IEEE transactions on components, hybrids, ... , December 92, vol. 15, n° 6, pp 1160-1164.

[Taylor, 1975]
J.R. Taylor - *A study of failures causes based on U.S. power reactor abnormal occurrence reports*, Proceedings of the symposium on Reliability of nuclear power plant, Innsbruck, Report n°IAEA-SM-195/16, 1975, pp 119-130.

Retour d'expérience

[Avizienis & Laprie, 1986]
A. Avizienis, J.C. Laprie – *Dependable computing : From concept to design diversity, ,* Proceedings of the IEEE, May 1986, vol. 74, n° 5, pp 629-638.

[Bourne, 1981]
A.J. Bourne, G.T. Edwards, D.M. Hunns, D.R. Poulter, I.A. Watson - *Defences against CMF in redundancy systems - A guide for management, designers and operators,* UK Atomic Energy Authority Report SRD-R196, 1981.

[Brilliant et al., 1990]
S.S. Brilliant, J.C. Knight, N.G. Leveson - *Analysis of faults in a n-version software experiment*, IEEE Transactions on software engineering, February 1990, vol. 16, n° 2, pp 238-247.

[Eckhardt & Lee, 1985]
D.E. Eckhardt, L.D. Lee - *A theorical basis of multiversions software subject to coincident errors*, IEEE Transactions on software engineering, December 1985, vol. 11, n° 12, pp 1511-1517.

[Knight & Leveson, 1986a]
J.C. Knight, N.G. Leveson – *An experimental evaluation of the assumption of independence in multi-version software*, IEEE Transactions on software engineering, January 1986, vol. 12, n° 1, pp 96-109.

[Knight & Leveson, 1986b]
J.C. Knight, N.G. Leveson – *An empirical study of failure probabilities in multi-version software*, 16th International Symposium on Fault Tolerant Computing, Vienne, July 1986, pp 165-170.

[Lala & Harper, 1994]
J.H. Lala, R.E. Harper - *Fault tolerance in embedded real-time systems : importance and treatment of common mode failures*, Congress "Hardware and software architectures for fault-tolerance : Experience and perspectives, Ed Springler-Verlag, June 1994, pp 263-282.

[Miller et al., 1981]
C.F. Miller, W.H. Hubble, D.W. Sams, W.E. Moore – *Data summaries of LER – US Nuclear Power Plants*, US Nuclear Regulatory Commission, NUREG/CR-1740 EGG-EA-5388, 1981, 573p.

[Mitra et al., 2000]
S. Mitra, N.R. Saxena, E.J. McCluskey – *Common-mode failures in redundant VLSI Systems : a survey*, IEEE transactions on reliability, September 2000, vol. 49, n° 3, pp 285-295.

[Mosleh, 1991]
A. Mosleh – *Common cause failures : an analysis methodology and examples*, Reliability engineering and system safety, 1991, vol. 34, pp 249-292.

[Preckshot, 1994]
G.G. Preckshot - *Method for performing diversity and defence-in-depth analyses of reactor protection systems*, Fission Energy and Systems Safety Program, Rapport UCRL-ID-119239, December 1994, 35p.

[Rutledge & Mosleh, 1995]
P.J. Rutledge, A. Mosleh – *Dependant-failures in spacecraft : Root causes, coupling factors, defences and design implications*, Proceedings of the IEEE annual reliability and maintainability symposium, Washington DC, 1995, pp 337-341.

[Smith, 1997]
D.J. Smith – *Reliability, maintainability and risk – Practical methods for engineers*, Butterworth-Heinemenn Editor, 1997, 317p. (Chap. 8.3 : Common mode effects, pp 99-106)

[Tokmachev, 1997]
G.V. Tokmachev – *VVER specific common cause failure data*, Congrès ESREL 97, Lisbonne, juin 97, pp 2189-2194.

Diversité

[Avizienis, 1985]
A. Avizienis – *The N-Version approach to fault-tolerant software,* IEEE Transactions on software engineering, December 1985, vol. SE-11, n° 12, pp 1491-1501.

[Eckhardt et al., 1991]
D.E. Eckhardt, A.K. Caglayan, D.F. McAllister, M.A. Vouk, J.P.J. Kelly - *An experimental evaluation of software redundancy as a strategy for improving reliability*, IEEE Transactions on software engineering, July 1991, vol. 17, n° 7, pp 692-702.

[Hourtolle, 1994]
Hourtolle C. - L*ogiciels sûrs de fonctionnement par diversification de la conception,* 9ième colloque international de fiabilité et de maintenabilité, lieu, Octobre 1994, pp 446-453.

[HSE, 1987]
HEALTH AND SAFETY EXECUTIVE - *PES in safety related applications general technical guidelines*, HMSO Books, London, 1987, 167 p.

[Kelly et al., 1986]
J.P.J. Kelly, A. Avizienis, B. T. Ulery, B.J. Swain, R.T. Lyu, A. Tai, K.S. Tso – *Multi-version software development,* Proceedings of the 5th SAFECOMP'86, Sarlat, October 1986, pp 43-49.

[Littlewood et al., 2000a]
B. Littlewood, P. Popov, L. Strigini – *Modelling software diversity : a review*, ACM Computing Survey, 2000, disponible sur le site http://www.csr.city.ac.uk/csr_city/staff/popov, 24p.

[Littlewood et al., 2000b]
B. Littlewood, P. Popov, L. Strigini – *Assessment of the reliability of fault-tolerant software : a Bayesian approach,* Proceedings of the 19th international conference on computer safety, reliability and security, SAFECOMP 2000, Rotterdam, 2000, Ed Springer-Verlag, 10 p.

[Littlewood & Miller, 1989]
B. Littlewood, D.R. Miller – *Conceptual modelling of coincident failure in multi-version software*, IEEE Transactions on software engineering, December 1989, vol. 15, n° 12, pp 1596-1614.

[Partridge, 1997]
D. Partridge, W. Krzanowski – *Software diversity : practical statistics for its measurement and exploitation*, Information and software technology, 1996, n° 39, pp 707-717.

[Popov et al., 1999]
P.Popov, L. Strigini, A. Romanovski – *Choosing effective methodes for design diversity - How to progress from intuition to science,* Proceedings of SAFECOMP'99, Toulouse, Springer-Verlag Berlin, 1999, pp 272-285.

[Summers & Raney, 1999]
A.E. Summers, G. Raney – *Common cause and common sense, designing failure out of your safety instrumented systems (SIS)*, ISA Transactions, 1999, vol. 38, pp 291-299.

Modélisation des défaillances de mode commun du logiciel :

[Littlewood, 1996]
B. Littlewood, *The impact of diversity upon common-mode failure,* Reliability Engineering and System Safety, 1996, vol. 51, n° 1, pp 101-113.

[Popov et al., 1998a]
P. Popov, L. Strigini – *Conceptual models for the reliability of diverse systems – New results*, Proceedings of FTCS'28, Munich, June 1998, pp 80-89.

[Popov et al., 1998b]
P. Popov, L. Strigini, M. Pizza – *The efficacy of diverse redundancy against design errors : Some practical considerations*, 3ième International conference on Control and Instrumentation in Nuclear Installations, Edinburg, 1998, 10p.

Modélisation des défaillances de mode commun du matériel :

[Blanchet, 2000]
C. Blanchet – *Contribution à la réalisation d'une architecture redondante hétérogène d'ordre 2*, Rapport INRS n° IET/00DT-057, Août 2000, 64p.

[Dewailly et al., 2002]
J. Dewailly, D. Vasseur, A. Voicu – *Traitement des défaillances de cause commune dans les systèmes hautement redondants, avec la méthode CLM*, 13ième colloque international de fiabilité et de maintenabilité, Lyon, Mars 2002, pp 703-708.

[Fleming, 1975]
K.N. Fleming K.N. - *A reliability model for common mode failures in redundant safety systems*, Proceedings of the 6th Conference on Modelling and Simulation - Part 1, Pittsburgh, 1975, vol. 6, pp 579-581.

[Fleming, 1989]
K.N. Fleming - *Parametric models for common cause failure analysis,* -Advanced seminar on Common Cause failure analysis in probabilistic Safety assessment – Ed. A. Amendola - Kluwer Academic Publishers, Dordrecht (Pays-Bas), 1989, pp 159-174.

[Fleming et al., 1986]
K.N. Fleming, A. Mosleh, R.K. Deremer - *A systematic procedure for the incorporation of common cause events into risk and reliability models*, Nuclear Engineering and Design, 1986, vol. 93, pp 245-273.

[Humphrey, 1987]
R.A. Humphrey – *Assigning numerical value to the beta factor common cause evaluation*, Proceedings of the 2nd National reliability conference, Birmingham, 1987, vol. 1, pp 2C/5/1-2C/5/8.

[ISAdTR84, 1998]
*Safety Instrumented Systems - Safety Integrity Level evaluation techniques, Part 3 : Determining the SIL of a SIS via Fault Tree Analysis,* dTR84.0.02 Version 3, March 1998, 42p.

[Modarres et al., 1998]
M. Modarres, M. Kaminskiy, V. Kristof - *Reliability engineering and risk analysis – A practical guide*, Marcel Dekker Editor, New York, N.Y., 1998, pp. 410-421.

[Mosleh & Fleming, 1988]
A. Mosleh, K.N. Fleming - *Procedure for treating common cause failures in safety and reliability studies : procedural framework and examples*, Pickard, Lowe, and Garrik, Inc, NUREG/CR-4780-V1, January 1988, 350p.

[Mosleh & Siu, 1987]
A. Mosleh, N.O. Siu - *A multiparameter event based common cause failure model*, Proceedings of the 9th international conference on structural mechanics in reactor technology, Lausanne, 1987, paper M7/3, pp 147-152.

[Berg et al., 2002]
H.P. Berg, R. Götz, E. Schimetschka - *A process oriented simulation model for common cause failures in systems of redundant components,* 13ième colloque international de fiabilité et de maintenabilité, Lyon, Mars 2002, pp 336-339.

[Brandt, 1998]
C. Brandt – *Quantification de la probabilité de défaillance dangereuse d'un système électronique par graphe de Markov*, DEA ATNS, Rapport de synthèse, September 1998, 45 p.

[Bukowski & Goble, 1995]
J.V. Bukowski, W.M. Goble - *Using Markov models for safety analysis of programmable electronic systems*, ISA Transactions 1995, vol. 34, pp 193-198.

[EXERA, 2000]
EXERA : Groupe de travail "Systèmes de sécurité à API" – *Automates Programmables dédiés à la sécurité : Panorama*, Rapport EXERA, Rédacteur J.F. Aubry, Décembre 2000, 68p.

[Goble, 1998a]
W.M. Goble – *Control systems safety evaluation & reliability,* ISA Resources for Measurement and Control Series Editor, Research and Triangle Park N.C., 1998, ISBN 1-55617-636-8, 513p.

[Goble, 1998b]
W.M. Goble – *The use and development of quantitative reliability and safety analysis in new product design*, Eindhoven University of Technology Editor, Eindhoven, 1998, ISBN 90-386-0870-5, 213p.

[Goble, 2000]
W.M. Goble – *Using simulation to characterise Common Cause*, disponible sur le site http://www.exida.com, 2000, 6p.

[Goble et al., 1998a]
W.M. Goble, J.V. Bukowski, A.C. Brombacher – *How common cause ruins the safety rating of a fault tolerant PES,* ISA Transactions, 1996, vol. 35, pp 59-65.

[Goble et al., 1998b]
W.M. Goble, J.V. Bukowski, A.C. Brombacher – *How diagnostic coverage improves safety in programmable electronic systems,* ISA Transactions, 1998, vol. 36, n° 4, pp 193-198.

[Goble & Brombacher, 1999]
W.M. Goble, A.C. Brombacher – *Using Failure Modes, Effects and Diagnostic Analysis (FMEDA) to measure diagnostic coverage in programmable electronic systems*, Reliability engineering & system safety, 1999, vol. 66, pp 145-148.

[Kaufman et al., 2000]
L.M. Kaufman, S. Bhide, B.W. Johnson – *Modelling Common-Mode failures in digital embedded systems*, Proceedings Annual reliability and maintainability symposium, Los Angeles, 2000, pp 350-357.

[Knegtering & Brombacher, 1999]
B. Knegtering, A.C. Brombacher – *Application of micro-Markov models for quantitative safety assessment to determine safety integrity levels as defined by the IEC 61508 standard for functional safety*, Reliability engineering & system safety, 1999, vol. 66, pp 171-175.

[Olf, 2001]
*OLF guideline on the application of IEC 61508 and IEC 61511 in the petroleum activities on the Norwegian Continental Shelf*, Rev. 01, January 2001, http://www.itk.ntnu.no/sil, 55p.

Évaluation quantitative

[Bukowski, 2001]
J.V. Bukowski – D*efining Mean Time To Failure in particular failure state for multi-failure-state systems,* IEEE transactions on reliability, June 2001, vol. 50, n° 2, pp 221-228.

[Desroches, 1995]
A. Desroches – *Concepts et méthodes probabilistes de base de la sécurité,* Editions Lavoisier Tech & Doc, Paris, 1995, 188p.

[Fouque, 1993]
J.P. Fouque - *Calcul des probabilités – Concepts et résultats de base*, Techniques de l'ingénieur, mai 1993, A 560, 29p.

[Ringler, 1996]
J. Ringler – *Précis de probabilités et de statistiques à l'usage de la fiabilité*, Octares Éditions, Toulouse, 1996, ISBN : 2-906769-30-4, 106p.

***Mise en oeuvre***

[EXERA, 1998]
EXERA - *Les Automates Programmables Industriels*, J'automatise, septembre / octobre / novembre 1998, pp 22-29.

[Mougeot & Fauconnet, 1993]
B. Mougeot, M. Fauconnet – *Contrôleurs de discordance : Bilan d'une évaluation des dispositifs disponibles sur le marché français*, Cahiers de Notes Documentaires, 3ième trim. 1993, n° 152, pp 435-444.

***Conclusion***

[Worsell, 1997]
N. Worsell, J. Wilday, T. Treble – *New developments in risk assessment: the application of risk assessment to safety in machinery de*sign - Annual meeting of the Society for Risk Analysis (Europe) Risk Analysis and Management in a Global Economy. **1** (1), 1997, pp. 386-409, ISBN 3-932013-20-4.

[RAMSEM, 2002]
Projet RAMSEM "*Development and validation of a risk assessment methodology for incorporation into EN 1050*" – Part b (34p), Part c (33p), Déposé à la CE en Février 2002, N° GRD1-2002-70005

## ANNEXE 1 - REPRÉSENTATION SCHÉMATIQUE D'UNE MACHINE

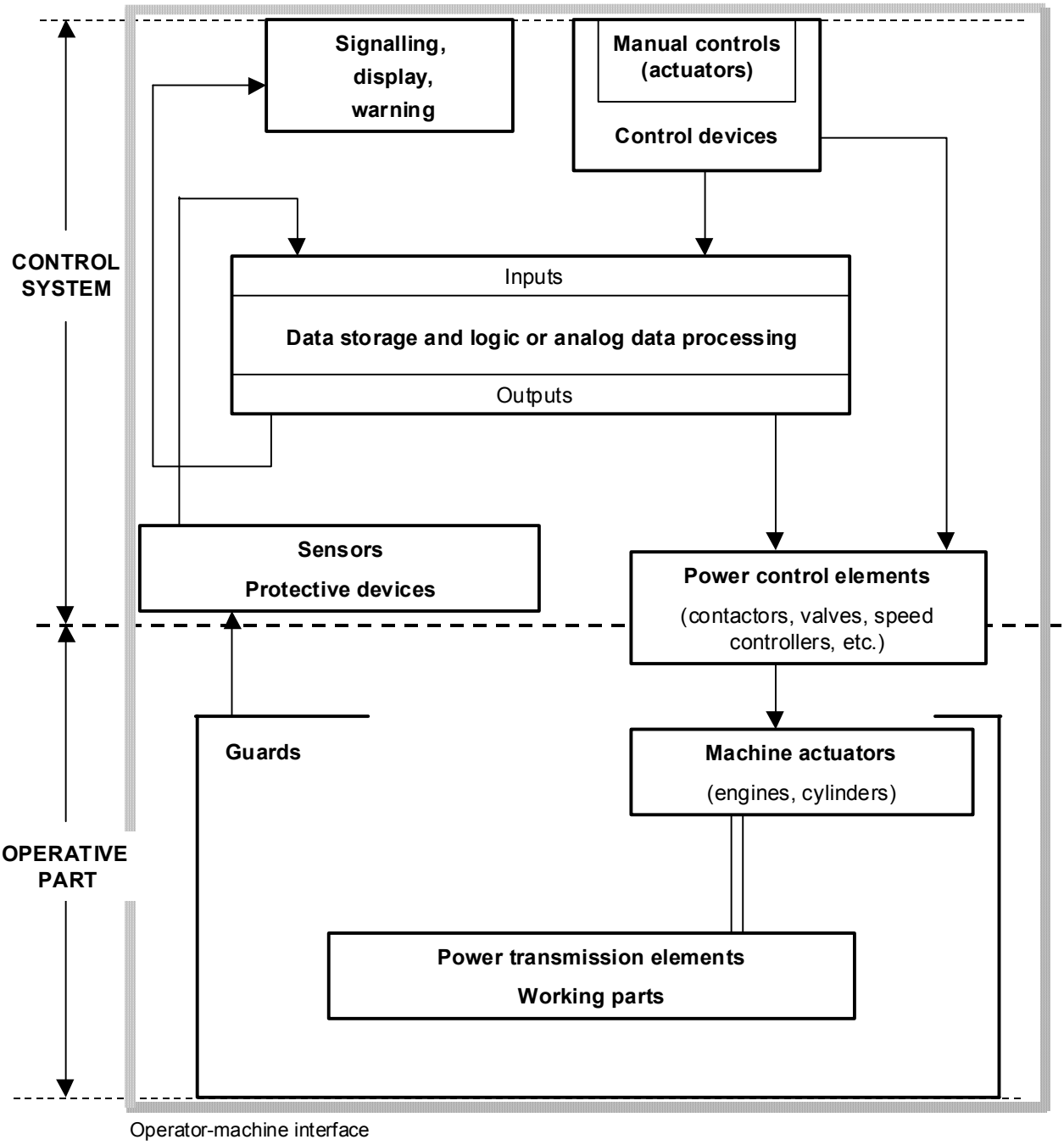



**Tableau 7-1 : Représentation schématique d'une machine**

Source EN 292-1

## ANNEXE 2 - REPRÉSENTATION SCHÉMATIQUE DU PROCESSUS ITÉRATIF DE RÉDUCTION DU RISQUE

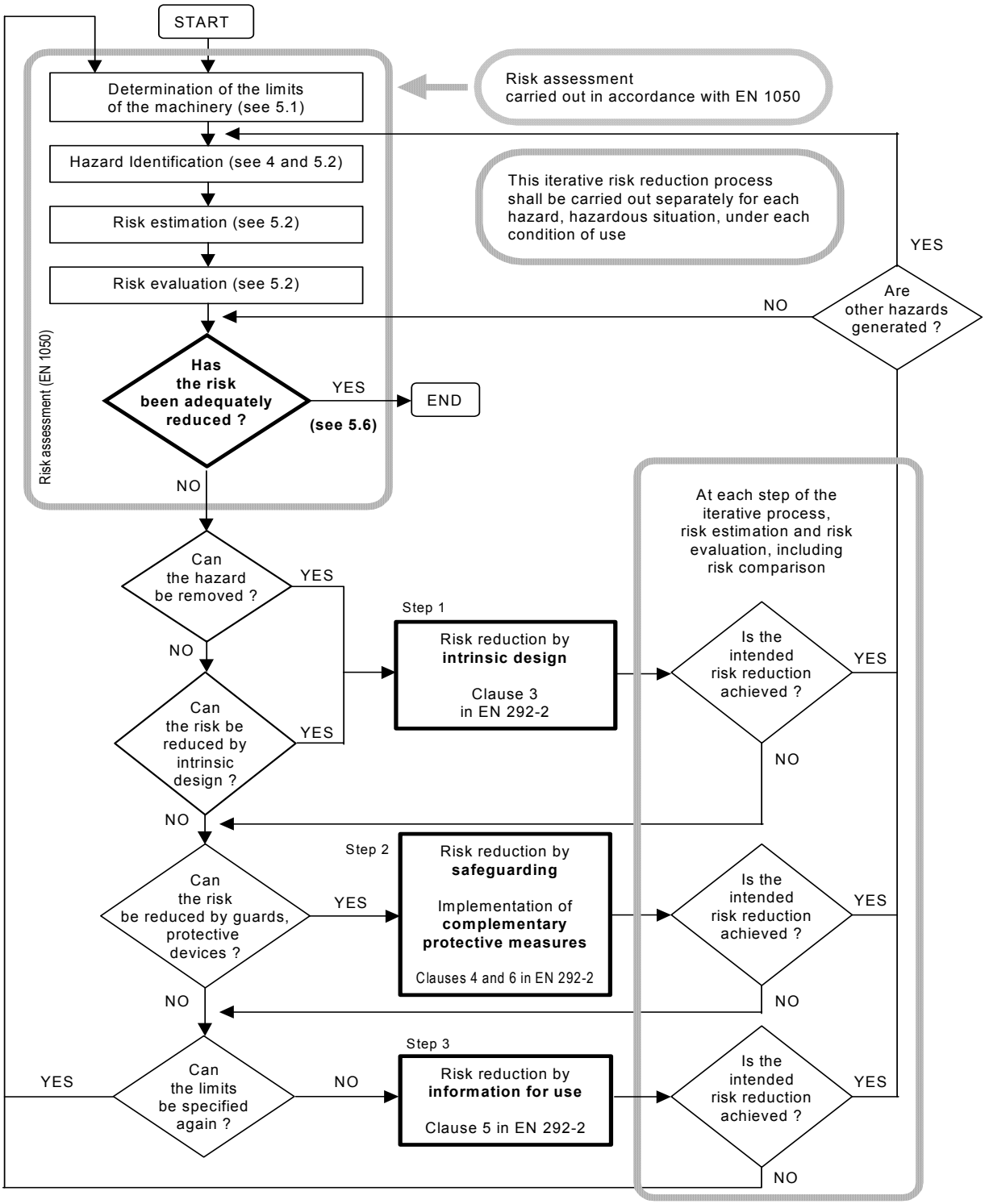



**Tableau 7-2 : Représentation schématique du processus de réduction du risque**

Source EN 292-1

# ANNEXE 3 - LES DIFFÉRENTES CATÉGORIES DE LA NORME EN 954

| Caté-gorie | Résumé des prescriptions | Comportement du système | Principes pour atteindre la sécurité |
|---|---|---|---|
| B | Les parties relatives à la sécurité du système de commande et/ou ses dispositifs de protection, ainsi que ses composants doivent être conçus, fabriqués, sélectionnés, montés et combinés selon les normes pertinentes afin de pouvoir faire face aux influences attendues. | Si un défaut se produit, il peut conduire à la perte de la fonction de sécurité. | Principalement caractérisés par la sélection des composants |
| 1 | Les prescriptions de B doivent s'appliquer.<br><br>Des composants éprouvés et des principes de sécurité éprouvés doivent être utilisés. | L'occurrence d'un défaut peut conduire à la perte de la fonction de sécurité mais la probabilité de cette occurrence est plus basse qu'en catégorie B | |
| 2 | Les prescriptions de B et l'utilisation des principes de sécurité éprouvés doivent s'appliquer.<br><br>La fonction de sécurité doit être contrôlée à intervalles convenables par le système de commande de la machine. | - L'apparition d'un défaut peut mener à la perte de la fonction de sécurité entre les intervalles de contrôle.<br><br>- La perte de la fonction de sécurité est détectée par le contrôle. | Principalement caractérisés par la structure |
| 3 | Les prescriptions de B et l'utilisation des principes de sécurité éprouvés doivent s'appliquer.<br><br>Les parties relatives à la sécurité doivent être conçues de façon à ce que :<br><br>- un défaut unique dans n'importe laquelle de ces parties ne conduise pas à la perte de la fonction de sécurité, et<br><br>- si cela est raisonnablement faisable, le défaut unique soit détecté. | - Lorsqu'un défaut unique se produit, la fonction de sécurité est toujours assurée.<br><br>- Certains défauts seront détectés, mais pas tous.<br><br>- L'accumulation de défauts non détectés peut conduire à la perte de la fonction de sécurité. | |
| 4 | Les prescriptions de B et l'utilisation des principes de sécurité éprouvés doivent s'appliquer.<br><br>Les parties relatives à la sécurité doivent être conçues de sorte que :<br><br>- un défaut unique dans n'importe laquelle de ces parties ne conduise pas à une perte de la fonction de sécurité, et<br><br>- le défaut unique soit détecté à, ou avant la prochaine sollicitation de la fonction de sécurité. Si cette détection n'est pas possible, une accumulation de défauts ne doit pas mener à une perte de la fonction de sécurité. | - Lorsque les défauts se produisent, la fonction de sécurité est toujours assurée.<br><br>- les défauts seront détectés à temps pour empêcher une perte de la fonction de sécurité. | Principalement caractérisés par la structure |

**Tableau 7-3 : Les catégories de la norme EN 954-1**

Important : La catégorie n'est pas destinée à être utilisée dans un ordre donné ou une hiérarchie donnée par rapport aux prescriptions de sécurité.

## ANNEXE 4 - DÉMARCHE DE CONCEPTION PROPOSÉE PAR LA NORME CEI 62061

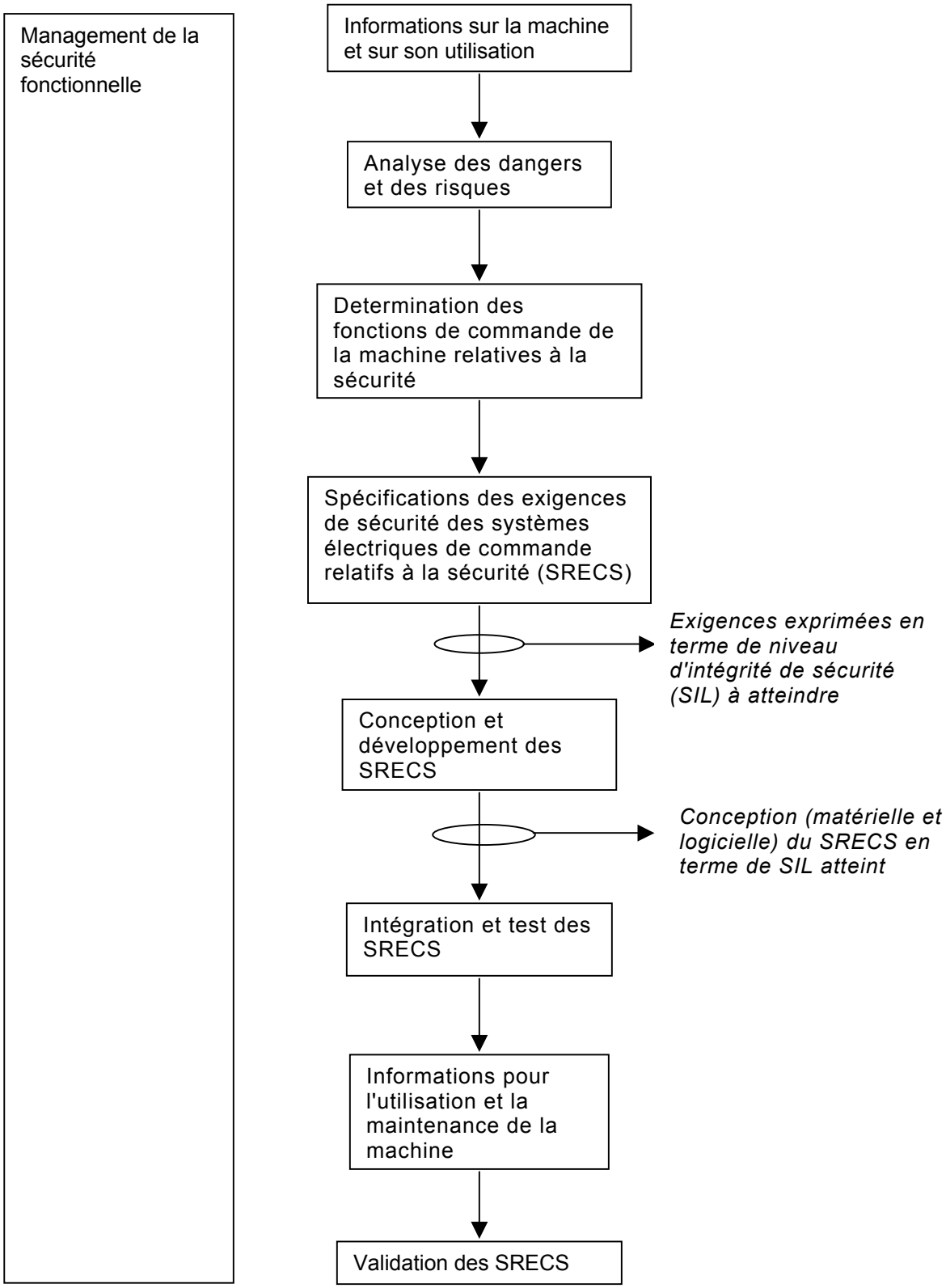



**Tableau 7-4 : Démarche de conception de la norme CEI 62061**

# ANNEXE 5 – CHECK LISTES POUR L'ÉVITEMENT DES DÉFAILLANCES DE MODE COMMUN

Les check listes proposées organisent les points à vérifier durant les différentes phases du cycle de vie du système, dans le but de s'assurer de la mise en oeuvre de techniques et de moyens de sûreté de fonctionnement (en particulier l'évitement) pour faire face aux défaillances de mode commun [HSE, 1987]. Le questionnaire sert de trame au concepteur et à l'analyste pour prendre en compte des CMF durant les différentes activités. Il est destiné à mettre en évidence de façon qualitative certains points sensibles, qui devront être analysées en détail.

L'objectif de ces check listes est de favoriser une analyse critique vis-à-vis des défaillances de mode commun en traitant :

- les spécifications de sécurité,
- la spécification, la conception, la fabrication, le test du matériel,
- la spécification, la conception, le codage, le test du logiciel.

Les différents items sont en grande partie destinés à analyser le degré d'indépendance et de diversité introduit pour les phases considérées du cycle de vie.

Certains items sont issus des check listes établies par le HSE [HSE, 1987].

Note : La convention suivante sera utilisée pour compléter la colonne relative aux résultats de l'évaluation des différents items : le score maximal est obtenu si une diversité maximale est atteinte.

Exemple : à la question "les spécifications de sécurité ont-elles été écrites sous différentes formes?", le score sera de 5 (maximal) si les langages de spécification utilisés sont totalement différents et de 0 s'il sont identiques.

## SPÉCIFICATION DU SYSTÈME

| N° | ITEM À ÉVALUER | RÉSULTAT | COMMENTAIRES |
|---|---|---|---|
| | Les spécifications de sécurité ont-elles été écrites et développées par différentes personnes? Si non, quels sont les liens entre ces personnes?<br>*Les langages de spécification peuvent être les mêmes.*<br>*Il est très difficile, voir impossible d'introduire une réelle diversité des spécifications d'un système, des liens étant souvent nécessaires entre les équipes chargées de les rédiger.* | ☐☐☐☐☐ | |

| | | | |
|---|---|---|---|
| | Les spécifications relatives à la sécurité ont-elles été vérifiées par des personnes différentes ? Si non, quels sont les liens entre ces personnes?<br>*La vérification détecte des erreurs de spécification (CMF et autres). Les spécifications constituant généralement le point commun à tout développement, même diversitaire, toute faute à ce niveau sera répercutée dans les différents canaux. Les systèmes de comparaison étant incapables de détecter de telles fautes, il est important d'en minimiser le nombre par une vérification approfondie des spécifications.* | | |
| | Les sources de défaillances de mode commun (matérielles et logicielles) susceptibles d'affecter le système ont-elles été déterminées ?<br>*La détermination des CMF est indispensable à une prise en compte correcte de ces phénomènes, dès la phase de spécification.* | | |

## SPÉCIFICATIONS DU MATÉRIEL

| N° | ITEM À ÉVALUER | RÉSULTAT | COMMENTAIRES |
|---|---|---|---|
| | Les spécifications du matériel ont-elles été écrites et développées par des personnes différentes? Si non, quels sont les liens entre ces personnes?<br>*Les spécifications peuvent être écrites sous différentes formes.*<br>*Comme pour les spécifications du système, il est très difficile, voir impossible, d'introduire une réelle diversité des spécifications du matériel, des liens étant souvent nécessaires entre deux équipes chargées de rédiger des spécifications.* | | |
| | Les fautes de mode commun susceptibles d'affecter le matériel ont-elles été déterminées ?. Quelles sont les sources potentielles de CMF susceptibles d'affecter le matériel?<br>*La détermination des CMF est indispensable à une prise en compte correcte de ces phénomènes.*<br>*On analysera notamment les influences possibles des perturbations externes éventuelles, des fautes systématiques et d'éventuels points communs.* | | |

**CONCEPTION DU MATÉRIEL**

| N° | ITEM À ÉVALUER | RÉSULTAT | COMMENTAIRES |
|---|---|---|---|
| | Les différents canaux ont-ils été développés par des personnes différentes? Si non, quels sont les liens entre ces personnes?<br>*Cette diversité permet notamment de s'affranchir de fautes de conception d'origine humaine.* | | |
| | Quelles sont les séparations physiques entre les canaux redondants pour éviter la propagation des fautes ?<br>- Isolation électrique des différentes voies?<br>- Attribution d'un équipement à une seule fonction (sans ressources partagées avec d'autres parties du système)?<br>- Non utilisation de composants commun à plusieurs canaux?<br>*La séparation est une technique constructive destinée à ne pas propager les défaillances d'une fonction à l'autre, donc de limiter les analyses aux points sensibles.* | | |
| | Quels sont les moyens utilisés pour éviter la propagation logique des fautes :<br>- Isolation physique des modules logiciels exécutés par deux microprocesseur?<br>- Interactions par des mémoires partagées?<br>- Absence de liens bidirectionnels entre les systèmes?<br>- Existence de protocoles de sécurisation des informations transférées par les bus?<br>*Par le point commun qu'ils constituent, les liens sont des sources potentielles de CMF, car susceptibles de propager des défaillances.*<br>*Les liens entre les canaux sont généralement destinés aux échanges d'informations, aux comparaisons et aux synchronisations : mémoires partagées, liaisons séries ou parallèles.* | | |

| | | | |
|---|---|---|---|
| | La diversité, lorsqu'elle est utilisée, est basée sur :<br>- Différents fournisseurs de produits fondamentalement différents?<br>- Le même fournisseur de produits fondamentalement différents?<br>- Différents fournisseurs de produits similaires?<br>- Différentes versions d'un même produit? | | |
| | Dans le cas où une diversité fonctionnelle existe, celle-ci est basée sur une différence :<br>• des mécanismes sous-jacents?<br>• des fonctions, logiques de contrôle ou moyens d'activation?<br>• des échelles de temps de réponse.<br>*La diversité fonctionnelle la plus efficace fait appel à des mécanismes sous-jacents différents.* | | |
| | Dans le cas où une diversité de signaux existe, celle-ci est basée sur :<br>- Une différence entre des effets physiques mesurés?<br>- Une différence entre des paramètres mesurés à l'aide des mêmes principes?<br>- Une redondance de capteurs identiques? | | |
| | Les canaux sont-ils conçus :<br>• avec des technologies différentes ?<br>*Numérique / analogique, TTL ou CMOS, ...*<br>• avec des composants électroniques d'architectures différentes ?<br>*Microprocesseurs RISC ou CISC, qui génèrent l'utilisation d'assembleur ou de compilateurs différents.*<br>• avec des composants électroniques de versions différentes ?<br>*Des composants différents prémunissent contre certaines fautes de conception ou d'utilisation.* | | |

| | | | |
|---|---|---|---|
| | Y a-t-il des blindages et protections contre les perturbations électromagnétiques ?<br>*Les blindages et protections constituent une barrière contre les fautes communes susceptibles d'être provoquées par des perturbations externes.* | ☐☐☐☐☐ | |
| | Les canaux redondants ont-ils une horloge commune ?<br>*Des horloges distinctes créent une diversité temporelle utile à l'évitement des CMF d'origine externe.* | ☐☐☐☐☐ | |
| | Les différentes ressources matérielles sont-elles testées périodiquement ?<br>*Les tests périodiques évitent les accumulations de fautes, et de ce fait, ont un effet sur l'évitement des fautes de mode commun.* | ☐☐☐☐☐ | |
| | Y a-t-il une alimentation commune à l'ensemble des canaux ? | ☐☐☐☐☐ | |
| | Les règles de l'art ont-elles été suivies lors de la conception du matériel ?<br>*Le respect des règles de l'art évite l'apparition de certaines fautes en opérationnel. Ces règles peuvent être : utiliser une conception éprouvée et des normes, éviter une complexité ou des difficultés non nécessaires, dimensionner correctement les composants, les capacités de mémorisation et les temps de traitement, faire travailler les composants dans leurs limites d'utilisation.* | ☐☐☐☐☐ | |

**FABRICATION DU MATÉRIEL**

| N° | ITEM À ÉVALUER | RÉSULTAT | COMMENTAIRES |
|---|---|---|---|
| | Y a-t-il eu une indépendance suffisante entre la réalisation des différents canaux ?<br>*Les sources de fautes communes dues à la réalisation du matériel sont moins nombreuses que pour le logiciel.* | ☐☐☐☐☐ | |
| | Les circuits imprimés de chaque canal sont-ils différents ?<br>*Une différence d'implantation des composants et de routage des signaux favorisera l'immunité aux fautes de mode commun d'origine externe.* | ☐☐☐☐☐ | |

**TEST DU MATÉRIEL**

| N° | ITEM À ÉVALUER | RÉSULTAT | COMMENTAIRES |
|---|---|---|---|
| | Les tests du matériel ont-ils été réalisés par des personnes différentes de celles qui ont spécifié et conçu ce matériel ?<br>*Les tests participent à l'évitement des fautes (de mode commun ou non).*<br>*Les personnes ayant spécifié et conçu le matériel ne sont pas les mieux à même de détecter les fautes lors de la phase de test.*<br>*A minima, les tests devront être spécifiés par des personnes autres que celles qui ont conçu et spécifié le matériel.* | ☐☐☐☐☐ | |
| | Y a-t-il eu suffisamment d'indépendance entre les tests des différents canaux ?<br>*Des tests identiques ou similaires peuvent s'avérer inappropriés à détecter des fautes matérielles.* | ☐☐☐☐☐ | |
| | En cas de défaillance révélée par les tests, a-t-on recherché d'éventuelles causes relatives au mode commun ? | ☐☐☐☐☐ | |

**SPÉCIFICATIONS DU LOGICIEL**

| N° | ITEM À ÉVALUER | RÉSULTAT | COMMENTAIRES |
|---|---|---|---|
| | Des spécifications du logiciel ont-elles été développées par des personnes différentes? Si non, quels sont les liens entre ces personnes?<br>*Les spécifications peuvent être écrites sous différentes formes.*<br>*Il est très difficile, voire impossible, d'introduire une réelle diversité des spécifications du matériel, des liens étant souvent nécessaires entre deux équipes chargées de rédiger des spécifications.* | ☐☐☐☐☐ | |
| | Les fautes de mode commun susceptibles d'affecter le logiciel ont-elles été déterminées ? Quelles sont les sources potentielles de CMF au niveau du système?<br>*La détermination des CMF est indispensable à une prise en compte correcte de ces phénomènes.*<br>*On analysera notamment les possibilités d'introduction de fautes systématiques au cours de l'ensemble des étapes du cycle de vie du logiciel.* | ☐☐☐☐☐ | |

| | | | |
|---|---|---|---|
| | La redondance logicielle choisie tient-elle compte des modes communs ?<br>*La prise en compte des CMF peut / doit influer le choix de la structure : diversités temporelles, structurelles, fonctionnelles, ...* | ☐☐☐☐☐ | |
| | Quel est le type de diversité spécifié pour le logiciel :<br>- Diversité humaine ?<br>- Diversité fonctionnelle ?<br>- Diversité des signaux d'entrées ?, de sorties ?, des traitements ?<br>- Diversité des équipements ?<br>*Les diversités choisies sont déterminées par les CMF susceptibles d'affecter le matériel.*<br>*La redondance homogène ne prémunira pas contre les fautes de conception.* | ☐☐☐☐☐<br>☐☐☐☐☐<br>☐☐☐☐☐<br>☐☐☐☐☐ | |
| | Le ou les logiciels ont-ils été développés en suivant un plan ou des prescriptions qualité ?<br>*Le suivi d'un plan qualité pour le développement d'un logiciel concoure à éviter les fautes (de mode commun ou non). Il est impératif, en particulier lorsque l'architecture logicielle est homogène. Une qualité logicielle minimale doit être assurée, même dans le cas d'une structure diversifiée. La qualité évite l'introduction de fautes logicielles, en particulier au stade de la maintenance.* | ☐☐☐☐☐ | |

**CONCEPTION DU LOGICIEL**

| N° | ITEM À ÉVALUER | RÉSULTAT | COMMENTAIRES |
|---|---|---|---|
| | En cas de diversité, les différents logiciels ont-ils été conçus par des personnes différentes?<br>*Attention aux liens possibles entre équipes chargées de concevoir les différentes versions d'un logiciel.* | ☐☐☐☐☐ | |
| | Compétences des concepteurs ?<br>*Une compétence insuffisante des concepteurs peut les amener introduire des erreurs similaires dans les différentes versions d'un logiciel.* | ☐☐☐☐☐ | |
| | Les algorithmes de traitement sont-ils identiques d'un canal à l'autre ? | ☐☐☐☐☐ | |

| | | | |
|---|---|---|---|
| | Les outils de développement utilisés sont-ils identiques d'un canal à l'autre ? | ☐☐☐☐☐ | |
| | Quelles sont les séparations logiques entre les canaux redondants ?<br>*La séparation est une technique constructive pour ne pas propager les défaillances d'une fonction à l'autre, donc de limiter les analyses aux points sensibles.* | ☐☐☐☐☐ | |
| | Y a-t-il des mémoires partagées qui pourraient propager une faute logicielle d'un canal à l'autre ? | ☐☐☐☐☐ | |
| | L'exécution des logiciels est-elle desynchronisée ?<br>*Une desynchronisation de l'exécution de deux logiciels crée un décalage favorable à la réduction de fautes ayant pour origine commune des perturbations externes, comme par exemple les perturbations électromagnétiques.* | ☐☐☐☐☐ | |

**CODAGE DU LOGICIEL**

| N° | ITEM À ÉVALUER | RÉSULTAT | COMMENTAIRES |
|---|---|---|---|
| | En cas de diversité, les différents logiciels ont-ils été codés par des personnes différentes?<br>*Attention aux liens possibles entre équipes chargées de coder les différentes versions d'un logiciel.* | ☐☐☐☐☐ | |
| | Différents langages de programmation ont-ils été utilisés pour chaque canal ? | ☐☐☐☐☐ | |
| | Différents compilateurs ou assembleur ont-ils été utilisés ? | ☐☐☐☐☐ | |

**TEST DU LOGICIEL**

| N° | ITEM À ÉVALUER | RÉSULTAT | COMMENTAIRES |
|---|---|---|---|
| | Des spécifications de sécurité ont-elles été développées par des personnes différentes?<br>*Des liens sont souvent nécessaires entre deux équipes chargées de rédiger des spécifications.* | ☐☐☐☐☐ | |
| | Les tests du logiciel ont-ils été réalisés par des personnes différentes de celles qui ont spécifié et conçu ce logiciel ? | ☐☐☐☐☐ | |

# ANNEXE 6 – Rappels de probabilité et fiabilité

Sources : [Fouque, 1993], [Desroches, 1995], [Ringler, 1996].

**Notion d'événement :**

On appelle expérience toute opération reproductible susceptible de sélectionner un élément A dans un ensemble Ă. Exemple : jet d'un dé
Un événement A est le résultat d'une expérience. Exemple : obtention d'un 3 aux dés.
Un événement peut être la combinaison (avec un OU logique) d'événements élémentaires résultant de plusieurs expériences. Exemple : obtention d'un 2 ou d'un 3 aux dés.
Un événement peut résulter de plusieurs expériences élémentaires simultanées. Exemple : obtention d'un 2 et d'un 5 en jetant deux dés.
Deux événements sont incompatibles s'ils ne peuvent se produire simultanément. Exemple : tirer un cœur est incompatible avec tire un pique.

**Notion de probabilité :**

Probabilité d'un événement A : Évaluer les chances d'apparition d'un événement A.
Axiomes liés aux probabilités :

- la probabilité d'un événement quelconque est toujours positive ;
- la probabilité d'un événement certain est égale à 1 ;
- si A et B sont incompatibles, alors on a $A \cap B = \phi$ et P(A ou B) = P(A) + P(B).

Théorèmes liés aux probabilités :

- La probabilité de l'événement contraire de A est 1 – P(A).
- P(A ou B) = P(A) + P(B) – P(A et B) quels que soient A et B qui se généralise pour donner la formule de Poincaré :

$$P\left(\bigcup_{m=1}^{n} Am\right) = p_1 - p_2 + ... + (-1)^{n-1} p_n$$

$$où \quad p_k = \sum_{1 \le i_1 \le ... \le i_k \le n} P(A_{i1} \cap ... \cap A_{ik})$$

- Probabilités conditionnelles : Si P(B/A) est la probabilité de l'événement B si l'événement A s'est produit, alors $P(B / A) = \frac{P(A \text{ et } B)}{P(A)}$
- Si les chances d'obtenir B ne dépendent pas de l'apparition de A, alors les deux événements A et B sont indépendants et P(B/A) = P(B) ce qui s'écrit aussi P(A et B) = P(A).P(B) .
- Formule de la probabilité totale. Si un événement B peut être décomposé en une suite exhaustive finie ou infinie d'événements An, alors il est possible d'écrire :
  $P(B) = \sum_n P(B \cap A_n) = \sum_n P(B / A_n) \cdot P(A_n)$
- Formule de Bayes. C'est une conséquence de la formule de la probabilité totale. Si un événement B peut être décomposé en une suite exhaustive finie ou infinie d'événements An, alors il est possible d'écrire :
  $P(A_n / B) = \frac{P(B \cap A_n)}{P(B)} = \frac{P(B / A_n) \cdot P(A_n)}{\sum_j P(B / A_j) \cdot P(A_j)}$

**Notion de variables aléatoires :**

Variable aléatoire : repérage mathématique d'un événement

Variable aléatoire discrète : utilise un sous-ensemble des entiers (nombre de points sur la face d'un dés, valeur de la carte).
L'événement A sera repéré par une valeur $X=x_1$ de la variable aléatoire X
$P(A) = P(X=x_1) = p_1$
$P(A \text{ ou } B) = P(X=x_1 \text{ ou } X=x_2) = p_1 + p_2$ si les événements sont incompatibles.
La fonction de répartition F(x) associée à la variable aléatoire X est définie par $F(x) = P(X \leq x)$.

Variable aléatoire continue : utilise un sous-ensemble des réels (instant d'apparition d'une défaillance).
La fonction de répartition est définie par $F(x) = P(X<x)$.
et la densité de probabilité par $F(x) = \int_{-\infty}^{x} f(t)dt \quad et \quad f(x)dx = P(x < X < x + dx)$

**Notion de fiabilité :**

La fiabilité R(t) est définie par $R(t) = P(T>t)$, si T est la variable aléatoire représentant l'instant de la défaillance d'un composant. La fiabilité représente la probabilité pour que le dispositif n'ait pas eu de défaillances entre 0 et t, donc qu'il fonctionne correctement.
La probabilité de défaillance ou défiabilité F(t) est définie par $F(t) = P(T \leq t)$ et $R(t) = 1 - F(t)$.
La densité de probabilité de panne f(t) est définie par $f(t) = dF/dt = - dR/dt$.
Le taux de défaillance h(t) est tel que $R(t) = e^{-\int_0^T h(t)dt}$

MTTF (Mean Time To Failure) : temps pour être en panne, pour un système non réparable [Bukowski, 2001]. Le MTTF est donc le temps moyen de fonctionnement **avant la première** défaillance.
La définition du MTTF est

$$MTTF = \int_0^{\infty} R(t)\,dt = \int_0^{\infty} \big(1 - F(t)\big)dt$$

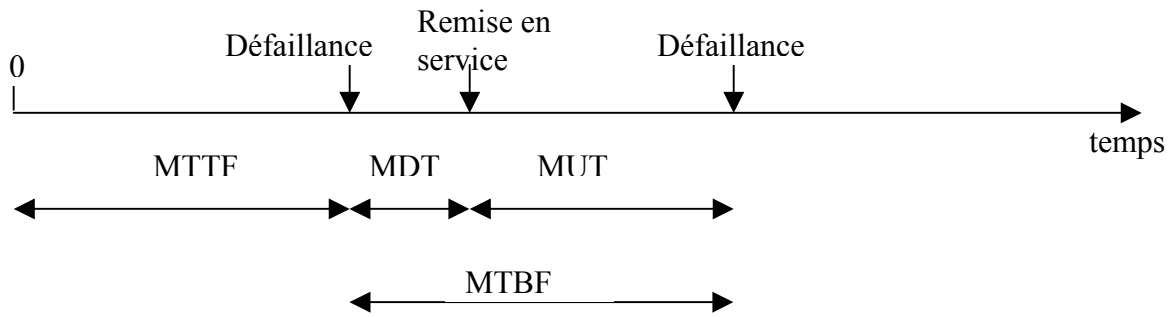
MTBF (Mean Time Between Failure) **:** temps entre pannes d'un **système que l'on répare**. MTBF ≈ MTTF = $1/\lambda$ dans le cas d'un composant simple / où le composant, après réparation, peut être considéré comme identique à ce qu'il était en début de vie (p 23 du précis de proba).

MTTR (Mean Time To Repair) **:** temps moyen mis pour réparer le système. Le MTTR est défini dans la CEI 61508 comme étant Mean Time To Restoration. Il faudrait alors préciser les termes $MTTR_{ep}$ pour Mean Time To Repair et $MTTR_{es}$ pour Mean Time To Restoration. La différence entre les deux est en fait lié au fait que l'on considère ou non le temps mis pour remettre en service l'équipement.

MDT (Mean Down Time) **:** durée moyenne d'indisponibilité ou de défaillance. Elle correspond à la détection de la panne, la réparation de la panne et la remise en service. Le MDT se décompose en : (temps mis pour détecter la défaillance) + (temps de réaction avant la mise en place des actions pour réparer) + ($MTTR_{es}$ ).
Remarque : la CEI 61508 confond à tort MDT et MTTR.

MUT (Mean Up Time) **:** durée moyenne de fonctionnement après réparation

On peut représenter ces différents temps par le graphique suivant :




Il est possible de confondre MTTF et MTBF en supposant MDT est faible devant le MUT. De plus, si on considère que le système est 'comme neuf' suite à réparation, MUT peut être considéré comme équivalent au MTTF.

**Modélisation des défaillances de composants**

La modélisation 'classique' des défaillances de composants se fait par l'intermédiaire de leurs instants d'apparition. On appellera expérience l'opération reproductible qui sélectionne un élément / événement A (la défaillance d'un composant) dans un ensemble Ă. . Des sous-ensembles de Ă peuvent être :

Ind = {défaillances indépendantes de composants}
CCF = {défaillances dépendantes de composants}
Ind/simul = {défaillances indépendantes simultanées de composants}

- Un événement peut être la combinaison (avec un OU logique) d'événements élémentaires résultant de plusieurs expériences (suite à une cause ou à un stress donné, obtention de la défaillance du composant C1 ou du composant C2).
- Un événement peut résulter de plusieurs expériences élémentaires simultanées (suite à un stress donné, obtention de la défaillance du composant C1 et du composant C2).
- Deux événements sont incompatibles s'ils ne peuvent se produire simultanément (Les défaillances indépendantes et les défaillances de mode commun sont incompatibles. C'est une ou l'autre, mais pas les 2 simultanément).

Les événements élémentaires peuvent être mathématiquement repérés par la variable aléatoire T "Instant d'apparition de la défaillance". On associera alors à l'événement A une valeur T = t1 de la variable aléatoire T telle que P(A) = P(T = $t_1$). P(A) est la probabilité que la défaillance apparaisse à l'instant $t_1$ et T est une variable aléatoire continue qui prend ses valeurs dans l'ensemble des réels.

La fonction de répartition de la variable aléatoire T est définie par F(t) = Q(t) = P(T<t). Elle correspond à la probabilité de défaillance du composant sur l'intervalle [0,t]. Si on considère que la densité de probabilité suit une loi exponentielle, on a :

$$f(t) = \lambda \cdot e^{-\lambda t}$$

d'où $R(t) = e^{-\lambda t}$ et $Q(t) = 1 - e^{-\lambda t} = P(T < t)$

où t est assimilable à l'instant d'apparition de la défaillance de ce composant.

Si λt est suffisamment petit, alors la probabilité de défaillance est donnée par :
$Q(t) = P(T \leq t) = \lambda \cdot t$ qui sera notée $Q_t(t)$

De plus $MTBF = \int t\lambda e^{-\lambda t} dt = \frac{1}{\lambda} \approx MTTF$ *si le système est identique après réparation*

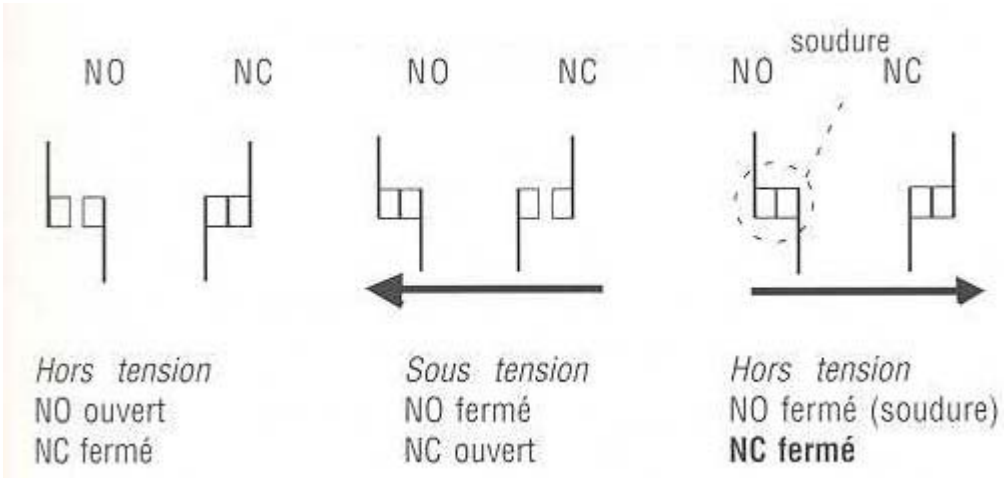
Remarque : Les opérations envisageables sur les événements (les défaillances de composants) portent sur les ensembles alors que les opérations envisageables sur les variables aléatoires (instant d'apparition de la défaillance) portent sur les probabilités. Travailler sur les probabilités de défaillances revient à travailler sur la fonction de répartition de la variable aléatoire T.

# ANNEXE 7 – Bases de la logique câblée

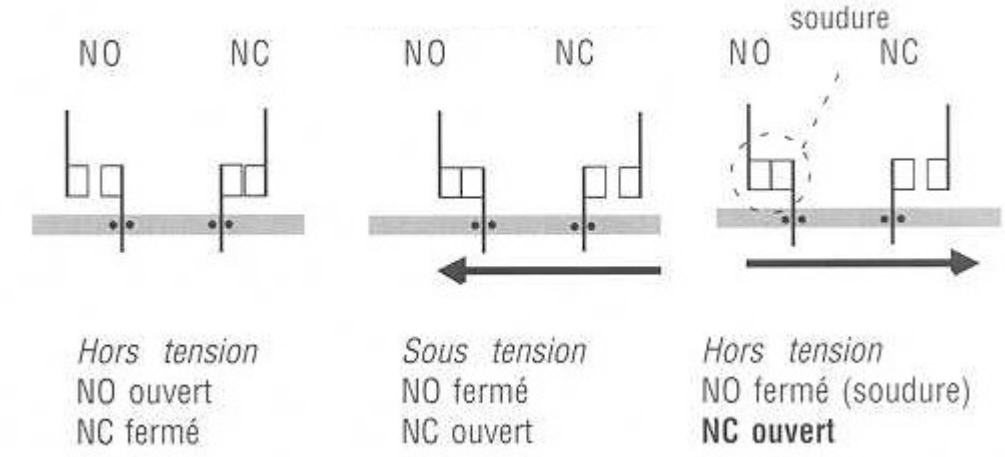
L'élément de base de la logique câblée de sécurité est le relais à contacts guidés. Il est constitué d'une bobine et de différents contacts N/O (normalement ouvert) ou N/F (normalement fermé).

- ✓ Contact N/O : si la bobine du relais n'est pas alimentée, le contact est ouvert, si la bobine du relais est alimentée le contact est fermé.
- ✓ Contact N/F : si la bobine du relais n'est pas alimentée, le contact est fermé, si la bobine du relais est alimentée le contact est ouvert.

La propriété d'un relais à **contacts guidés** réside dans la liaison qui existe entre les différents contacts ; ceci permet de rendre la fermeture simultanée des contacts N/O et N/F impossible. Ainsi, les contacts N/F d'un relais à contacts guidés peuvent être utilisés pour contrôler l'état des contacts N/O associés.




**Situations rencontrées pour certains relais à contacts non-guidés**



**Situations rencontrées pour des relais à contacts guidés**


Soudure d'un contact de relais à liaison mécanique

*En cas de soudure d'un contact à ouverture, les contacts à fermeture ne doivent plus pouvoir se fermer lors de la désexcitation de la bobine.*

*En cas de soudure d'un contact à fermeture, les contacts à ouverture ne doivent plus pouvoir se fermer lors de l'excitation de la bobine.*

Le guidage des relais va donc permettre de considérer le collage simultané de deux relais sensés être d'états contraires comme impossible. Ceci rejoint la démarche qualitative proposée par la norme **EN 954**.

# ANNEXE 8 - Taux de défaillances d'un canal

Cette annexe a pour objectif de déterminer le taux de défaillance d'un canal en prenant l'exemple de l'automate S7-216 de SIEMENS, constitué d'une unité centrale (6ES7 216-2BD00-0XB0), et d'un module d'extension entrée/sortie tout ou rien (6ES7 223-1PL00-0XA0).

Les contacts établis avec la société SIEMENS ont permis d'obtenir des MTBF de composants. Les caractéristiques du module d'extension choisi n'ayant pas été précisées, nous avons choisi de considérer le MTBF d'un module proche : le 6ES7 223-1HF00-0XA0 (4 entrées 24 V.c.c., 4 sorties relais 2 A).

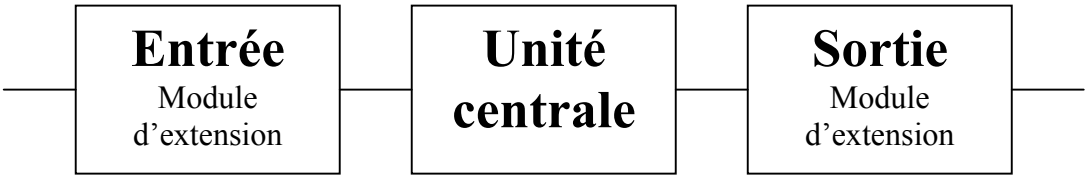



Diagramme de fiabilité de l'automate S7-216

Le diagramme ci-dessus se traduit de la façon suivante : $\lambda_{Architecture}=\lambda_{Entrée}+\lambda_{UC}+\lambda_{Sortie}$ .

On a :

$$\lambda_{Entrée}=\lambda_{Sortie}=\frac{\lambda_{Module}}{Nombred'E/S}=\frac{1}{MTTF_{Module}*Nombred'E/S}\approx\frac{1}{MTBF_{Module}*Nombred'E/S}.$$

Et, $\lambda_{UC}=\frac{1}{MTTF_{UC}}\approx\frac{1}{MTBF_{UC}}$

Les données fournies par SIEMENS sont :

$MTTF_{Module}=228ans$ , et $MTTF_{UC}=25ans$

donc

$$\lambda_{Architecture}\approx 2*\frac{1}{MTBF_{Module}*Nombred'E/S}+\frac{1}{MTBF_{UC}}$$

$$\lambda_{Architecture}\approx\frac{2}{228*8}+\frac{1}{25}\approx 4{,}11.10^{-6}\text{ défaillances par heure}$$

# ANNEXE 9 – Caractéristiques de l'API PSS 3056 e la société PILZ

| | |
|---|---|
| **Sorties (Dual-pole Outputs)** | |
| Nombre | 8 |
| Courant de sortie à "1" | 2 A |
| Intervalle autorisé | 0 … 2,5 A |
| Courant résiduel pour le signal "0" | Signaux de test 5 mA (pendant 500 µs max.) |
| Niveau du signal à "0" | 12 VDC |
| Niveau du signal à "1" | De -2 à 2,5 VDC |
| Retard de commutation de sortie | 50 µs |
| Temps mort pendant l'auto-test | < 200 µs |
| **Entrées** | |
| Nombre | 32 |
| Niveau du signal à "0" | -3 à +5 VDC |
| Niveau du signal à "1" | +15 à +30 VDC |
| Courant d'entrée | 6 mA |
| Retard d'entrée | 1 ms |
| **Caractéristiques mécaniques** | |
| Taille (H × L × P) | 223 × 277 × 155 mm |
| Poids | 3740 g |

# ANNEXE 10 - Détection des fautes sur les API

**TSX Micro** de Schneider

- Débordement ou erreur arithmétique
- Débordement période de tâches
- Débordement d'index
- Débordement du chien de garde
- Perte d'évènements
- Défaut horodateur
- Défaut d'E/S de tâche
- Défaut E/S
- Défaut CPU
- Test du câblage
- Contrôle d'acquisition des entrées des tâches
- Contrôle de la mise à jour des sorties des tâches
- Validité de la sauvegarde du prg application
- État pile cartouche
- État pile processeur

**S7-200** de Siemens

- débordement ou valeur numérique illicite
- division par 0
- débordement de la table
- lecture dans table vide
- conversion d'une valeur non DCB en valeur binaire
valeur ASCII non convertible en valeur hexa. correcte
- file d'attente des IT de comm, d'entrées, cycliques débordée
- erreur de prg° détectée à l'exécution
- trop d' E/S TOR ou analogiques
- erreur de configuration
- réception interrompue : erreur de parité, nb max de caractères atteint, expiration de temporisation, paramètres d'entrée erronés / condition de début ou fin manquante, par commande d'inhibition utilisateur
- Erreur totale de contrôle dans le prg compilé
- Dépassement du temps de cycle
- EEPROM interne défaillante : erreur totale de ctrl dans le prg utilisateur, dans les paramètres de configuration, dans les données utilisateur, dans les valeurs par défaut de la table des sorties
- Cartouche mémoire défaillante : erreur totale de ctrl dans le prg utilisateur, dans les paramètres de configuration, dans les données utilisateur, dans les valeurs par défaut de la table des sorties
- Erreur logicielle interne
- Erreur d'adressage indirect
- Cartouche vide ou prg non compris par cette CPU
- Erreur de plage ( adresse ou lors de l'écriture en mémoire non volatile)
- Erreur dans le champ de comptage d'une opération
- Prg trop grand pour la compilation
- Débordement bas de la pile

- Opération, repère, paramètre illicite
- Type de bloc inconnu
- Paramètre incorrect dans les informations de configuration
- Bloc trop grand pour la mémoire de l'AP
- Longueur inattendue pour la zone de données communication
- Temps incorrect
- Débit en bauds incorrect
- Numéro d'interface incorrect
- Erreur lors de transmission / réception de données à l'AP
- Dépassement du délai de communication
- Erreur d'adresse dans le réseau
- Horloge de l'AP non définie
- Défaillance matérielle de l'AP
- Type de données non prise en charge
- Objet inexistant ou erreur longueur
- Demande de communication incorrecte
- Erreur de communication : unité MPI introuvable
- Erreur de communication : IT d'unité occupée
- Horodateur dans le prg ≠ horodateur dans l'AP

**RÉSUMÉ :** La conception de la sécurité du système de commande d'une machine entre dans le cadre général de la sûreté de fonctionnement et doit porter sur la tolérance, l'évitement et la prévision des fautes. Après avoir rappelé le contexte particulier de la sécurité des machines, diverses techniques sont données pour traiter ces différents aspects vis-à-vis des fautes indépendantes.

On aborde ensuite des défaillances de mode commun inhérentes aux structures redondantes. Suite au rappel des principales définitions et du processus de 'création' des défaillances de mode commun, une étude bibliographique recense les principales méthodes pour prendre en compte ces défaillances. Une attention particulière est portée à la modélisation des défaillances matérielles de mode commun. La comparaison de différents modèles conduit à retenir le facteur β pour les applications en sécurité des machines. Les conditions d'utilisation de ce modèle sont données.

Les différentes architectures classiquement utilisées pour assurer la Sûreté de Fonctionnement d'un système de commande ont été analysées de façon qualitative (comportement prévisible en présence de défaillances) et quantitative (détermination de la probabilité de défaillance dangereuse PDF). L'influence du facteur β et du taux de couverture des diagnostics sur la PDF a été étudiée pour l'architecture 1oo2D. Enfin, deux conditions de mise en œuvre d'une architecture hétérogène de type 1oo2 ont été étudiées : synchronisation les deux canaux et développement d'un comparateur.

Les travaux menés font apparaître l'incapacité des modèles existants à représenter de façon satisfaisante les défaillances de mode commun de deux canaux hétérogènes. Ils montrent aussi les limites de la quantification par le calcul de la PDF, dues aux difficultés à évaluer précisément les taux de défaillances des composants, les taux de couverture des tests de diagnostic ou encore le facteur β.

**Mots clés :** Défaillances de mode commun, Architecture, Sécurité des machines, SIL

---

**SUMMARY :** The safety of machine control systems has to be treated within the general framework of the dependability and must relate to the faults tolerance, avoidance and forecasting. After having described the particular context of the safety of machinery, some techniques are given to deal with these various aspects for the independent faults.

Then the common mode failures inherent to the redundant structures are dealt with. Following the recall of the main definitions and the process of creation of common mode failures, a bibliographic study makes an inventory of the principal methods to take into account these failures. A detailed attention is paid to the modelling of the hardware common mode failures. The comparison of various models shows that the β factor is appropriate for the applications in safety of machinery. The conditions for the use of this model are given.

Different architectures classically used to ensure the dependability of a control system were analysed in a qualitative way (foreseeable behaviour in the presence of failures) and quantitative (determination of the probability of dangerous failure PDF). The influence of the β factor and the coverage rate of the diagnostic tests on the PDF was studied for the 1oo2D architecture. At least, two conditions for the design of a type 1oo2 heterogeneous architecture were studied : synchronisation of the two channels and design of a comparator.

This work reveals the incapacity of the existing models to represent in a satisfactory way the common mode failures of two heterogeneous channels. It also shows the limits of the quantification by the calculation of the PDF, due to the difficulties of precisely evaluating the failure rates of the components, the coverage rates of the diagnostic tests or the β factor.

**Key words :** Common Mode Failures, Architecture, Safety of machinery, SIL