Probabilité de défaillance dangereuse d'un système : explications et exemple de calcul.

P. Lamy

INRS

INSTITUT NATIONAL DE RECHERCHE ET DE SÉCURITÉ
pour la prévention des accidents du travail et des maladies professionnelles

Probabilité de Défaillance Dangereuse d'un système : explications et exemple de calcul

Note Scientifique et Technique n° 225

Pascal LAMY N / IET

septembre 2002

Résumé

Dans le domaine de la sécurité des machines, de nouvelles normes (CEI 61508 et CD CEI 62061) sont apparues et visent à quantifier la probabilité de défaillance dangereuse, nommée PFD, des pannes matérielles d'un système. Cette quantification est un des moyens qui permet de classer les systèmes par niveau de sécurité (SIL).
Ce document rappelle, dans un premier temps, les différentes notions utiles pour calculer la probabilité de défaillance dangereuse, notamment l'explication des différents paramètres intervenants dans l'estimation de la PFD. Il précise ensuite la démarche à suivre pour réaliser ce calcul.
Finalement, un exemple d'application sera détaillé en insistant sur une architecture à base d'automates identiques et une autre architecture à base d'automates hétérogènes. Cette note scientifique et technique se veut comme un support permettant de mieux appréhender le calcul de la probabilité de défaillance dangereuse. Les hypothèses prises seront détaillées et plusieurs méthodes de calcul seront explicitées.

Sommaire

1 Introduction

L'utilisation de systèmes électroniques complexes (par exemple programmables) dans les systèmes de commande des machines a nécessité de modifier l'approche au niveau de la conception de ces machines.

De nouvelles normes[1] [2] proposent ainsi des prescriptions techniques pour la conception et l'intégration de systèmes électroniques complexes dans la partie commande des machines. Pour caractériser **les pannes aléatoires matérielles**, il est de plus proposé de quantifier les défaillances dangereuses de ces systèmes en utilisant la probabilité de défaillance dangereuse (PFD).

Ce document se propose d'expliquer les différentes méthodes permettant de calculer **la moyenne de la probabilité de défaillance dangereuse sur demande due aux pannes matérielles aléatoires (cas de la faible sollicitation)**. On expliquera aussi le calcul de la probabilité de défaillance dangereuse par unité de temps (cas de la forte sollicitation).
Ce guide s'utilise dans le cas de systèmes de commande pour des machines (au sens de la norme CD CEI 62061[1]) constitués d'un ensemble de sous-systèmes dont on connaît les caractéristiques de fiabilité.

Dans un premier temps, nous ferons un rappel sur la notion de probabilité et nous définirons les termes nécessaires au calcul.
Le calcul de la probabilité de défaillance dangereuse commence avant tout par une étude fonctionnelle du système qui permet de décomposer l'ensemble afin d'identifier les fonctions de sécurité. Cette étude fonctionnelle va aussi permettre d'obtenir une vue architecturale (matérielle) du système.
En fonction de la décomposition, il faut ensuite quantifier les défaillances de chaque partie. Les modèles de calcul proposés permettront ensuite de prendre en compte les apports des différents sous-systèmes et de fournir la probabilité de défaillance dangereuse de l'ensemble.

2 Définitions

Système [3]: Ensemble déterminé d'éléments interconnectés ou en interaction.
Exemples :

- Arrêt d'une machine suite à la détection d'une personne. Le système pourrait être constitué d'un barrage immatériel qui détectera l'intrusion d'une personne, d'un module de traitement (APIdS Automate Programmable Industriel dédié à la Sécurité ou autre) et de la commande d'arrêt.
- Si on utilise un autre point de vue, on peut aussi considérer comme système un automate programmable dont la fonction sera de positionner une de ses sorties en fonction de ses entrées. Dans ce cas, le système API (Automate Programmable Industriel) sera constitué très schématiquement de la carte d'acquisition, de l'unité centrale (CPU) et de la carte de sortie.

On constate donc que la notion de système dépend fortement du niveau auquel on se place. Dans la suite du document, on s'intéressera plus particulièrement aux fonctions de sécurité et

donc aux systèmes tels que, par exemple, celui réalisant l'arrêt d'une machine suite à la détection d'une personne.

Sous-système : Ensemble de modules (automate programmable par exemple). Selon la norme CEI 61508[2], un élément d'un système peut-être un autre système appelé dans ce cas sous-système. Les sous-systèmes peuvent être eux-mêmes soit un système de commande, soit un système commandé composé de matériel et de logiciel en interaction avec l'être humain.

Module[3]: Ensemble fonctionnel de composants encapsulés formant un tout (circuit d'entrée ou de sortie, carte électronique).

Composant[2] : La plus petite partie d'un module, d'un sous-système ou d'un système qu'il est nécessaire et suffisant de considérer pour l'analyse du système. Cette plus petite partie pourra être limitée par les données disponibles donnant les caractéristiques du composant. On sera parfois obligé de rester au niveau module pour l'analyse.
La décomposition proposée est donc :
composant / module / sous-système / système.

Redondance[3]: Existence dans une entité de plus d'un moyen pour accomplir une fonction requise. On parlera de canaux en redondance.

Canal **(selon CEI 61508[2])**: Module ou sous-système exécutant une fonction indépendante. Un canal peut être constitué de plusieurs modules.

Fonction de sécurité (selon [2] ***et 62061***[1]***)***: fonction réalisée par un système relatif à la sécurité pour assurer ou maintenir un état de sécurité de la machine par rapport à un évènement dangereux spécifique.

Etat de sécurité (selon [2]): Etat de la machine lorsque la sécurité est réalisée.

Panne aléatoire [2]: Panne en général due uniquement au matériel (hardware), résultant de divers mécanismes de dégradation et dont l'instant exact d'occurrence n'est pas prévisible. Par contre, la moyenne de la probabilité de défaillance dangereuse due aux pannes aléatoires matérielles peut être quantifiée de façon précise, par le taux de défaillance λ notamment.

Panne systématique [2]: Par opposition aux pannes aléatoires, ce sont des défaillances reliées de façon systématique à une certaine cause, ne pouvant être éliminées que par une modification de la conception, du processus de fabrication, des procédures d'exploitation, de la documentation ou d'autres facteurs.

Test de diagnostic : Selon [1] et [2], test en ligne (en fonctionnement) pour détecter des défauts (comme un WatchDog – ou chien de garde - selon ISA-S84.01-1996[5], un test de la mémoire vive (RAM) ou de la mémoire programme).
D'après ISA-S84.01-1996[5] (page 70), les tests de diagnostic sont effectués périodiquement et automatiquement pour détecter les défauts latents cachés qui empêchent le SIS (Safety Integrated System) de répondre à une demande.
Les tests de diagnostic agissent au niveau composant/interne (et non pas au niveau de la fonction de sécurité) et permettent de détecter les erreurs aléatoires (dues au matériel). Dans

certains cas, ils permettent de détecter les erreurs systématiques (telles que les bugs logiciels : cf. ISA[5] B.9.3.6). Un WatchDog peut détecter des sauts de programme dus à des bugs logiciel.
Les tests de diagnostics permettent de réparer rapidement et de limiter le temps passé en mode dégradé.
Il existe 2 types de tests de diagnostics :

- les diagnostics de référence : comparaison par rapport à une valeur prédéterminée comme la mesure de la période de temps (watch dog), le calcul de la somme de contrôle d'une mémoire programme (checksum), le rebouclage de toutes les sorties sur une entrée (détection des pannes des préactionneurs en cas de discordance). Selon Goble[4] (p.86), le taux de couverture de ces tests est généralement bon (entre 0.9 et 0.999),
- les diagnostics par comparaison à une unité opérationnelle (comparaison de tables de données logiques entre 2 ou plusieurs canaux).

Taux de couverture des tests de diagnostic (ou Diagnostic Coverage DC) : Historiquement, le taux de couverture a été introduit lors des programmes spatiaux américains pour caractériser la tolérance aux fautes logicielles. Le taux de couverture a été ainsi défini comme étant la probabilité (nombre compris entre 0 et 1) qu'une panne soit détectée. La définition la plus générale est :

$$taux\ de\ couverture = \frac{somme\ des\ taux\ de\ défaillances\ des\ composants\ détectés\ en\ panne}{somme\ des\ taux\ de\ défaillances\ de\ tous\ les\ composants}$$

La norme CEI61508[1] définit ce taux de couverture pour les tests automatiques de diagnostic comme étant le rapport du taux de défaillance des pannes dangereuses détectées (par un test de diagnostic) sur le taux de défaillance total des pannes dangereuses (détectées et non détectées).

$$DC = \frac{\sum \lambda_{DD}}{\sum \lambda_{total\ dangereuses}}$$

L'évaluation de DC se fait par exemple (cf. ISA[4] p 201) par une analyse des modes de défaillances et des diagnostics. On cherchera ainsi à déterminer les défaillances possibles et à savoir si elles peuvent être détectées, puis on calculera le rapport ci-dessus à l'aide des taux de défaillances et non pas uniquement à l'aide du nombre des défaillances.

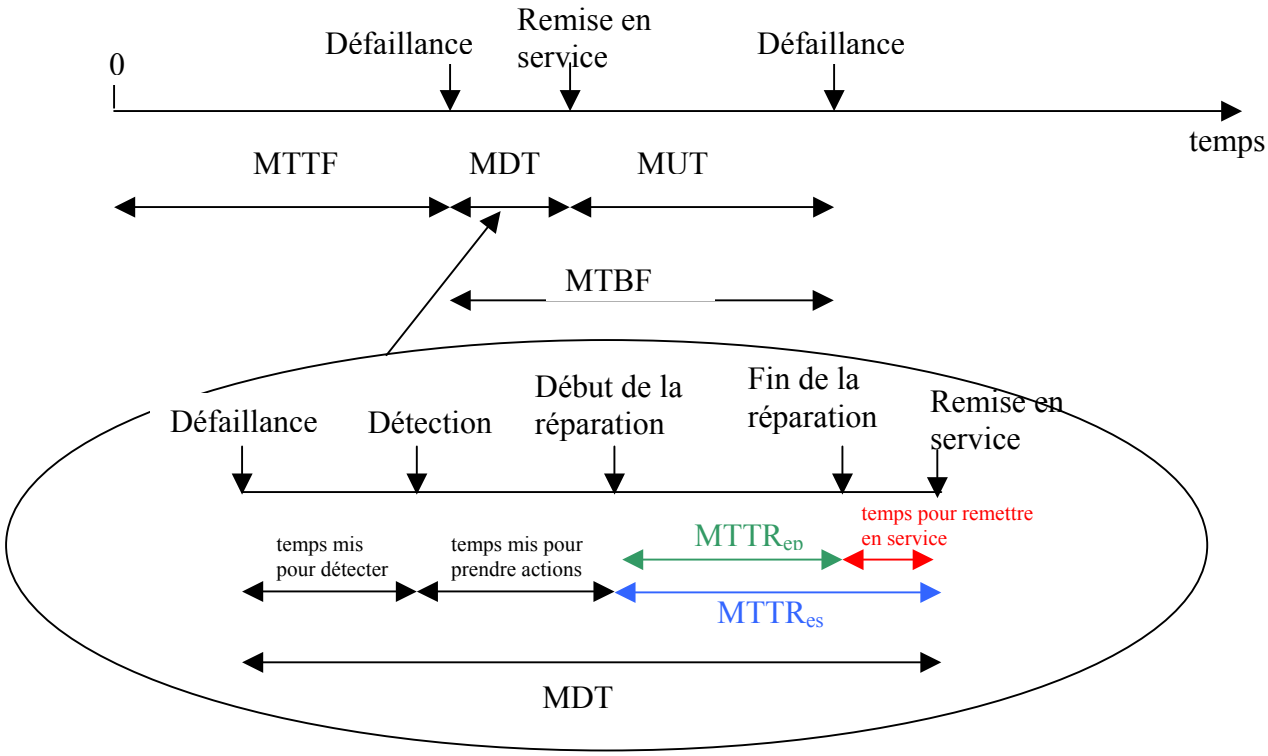
Proof Test : Définition de CEI62061 et de CEI61508: test périodique hors ligne réalisé pour détecter des pannes dans un système de telle sorte que le système puisse être réparé afin de revenir dans un état équivalent à son état initial.
Selon ISA-S84.01-1996[5], dans le cas où le diagnostic coverage serait minimum ou insuffisant (si on ne peut pas ou ne sait pas réaliser un test de diagnostic satisfaisant), on pourra augmenter la fréquence du proof test. En augmentant la fréquence du proof test, on vérifiera plus souvent que la fonction de sécurité est bien disponible.
Le proof test est exécuté au niveau du système. C'est un test fonctionnel de la fonction de sécurité hors fonctionnement automatique (activité périodique devant être conduite selon une procédure afin de détecter les défauts latents qui empêchent le système de sécurité de remplir sa fonction de sécurité ; le système de sécurité entier doit être testé) alors que le test de diagnostic est plutôt une détection interne en fonctionnement. Le proof test permet de détecter les pannes latentes qui n'ont pas été vues par les tests de diagnostics. On peut aussi effectuer des proof tests au niveau des sous-systèmes mais cela s'applique plus à l'industrie du process.

Temps de mission : Temps pendant lequel le système est utilisé pour son application de sécurité. Lorsque des proof tests sont réalisés, ce temps de mission sera considéré comme égal au temps entre deux proof tests consécutifs. Dans les cas où il n'y a pas de proof tests, on prendra le temps de mission égal au temps de vie de l'appareil (MTTF).

Fiabilité (reliability en anglais)[4] : Probabilité qu'un dispositif accomplisse sa fonction voulue lorsque ce dispositif travaille dans les limites prévues et à un instant donné. La fiabilité se note R. C'est une fonction du temps telle que :

R(t) = P(T>t); probabilité que le temps T (temps où la panne intervient) soit supérieur au temps t (temps opérationnel ou temps de fonctionnement; définition de l'ISA[5]); on peut aussi la définir comme la probabilité que le système ne soit pas défaillant pendant l'intervalle de temps [0,t]. La fiabilité n'a de sens que si elle est associée au temps de mission. C'est un nombre sans dimension.

Différents temps caractéristiques :



MTTF[3] *:* Mean Time To Failure. Pour un système que l'on ne répare pas, le MTTF est le temps moyen de fonctionnement **avant la première** défaillance.
La définition du MTTF est :

$$MTTF = \int_0^{\infty} R(t)\,dt$$

MTBF[3]**:** Mean Time Between Failure. Le MTBF n'a de sens que pour un système réparable.. C'est la durée moyenne entre deux défaillances consécutives. MTBF ≈ MTTF = 1/λ dans le cas d'un composant simple de taux de défaillance λ qui, après réparation, peut-être considéré comme identique à ce qu'il était en début de vie.

MTTR[3] ***:*** Mean Time To Repair. C'est le 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 $MTTR_{ep}$[6] pour Mean Time To Repair et $MTTR_{es}$[6] pour Mean Time To Restoration. La différence entre les deux est liée au fait que l'on considère ou non le temps mis pour remettre en service l'équipement, le $MTTR_{es}$ l'incluant.

MDT[3] (donné équivalent à tort au MTTR comme dans la CEI 61508): Mean Down Time. C'est la 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 + $MTTR_{es}$.
Pour les systèmes où la réparation s'effectue hors-ligne (il y a possibilité d'enlever l'élément défaillant sans perturber le fonctionnement du système), le $MTTR_{es}$ ne comprend plus le $MTTR_{ep}$. Cela permet de diminuer le temps d'indisponibilité du système.

MUT[3] **:** Mean Up Time. C'est la durée moyenne de fonctionnement après réparation.

En règle générale, on confond le MTTF et le MTBF car on suppose que le MDT est faible devant le MUT. Le MUT est de plus considéré comme équivalent au MTTF (le système réparé est considéré comme neuf alors que ce n'est pas forcément le cas (tout n'a pas été changé). Sans approximation, MTBF = MUT + MDT.

Intervalle entre tests de diagnostics : Les tests de diagnostics doivent être effectués à une certaine fréquence afin de détecter les pannes. Cet intervalle de temps intervient dans le calcul de la probabilité de défaillance dangereuse. En fait la norme CEI 61508 précise les cas :

- pour les systèmes à faible sollicitation, le $MTTR_{es}$ est considéré comme étant la somme de l'intervalle entre les tests de diagnostic et le temps de réparation,
- pour les systèmes à forte sollicitation, pour lesquels l'intervalle entre tests de diagnostics n'est pas d'un ordre de grandeur inférieur au taux de demande moyen, l'intervalle entre tests de diagnostic s'ajoute au $MTTR_{es}$.

Dans la mesure où les pannes sont considérées comme étant détectées et réparées rapidement, on a vu ci-dessus que le $MTTR_{es}$ est négligeable devant le temps entre deux pannes consécutives. En fait, il faut s'assurer que l'intervalle entre tests de diagnostics est suffisamment petit pour que le temps d'indisponibilité du système soit petit devant le MTBF.
Finalement, on pourra envisager un intervalle entre tests de diagnostics inférieur d'un facteur 100 (valeur d'expert) au MTBF de l'équipement afin de s'assurer que l'indisponibilité de l'équipement sera négligeable devant le MTBF.

Probabilité d'une défaillance par unité de temps (heure, année) ou taux de défaillance*:*
Notée λ(t) pour un fonctionnement continu du système. C'est la probabilité pour que le système soit défaillant. Cette définition s'applique pour tout type d'éléments (système, sous-système, module, composant, canal).

Probabilité de défaillance sur demande PFD(t) (probability failure on demand) *:* C'est la probabilité sur l'intervalle de temps [0,t] que le système ne puisse pas exécuter la fonction pour laquelle il a été conçu au moment où la demande de cette fonction est faite. C'est un nombre sans dimension.
Dans le cas d'un système mono canal :
PFD(t) = 1 - Fiabilité = 1- R(t); cette notion est en fait à la base des probabilités. La somme de la probabilité d'un événement et de son complément doit faire 1.

Probabilité moyenne de défaillance sur demande PFD_{avg} (Average of the probability failure on demand) : C'est la valeur moyenne par rapport à l'intervalle de temps entre proof test (test fonctionnel) de la probabilité de défaillance sur demande[5]. Selon l'existence de proof test ou non, la valeur moyenne se calculera par rapport à l'intervalle de temps T_i entre ces proof tests ou par la valeur moyenne par rapport au temps de vie du composant (s'il n'y a pas de proof test, T_i = MTTF).

$$PFD_{avg}(T_i) = \frac{1}{Ti}\int_0^{T_i} PFD(t)\,dt$$

Cette grandeur s'utilise dans le cas des systèmes à faible sollicitation et c'est un nombre sans dimension.

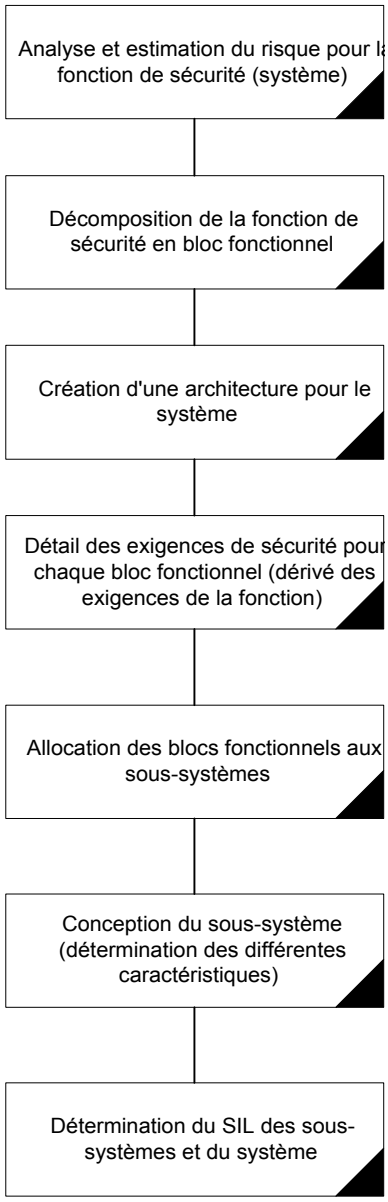
Probabilité d'une défaillance dangereuse par unité de temps (heure, année) ou taux de défaillance dangereuse : Noté $\lambda^D(t)$ pour un fonctionnement continu du **système**. C'est la probabilité que le **système** soit défaillant de telle sorte qu'il soit incapable d'exécuter la fonction de sécurité attendue. $\lambda^D(t)\,dt$ représente la probabilité d'avoir une défaillance entre (t,t + dt) sachant qu'il n'y a pas de défaillance sur l'intervalle [0,t].

Probabilité d'une défaillance dangereuse par heure PFH (probability of a dangerous failure per hour):
Cette notion prend en compte les intervalles entre tests. S'il n'y a pas de moyens de réparation, $PFH = \frac{PFD}{T_i}$. Cette grandeur s'utilise dans le cas des systèmes à forte sollicitation.

Disponibilité (availability en anglais)[4] : Probabilité pour qu'un dispositif soit opérationnel au temps t. Le système peut avoir été réparé dans le passé.

3 Logigramme pour le processus de conception

Le logigramme suivant représente les différentes étapes de conception d'une fonction de sécurité. Il suit la démarche proposée par les normes CEI 61508 et CD CEI 62061.



4 Etude fonctionnelle du système

Dans un premier temps, il faut procéder à une estimation du risque afin de déterminer les dangers possibles pour l'opérateur et donc les fonctions de sécurité qui devront être mises en place pour s'en affranchir.

4.1 Les fonctions de sécurité

Il faut recenser les fonctions de sécurité que le système doit réaliser. On décrira chacune de ces fonctions. On doit définir chaque fonction de sécurité en une structure de blocs fonctionnels et donner pour chaque bloc:

- la description de sa structure,
- les exigences fonctionnelle,
- la définition des entrées/sorties (informations transmises telles que la vitesse, la position,…).

4.2 Etude de l'architecture du système

Il faut décomposer le système en sous-systèmes puis en modules afin d'avoir une finesse d'analyse suffisante en fonction des données disponibles des différents éléments. (Voir la figure d'allocation des exigences de sécurité des blocs fonctionnels aux sous-systèmes de la norme CD CEI 62061).

A chaque niveau de décomposition, on essaiera de descendre un niveau plus bas. On arrêtera la décomposition en fonction des informations disponibles (connaissance des taux de défaillances par exemple).

4.3 Allocation des blocs fonctionnels aux sous-systèmes

Il faut ensuite affecter à chaque bloc fonctionnel un sous-système.
La représentation en bloc fonctionnel correspond à une vue de l'esprit du système. Par contre, le passage aux sous-systèmes correspond à la vue matérielle.
On peut allouer plus d'un bloc fonctionnel à un simple sous-système mais on ne peut pas allouer un bloc fonctionnel à plusieurs sous-systèmes si ces sous-systèmes ont des exigences fonctionnelles différentes (selon CD CEI 62061).
Selon la norme CD CEI 62061[1], les fonctions de diagnostic ne doivent pas être inclues dans ces blocs fonctionnels (ce doit être des blocs à part).

5 Etude probabiliste

5.1 λ d'un composant

Le taux de défaillance d'un composant électronique élémentaire sera donné par le constructeur de ce composant.

Dans la plupart des cas, le constructeur ne fournit qu'un taux de défaillance λ global (λ du système ou λ des sous-systèmes). Toutefois, il ne sera pas nécessaire, en général, de descendre au niveau composant, ce guide étant fait pour un assemblage de sous-systèmes.

5.2 Défaillance dangereuse et défaillance sûre

5.2.1 Préambule

La description ci-dessous est abordée au sens de la norme CEI 61508 ou CD CEI 62061 qui considère que l'on peut déterminer le taux de couverture des pannes dangereuses (DC) et des pannes sûres (CS). Cela suppose donc au préalable la détermination des défaillances dangereuses des fonctions de sécurité du système.

5.2.2 Définitions des états

Le système peut se trouver dans 4 types d'états

- un état normal,
- un état normal dégradé,
- un état de sécurité,
- un état de défaillance dangereuse.

Etat normal

Dans l'état normal, la fonction de sécurité du système est valide (le système peut répondre à une sollicitation) et il n'existe pas de défaillance.

Etat normal dégradé

Dans l'état normal dégradé, la fonction de sécurité du système est valide (en fonctionnement ou hors ligne), des composants du système pouvant être défaillants. Le système peut réagir lors de l'arrivée d'un événement dangereux (intrusion d'une personne dans une zone contrôlée par exemple).
On peut avoir des fautes latentes ou une accumulation de fautes. Un état non dangereux peut très bien comprendre une accumulation de fautes (cela correspond à la catégorie 4 de la norme EN 954). Cette accumulation de fautes ne va pas entraîner un état dangereux. La fonction de sécurité est globalement respectée mais le temps de réponse a peut-être évolué, la finesse de détection également.

Etat de sécurité

Il s'agit d'un état du système pour lequel la sécurité est réalisée. Cet état peut faire suite à l'apparition d'un événement dangereux ayant entraîné l'activation de la fonction de sécurité, on est donc dans le cas du fonctionnement nominal du système.
Il peut aussi faire suite à l'apparition d'une défaillance. Le système peut ainsi entrer dans cet état (arrêt d'une machine par exemple) dès qu'une défaillance d'un ou plusieurs composants se produit:
- soit le système a détecté cette défaillance,
- soit la défaillance n'a pas eu d'action néfaste vis à vis de la sécurité et a placé le système dans un état sûr
  ➥ on parle alors de *défaillance sûre* ou *défaillance en sécurité.*

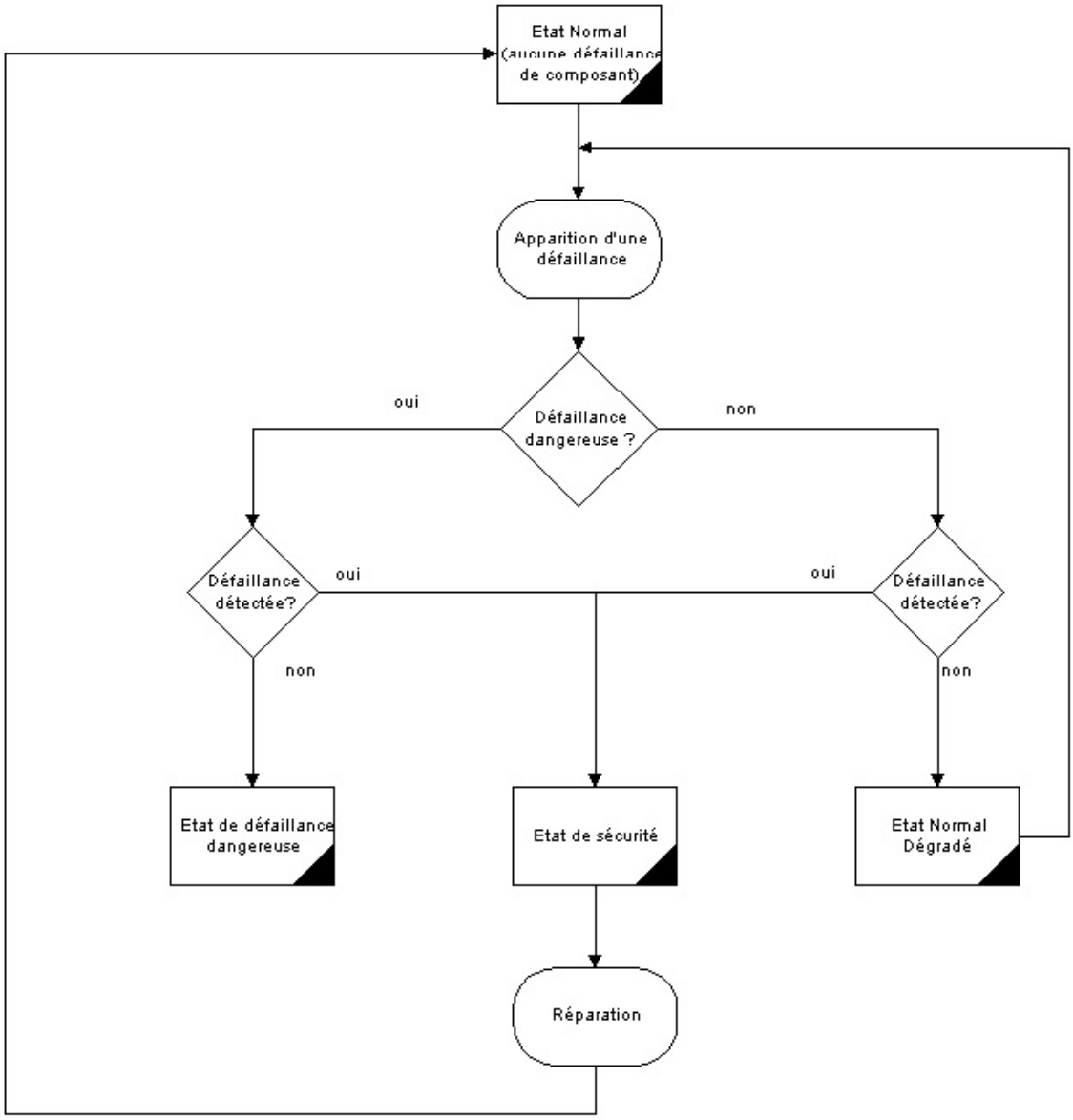
Etat de défaillance dangereuse

C'est un état du système où la fonction de sécurité n'est plus réalisée, un ou plusieurs composants étant défaillant.
Le système entre dans cet état dès qu'il présente un risque d'accident et manque de répondre à une demande d'activation de la fonction de sécurité.

➥ on parle alors de *défaillance dangereuse.*

5.2.3 Action des défaillances

Toutes les défaillances détectées en ligne par les tests de diagnostic sont qualifiées de défaillances *détectées*. Celles qui ne sont pas détectées sont qualifiées de défaillances *non détectées*.



Les défaillances sûres et détectées font passer le système de l'état normal à l'état de sécurité.

→ on leur associe le taux de défaillance λ^{SD}.

Les défaillances sûres et non détectées font passer le système de l'état normal à l'état dégradé.

→ on leur associe le taux de défaillance λ^{SU}.

Les défaillances dangereuses et détectées auraient la potentialité de faire passer le système de l'état normal à l'état de défaillance dangereuse mais leur détection associée à une stratégie

sécuritaire (passage en position de repli, arrêt, alarme) permet au système de le rediriger vers l'état de sécurité.

→ on leur associe le taux de défaillance λ^{DD}.

Les défaillances dangereuses et non détectées - et uniquement celles-ci - font passer le système de l'état normal à l'état de défaillance dangereuse

→ on leur associe le taux de défaillance λ^{DU}.

On peut aussi améliorer la sécurité du système en effectuant des proof tests de période plus petite. Le proof test vérifie la fonction de sécurité alors que les tests de diagnostic vérifient le fonctionnement interne du système. Ces proof tests sont effectués hors-ligne afin de tester la fonction de sécurité sans perturber le process (temps masqué) et de pouvoir ainsi replacer le système dans son état initial (état normal) après réparation. On voit donc apparaître ici 2 types de proof test :

- soit un proof test qui va vérifier la fonction de sécurité dans sa totalité (jusqu'à la mise en repli) avec donc un risque de blocage de la production et impossibilité de faire ce test sauf en fin de poste ;
- soit un proof test qui va vérifier, lors d'un temps mort du process, que la fonction de sécurité est efficace (par exemple test de la commutation d'une sortie sans mise en repli, test de la fonction de sécurité s'il n'y a pas de danger comme lors de la remontée du coulisseau d'une presse).

En règle générale, un proof test a une périodicité beaucoup plus importante (intervalle entre test plus grand) qu'un test de diagnostic et cela dû au fait qu'il est effectué hors-ligne. On souhaite ainsi détecter une panne qui ne serait pas forcément vue . Par exemple, le fait qu'un barrage immatériel ne remplisse plus sa fonction de détection et que cela n'ait pas été vu par des tests de diagnostic. Si on simule, par un proof test, l'intrusion d'une personne passant à travers le barrage immatériel, la fonction de sécurité sera sollicitée et on se rendra compte de l'inefficacité du barrage. Un proof test permettra de s'assurer que cette fonction de sécurité est bien remplie et donc que la défaillance dangereuse devient détectée.

5.2.4 Classification / quantification des défaillances

Si on considère que toutes les défaillances sont réparties en défaillances sûres et défaillances dangereuses, il faut bien préciser ce que recouvrent ces deux termes :

- *pour les défaillances dangereuses*, s'agit-il des défaillances dangereuses détectées seules ou des défaillances potentiellement dangereuses ?
- *pour les défaillances sûres*, s'agit-il des défaillances sûres détectées et non détectées ou des défaillances qui permettent d'entrer dans un état de sécurité ?

Ce paragraphe est destiné à éviter d'éventuelles confusions.

La norme CEI 61508[2] (partie 6, annexe C) considère que la répartition entre défaillances sûres et défaillances dangereuses peut être déterministe pour des composants simples. Pour des composants complexes, dont il n'est pas possible d'effectuer une analyse détaillée de chaque mode de défaillance, une répartition des défaillances en 50% sûres et 50% dangereuses est acceptée :

$\lambda^D = \lambda^S = 0{,}5 \times \lambda$

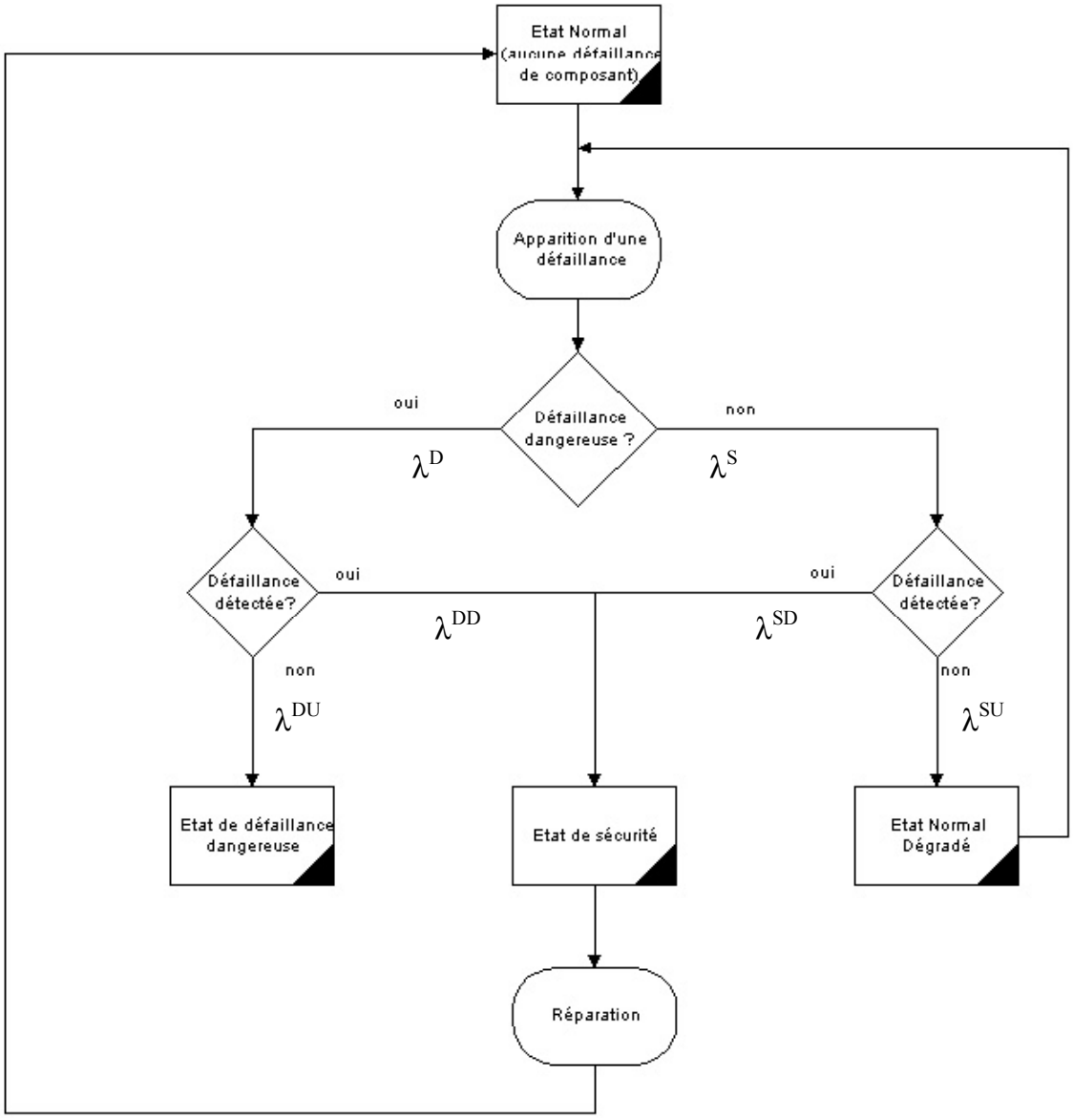
où λ est le taux de défaillance du composant.

La structure des dispositifs de sécurité actuels minimise les défaillances dangereuses. Dans notre cas, nous pourrons considérer des architectures où une seule défaillance est dangereuse, le but étant de déterminer la probabilité de défaillance dangereuse du système. Pour les systèmes ayant plusieurs canaux pour accroître la sécurité, il est moins probable qu'une défaillance dangereuse du matériel conduise à un état dangereux de l'ensemble ou à un état dans lequel la fonction de sécurité ne peut plus être exécutée.

Ici, il faut entendre par défaillances dangereuses les *défaillances potentiellement dangereuses* (défaillances dangereuses non détectées et défaillances dangereuses détectées non réparées) c'est-à-dire :

$$\lambda^{DU} + \lambda^{DD} = 0{,}5 \times \lambda$$

En fait, les défaillances dangereuses détectées et non réparées peuvent être considérées comme des défaillances en sécurité si le temps de réparation est court. Elles sont dangereuses uniquement tant qu'elles ne sont pas réparées.



En conclusion, nous décidons de prendre le raccourci suivant :

DEFAILLANCE DANGEREUSE = DEFAILLANCE DANGEREUSE NON-DETECTEE

DEFAILLANCE EN SECURITE = DEFAILLANCE DANGEREUSE DETECTEE ET REPAREE
+ DEFAILLANCE SURE NON-DETECTEE
+ DEFAILLANCE SURE ET DETECTEE

Compte tenu de l'hypothèse ci-dessus, il en résulte :

taux de défaillance dangereuse = $\lambda^{DU} = \lambda^{D}$
taux de défaillance en sécurité = $\lambda^{SU} + \lambda^{SD} + \lambda^{DD} = \lambda^{S}$

Et, si on considère qu'il y a équi-répartition entre les défaillances dangereuses et les défaillances sûres :

$$\lambda^{D} = \lambda^{S} = 0{,}5 \times \lambda$$

5.3 λ d'un module

Pour simplifier les calculs, la détermination du taux de défaillance d'un module (carte électronique, circuit d'entrée, circuit de sortie) se fera en utilisant l'égalité suivante :

$$\lambda_{\mathrm{mod}\,ule} = \sum \lambda_{composant}$$

Ce cas correspond au pire cas puisque l'on met tous les composants en série (sans tenir compte des éventuelles redondances).
Il nécessite de connaître les différents composants et leurs taux de défaillance.

On peut utiliser des logiciels tel que RAM Commander pour évaluer le λ des modules à partir des λ des composants.

5.4 Détermination des facteurs

5.4.1 Taux de couverture

En fonction des tests de diagnostic effectués, on pondère le taux de défaillance. On définit ainsi :

$\lambda^{DU} = \lambda^{D}$ = taux de défaillance dangereuse non détectée= 0.5 x (1-DC) x λ où DC est le taux de couverture des défaillances dangereuses.

La détermination du taux de couverture des diagnostics résulte dans la plupart des cas d'un travail d'expertise, pouvant être guidé par l'expérience ou par estimation. On les classe généralement en 3 catégories (CF CEI 61508[2]) : faible (inférieur à 60 %), moyen (compris entre 60 et 90 %), élevé (supérieur à 99 %).

En toute rigueur, la définition des taux de couverture donne pour les taux de défaillances dangereuses détectées :

$\lambda^{DD} = 0.5 \times DC \times \lambda$

Remarque:
Du fait de l'hypothèse que l'on a faite ($\lambda^{DU} = \lambda^{D}$), l'expression de λ^{DD} est à interpréter avec précaution. Dans notre cas, nous ne nous intéressons qu'à λ^{DU}.

5.4.2 Modes communs

Dans le cas de structure redondante (multiple canaux), les modes communs représentent les défaillances qui peuvent apparaître dans les canaux suite à une même cause (par exemple une erreur de conception ou une perturbation agissant sur les deux dispositifs).

L'introduction des défaillances de mode commun est généralement modélisée par le facteur β tel que :

$\lambda^{C} = \beta \lambda$: taux de défaillances de mode commun
$\lambda^{N} = (1-\beta) \lambda$: taux de défaillances indépendantes

$\lambda = \lambda^{C} + \lambda^{N}$

Dans le cas des défaillances dangereuses non détectées, on a
$\lambda^{DUC} = \beta \lambda^{DU}$: taux de défaillances dangereuses non détectées de mode commun
$\lambda^{DUN} = (1-\beta) \lambda^{DU}$: taux de défaillances dangereuses non détectées de mode normal.

Le choix du facteur β est laissé à l'appréciation subjective de l'utilisateur. Il varie généralement entre 0.5 et 5 %. (Voir la Norme CEI 61508[2] annexe D partie 6 qui essaie de formaliser l'évaluation de β et voir aussi la thèse de P. Charpentier[9]).

5.4.3 Intervalle de temps entre proof test

L'intervalle de temps entre proof test intervient dans le calcul de la PFD_{avg}. Cet intervalle de temps influe sur le niveau de sécurité. (cf. B15.2 ISA-S84.01-1996[5]).
Pour le calcul de PFD_{avg}, on utilise un intervalle de temps entre proof test Ti qui correspond à la fréquence à laquelle on teste tout le système et on suppose qu'après réparation, le système est comme neuf.
Pour la machinerie où les temps de cycle sont courts, on utilise généralement un seul intervalle de temps entre proof test pour vérifier la fonction de sécurité de l'ensemble du système.

Remarque : Dans certaines applications, l'industrie du process utilise des intervalles de temps entre proof test propres à chaque sous-système. On suppose ainsi que l'on teste fonctionnellement chaque sous-système indépendamment les uns des autres.

5.5 Détermination du λ d'un sous-système

Après avoir décomposé le système en sous-système, on calcule le taux de défaillance du sous-système lorsque cette donnée n'est pas fournie par le constructeur. Dans le cas d'un système utilisant un capteur d'acquisition, un automate programmable, et la commande de la partie opérative, l'automate programmable sera considéré comme un sous-système. Le sous-système automate programmable est constitué d'une carte d'acquisition, d'un ensemble logique de traitement, d'une carte de sortie.
On peut représenter un automate programmable comme étant constitué de 3 composants :

Unité d'acquisition	Unité de traitement	Unité de sortie

$$\lambda_{\mathrm{mod}\,ule} = \sum \lambda_{composant} = \lambda_{unité\ d'acquisition} + \lambda_{unité\ \ de\ \ traitement} + \lambda_{unité\ de\ sortie}$$

Il existe aussi des outils logiciels permettant de calculer le taux de défaillance d'un ensemble à partir des éléments constituant cet ensemble (par exemple RAM Commander, BQR Suite).

5.6 PFD (Probabilité de défaillance dangereuse) du système

Le calcul de la probabilité de défaillance de l'ensemble sera déterminé après avoir modélisé le système par un arbre des causes, un diagramme de Markov ou des blocs diagramme (cf. rapport de synthèse INRS[7], logiciel SOFIA ou Risk Spectrum pour l'arbre des causes ou MARK-EXD pour Markov).

L'arbre des causes est un modèle statique basé sur la détermination des causes amenant à l'événement dangereux. Le logiciel Risk Spectrum permet de calculer le taux de défaillance global de l'ensemble. Pour déterminer le PFD_{avg}, il faudra introduire les intervalles de temps entre proof test Ti et reprendre un calcul formel (un outil logiciel n'est alors pas nécessaire).

Le graphe de Markov sera utile lorsque les taux de défaillance sont constants et si on veut éventuellement faire des calculs en considérant les taux de réparation. Il permet d'avoir une approche dynamique (variation en fonction du temps) : à un instant donné, l'état de l'élément est suffisant pour connaître toute sa vie future. Le graphe de Markov donnera une probabilité de défaillance dangereuse. C'est le modèle le plus délicat à mettre en œuvre mais certainement le plus exact (si la modélisation est correcte).

Les blocs diagrammes présentent une approche plus fonctionnelle dont l'utilisation est plus facile. Ils ne permettent pas la modélisation de système incluant des temps de réparation. Ils permettent d'utiliser des lois de calculs de probabilité (modélisation série, parallèle,…). L'utilisation de cette méthode permet de calculer le PFD ou PFD_{avg}. Il existe aussi des logiciels de calcul (SafeCalc par exemple) qui utilisent cette modélisation. La modélisation du

mode commun se fera en plaçant un bloc supplémentaire dans le modèle qui représentera la part des modes communs.

Remarque :
Il ne faut pas confondre taux de défaillance et PFD/PFD_{avg}. Le taux de défaillance donne une probabilité de défaillance par unité de temps alors que le PFD est la probabilité que le système n'exécute pas la fonction pour laquelle il a été conçu, au moment où on le demande. On travaille ainsi pendant une période de temps donné.
Le PFD fait intervenir le taux de défaillance mais aussi le taux de couverture et l'intervalle de temps entre les proof tests.

6 Exemple de calcul

L'exemple choisi ne vise qu'à montrer une application pratique de la méthode et ne doit pas être considéré comme la validation d'un processus industriel existant.

Un opérateur commande la mise en marche d'une machine via un dispositif de double commande. Plus précisément, lorsque la double commande est sollicitée, la machine peut démarrer. L'opérateur doit laisser constamment ses 2 mains sur la double commande. Lorsque la machine a terminé son travail, l'opérateur peut retirer ses mains des boutons poussoirs.

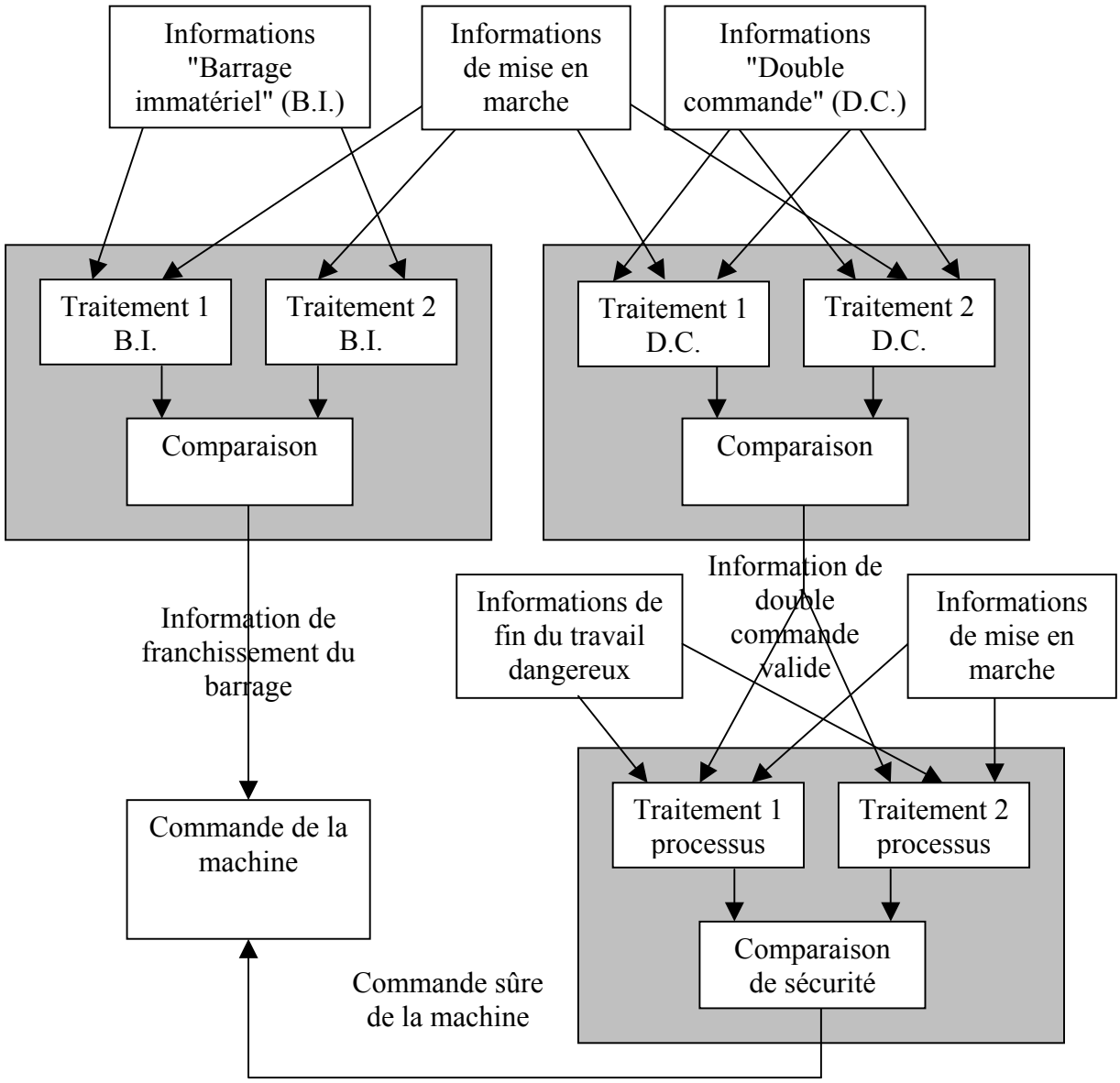
Une protection ultime est prévue sous la forme d'un barrage immatériel pour assurer une protection si la double commande ne joue plus son rôle.

Remarque : Dans les applications actuelles en machinerie, ce barrage sert pour les tiers et non comme protection ultime car on fait confiance à la double commande qui possède déjà un SIL ou une catégorie.

6.1 Fonctions de sécurité

Dans l'exemple de l'architecture double hétérogène étudiée par C. Blanchet[8], on peut distinguer 2 fonctions de contrôle de la sécurité:

- la fonction de la commande bi-manuelle qui empêche le démarrage de la machine ou arrête la machine,
- la fonction de détection de personne qui coupe l'alimentation de la machine.



6.1.1 Détection de personne

Cette fonction a lieu lorsque le barrage immatériel détecte le franchissement ou l'intrusion dans la zone. La sortie de ce barrage est dupliquée. Chaque information est envoyée vers un API. Le résultat du traitement est ensuite comparé et envoyé directement vers la machine pour couper l'alimentation (envoi vers l'actionneur).
La fonction de sécurité est donc l'arrêt de la machine lors de la détection du franchissement à travers le barrage immatériel.

6.1.2 Commande bi-manuelle

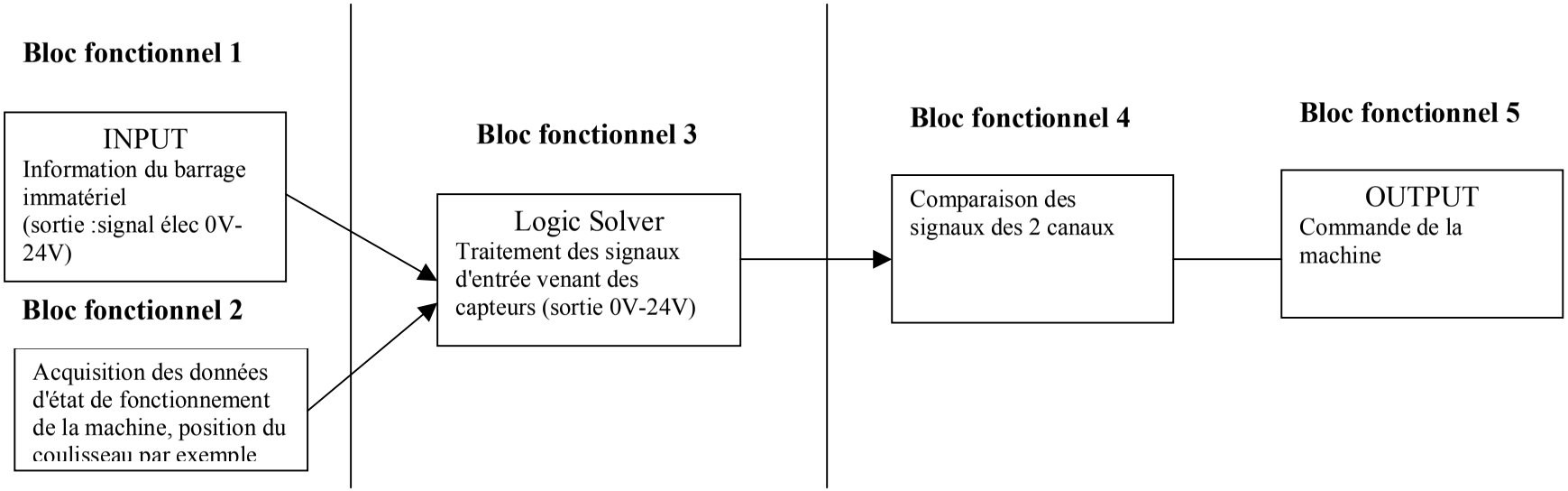
La double commande doit être actionnée pour enclencher le démarrage de la machine et maintenue en position pour continuer le processus. La fonction de sécurité est donc l'arrêt de la machine si l'on relâche l'appui sur 1 ou les 2 boutons de la double commande pendant le mouvement dangereux (et non pas, comme on pourrait le penser, le démarrage d'un cycle).

On a aussi comme fonction de sécurité l'empêchement de démarrage de la machine en cas d'appui désynchronisé.
Les signaux de la double commande sont envoyés vers 2 API, le résultat du traitement est comparé pour permettre la commande de l'alimentation de la machine.

On traitera dans la suite la fonction de sécurité relative à la détection de personnes.

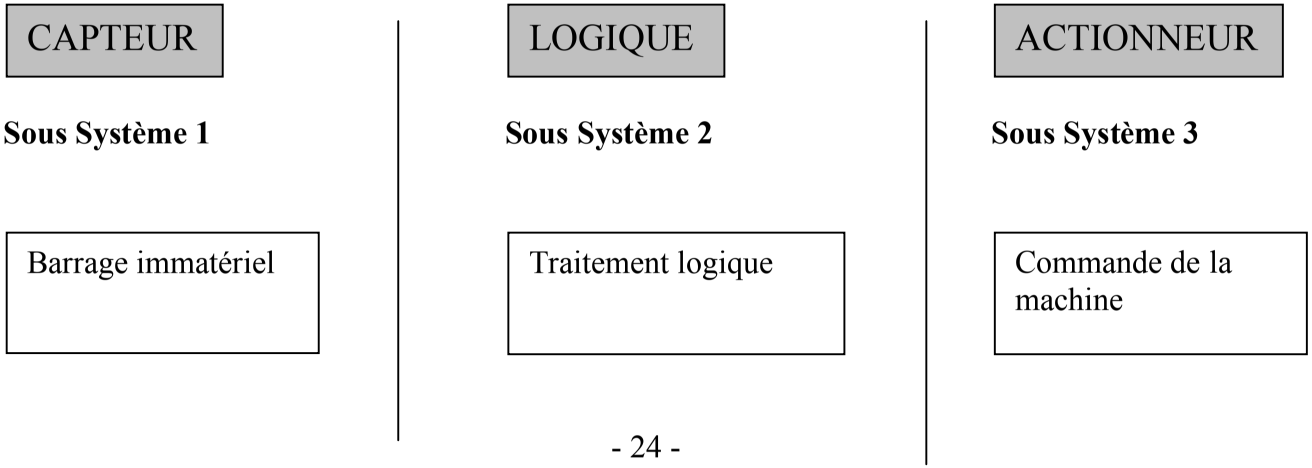
6.2 Décomposition en blocs fonctionnels

On doit donner un titre fonctionnel à chaque bloc.



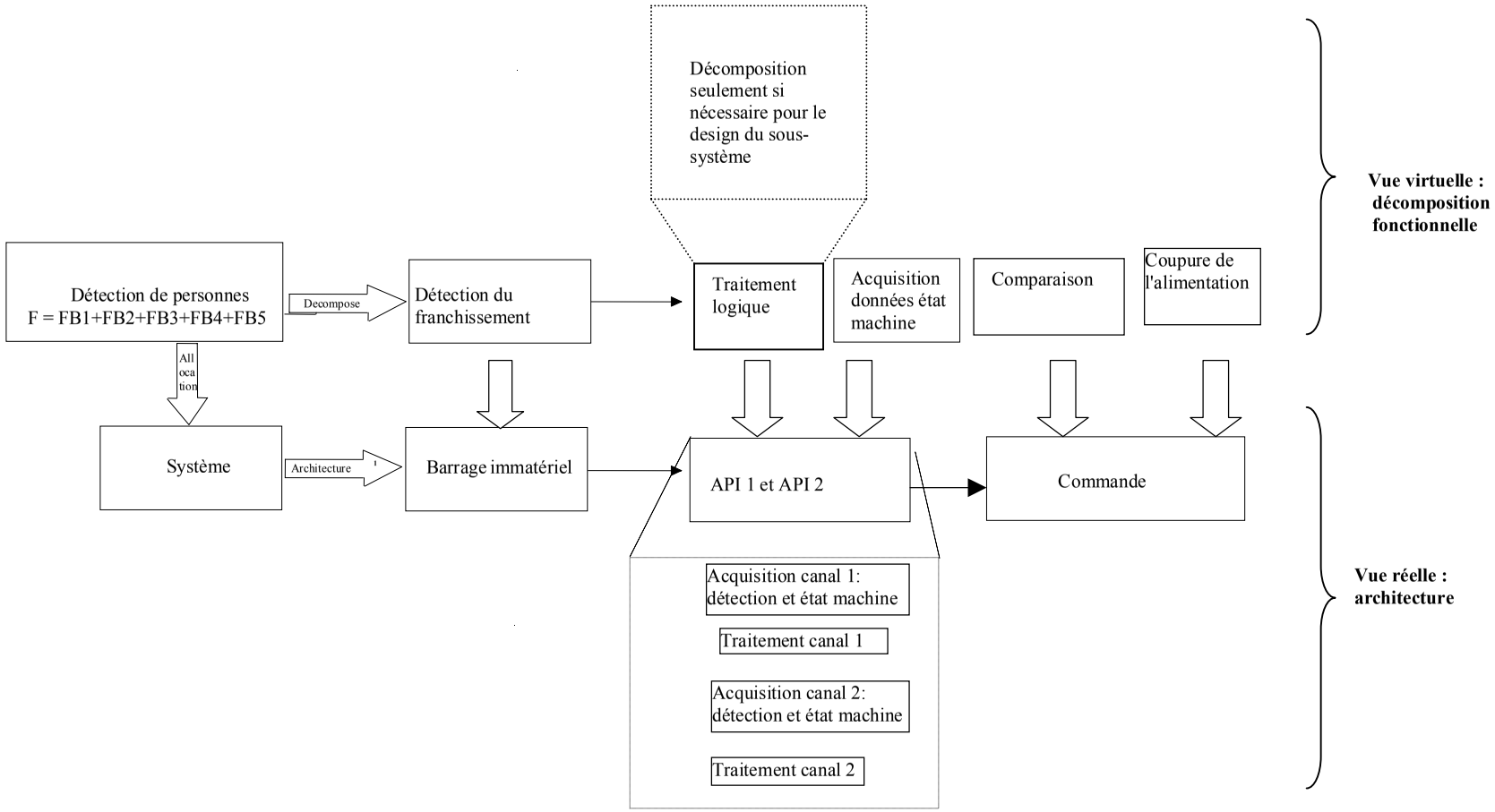
6.3 Architecture du système

L'architecture doit être définie en fonction du design et de la décomposition fonctionnelle faite ci-dessus.



6.4 Allocation des blocs fonctionnels aux sous-systèmes

On peut utiliser un schéma du type tel que celui proposé dans la norme CEI62061[1]
Remarque : On peut associer plusieurs blocs fonctionnels à un sous-système.



6.5 Exigences de sécurité pour chaque bloc fonctionnel

Voir CEI62061[1]

Note:
Les exigences de sécurité consistent à déterminer le SIL requis. La norme CD CEI 62061 ne donne pas la méthode pour déterminer le SIL requis pour un bloc fonctionnel. Il faut juste s'assurer que $SIL_{sys} <= (SIL_{subsytem})_{lowest}$.
Par contre, pour la fonction de sécurité, la norme CD CEI 62061 précise qu'il faut faire une estimation du risque pour déterminer le SIL requis (exigé).

6.6 Exigences fonctionnelles

6.6.1 Bloc 1 : détection de franchissement

Exigences fonctionnelles
Entrée : Franchissement ou non-franchissement du barrage.
Fonctionnalité : Détection du franchissement du barrage par une personne.
Sortie : la sortie d'un barrage immatériel est composée soit d'une sortie statique (en fait un transistor interne au barrage qui est alimenté par l'extérieur et qui va pouvoir générer le signal) ou par une sortie à relais. La sortie est donc du type à contact libre de potentiel, l'utilisateur choisissant sa propre alimentation suivant ses besoins.

6.6.2 Bloc 2 : acquisition des données état machine

Exigences fonctionnelles
Entrée : état de la machine (position du coulisseau et non pas mode de fonctionnement).
Fonctionnalité : donner la position de la machine.
Sortie : état de fonctionnement de la machine.

6.6.3 Bloc 3 : traitement logique

Ces blocs correspondent aux API.
Exigences fonctionnelles
Entrée : Signal 0V-24V venant du barrage immatériel et signal de fonctionnement de la machine.
Fonctionnalité : Si la machine est en fonctionnement et signal du barrage à 0V, alors il faut couper l'alimentation (intrusion dans la zone lors du fonctionnement). De plus, si la machine est arrêtée, l'opérateur actionne la double commande et le signal du barrage passe à 0V, il faut couper l'alimentation de la machine dangereuse.
Sortie: Désactiver la commande de la machine.

6.6.4 Bloc 4 : comparateur

Exigences fonctionnelles
Entrée : signal 0V-24V venant des sorties des API.
Fonctionnalité : s'il y a une discordance entre les 2 signaux, il faut couper l'alimentation de la machine. Sinon, le signal est envoyé tel quel vers la commande de la machine.

Sortie : signal 0V-24V coupure de l'alimentation.

6.6.5 Bloc 5 : Arrêt de la machine (ou mécanisme d'arrêt)

Exigences fonctionnelles
Entrée: signal venant du comparateur.
Fonctionnalité : déclencher le mécanisme d'arrêt.
Sortie: machine arrêtée.

6.7 Détermination des taux de défaillances

Le MTBF d'un équipement peut être fourni par son constructeur. En général, ce MTBF est prévisionnel (calculé par utilisation de bases de données composant). Parfois, les constructeurs disposent de suffisamment de retour d'expérience sur leurs matériels en exploitation pour fournir un taux de défaillance réel.

6.7.1 Sous-système 1

Le taux de défaillance du barrage immatériel est fourni par le constructeur. Le barrage immatériel est un barrage constitué de 2 cellules photoélectriques.
Attention, en fonction de la charge en sortie, la durée de vie du barrage immatériel pourra être fortement diminuée. En effet, des relais sont utilisés en sortie pour transférer l'information du barrage. Or il est communément admis que des relais peuvent subir en fonctionnement normal 10^6 manœuvres. En fonction du courant appliqué sur ces contacts et de la nature de la charge (capacitive ou inductive donc risque d'étincelles), le nombre de manœuvre sera fortement réduit et le MTBF de l'ensemble pourra ne pas correspondre au MTBF calculé. De plus, si l'on considère que le système est sollicité toutes les 10 s, au bout de 2780h de fonctionnement, le relais aura atteint les 10^6 commutations. Par contre, si on sollicite le barrage toutes les 2h, on aura atteint les 10^6 commutations au bout de 228 ans de fonctionnement.

Pour information, 1,9% des barrages de ce type reviennent en usine pour réparation.

6.7.2 Sous-système 2

Ce sous-système est composé de deux automates programmables différents.

6.7.2.1 **Automate A**

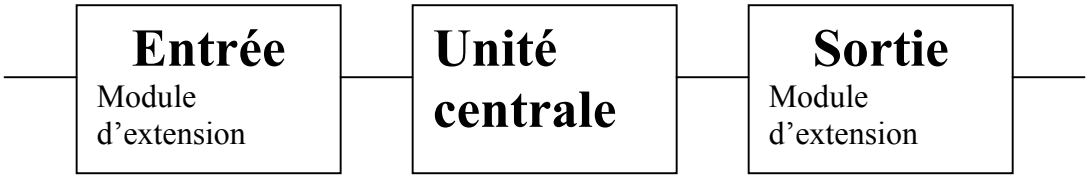


Diagramme de fiabilité de l'automate A

En terme de taux de défaillance, le diagramme ci-dessus se traduit de la façon suivante : $\lambda_{API} = \lambda_{Entrée} + \lambda_{UC} + \lambda_{Sortie}$.

D'après les définitions du §2 concernant le MTBF, on a

$$\lambda_{Entrée} = \lambda_{Sortie} = \frac{\lambda_{Module}}{Nombred'E/S} = \frac{1}{MTTF_{Module} * Nombred'E/S} \approx \frac{1}{MTBF_{Module} * Nombred'E/S}$$

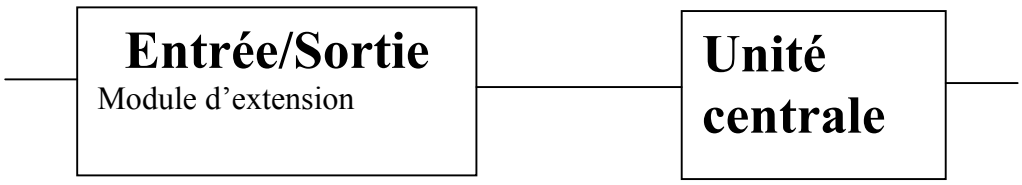
.

Et, $\lambda_{UC} = \frac{1}{MTTF_{UC}} \approx \frac{1}{MTBF_{UC}}$

Pour l'automate A, on a les valeurs suivantes fournies par le constructeur :
$MTTF_{Module} = 228 ans$, Et, donc

$$\lambda_{API} \approx 2 * \frac{1}{MTBF_{Module} * Nombred'E/S} + \frac{1}{MTBF_{UC}} = \frac{2}{228 * 8} + \frac{1}{25} \approx 4{,}11.10^{-6}$$ défaillances /h

6.7.2.2 **Automate B**



$$\lambda_{API} = \lambda_{Entrée/Sortie} + \lambda_{UC}$$

Or le constructeur nous a fourni
$\lambda_{UC} = 22.1\ 10^{-6}$
$\lambda_{E/S} = 1.1\ 10^{-6}$ (pour une carte)

d'où $\lambda_{API} = 23.1\ 10^{-6}$ défaillances par heure (avec une carte E/S)

6.7.3 Sous-système 3

Le sous-système 3 peut être réalisé à partir :

- D'un comparateur. Le bloc fonctionnel "commande de la machine" n'est plus nécessaire. On considèrera que le signal de commande est envoyé par le comparateur et le sous-système 3 n'existe plus. La fonction de sécurité s'arrête à l'ordre envoyé.
- D'une électrovanne double corps. La commande ainsi que la fonction comparateur sont effectuées par l'électrovanne. Il faudra alors obtenir le taux de défaillance de cette électrovanne.

6.8 Détermination des facteurs

6.8.1 Sous-système 1

Le fournisseur de ce sous-système ne peut pas nous donner les taux de couverture, la proportion de pannes dangereuses et sûres ainsi que l'intervalle de temps entre proof test.

On ne dispose que du λ du barrage, ce qui conduit à prendre arbitrairement :

$$\lambda^{D}_{barrage} = 0.5 \times (1\text{-}DC_{barrage}) \times \lambda_{barrage}$$

Le barrage étant de catégorie 4, on considérera que le taux de couverture des tests de diagnostics est égal à $DC_{barrage} = 99\%$.

Le barrage immatériel n'est pas redondant donc il n'y a pas de modes communs pour ce barrage.

Le temps entre deux proof tests consécutifs (test fonctionnel) sera pris arbitrairement égal à T_i = 168 h (1 mois).

6.8.2 Sous-système 2

On considérera pour les calculs que les deux APIs ont les mêmes caractéristiques de fiabilité et les mêmes taux de couverture, proof tests, … Ces valeurs seront celles considérées pour l'automate A.

On ne dispose que du λ global de l'API, ce qui conduit à prendre

$$\lambda^{D}_{API} = 0.5 \times (1\text{-}DC_{API}) \times \lambda_{API}$$

Le fournisseur de l'API ne nous a pas donné un taux global mais a simplement donné les taux de couverture des tests internes (test de la RAM par exemple).
Pour l'API, on considère que son taux de couverture des défaillances dangereuses est de $DC_{API} = 90\%$. Ce taux de couverture est donné à titre indicatif pour permettre d'évaluer la probabilité de défaillance. Il doit certainement être supérieur au taux de couverture réel.

On considérera tout d'abord les deux automates comme étant homogènes (les paramètres de calculs retenus seront ceux de l'automate A et on ne considérera pas les modes communs).

Pour les API, on considère que l'intervalle de temps entre proof test est égal à la vie de l'équipement soit 10 ans, ce qui revient à dire que l'automate n'est jamais testé complètement fonctionnellement durant toute sa durée de vie;
$T_i = 87700$ h.

Nous allons à présent insister sur l'influence des paramètres « taux de couverture » et « intervalle entre proof test » sur la détermination de la PFD.

L'intervalle de temps entre proof test ou temps de mission T_i sert comme base de calcul pour la valeur moyenne de la PFD. Les tests de diagnostics interviennent dans le calcul du taux de défaillance par l'intermédiaire du taux de couverture.

En effet, dans la mesure où après un proof test, le système est considéré comme neuf, il ne doit plus y avoir de défaillance à la demande. PFD_{avg} étant une valeur moyenne par rapport au T_i, il peut bien y avoir une indépendance entre le taux de couverture des tests de diagnostics et le temps entre deux proof tests consécutifs T_i.

Pour l'impact de l'intervalle entre tests de diagnostics, voire la partie définition. On rappellera ici que l'intervalle entre tests de diagnostics doit être suffisamment petit devant le temps entre 2 défaillances MTBF (facteur 100 au moins) pour que l'on puisse négliger le temps de réparation devant le temps entre deux défaillances.

6.8.3 Sous-système 3

On considère que la machine est commandée par une électrovanne.

$$\lambda^D_{commande} = 0.5 \times (1\text{-}DC_{commande}) \times \lambda_{commande}$$

De façon arbitraire, on prendra un taux de couverture de 90% et un temps Ti entre proof tests égal à 12 mois soit 8770h. Les défaillances de mode commun n'ont pas de sens.

6.9 Architecture avec 2 API identiques : cas de la faible sollicitation

On considère ici que les deux automates ont le même taux de défaillances, sans modes communs (β=0)

sous-système	**mode commun β**	**taux de couverture DC**	**intervalle de temps entre proof test T_i**		**λ**	**λ^D**
			en heures	**en mois**		
barrage immatériel	0	99%	T_1 = 168	1	$5.0\ 10^{-9}$	$2.5\ 10^{-11}$
API1	0	90%	T_2 = 87700	120	$4.1\ 10^{-6}$	$2.05\ 10^{-7}$
API2	0	90%	T_2 = 87700	120	$4.1\ 10^{-6}$	$2.05\ 10^{-7}$
commande	0	90%	T_3 = 8770	12	$5.0\ 10^{-8}$	$2.5\ 10^{-9}$

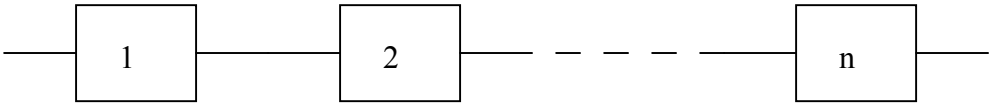
6.9.1 PFD_{avg} du système par la méthode des blocs diagrammes

6.9.1.1 Calcul manuel

Rappels :

- On nommera r_i la fiabilité de l'élément i du diagramme.

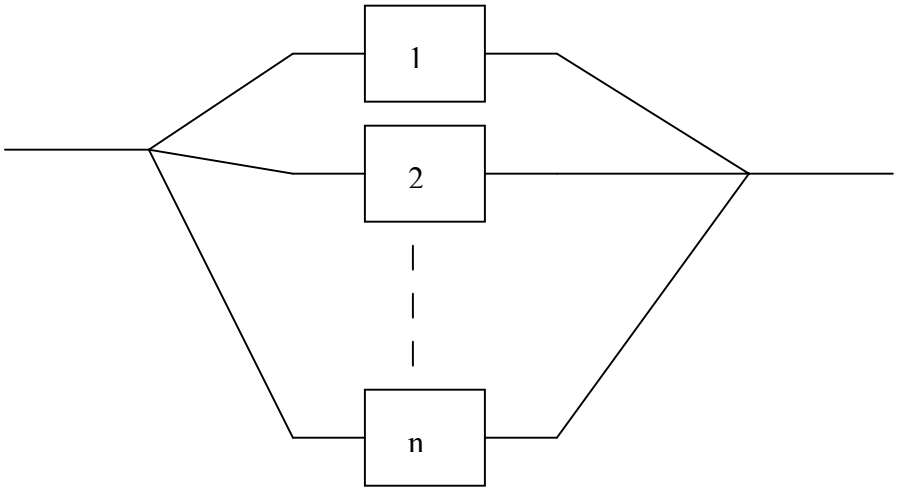
  Pour des éléments en série,



  la fiabilité est donnée par

$$R(t) = \prod_{1}^{n} r_i(t)$$

  Pour des éléments en //,



  la fiabilité est

$$R(t) = 1 - \prod_{1}^{n} \left(1 - r_i(t)\right)$$

- Les différents types d'architecture d'un système redondant ont été modélisés dans la norme CEI 61508[2]. On rappelle ici les principaux types d'architecture avec deux canaux[9]:
  - 1oo2 (1 out-of 2) qui est une architecture constituée de deux canaux connectés en parallèle réalisant chacun la fonction de sécurité. Les tests de diagnostics qui

existent ne changent pas l'état de la sortie et il faut une défaillance des deux canaux pour avoir une défaillance dangereuse.

- 1oo2D (1 out-of 2D) qui est une architecture constituée de deux canaux connectés en parallèle. En fonctionnement normal, les deux canaux doivent demander la fonction de sécurité avant que celle-ci ne soit activée. Si les tests de diagnostic détectent une défaillance sur un des canaux, la sortie considérée sera celle du canal exempt de faute. Si les tests de diagnostic détectent une faute ou une différence entre les deux canaux ne pouvant être attribuée à l'un des canaux, la sortie sera positionnée dans un état sûr. Cette architecture est plus appropriée pour du process ou pour privilégier la disponibilité.
- 2oo2 (2 out-of 2) qui est une architecture dans laquelle les deux canaux sont connectés en parallèle de telle sorte que les deux canaux doivent demander la fonction de sécurité avant que celle-ci ne soit activée. Les tests de diagnostics signalent les fautes mais ne changent pas l'état de la sortie.

Dans notre cas, la partie logique est constituée de 2 API montés en parallèle. Pour des raisons de sécurité, on souhaite que dès qu'un des deux canaux demande la fonction de sécurité, le système soit arrêté. De plus, le système passe en sécurité en cas de désaccord des 2 sorties et un signal est renvoyé vers l'entrée des API. La logique correspondante est soit 1oo2, soit 1oo2D. On peut associer la discordance à une architecture 1oo2D (si une différence existe entre les deux canaux alors on positionne la sortie dans un état sûr) car on ne sait pas distinguer quel est le canal en panne. Il faudrait demander à l'opérateur un test supplémentaire pour déterminer le canal en panne. Par contre, l'état de la sortie n'est pas modifié par les tests de diagnostics.
On a donc ici une logique 1oo2 puisque les tests de diagnostics ne modifient pas directement l'état de la sortie (c'est la comparaison entre les deux API qui oriente l'état de la sortie vers l'état sûr) et il faut une défaillance des deux canaux pour avoir une défaillance dangereuse.

Le problème est de quantifier le taux de couverture de la détection des discordances au niveau du comparateur. Il paraît impossible de déterminer tous les défauts dont les conséquences se manifestent par un écart de temps de traitement entre les deux voies (pas de désynchronisme). On ne détecte pas les pannes latentes qui se révèleront un cycle plus tard ou dans un autre mode de marche.

Les taux de couverture sont donnés pour caractériser les tests de diagnostics faits en interne (dans un API ou dans le barrage) comme les tests dédiés (test CPU, test de la mémoire). Ils ne prennent pas en compte des mécanismes de détection tels que celui fait par le comparateur pour détecter une incohérence entre les 2 canaux. De même, si on introduit du dynamisme comme mécanisme de détection de fautes, il sera difficile de quantifier cet apport par un taux de couverture. Comment estimer alors le taux de couverture de ce mécanisme pour la détection de pannes dangereuses? En effet, nous travaillons dans notre cas sur des états établis donc nous ne voyons pas forcément si une entrée ne change pas d'état.
La quantification des taux de couverture reste donc aujourd'hui très approximative.

Dans les calculs qui suivent, on ne peut pas parler en toute rigueur de fiabilité puisque l'on considère uniquement les pannes dangereuses non détectées. Toutefois, pour des raisons de simplicité, nous emploierons le terme fiabilité.

La fiabilité R(t) de la partie logique, dans ce cas de redondance parallèle, est donnée par :
$R_{logique} = 1-[(1-r_{API1})(1-r_{API2})]$.

En considérant une loi de distribution exponentielle,
$1\text{-}r_{API1}(t) = 1\text{-}e^{-\lambda api1\ t} \approx \lambda^D_{API1} * t$
$1\text{-}r_{API1}(t) = 1\text{-}e^{-\lambda api2\ t} \approx \lambda^D_{API2} * t$
d'où $R_{logique}(t) = 1\text{-}\lambda^D_{API1}\lambda^D_{API2} * t^2$

La fiabilité du barrage est donnée par
$R_{barrage}(t) = e^{-\lambda^d_{barrage} t} \approx 1 - \lambda^D_{barrage}\ t$

La fiabilité de la commande est donnée par
$R_{commande}(t) = e^{-\lambda^d_{commande} t} \approx 1 - \lambda^D_{commande}\ t$

Pour le système complet, la fiabilité de l'ensemble (association en série des 3 sous-systèmes) est
$R_{système} = R_{barrage} * R_{logique} * R_{commande}$

$R_{système}(t) = (1\text{-}\lambda^D_{barrage}\ t)\ (1\text{-}\lambda^D_{commande}\ t)(\ 1\text{-}\lambda^D_{API1}\lambda^D_{API2} * t^2)$
et
$PFD_{système}(t) = 1\text{-}\ R_{système}(t) = 1\text{-}(1\text{-}\ \lambda^D_{barrage}\ t)\ (1\text{-}\lambda^D_{commande}\ t)(1\text{-}\lambda^D_{API1}\lambda^D_{API2} * t^2)$
$PFD_{système}(t) = \lambda^D_{API1}\lambda^D_{API2} * t^2 + \lambda^D_{barrage}\ t + \lambda^D_{commande}\ t - \lambda^D_{barrage}\ \lambda^D_{commande}\ t^2 - \lambda^D_{barrage}\lambda^D_{API1}\lambda^D_{API2}\ t^3 - \lambda^D_{commande}\ \lambda^D_{API1}\lambda^D_{API2}\ t^3 + \lambda^D_{barrage}\ \lambda^D_{commande}\ \lambda^D_{API1}\lambda^D_{API2}\ t^4$

6.9.1.1.1 Intervalle de temps entre proof test identique pour tous les sous-systèmes

Pour avoir le PFD_{avg}, il faut calculer

$$PFD_{avg} = \frac{1}{T_i}\int_0^{T_i} PFD\,(t)\ dt$$

d'où
$PFD_{avg} = (\lambda^D_{barrage} + \lambda^D_{commande})\ T_i/2 + (\lambda^D_{API1}\lambda^D_{API2} - \lambda^D_{commande}\lambda^D_{barrage})T_i^2/3 - (\lambda^D_{barrage} + \lambda^D_{commande})\ \lambda^D_{API1}\lambda^D_{API2}\ T_i^3/4 + \lambda^D_{barrage}\lambda^D_{commande}\lambda^D_{API1}\lambda^D_{API2}\ T_i^4/5$

On considère ci-dessus que le temps T_i entre proof test est le même pour tous les composants.

On prendra un intervalle entre proof test T_i de 87700h soit le temps de mission.
On obtient:

	Blocs diagrammes
PFD_{avg}	$2.18445\ 10^{-4}$

Si on prend un intervalle entre proof test Ti de 168h, on a le résultat suivant :

	Blocs diagrammes
PFD_{avg}	$2.1249\ 10^{-7}$

6.9.1.1.2 Intervalle de temps entre proof test différent pour chaque sous-système

Si on veut calculer la valeur moyenne pour différents intervalle de temps entre proof test (de chaque sous-système), on peut utiliser le calcul suivant :

$$PFD_{avg} = \frac{1}{T_1 T_2 T_3} \iiint \left[1 - (1 - \lambda^D_{barrage}\ t_1)(1 - \lambda^D_{commande}\ t_3)(1 - \lambda^D_{API1}\ \lambda^D_{API2}\ t_2^2)\right] dt_1\ dt_2\ dt_3$$

avec

T_1 = 1 mois soit 168h pour le barrage
T_2 = 120 mois (10 ans) soit 87700h pour les API
T_3 = 12 mois soit 8770h pour la commande

soit

$$PFD_{avg} = 1 - \left(1 - \lambda^D_{barrage}\ \frac{T_1}{2}\right)\left(1 - \lambda^D_{commande}\ \frac{T_3}{2}\right)\left(1 - \lambda^{\ D}_{API1}\ \lambda^D_{API2}\ \frac{T_2^2}{3}\right)$$

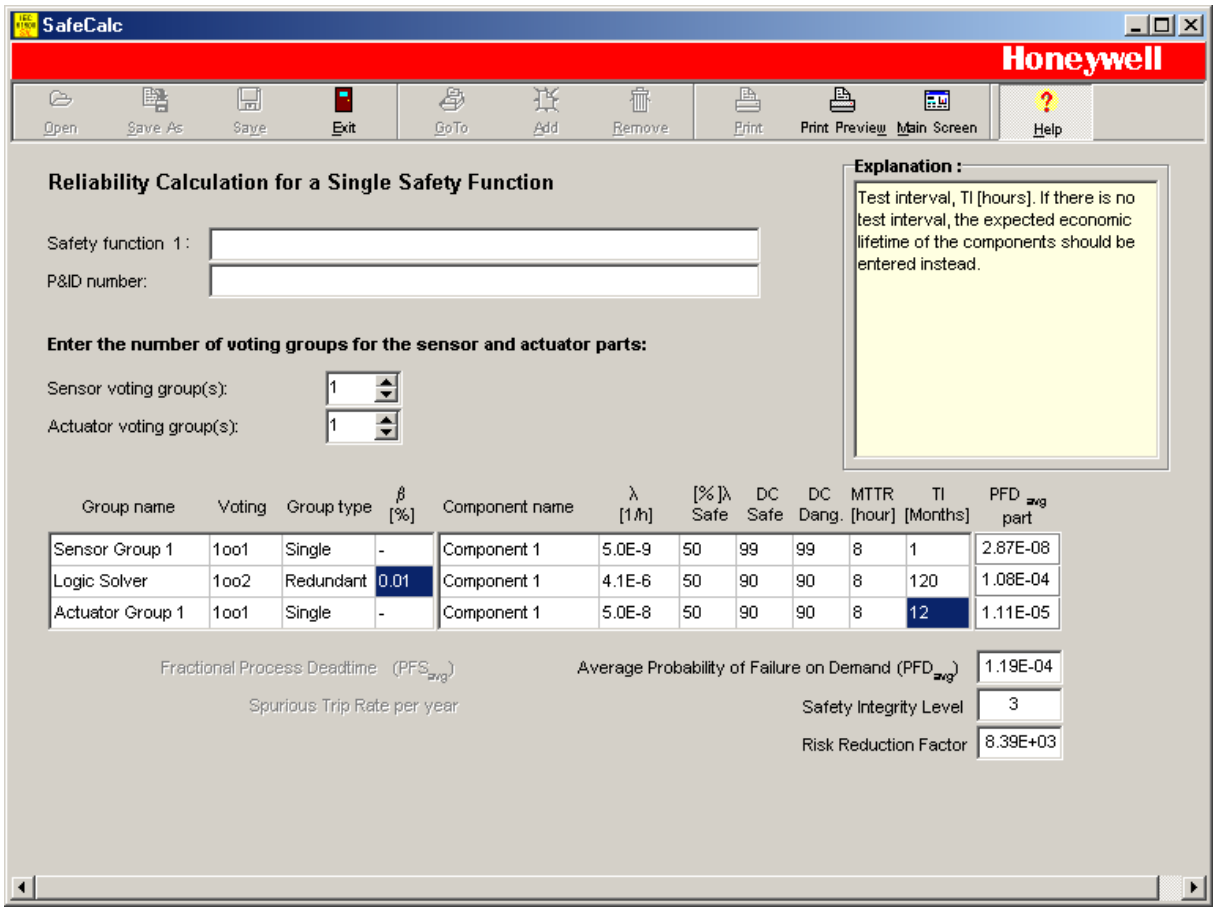
et donc, avec les valeurs présentées au début de ce chapitre, on a
PFD_{avg} = 1-(1-2.1 10^{-9})(1-1.09625 10^{-5})(1-1.07742154083 10^{-4})
PFD_{avg} = 1.1871 10^{-4}

6.9.1.2 Utilisation du logiciel SafeCalc

Ce logiciel ne fait que des calculs numériques associés à une modélisation par blocs diagramme. On supposera par exemple que les taux de défaillances sont :

- pour le barrage immatériel, on prendra comme précédemment $\lambda_{barrage}$ = 5 10^{-9}
- pour la commande, on prendra un $\lambda_{commande}$ = 5 10^{-8}

Ce logiciel utilise les taux de pannes de base du composant, on entre les taux de couverture et le SIL obtenu est de 3.



Dans ce cas, T_i pour le sensor group est de 1 mois soit 168h, 120 mois correspond à 10 ans pour la partie logique et 12 mois pour l'actionneur. On rappelle que le T_i caractérise l'intervalle de temps entre chaque proof test (test de maintenance qui remet le système comme neuf).

On a donc les résultats suivants:

	PFD_{avg}
Calcul manuel	$1.18\ 10^{-4}$
Outil informatique SafeCalc	$1.19\ 10^{-4}$

Remarque:
Dans le logiciel SafeCalc, il faut entrer une valeur pour le MTTR. Compte tenu de la valeur élevée des temps considérés par rapport au temps de réparation, l'impact du MTTR sur la valeur du PFD_{avg} est très faible. L'utilisation du MTTR par le logiciel SafeCalc doit être utile pour déterminer les temps d'indisponibilité du système. On ne peut pas utiliser le logiciel SafeCalc avec 2 parties logiques différentes (« logic solver » ayant des valeurs de taux de défaillances différents).

6.9.1.3 Utilisation d'autres outils

La description détaillée de ces outils sera faite en Annexe. Tous ces outils ne donnent pas la PFD mais le MTBF ou le taux de défaillance.

Exemple de l'outil Reliability Workbench
On obtient un MTBF = 3.96 10^8 soit un taux de défaillance λ de 2.53 10^{-9}. On obtient ici un taux de défaillance et non pas la PFD_{avg}. Ces deux notions ne sont pas les mêmes puisque dans la PFD_{avg} intervient l'intervalle de temps entre proof test.

6.9.1.4 Remarques concernant les normes CEI 61508 et CD CEI 62061

Les normes CEI 61508 et CD CEI 62061 proposent pour le calcul de la PFD du système :

$$PFD_{système} = \Sigma\, PFD_{sous\text{-}système}$$

Dans ce cadre, on a donc

$$PFD_{système} = PFD_{barrage} + PFD_{logique} + PFD_{commande}$$

$PFD_{barrage}(t) = 1\text{-}R_{barrage}(t) = \lambda^D_{barrage}\, t$
$PFD_{logique}(t) = 1\text{-}R_{logique}(t) = \lambda^D_{API1}\lambda^D_{API2}\, t^2$
$PFD_{commande}(t) = 1\text{-}R_{commande}(t) = \lambda^D_{commande}\, t$
soit $PFD_{système}(t) = \lambda^D_{barrage}\, t + \lambda^D_{API1}\lambda^D_{API2}\, t^2 + \lambda^D_{commande}\, t$

Dans le cas d'un intervalle de temps entre proof test différent pour chaque sous-système et pour des taux de défaillance identiques pour les automates, on a

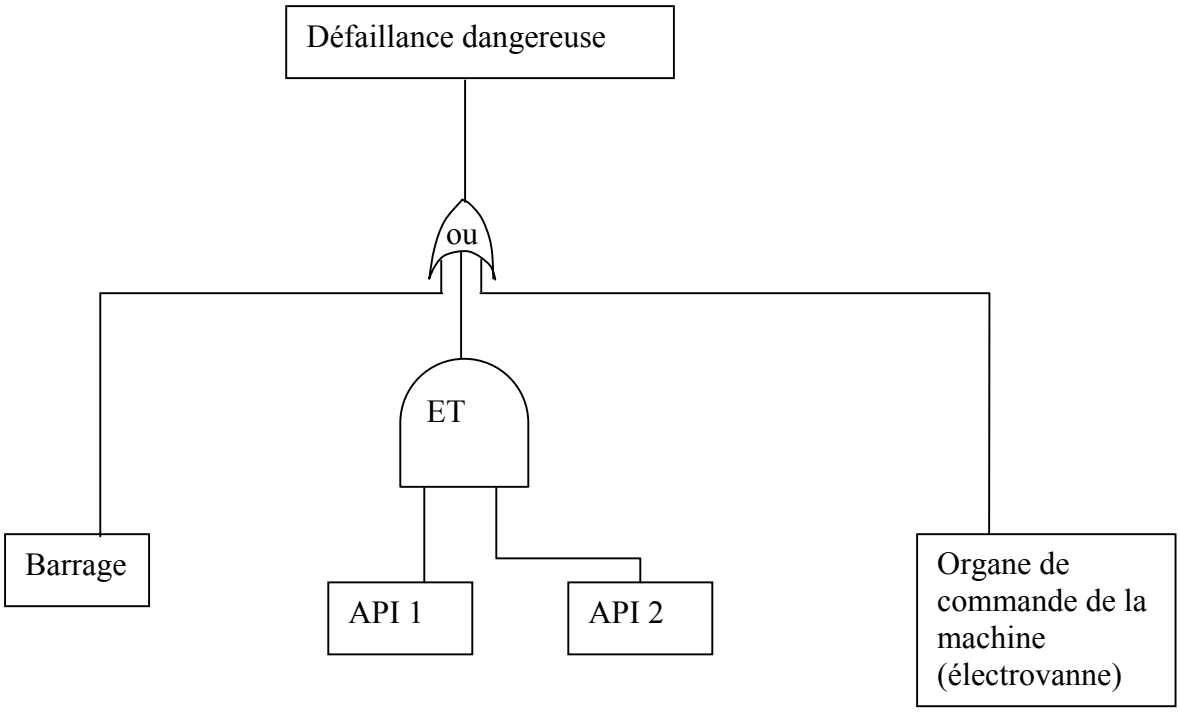
$$PFD_{avg} = \frac{1}{T_1 T_2 T_3} \iiint \left[\lambda^D_{barrage}\, t_1 + \lambda^D_{commande}\, t_3 + \left(\lambda^D_{API}\, t_2\right)^2 \right] dt_1 dt_2 dt_3$$

$$PFD_{avg} = \lambda^D_{barrage}\, T_1/2 + \lambda^D_{commande}\, T_3/2 + \frac{(\lambda^D_{API} T_2)^2}{3}$$

soit
$PFD_{avg} = 1.187\ 10^{-4}$

6.9.2 PFD_{avg} du système par arbre des causes

On peut représenter l'arbre des causes de la façon suivante:



Cet arbre prend en compte uniquement les défaillances dangereuses non détectées puisque ce sont celles qui contribuent à la valeur de la PFD.

On utilisera le logiciel Risk Spectrum pour obtenir le taux de défaillances.

6.9.2.1.1 Intervalle de temps entre proof test différent pour chaque sous-système

Dans ce cas, en utilisant la façon de calculer la PFD telle que dans l'ISA[4], on a :
$PFD = \lambda^D_{barrage}\ T_{i\ barrage} + \lambda^D_{commande}\ T_{i\ commande} + (\lambda^D_{API}\ T_{i\ API})^2$.

Cette expression est obtenue en regardant dans l'arbre des fautes les contributions à l'événement dangereux. Un "OU" donne une addition, un "ET" donne un produit. Chaque partie étant le produit du taux de défaillance dangereuse non détectée par l'intervalle de temps entre proof test.

On a donc

$$PFD_{avg} = \frac{1}{T_1 T_2 T_3} \iiint \left[\lambda^D_{barrage}\, t_1 + \lambda^D_{commande}\, t_3 + \left(\lambda^D_{API}\, t_2\right)^2 \right] dt_1 dt_2 dt_3$$

$$PFD_{avg} = \lambda^{D}_{barrage}\, T_1/2 + \lambda^{D}_{commande}\, T_3/2 + \frac{(\lambda^{D}_{API} T_2)^2}{3}$$

soit
$PFD_{avg} = 1.187\ 10^{-4}$

6.9.2.1.2 Intervalle de temps entre proof test identique pour chaque sous-système

$$PFD_{avg} = \lambda^{D}_{barrage}\, T_i/2 + \lambda^{D}_{commande}\, T_i/2 + \frac{(\lambda^{D}_{API}\ T_i)^2}{3}$$

On prendra un intervalle entre proof test T_i de 87700h soit le temps de mission.
On obtient:

	Arbre des fautes
PFD_{avg}	$2.1846\ 10^{-4}$

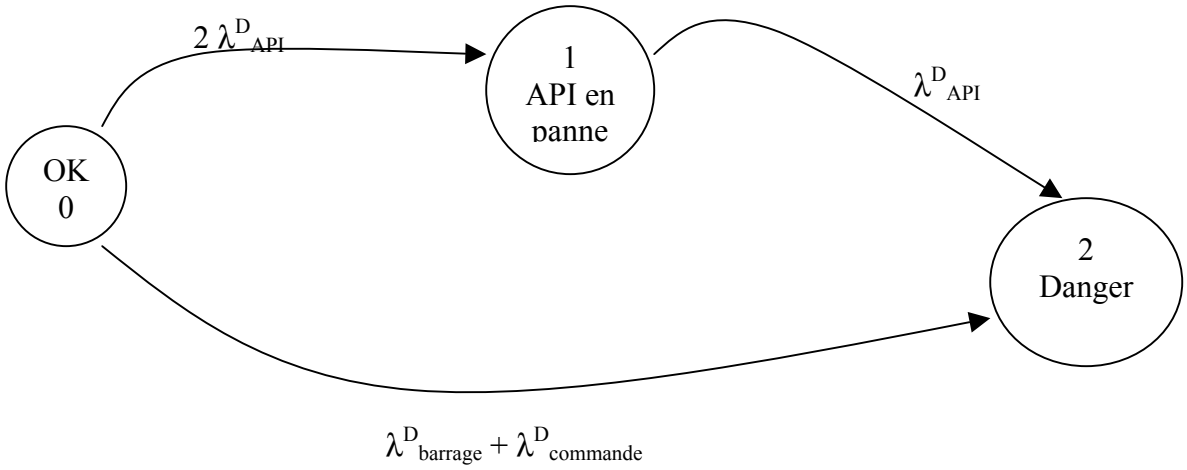
Si on prend un intervalle entre proof test Ti de 168h, on a les résultats suivants :

	Arbre des fautes
PFD_{avg}	$2.1249\ 10^{-7}$

6.9.3 PFD_{avg} du système par diagramme de Markov

État dangereux: état n°2.

Ce diagramme est un cas simple, il faudrait tenir compte des défaillances détectées et non détectées, dangereuses et non dangereuses.



On obtient ainsi la matrice de transition de l'ensemble

$$P = \begin{pmatrix} 1 - 2\lambda_{API}^{D} - \lambda_{barrage}^{D} - \lambda_{commande}^{D} & 2\lambda_{API}^{D} & \lambda_{barrage}^{D} + \lambda_{commande}^{D} \\ 0 & 1 - \lambda_{API}^{D} & \lambda_{API}^{D} \\ 0 & 0 & 1 \end{pmatrix}$$

Il faut ensuite à partir de l'état de départ S = (1 0 0), calculer par itérations successives la matrice $S_i = S_{i-1}$ P et la PFD associée.

Pour obtenir le PFD_{avg}, il faut en même temps calculer la moyenne des PFD donc calculer pour chaque itération le $PFD_i = PFD_{i-1} + Proba_i$(état 2).
(PFD_{avg} = ΣPFD/nombre de pas de calculs).

Dans ce cas, on ne peut pas avoir plusieurs intervalles de temps entre proof test. La valeur que l'on va choisir pour le nombre d'itérations correspond à la valeur du T_i et on ne peut prendre en compte qu'une seule valeur pour tous les sous-systèmes.

On peut représenter cela par l'algorithme suivant:
S = (1 0 0)
Sint = S*P
PFDavgint = Sint(3); cela correspond à la troisième composante de la matrice

```
for i=1 to Ti
        S=Sint*P
        Sint=S
        PFDavgint = PFDavgint + S(3)
endfor
```

PFD_{avg} = PFDavgint/Ti

La valeur numérique de la matrice est :

$$P = \begin{pmatrix} 0.9999995875 & 4.1\ 10^{-7} & 2.525\ 10^{-9} \\ 0 & 0.999999795 & 2.05\ 10^{-7} \\ 0 & 0 & 1 \end{pmatrix}$$ en prenant les λ^D incluant les taux de couverture.

On obtient les valeurs suivantes en fonction de T_i :

T_i	PFD_{avg}
168	$2.13753\ 10^{-7}$
8770	$1.2136\ 10^{-5}$
87700	$2.1569\ 10^{-4}$

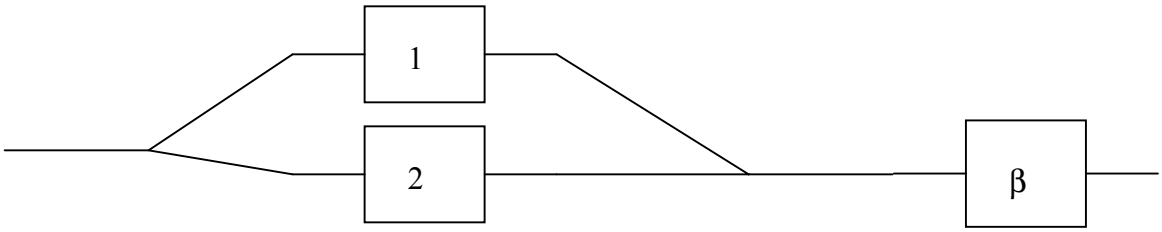
On trouvera le détail de la démarche dans le chapitre 8 de Goble[4].

6.9.4 Introduction des modes communs

On présentera succinctement les cas pour le bloc diagramme et l'arbre des fautes. Pour de plus amples informations, on pourra se référer à Goble[4] et Charpentier[9].

6.9.4.1 Bloc diagramme

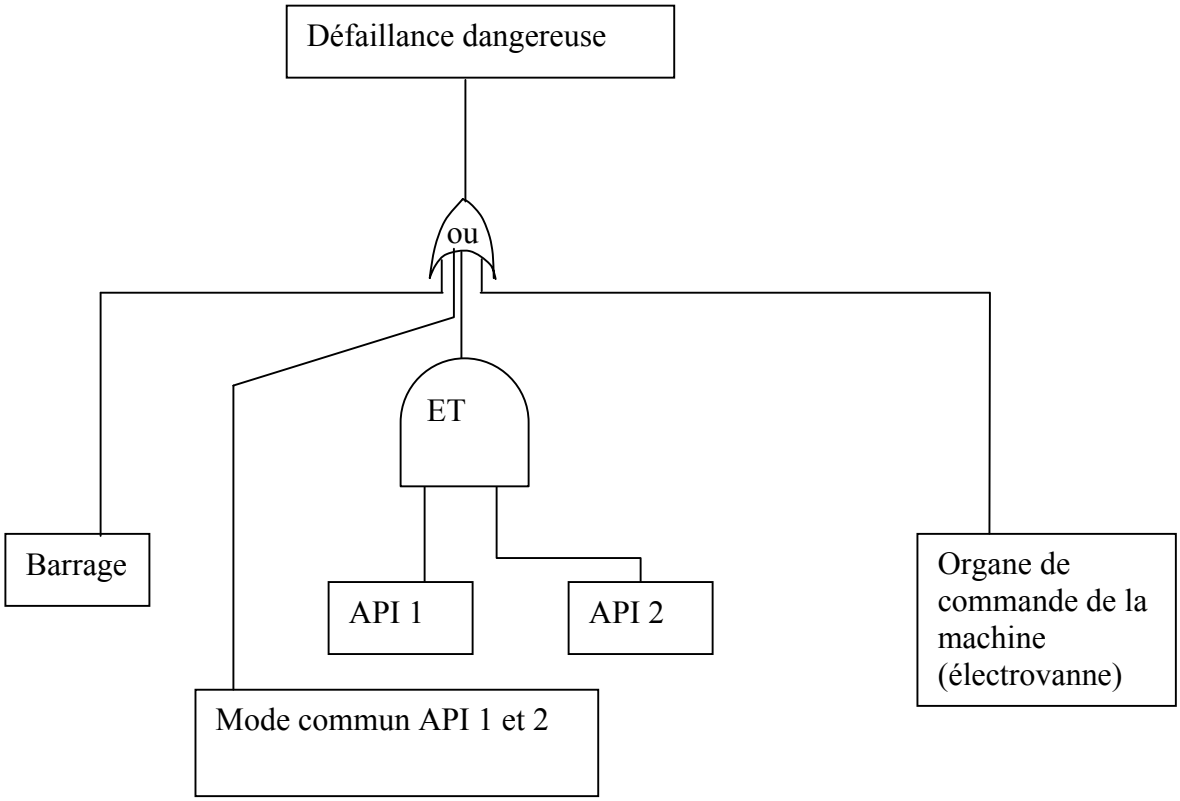
Il faut ajouter un bloc en série avec les API qui modélisera les modes communs.



La fiabilité de cette partie logique s'obtiendra par le calcul suivant :

$R_{logique}$ (t) = [1-(1-r indépendant $_{API}$ (t))(1-r indépendant $_{API}$(t))]r commun $_{API}$(t)
Avec r commun $_{API}$ (t) = $e^{-\beta \lambda^{d}_{API} t} \approx 1 - \beta \lambda^{D}_{API}\, t$
r normal $_{API}$ (t) = $1-e^{-\beta\lambda API\ t} \approx \beta\, \lambda^{D}{}_{API} * t$

6.9.4.2 Arbre des causes



Dans ce cas, en utilisant la façon de calculer le PFD telle que dans l'ISA[4] et en considérant les deux automates homogènes de taux de défaillance λ_{API}, on a :
$PFD = \lambda^D_{barrage}\ T_{i\ barrage} + \lambda^D_{commande}\ T_{i\ commande} + ((1-\beta)\ \lambda^D_{API}\ T_{i\ API})^2 + \beta\ \lambda^D_{API}\ T_{i\ API}$

6.10 Architecture avec 2 API différents : cas de la faible sollicitation

On considère maintenant deux APIs 1 et 2 différents, sans modes communs. Nous utiliserons les valeurs suivantes :

	Barrage	API 1	API 2	Commande
Taux de défaillance dangereuse non détectée	$2.5\ 10^{-11}$	$2.05\ 10^{-7}$	$1.15\ 10^{-6}$	$2.5\ 10^{-9}$
Intervalle entre proof test (h)	168	87 700	87 700	8770

La valeur du taux de défaillance pour l'API2 est obtenue en prenant les valeurs $\lambda = 23.1\ 10^{-6}$, DC = 0.9 de l'automate B.

6.10.1 PFD_{avg} par la méthode des blocs diagrammes

6.10.1.1 Intervalle de temps entre proof test différents

Il suffit de reprendre la formule

$$PFD_{avg} = 1 - \left(1 - \lambda^D_{barrage}\frac{T_1}{2}\right)\left(1 - \lambda^D_{commande}\frac{T_3}{2}\right)\left(1 - \lambda^D_{API1}\,\lambda^D_{API2}\frac{T_2^2}{3}\right)$$ et l'on a :

$PFD_{avg} = 6.1536\ 10^{-4}$

6.10.1.2 Intervalle de temps entre proof test unique

On utilise l'expression trouvée au paragraphe précédent:
$PFD_{avg} = (\lambda^D_{barrage} + \lambda^D_{commande})\ T_i/2 + (\lambda^D_{API1}\lambda^D_{API2} - \lambda^D_{commande}\lambda^D_{barrage})T_i^2/3 - (\lambda^D_{barrage} + \lambda^D_{commande})\ \lambda^D_{API1}\lambda^D_{API2}\ T_i^3/4 + \lambda^D_{barrage}\lambda^D_{commande}\lambda^D_{API1}\lambda^D_{API2}\ T_i^4/5$

	Blocs diagrammes
PFD_{avg} (T_i 87 700 heures)	$7.1503\ 10^{-4}$
PFD_{avg} (T_i 168 heures)	$2.1432\ 10^{-7}$

6.10.2 PFD_{avg} par la méthode de l'arbre des causes

6.10.2.1 Intervalle de temps entre proof test différents

On reprend l'expression trouvée dans le paragraphe précédent :

$$PFD_{avg} = \lambda^{D}_{barrage}\, T_1/2 + \lambda^{D}_{commande}\, T_3/2 + \frac{\lambda^{D}_{API1}\,\lambda^{D}_{API2}\,T_2^{\,2}}{3}$$

d'où $PFD_{avg} = 6.1537\ 10^{-4}$

6.10.2.2 Intervalle de temps entre proof test unique

Pour la méthode utilisant l'arbre des causes, on utilise :

$$PFD_{avg} = \lambda^{D}_{barrage}\, T_i/2 + \lambda^{D}_{commande}\, T_i/2 + \frac{\lambda^{D}_{API1}\,\lambda^{D}_{API2}\,T_i^{\,2}}{3}$$

On obtient

	Arbre des causes
PFD_{avg} (T_i 87 700 heures)	$7.1513\ 10^{-4}$
PFD_{avg} (T_i 168 heures)	$2.1432\ 10^{-7}$

6.10.3 PFD_{avg} par la méthode du graphe de Markov

La méthode « matricielle » par graphe de Markov ne permet pas de donner la PFD_{avg} dans le cas d'intervalles entre proof test multiples.
Le calcul avec des taux de défaillances différents ne sera pas exécuté ici. La modélisation du comportement est plus compliquée et n'apportera rien concernant l'application de la méthode.

6.11 Récapitulatif des résultats pour PFD_{avg}

	Bloc diagramme	Arbre des causes	Markov
PFD_{avg} (T_i différent, λ_{API} égaux)	$1.186\ 10^{-4}$	$1.187\ 10^{-4}$	N/A
PFD_{avg} (T_i unique = 168h, λ_{API} égaux)	$2.1249\ 10^{-7}$	$2.1249\ 10^{-7}$	$2.137\ 10^{-7}$
PFD_{avg} (T_i unique = 87 700h, λ_{API} égaux)	$2.1844\ 10^{-4}$	$2.1846\ 10^{-4}$	$2.1569\ 10^{-4}$
PFD_{avg} (T_i différent, λ_{API} différents)	$6.1536\ 10^{-4}$	$6.1537\ 10^{-4}$	N/A
PFD_{avg} (T_i unique = 168h, λ_{API} différents)	$2.1431\ 10^{-7}$	$2.1431\ 10^{-7}$	-
PFD_{avg} (T_i unique = 87 700h, λ_{API} égaux)	$7.150\ 10^{-4}$	$7.151\ 10^{-4}$	-

Il faudra faire attention lors du calcul matriciel au nombre d'itérations qui seront prises en compte. En effet, on approxime la valeur de PFD_{avg} en faisant une somme destinée à calculer la valeur moyenne. Or, une valeur moyenne étant une intégrale, le calcul de cette valeur par itérations sera plus précis si le nombre de pas de calcul est important. Plus le Ti est grand, plus le nombre d'itérations est grand et meilleur sera le résultat obtenu.

6.12 Cas de la forte sollicitation

Pour les systèmes en forte sollicitation, il faut déterminer la probabilité de défaillance dangereuse par heure PFH. On rappelle que
PFH = PFD/Temps de mission.
Dans ce cas, on ne peut qu'utiliser des intervalles entre proof tests uniques ; il ne faudra donc pas comparer avec les valeurs de PFD_{avg} obtenues pour des intervalles de temps entre proof tests multiples.

6.12.1 Méthode des blocs diagrammes

On rappelle ci-dessous l'expression trouvée pour PFD

$$PFD_{système}(t) = \lambda^D_{API1}\lambda^D_{API2} * t^2 + \lambda^D_{barrage}\, t + \lambda^D_{commande}\, t - \lambda^D_{barrage}\lambda^D_{commande}\, t^2 - \lambda^D_{barrage}\lambda^D_{API1}\lambda^D_{API2}\, t^3 - \lambda^D_{commande}\lambda^D_{API1}\lambda^D_{API2}\, t^3 + \lambda^D_{barrage}\lambda^D_{commande}\lambda^D_{API1}\lambda^D_{API2}\, t^4$$

Si on utilise un intervalle de temps entre proof test unique, on a l'expression

$$PFD = \lambda^D_{API1}\lambda^D_{API2} * T_i^2 + \lambda^D_{barrage}\, T_i + \lambda^D_{commande}\, T_i - \lambda^D_{barrage}\lambda^D_{commande}\, T_i^2 - \lambda^D_{barrage}\lambda^D_{API1}\lambda^D_{API2}\, T_i^3 - \lambda^D_{commande}\lambda^D_{API1}\lambda^D_{API2}\, T_i^3 + \lambda^D_{barrage}\lambda^D_{commande}\lambda^D_{API1}\lambda^D_{API2}\, T_i^4$$

D'où

$$PFH = PFD/T_i = \lambda^D_{API1}\lambda^D_{API2} * T_i + \lambda^D_{barrage} + \lambda^D_{commande} - \lambda^D_{barrage}\lambda^D_{commande}\, T_i - \lambda^D_{barrage}\lambda^D_{API1}\lambda^D_{API2}\, T_i^2 - \lambda^D_{commande}\lambda^D_{API1}\lambda^D_{API2}\, T_i^2 + \lambda^D_{barrage}\lambda^D_{commande}\lambda^D_{API1}\lambda^D_{API2}\, T_i^3$$

Pour un intervalle entre proof test de 10 ans et pour une architecture API identique,

$$PFH = 6.21\ 10^{-9}$$

6.12.2 Arbre des causes

PFH = PFD/temps de mission

L'expression de PFD a été obtenue dans le paragraphe PFD_{avg} du système par arbre des causes

$$PFD = \lambda^D_{barrage}\, T_i + \lambda^D_{commande}\, T_i + \lambda^D_{API1}\, \lambda^D_{API2}\, (T_{i\,API})^2$$

$$PFH = \lambda^D_{barrage} + \lambda^D_{commande} + \lambda^D_{API1}\, \lambda^D_{API2}\, T_i$$

Pour un intervalle entre proof test de 10 ans et pour une architecture API identique,

$$PFH = 2.5\ 10^{-11} + 2.5\ 10^{-9} + (2.05\ 10^{-7})^2\ \ 87700$$

$$PFH = 6.21\ 10^{-9}$$

6.12.3 Graphe de Markov

On rappelle la matrice de transition de l'ensemble

$$P = \begin{pmatrix} 1 - 2\lambda^D_{API} - \lambda^D_{barrage} - \lambda^D_{commande} & 2\lambda^D_{API} & \lambda^D_{barrage} + \lambda^D_{commande} \\ 0 & 1 - \lambda^D_{API} & \lambda^D_{API} \\ 0 & 0 & 1 \end{pmatrix}$$

Il faut ensuite à partir de l'état de départ S = (1 0 0), calculer par itérations successives la matrice $S_i = S_{i-1}\, P$ et la PFD associée qui est la $3^{ième}$ composante de la matrice S.

Dans ce cas, on ne peut pas avoir plusieurs intervalles de temps entre proof test. La valeur que l'on va choisir pour le nombre d'itérations correspond à la valeur du T_i et on ne peut prendre en compte qu'une seule valeur pour tous les sous-systèmes.

On peut représenter cela par l'algorithme suivant :
S = (1 0 0)
Sint = S*P
PFD = Sint(3); cela correspond à la troisième composante de la matrice

```
for i=1 to Ti
        S=Sint*P
        Sint=S
endfor
PFHint = S(3)

PFH = PFHint/Ti
```

La valeur numérique de la matrice est :

$$P = \begin{pmatrix} 0.9999995875 & 4.1\ 10^{-7} & 2.525\ 10^{-9} \\ 0 & 0.999999795 & 2.05\ 10^{-7} \\ 0 & 0 & 1 \end{pmatrix}$$ en prenant les λ^D incluant les taux de couverture.

On obtient les valeurs suivantes en fonction de T_i :

T_i	PFH
168	$2.531\ 10^{-9}$
8770	$2.888\ 10^{-9}$
87700	$6.099\ 10^{-9}$

6.12.4 Comparaison entre les différentes méthodes

Pour un temps entre proof test de T_i = 87700 h, on a

	Bloc diagramme	Arbre des fautes	Markov
PFH	$6.21\ 10^{-9}$	$6.21\ 10^{-9}$	$6.10\ 10^{-9}$

7 Récapitulatif des hypothèses faites

Rappel :
Les notions abordées dans ce document (taux de défaillance, probabilité de défaillance dangereuse) ne concernent que les pannes matérielles aléatoires.

1) Le temps d'indisponibilité (temps de détection de la panne, réparation et remise en service) est considéré petit devant le MTBF donc on confond le MTTF et le MTBF. Les pannes dangereuses détectées sont donc considérées réparées rapidement.
L'intervalle de temps entre test de diagnostic est petit devant le MTBF (facteur 100).

Ces deux hypothèses permettent de considérer comme dangereuse uniquement les pannes dangereuses non détectées.
Les pannes potentiellement dangereuses qui sont détectées sont considérées comme réparées suffisamment rapidement pour ne pas être dangereuses.

2) On suppose une répartition 50% pannes dangereuses, 50% pannes sûres; ceci permet d'utiliser dans le calcul du PFD le taux de défaillance λ donné par le constructeur ou calculé à l'aide des bases de données de fiabilité pour obtenir le taux de défaillance des pannes dangereuses.

$$\lambda^D = 0.5\ \lambda$$

3) Le taux de couverture des tests de diagnostic est approximé dans la plupart des cas. La méthode plus rigoureuse consisterait à faire une analyse des modes de défaillances (qui nécessite en toute rigueur de connaître les modes de défaillance des composants ou modules).

4) La détermination de la valeur de β caractérisant le mode commun reste à l'heure actuelle une estimation d'expert.

5) Pour le calcul de PFD_{avg}, le temps entre deux proof tests consécutifs T_i doit être déterminé en fonction des contraintes d'exploitation ainsi que des spécifications de maintenance du système. Si aucune spécification n'a été faite, il faudra estimer ce temps T_i. L'impact de cet intervalle de temps entre proof test est important sur le calcul de PFD_{avg}.

8 Conclusion et remarques

Il faut ici rappeler l'objectif de ce document qui est de présenter les différentes méthodes permettant de calculer la valeur moyenne de la Probabilité de Défaillance Dangereuse d'un système (cas des faibles sollicitations). Nous avons étendu le calcul au cas de la forte sollicitation.

Les résultats numériques de l'exemple ne sont en aucun cas à prendre comme référence pour d'autres applications. En effet, chaque fonction de sécurité nécessite une analyse et un calcul propre.

La méthode des **blocs diagrammes** est celle qui se rapproche le plus d'une représentation fonctionnelle du système. Cette méthode s'applique donc bien dans des cas simples et ne nécessite qu'une analyse fonctionnelle.

La méthode de **l'arbre des causes** nécessite en plus de l'analyse fonctionnelle, la détermination des défaillances qui vont amener à la perte de la fonction de sécurité. Cette méthode est plus complète que la méthode des blocs diagrammes.

La méthode des **graphes de Markov** est la plus compliquée des trois, tant au niveau de la modélisation que de l'exploitation du modèle pour la détermination de la probabilité de défaillance dangereuse. Toutefois, cette méthode, bien que plus complète pour modéliser les différents états du système, ne permet pas de considérer des intervalles de temps entre proof tests différents pour chacun des constituants du système.

La probabilité de défaillance dangereuse d'un système n'est qu'un des éléments qui permet de juger une fonction de sécurité. Elle ne concerne que les pannes matérielles aléatoires et fait partie des exigences demandées par la norme CEI 61508 ou CD CEI 62061 pour les systèmes complexes.

Annexe 1

"Les logiciels de calcul pour les blocs diagrammes"

SafeCalc

Cet outil est développé par Honeywell. Une version de démonstration est disponible sur le site http://europe.iac.honeywell.com/products/fsc/2product/safecalc.htm?session_id=1&timestamp=975939478290

Cet outil a été réalisé pour correspondre à la norme CEI 61508. Il permet, à partir d'une schématisation bloc diagramme, de calculer le SIL pour une installation. Il faut pour cela introduire le taux de défaillance, le taux de couverture, l'intervalle entre proof test. Cet outil ne permet pas d'utiliser plus d'une logique de traitement par fonction de sécurité.

PDS Tool

Cet outil a été développé par le SINTEF, organisme norvégien (voir sur le site http://www.sydvest.com/Products/products.htm).

Cet outil est de conception plus ancienne et reprend un concept propre à cet organisme. Il utilise tout de même les blocs diagrammes comme principe de modélisation.

SILK

Cet outil permet de déterminer un SIL en accord avec la norme CEI 61508. Il n'y a pas de version de démonstration. Pour tout renseignement, voir http://www.safety-sc.nl/software/silk.htm.

Reliability Workbench

Une version de démonstration de ce logiciel est disponible sur le site http://www.isograph.com/

Cet outil modélise les diagrammes blocs (modèle en série et parallèle) et avec la possibilité de modéliser des voteurs.

BQR Suite

Une version de démonstration est disponible sur le site http://www.bqr.com. Cet outil permet de mieux modéliser les systèmes série, //, voteur,… Ce logiciel utilise des bases de données de fiabilité (type MIL-HDBK ou RDF) et permet aussi de faire de la modélisation par arbre des fautes.

L'avantage de cet outil est de pouvoir décomposer un système en sous-système

Ce logiciel ne permet pas d'utiliser des taux de défaillances plus petits que 10^{-9}.

Relex Reliability Software

Une version de démonstration est disponible sur le site http://www.relexsoftware.com. Cet outil permet de modéliser des blocs diagrammes très sommairement (modèle parallèle ou série) en utilisant une base de données composant.

SUPERCAB

Un aperçu des possibilités de ce logiciel est disponible sur le site http://www.cabinnovation.fr/

Ce logiciel permet de modéliser par l'intermédiaire des blocs diagrammes un système et il fournit les données classiques de fiabilité (MTBF, …). Apparemment, il n'y a pas de bases de données associée avec ce logiciel.

Item Toolkit Software

Voir sur le site http://www.itemsoft.com/download.html. Cet outil n'a pas été essayé.

C'est un outil qui permet aussi de faire des blocs diagrammes et qui utilise des bases de données de composant type MIL, Bellcore.

RAM Commander

La version de base de ce logiciel fournit exclusivement une base de données de fiabilité de composants électroniques. Il existe en option la modélisation par blocs diagrammes.

Annexe 2

"Les logiciels de calcul pour les arbres des fautes et Markov"

On donnera ici pour information quelques logiciels existant sur le marché. Ces logiciels permettent de déterminer le taux de défaillance de l'événement principal (celui au sommet de l'arbre).
On dispose ainsi comme outil pour les arbres de fautes :

- Risk Spectrum; logiciel permettant de traiter les modes communs sans base de données propre; les valeurs des taux de défaillance sont à entrer par les utilisateurs. Cet outil est vendu par Sector (cf. http://www.sector-sa.com/)
- Simtree; logiciel simple d'arbre de défaillance
- Cabtree disponible sur le site http://www.cabinnovation.fr/
- BQR Suite
- Reliability Workbench
- Relex reliability software

Pour les outils pour les graphes de Markov:

- MARK_Exd
- Carms (voir le site http://rac.iitri.org/DATA/RMST/rel_model.html#CARMS)
- Reliability Workbench
- BQR Suite

Bibliographie

[1] CD CEI 62061 : Sécurité des machines – Sécurité fonctionnelle des systèmes électriques / électroniques / électroniques programmables relatifs à la sécurité, Version n° 44/380/CD, mai 2002, 90p

[2] CEI 61508 – Sécurité fonctionnelle des systèmes électriques / électroniques / électroniques programmables relatifs à la sûreté.
Partie 1 : Prescriptions générales, UTE C 46-061, avril 2000, 59 p.
Partie 2 : Exigences pour les systèmes électriques / électroniques / électroniques programmables, UTE C 46-062, avril 2000, 71 p.
Partie 3 : Prescriptions concernant les logiciels, UTE C 46-063, avril 2000, 49 p.
Partie 4 : Définitions et abréviations, UTE C 46-064, avril 2000, 26 p.
Partie 5 : Exemples de méthodes pour la détermination des niveaux d'intégrité de sécurité, UTE C 46-065, avril 2000, 28 p.
Partie 6 : Guide pour l'application des parties 2 et 3, UTE C 46-066, avril 2000, 75 p.
Partie 7 : Bibliographie des techniques et des mesures, UTE C 46-067, avril 2000, 115 p.

[3] Sûreté de fonctionnement des systèmes industriels – Fiabilité – Facteurs humains – Informatisation A. Villemeur 1988

[4] Control Systems Safety Evaluation & Reliability 2nd Edition William M.Goble 1998

[5] ANSI/ISA – 84.01 – 1996 : Application of Safety Instrumented Systems for the Process Industries

[6] Systèmes à haute disponibilité – Concepts – Technique de l'ingénieur – R. J. Chevance – 08/1999

[7] Document de travail INRS S/00DT – 003 : Quantification du niveau de sécurité de dispositifs à composants complexes : étude critique

[8] Contribution à la réalisation d'une architecture redondante d'ordre 2 hétérogène. Stage de fin d'étude de C. Blanchet, élève ENSEM, août 2000

[9] Philippe Charpentier – Architecture d'automatisme en sécurité des machines : Etude des conditions de conception liées aux défaillances de mode commun, Rapport de thèse, juillet 2002, 156 p.