Institut National
Polytechnique de Lorraine

Ecole doctorale IAEM Lorraine
Département de Formation Doctorale en Automatique

THÈSE

présentée et soutenue publiquement le 13 Janvier 2011
pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine
spécialité Automatique, Traitement du signal et Génie Informatique

Conception sûre des systèmes mécatroniques intelligents pour des applications critiques

Par
HICHAM BELHADAOUI

Composition du jury

Président :	Pr. José RAGOT (CRAN, INPL)
Rapporteurs :	HDR Dr. Zineb SIMEU-ABAZI (G-SCOP, INPG) Pr. Frédéric KRATZ (PRISME, ENSI de Bourges)
Examinateurs :	Pr. Luc HEBRARD (InESS, UdS) Pr. Jean-François AUBRY (CRAN, INPL), *Directeur de thèse* Dr. Olaf MALASSE (A3SI, Arts et Métiers ParisTech, Metz) , *co-Directeur*
Invités :	HDR Dr. Nicolas HERAUD (SPE, Université de Corse) Jean-Claude BOEHM (CETIM) Dr. Karim HAMIDI (IRSN)

Centre de Recherche en Automatique de Nancy
UMR 7039 - Nancy Université - CNRS
2, Avenue de la Forêt de Haye
54516 Vandœuvre-lès-Nancy
Tel. +33 (0) 83 59 59 59 - Fax +33 (0) 3 83 59 56 44

Remerciements

Ces travaux ont été effectués au sein de l'équipe A3SI (dirigée par Mr Olaf MALASSE) du centre Arts et Métiers ParisTech de Metz qui a cofinancée ce travail, et l'équipe Architecture des Systèmes de l'ENSEM de Casablanca.

Cette thèse a été dirigée par le Professeur Jean-François AUBRY, mon codirecteur de thèse français, à qui je tiens à exprimer ma reconnaissance et mes profonds remerciements pour son encadrement, pour sa disponibilité pendant nos réunions à l'INPL, ou dans les diverses réunions avec les membres du projet de la fondation CETIM. Durant la période de ma thèse, j'ai assisté à toutes les réunions effectuées au sein de l'équipe SACSS (Systèmes Automatisés Contraints par la Sûreté de Fonctionnement et la Sécurité) dirigée par le Professeur Aubry, et qui fait partie du groupe thématique SURFDIAG (Sûreté de Fonctionnement et Diagnostic des systèmes) du Centre de Recherche en Automatique de Nancy (CRAN). Merci monsieur le professeur pour vos efforts et l'intérêt que vous avez apporté à mon travail.

Je tiens à témoigner toute ma gratitude et ma sincère reconnaissance à Mr Olaf MALASSE Directeur de centre de compétence A3SI, de m'avoir offert l'opportunité d'un PFE et l'occasion de poursuivre par une thèse, ainsi que pour l'encadrement direct de ce travail. Merci de votre patience, et de votre implication matérielle et morale.

Je tiens également à exprimer mes remerciements à Mr Grégory BUCHHEIT, ingénieur de recherche au centre A3SI, pour sa coopération et sa participation à ce travail.

Je remercie le Professeur José RAGOT, Professeur d'enseignement supérieur (CRAN, INPL), pour l'honneur qu'il me fait en présidant ce Jury de thèse.

Je tiens à remercier également :

- Madame Dr.HDR Zineb SIMEU-ABAZI,
- Monsieur Pr. Frederic KRATZ,
- Monsieur Pr. Luc HEBRARD,
- Monsieur Dr. HDR Nicolas HERAUD,
- Monsieur Dr. Karim HAMIDI-GEORGES,
- Monsieur Dr. Nicolae BRINZEI,

Pour l'honneur qu'ils me font en participant à mon Jury de thèse, et particulièrement Madame Zineb Simeu-Abazi Maître de Conférences Habilité du G-SCOP de l'INP de Grenoble, Frédéric Kratz Professeur à l'ENSI de Bourge qui ont accepté d'être rapporteurs de mes travaux.

Merci à tous les membres du consortium CETIM pour leur disponibilité, leur bonne humeur et leur ambiance scientifique très riche pendant les réunions. Je veux remercier l'ensemble des

permanents pour leurs critiques et conseils, et l'ensemble des doctorants : Mehdi JELLOULI et Benoît DUBOIS pour l'ambiance conviviale qu'ils y mettent.

Je remercie la Fondation CETIM, Fondation de France pour son soutien à ce projet, et par conséquent à ma formation.

Je remercie Monsieur Jean Claude BOEHM le représentant de la Fondation CETIM qui nous a accompagnés durant toute cette expérience.

Il me reste à remercier toutes les personnes que j‟ai rencontrées durant cette aventure et avec qui j‟ai travaillé durant ces trois années.

Enfin, je dédie ce paragraphe à tous ceux qui m‟ont aidé et supporté dans cette épreuve. Je pense à mes amis et à ma famille grâce à qui j‟ai pu en arriver jusque là.

À tous les êtres chers
dont le soutien m'a été indispensable.

Table des matières

Chapitre 4 : Evaluation fiabiliste de l'architecture d'un capteur mécatronique

Liste des figures

Chapitre 0

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Conclusion et annexes

Liste des tableaux

Chapitre 3

Chapitre 4

Annexes

Chapitre 0 :
Introduction générale

Le mirage de l'ère du tout numérique entraîne notre société vers plus d'inconscience et d'insouciance envers les systèmes intelligents qui sont magnifiés à des fins mercantiles. La généralisation du Plug and Play (ajout facilité et transparent de fonctionnalités aux systèmes) impacte inévitablement la gestion et le traitement des fonctions précédemment installées: nouvelles interactions ; nouveaux partages des temps… La complexité des systèmes devient transparente en apparence avec la miniaturisation et l'informatisation. Elle existe toujours, mais n'est plus aisément appréhendable et visible par l'homme (cela résulte aussi des outils de CFAO qui permettent de concevoir et de réaliser sans parfaite maîtrise les systèmes). Dans une société mercantile, le Bug est banalisé (1 Bug ≡ 1 Reset, ou on jette : c'est bon pour le commerce !). Le système industriel a, quant à lui, une durée de vie de 4 à 50 ans !! Et si un Reset est possible sur un système isolé, il existe ici un risque d'engagement de la sécurité pouvant concerner plusieurs milliers de personnes (plusieurs millions dans le cas du Nucléaire).

L'ingénierie des systèmes complexes, avec ses outils de conception et développement, fait partie des marchés applicatifs reconnus à forte valeur ajoutée (System@tic, ITEA…). Ces systèmes (6 Md $ en 2007) embarqués et distribués, à haute exigence en sûreté de fonctionnement, font massivement recours à des architectures (circuits et systèmes) à électronique programmable, où le logiciel est devenu important. Une demande toujours croissante en performances, des contraintes économiques plus tendues, une croissance exponentielle de la taille des circuits/systèmes et du code gérant ces circuits ont pour conséquence de rendre la conception, la réalisation et la mise au point de ces systèmes de plus en plus délicate (que penser alors des conditions d'un rétrofit ?). Même les plus grandes sociétés ne sont plus à même de circonscrire la durée de la phase pré-commerciale et plus grave encore, de commercialiser un circuit/système exempt d'erreurs. La vérification fonctionnelle des systèmes combinant à la fois matériel et logiciel ne s'effectue actuellement qu'à la fin du processus de conception. Cette phase se révèle excessivement coûteuse et pénalisante au sein du processus d'intégration produit/système. Les bureaux d'ingénierie ont tendance à développer en premier lieu un modèle abstrait du système, pour ensuite en tester les fonctionnalités, la fiabilité et les performances que des composants basés sur un tel modèle seraient à même d'offrir. L'enjeu économique est d'importance [Bergeron et al., 2010], la validation peut représenter jusqu'à 70% de l'effort de conception, 80% des lignes de code d'un projet, et il existe plus d'ingénieurs de vérification que de concepteurs.

	Automobiles (USA)	Avions commerciaux (USA)
Unités déployées	100 000 000	10 000
Heures opérationnelles / an	30 000 millions	55 millions
Coût par véhicule	20 000 $	65 millions $
Mortalité / an	42 000	350
Accidents / an	21 millions	170
Mortalité / million d'heures	0,71	6,4
Qualification des opérateurs	Faible	Élevée
Niveau de redondance	Uniquement sur les freins	Tout système critique

Figure 0.1 : Eléments de comparaison entre l'avionique et l'automobile (source P. Koopmann – Carnegie Mellon)

L'ingénierie de la sûreté ne peut se satisfaire d'a priori, aussi crédibles et partagés soient-ils ! Le risque associé aux transports de masse reste très supérieur à celui du transport individuel (Figure 0.1). Les précautions et investissements dédiés à la sûreté et la sécurité dans la conception et l'exploitation des aéronefs sont parfaitement justifiés au regard de la mortalité comparée au nombre d'heures de fonctionnement. La massification des transports, en l'état actuel des techniques et organisations industrielles, représente un risque important mal appréhendé par les gestionnaires et la technocratie.

Les ingénieries de l'électronique numérique et du logiciel occupent donc une place prépondérante dans le processus de conception et de vérification des systèmes informatisés. Les niveaux d'intégration des composants électroniques, de complexité et de taille du code embarqué croissent continuellement et cette croissance s'accompagne d'une plus grande sensibilité aux perturbations (les environnements naturel et technologique devenant eux aussi plus agressifs) et aux autres attaques (exigence forte quant à la capacité des systèmes à communiquer).

Un système est un ensemble complexe de fonctions, soumis à des aléas (déclenchement d'erreurs systématiques, bit flips, défaillances matérielles), devant rendre un service défini, quelque soient son état interne, l'état de son environnement et le niveau de stress appliqué (Figure 0.2).

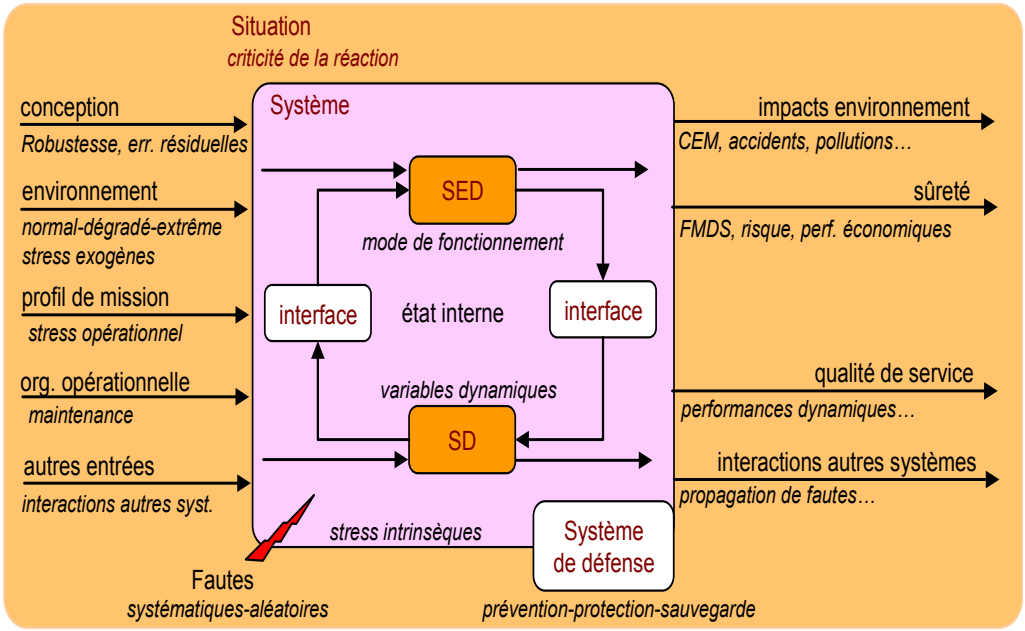


Figure 0.2: Interaction du système et son environnement.

Tous les systèmes mécatroniques (Figure 0.3) sont complexes. Le nombre des combinaisons possibles des états des parties pose un problème d'explosion combinatoire qui conduit à des nombres gigantesques d'états possibles. Cette complexité est un défi pour la sûreté de fonctionnement. Pour l'augmenter on a souvent recours à la redondance, déploiement d'une multitude de versions différentes d'un même schéma ou motif (pattern) qui participe elle-même à l'augmentation de la complexité. On parle de redondance fonctionnelle (exemple : la reprise en mode dégradé ou de secours d'une fonction) ou de redondance structurelle qui désigne des structures différentes pour exécuter une même fonction (comme le double circuit

de freinage d'une voiture automobile ou plusieurs ateliers différents ou usines différentes pour fabriquer une même pièce ou un même engin).

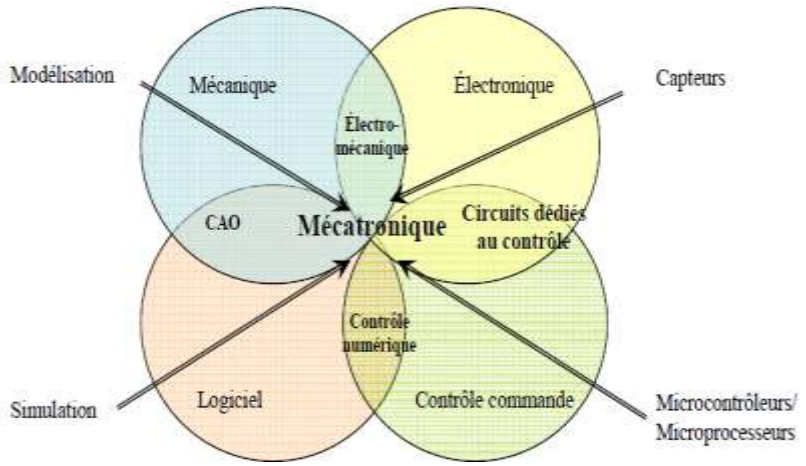


Figure 0.3 : Synergie des technologies pour construire les systèmes mécatroniques.

La structure fonctionnelle des systèmes (ensemble logiciel et matériel) est de facto devenue complexe et variable. Prévenir et éliminer les fautes font partie des attendus des processus de développement et de vérification. Les fautes issues d'erreurs résiduelles à la conception, ou réalisation des composants électroniques, ou du logiciel, sont certes localisables, identifiables et reproductibles, mais difficilement détectables ! Les causes potentielles de défaillance se révèlent multiples : matérielles, logicielles, environnements de développement…La non cohérence, les combinaisons d'erreurs potentiellement latentes et dormantes dépendant de l'état du système et la complexité des applications rendent l'analyse difficile. Les types de défaillance [Aiguo L. et Bingrong H., 2007] peuvent également se combiner : défaillance d'exécution (non progression, boucle infinie, plantage), fonctionnelle (mauvais traitement), opérationnelle (délai de traitement non respecté, indisponibilité).

Le traitement (détection et recouvrement) des erreurs apporte, au prix d'une augmentation des niveaux de complication et de complexité du système, la garantie d'une meilleure probabilité de comportement satisfaisant du système. L'intégration de mécanismes (matériels et logiciels) de tolérance aux fautes intrinsèques par conception, par la commande ou par diagnostic et reconfiguration, modifient sensiblement l'architecture matérielle et logicielle (firmware, application) et impactent la dynamique et la FMDS (fiabilité, maintenabilité, disponibilité et sécurité) des systèmes. Les aspects fonctionnels et dysfonctionnels doivent être pris en considération pour l'évaluation des performances (en terme fiabiliste et de qualité de service) d'un système. De nombreux référentiels encadrent le développement des systèmes contraints par des impératifs de sûreté et de sécurité, tel la norme générique CEI 61508 [CEI 61058] décrivant le processus de développement des systèmes E/E/EP dédiés aux applications relatives à la sécurité des systèmes industriels.

La sûreté de fonctionnement d'un système peut-être définie comme l'aptitude d'une entité à assumer une ou plusieurs fonctions requises dans des conditions données [CEI 50191], dont la variation sur occurrence d'évènement(s) peut induire une défaillance du système. Les dépendances (réelles ou potentielles, dans leur acception explicite ou implicite) entre parties d'un système, induisent des interactions entre elles, rendant difficile la complétude des spécifications du système. Cette complexité, intrinsèque à l'ingénierie systèmes, la confronte de fait aux problématiques des interactions multi-physiques et inter-disciplinaires. La

mécatronique, combinaison synergique et systémique de mécanique des solides et des fluides, d'électronique, et d'informatique temps réel…procède de cette complexité (Figure 0.3)

Les moyens de garantie de la Sûreté de Fonctionnement (Dependability) (Figure 0.4) sont généralement assimilés à l'élimination, la prévention et la tolérance aux fautes [Lap 04]. L'étude de ses propriétés mêle analyse de la Fiabilité (Reliability), de la Maintenabilité (Maintenability), de la Disponibilité (Availability), de la Sécurité innocuité (Safety), de la Confidentialité (Security) et de l'Intégrité (Intégrity) de l'information. Des attributs secondaires ont également été définis : la Robustesse (persistance d'un comportement correct en présence de fautes), la Résilience (capacité d'adaptation), l'Authenticité, la Survivabilité…

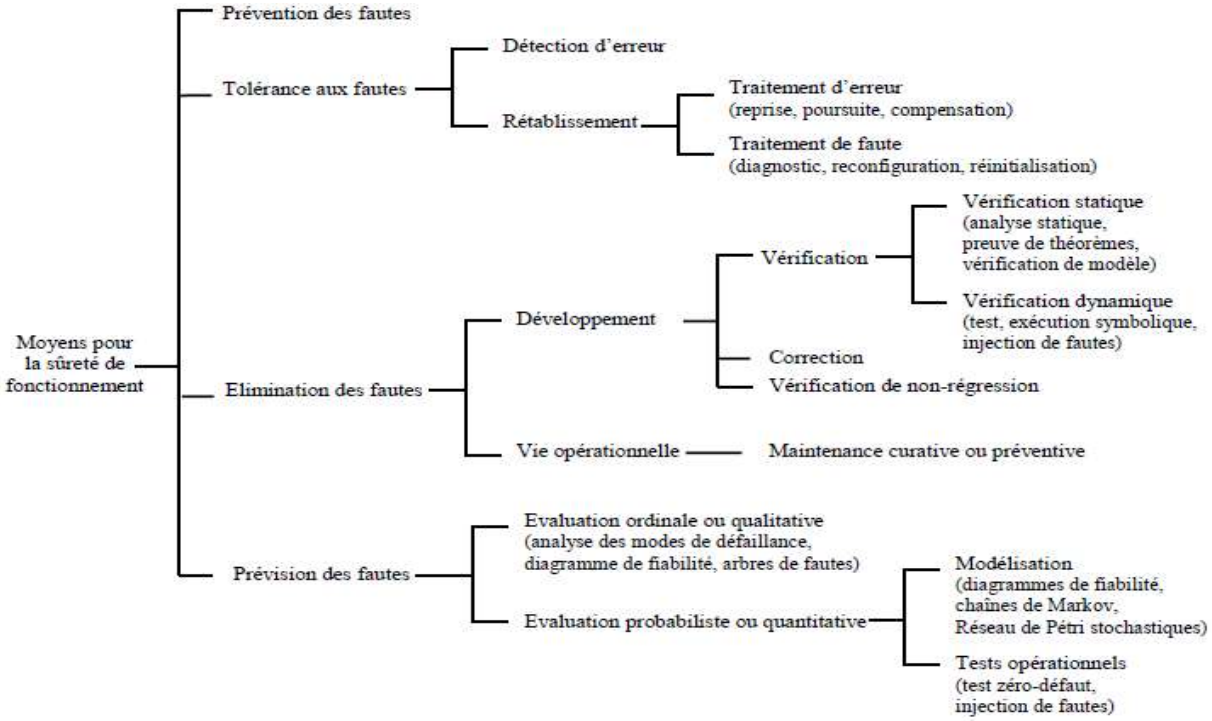


Figure 0.4 : Moyens d'intégration de la sûreté de fonctionnement [Lyonnet P., 2006].

L'évaluation des performances fiabilistes (FMDS) des systèmes embarqués complexes nécessite le développement de nouvelles approches. Dans les systèmes intégrant du logiciel, la structure fiabiliste des fonctions dépend du logiciel. La recherche des séquences d'événements menant à la défaillance du système doit donc associer logiciel et matériel. La méthode doit contribuer à l'analyse qualitative et quantitative de la sûreté des systèmes et microsystèmes.

Ce rapport de thèse est structuré pour aborder en détail chacun de ces volets :

- Les définitions indispensables à la compréhension des concepts de base de la sûreté de fonctionnement sont abordées au début du chapitre I.

- Les méthodes et les outils nécessaires pour entreprendre les analyses prévisionnelles de sûreté de fonctionnement pour des systèmes en cours de conception sont classés selon la mention qualitative et quantitative dans le chapitre I.

- L'intégration de la sûreté de fonctionnement en ingénierie système et notamment dans la conception des composants d'automatismes intelligents [Seung J.R. et Kosuke I., 2003], [Wen Y.K. 2001] est présentée dans le chapitre II. Nous proposons une architecture de capteur intelligent pour une application mécatronique. Nous détaillons dans le même chapitre les problématiques liées à la sûreté de fonctionnement des capteurs intelligents.

- L'analyse fiabiliste pour la conception des systèmes intelligents à signaux mixtes, et la généralisation de la méthode du flux informationnel [Hami et al.,2005] dans la partie logique programmée du capteur sont des points traités dans le chapitre III.

- L'étude globale de l'architecture d'un capteur intelligent, illustrée par une évaluation fiabiliste de l'exécution d'une application logicielle, embarquée sur une architecture de processeur à pile, est réalisée en détail dans le chapitre IV. La robustesse de ce type de capteur est étroitement liée aux techniques de tolérance aux fautes utilisées. Nous consacrons la dernière partie du chapitre IV à l'étude de l'impact de ces techniques sur les performances.

Chapitre 1 :

Sûreté de fonctionnement et ingénierie système

1.1 Introduction

Les systèmes numériques de contrôle/commande ont conquis quasiment tous les domaines d'applications [Barana et al., 2004] [Khobare et al., 1998], quelque soit leur niveau de criticité. Parmi les caractéristiques partagées par ces systèmes nous citons leur complexité intrinsèque croissante (intelligence distribuée, redondances, fonctionnalités hiérarchisées...) et leur interaction avec l'environnement (contraintes temps réel, applications critiques). Cette complexité croissante rend obsolète les méthodes de conception et d'évaluation actuelles. Il est illusoire de maîtriser complètement la conception et la pratique des systèmes complexes. Les méthodes de conceptions ne permettent qu'une approche partielle et parcellaire des problèmes : simplifications des contraintes mal maîtrisées, modèles inadéquats, complétude du prototypage, outils de conception et technologies limités… Il n'existe pas de solution entièrement satisfaisante pour la résolution de ces problèmes. C'est la raison pour laquelle nous recherchons à améliorer et valider sa conception. L'enjeu actuel est de définir un système, intégrant des interactions internes (matériel/logiciel, fonctionnelles, dysfonctionnelles) et externes (environnements).

Lors de la conception d'un système, les concepteurs doivent respecter des exigences (les délais, les coûts, la sûreté de fonctionnement, les performances) exprimées par différentes parties : concepteur (chercheurs et ingénieurs), maître d'œuvre (intégrateurs), maîtres d'ouvrages, exploitants…

Dans le domaine des systèmes programmables, de nombreuses méthodes et outils ont été développés pour faire face à la complication et à la complexité croissante. Une méthodologie systémique s'avère nécessaire pour garantir que le système conçue respecte les exigences du cahier des charges.

Une recherche bibliographique montre que les principaux défauts de conception sont dus à:

- des besoins mal spécifiés, incomplets ou erronés ou des exigences (cahier des charges) mal formulées,
- l'évolution des besoins ou des exigences dans le temps,
- la non accumulation de savoir-faire et le manque de retour d'expérience,
- l'évolution technologique,
- la définition erronée d'interfaces,
- la pression de la concurrence,
- l'extension d'exigences fonctionnelles.

Généralement, dans la conception des systèmes complexes, chaque fonction pouvait être étudiée et développée indépendamment des autres et l'implication de la sûreté de fonctionnement se résumait à la réutilisation de modèles génériques basés sur le retour d'expérience. Cette approche conventionnelle ne permet pas la prise en compte des risques liés à l'interaction entre ces fonctions matériel/logiciel. Il est donc important de formuler les exigences de sûreté de fonctionnement non seulement localement (pour chaque sous système), mais globalement (pour le système entier). Cela revient à formuler ces exigences au niveau du système complet et, ensuite, à les décliner à des niveaux plus bas (jusqu'aux simples composants).

Le «cycle en V» est traditionnellement utilisé dans la description du cycle de développement d'un système embarqué. Le cycle est composé de deux branches. La branche descendante, qui

correspond à une démarche de raffinements successifs, décrit les phases de conception allant de l'abstrait, qui démarre avec l'expression des besoins, au particulier. La branche ascendante détaille les phases d'intégration et de validation correspondant à chaque phase de conception.

La conception d'un système suivant le cycle en V génère un retarder du au choix de la technologie de réalisation. Dans cette optique, les efforts sont concentrés que sur la spécification et non sur les interactions au moment de l'intégration, ce qui entraîne le non maîtrise du comportement réel du système intégré.

Les systèmes actuels sont de plus en plus complexes car ils intègrent des fonctionnalités supplémentaires, des technologies variées et doivent être opérationnels sur des périodes longues. Ces systèmes demandent aussi de longues périodes de développement pendant lesquelles les besoins et les technologies évoluent. Résultat les coûts et les délais deviennent rédhibitoires compte tenu d'un environnement où la compétitivité mondiale est de rigueur. Pour prendre en considération ces conditions, une discipline a été définie, il s'agit de l'Ingénierie Système (IS).

La particularité de l'IS est de considérer à la fois les aspects techniques et les aspects logistiques (financiers, stockage, transport). L'IS a été traditionnellement une discipline appliquée aux systèmes physiques ; depuis une dizaine d'années, l'IS s'applique dans la conception des systèmes complexes. Les domaines concernés sont la télécommunication, les systèmes d'information, les systèmes de transport, les services financiers et autres applications allant du domaine médical au génie des procédés.

L'ingénierie système, science transdisciplinaire par obligation [Angeli A.M., 1996], a été définie dès les années soixante dans le standard Mil-Std-499. Elle doit intégrer des spécialistes de disciplines différentes dans un projet commun. Une grande partie des secteurs de l'ingénierie est concernée (électronique, automatique, informatique, mécanique…). Ces domaines sont concernés par cette discipline agrégative qu'est l'ingénierie système, et qui a engendré de nouvelles appellations, comme la mécatronique [(1)]… Elle propose un processus structuré de la conception d'un système. Elle a pour objectif de formaliser et d'appréhender la conception des systèmes complexes. Les premiers organismes à s'y intéresser ont donc été les grandes institutions de la défense américaine (NASA, USAF), afin de cadrer le développement de leurs programmes au travers d'approches industrielles plus rationnelles. Ils ont donné une définition assez ambitieuse à l'ingénierie système :

"Systems engineering is a robust approach to the design, creation, and operation of systems. In simple terms, the approach consists of identification and quantification of system goals, creation of alternative system design concepts, performance of design trades, selection and implementation of the best design, verification that the design is properly built and integrated, and post-implementation assessment of how well the system meets (or met) the goals".

[(1)] *Mécatronique : néologisme formé en 1969 par un ingénieur de Yasukawa, ce terme désigne une démarche qui regroupe les savoir-faire mécanique, électronique et informatique pour concevoir des produits plus performants, plus compacts et moins onéreux.*

Si l'ingénierie des systèmes matériels et celle des logiciels ont évolué en toute indépendance, les normes et directives récentes mettent l'accent sur la nécessité d'intégrer ces deux processus [Al-Dhaher A. H. G., 2001] (figure 1.1).

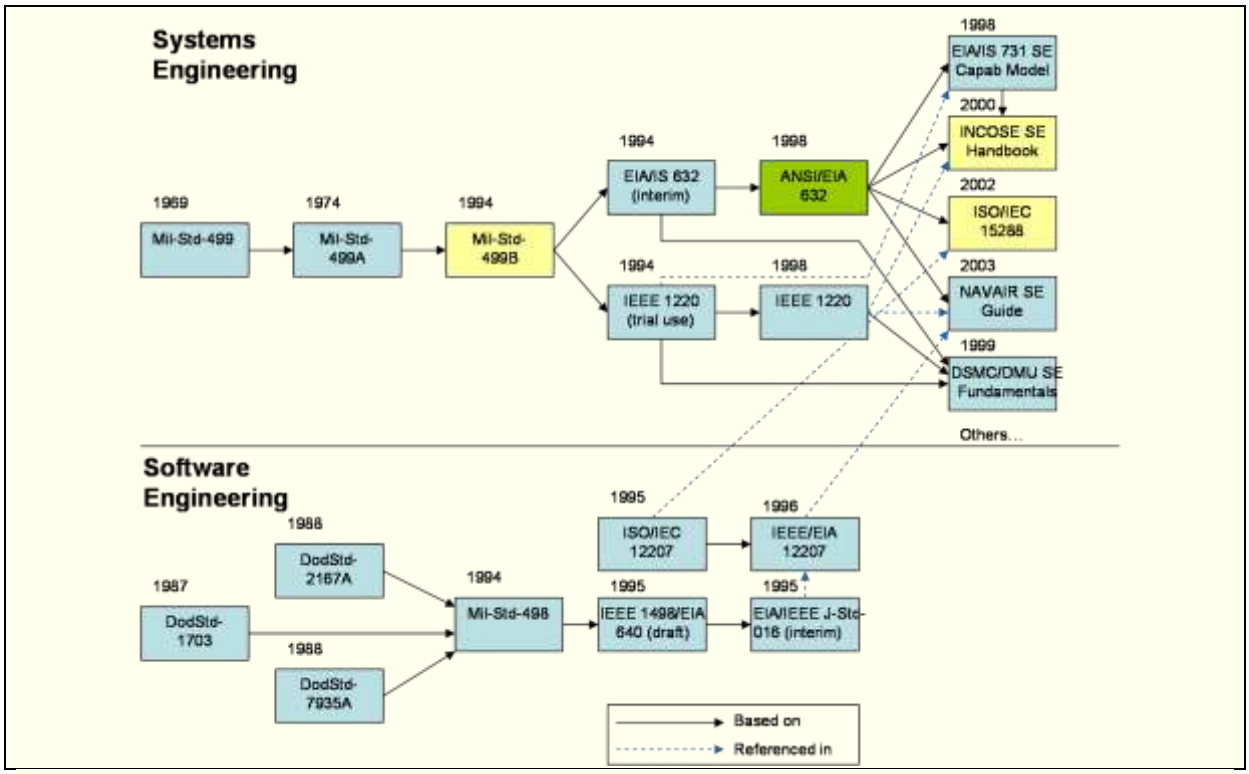


Figure 1.1 : Intégration des processus de développement matériel et logiciel (USA)

L'accélération des développements technologiques, ces dernières années, s'est traduit par l'augmentation du nombre des composants élémentaires dans les systèmes. Un système complexe est un système composé d'un grand nombre d'entités en interaction locale et simultanée. Il n'existe pas de définition formelle de ce qu'est un système complexe, mis à part un seul consensus qui s'établit autour de certaines propriétés comme la complexité des interactions, boucles de rétroaction, et intégration de sous-systèmes. Un système complexe est un système structuré. Sa structure subit une variation selon l'application [Goldenfeld et al., 2006], dans laquelle le nombre de composants indépendants en interaction est grand. Il existe de multiples voies par lesquelles le système peut évoluer [Whitesides et al., 2007], elles sont très sensibles aux conditions initiales ou aux petites perturbations. L'ingénierie système intègre le développement et l'organisation des systèmes complexes.

1.2. Processus de conception

Les systèmes complexes requièrent une grande rigueur, lors de leur conception, dans les choix des architectures et l'intégration de leurs composants. D'où la nécessité d'activités d'ingénierie complémentaires telles l'ingénierie du contrôle/commande, de la conception d'interface (extensibilité des systèmes), l'ingénierie de la performance, le génie logiciel (SysML, CMMI, méthodes orientées objet, ingénierie des exigences, langage formels), l'ingénierie de la fiabilité (s'applique à tous les aspects du système, élément essentiel de l'ingénierie de la sécurité, s'appuie très largement sur les statistiques, la théorie des probabilités), l'ingénierie de la sécurité (identification, minimisation des risques essentiels à

la sécurité), l'ingénierie de la sûreté (domaine interdisciplinaire qui intègre l'ingénierie du contrôle/commande, de la fiabilité, de la sécurité et des systèmes), l'ingénierie de la maintenance (maintien opérationnel des fonctions).

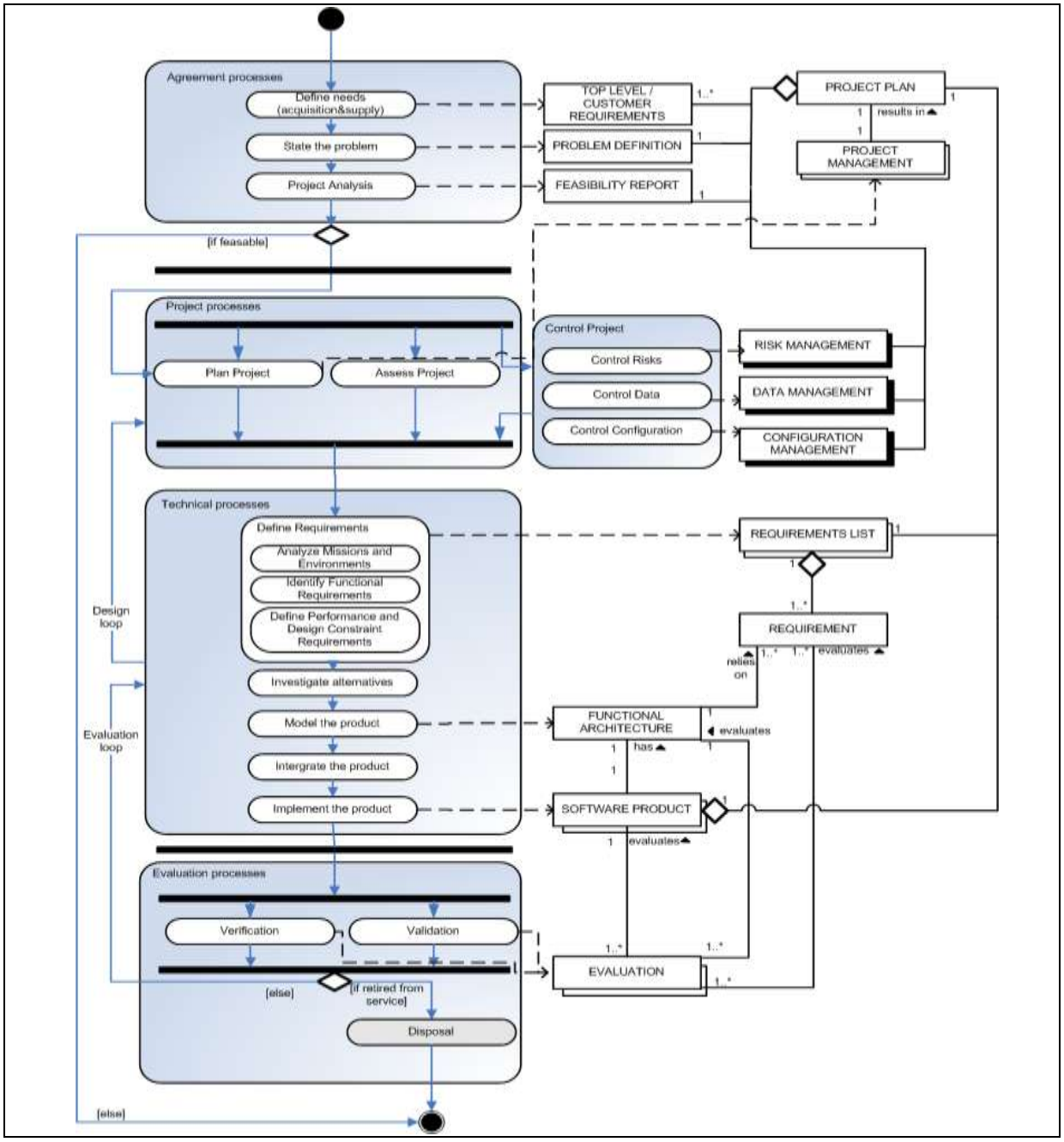


Figure 1.2 : Processus de développement d'une architecture programmable

Le processus de développement au sens de la MIL-STD-499A (Figure 1.2) est constitué de :

- Transformation d'un besoin opérationnel en termes de performances, la configuration d'un système est le fruit d'un processus itératif allant de sa définition aux essais et à son évaluation, via les phases de synthèse, d'analyse et de conception.
- Intégration des paramètres techniques permettant d'assurer la compatibilité physique, fonctionnelle, et informatique optimisante.

➢ Intégration de la fiabilité, la maintenance, la sécurité, et tout facteur influençant les coûts de possession financier et techniques du système.

Lors de la conception d'un système complexe incluant de nouvelles technologies, afin de mener dans les meilleures conditions le projet, les concepteurs utilisent des normes qui les aident à accomplir leur mission durant le développement. Parmi ces normes, on peut citer la norme *EIA-632* [STA-EIA], *ISO 15288* et *IEEE 1220*.

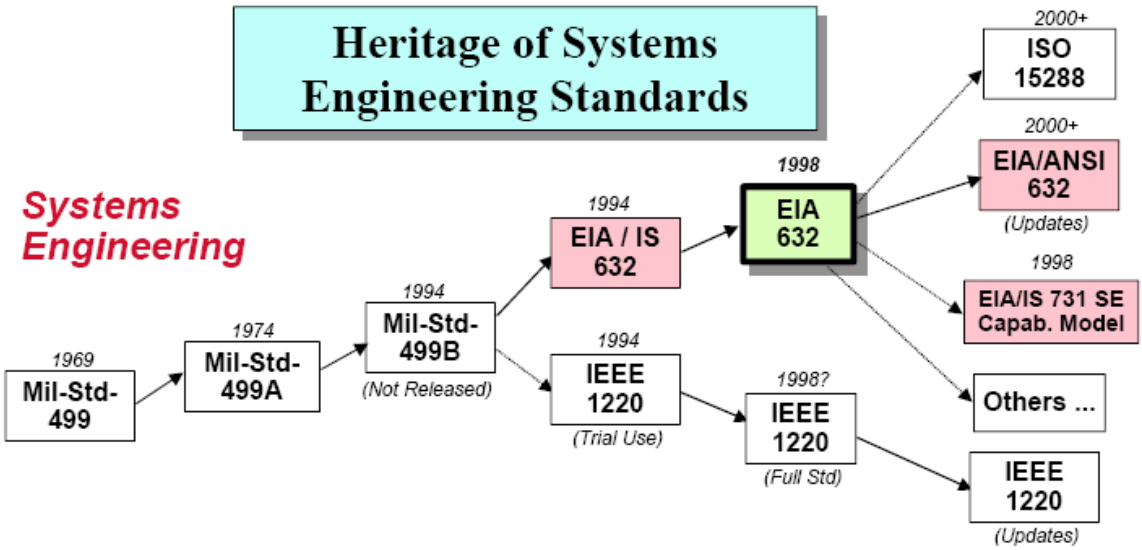


Figure 1.3 : Evolution des principales normes de l'IS [Verri10].

1.2.1. Structure d'un système selon l'EIA-632

La norme EIA-632 est l'une des normes les plus utilisées dans le domaine de l'Ingénierie Système [Sadou et al., 2005]. Une telle norme a été déployée principalement dans des industries de l'aéronautique, de production et militaires. Cette norme complète les processus techniques de définition du système en couvrant la réalisation des produits jusqu'à leur mise en service (mise en utilisation). De plus, elle inclut les processus contractuels d'acquisition et de fourniture.

Un processus est un ensemble d'activités interactives [AFIS] coordonnées pour transformer progressivement des éléments d'entrée en produits. Un processus normalisé décrit les types d'activités à réaliser et de résultats attendus de ces activités. Ces processus doivent répondre à un certain nombre d'exigences selon des normes [EIA 1999].
La norme *EIA-632* définit 13 processus de développement :

- 3 processus de Management Technique
- 2 processus d'Acquisition et de Fourniture
- 2 processus de Conception
- 2 processus de Réalisation
- 4 processus d'Evaluation Technique

Ces processus sont utilisés à chaque niveau de hiérarchisation des produits finaux, pour chaque module, pour les spécifier, les modéliser et, bien sûr, pour les développer.

Selon les informations recueillies sur l'échec de projets industriels [Sahraoui A.E.K.,2006], généralement la cause de l'échec du projet est la négligence des produits qui contribuent à la conception du système en question. En effet, la plupart des partenaires se focalisent uniquement sur le développement des produits finaux (*End products*). La norme *EIA-632*, introduit le concept de produit capacitant (enabling product). Il s'agit des étapes qui permettent d'obtenir un produit final et peuvent être utilisés par plusieurs produits finaux.

Le développement du système selon la norme industrielle (*EIA-632*) consiste à décomposer le système entre produits finaux et étapes à suivre et d'explorer un ensemble de processus qui sera appliqué pour le développement des produits finaux.

L'infrastructure pour le développement se base sur les problèmes de la logistique, qui sont considérés comme des produits capacitants dans le cadre de l'*EIA-632*. Ces produits capacitants seront utilisés pour exécuter l'ensemble des processus associés au développement, à la production, au déploiement, et à la formation des opérateurs afin d'utiliser les produits finaux.

Généralement il y a sept types de produits capacitants (étapes à suivre) contribuant à la production d'un produit final : la phase de développement, la production, le test, le déploiement, la formation des opérateurs pour savoir utiliser le produit, le support du produit, et la retraite du produit (recyclage).

Le développement d'un système (produit désiré) est décomposé en une hiérarchie de sous-systèmes définis en tant que modules. Le développement d'un module de niveau inférieur est lancé dès que le module de niveau supérieur est complètement spécifié. A partir de là, le module est considéré comme un produit qui a des caractéristiques et des exigences identifiées et fait l'objet d'un développement de même type que le système désiré.

La décomposition se poursuit jusqu'à l'identification de trois catégories de produits finaux :

- Produits finaux sur étagère,
- Produits finaux pouvant être implémentés directement,
- Produits finaux pouvant être fournis par un sous-traitant.

1.2.2. Prise en compte de la sûreté de fonctionnement (SdF)

Dans plusieurs cas d'étude les attributs de la *SdF* sont pris en compte à une échelle composant/équipement, ou plus généralement pour chaque sous-système homogène (mécanique, électronique...) d'une manière ascendante. Beaucoup d'autres pistes sont ouvertes dans la démarche systématique de type descendante.

Dans l'approche système [Sadou et al., 2005], la prise en compte de la sûreté de fonctionnement doit être faite à toutes les étapes de conception. Si on considère, une exigence de fiabilité définie globalement sur le système, sa formalisation et son analyse devront permettre de s'assurer que les solutions techniques choisies au fur et à mesure de l'avancement de la conception et de la réalisation permettent de prendre en compte correctement cette exigence au niveau des sous systèmes et de leur intégration. Le processus de vérification et de validation doit apporter les techniques et les solutions pour garantir le degré de fiabilité spécifiée. Les différentes méthodes et outils de l'analyse de la sûreté de fonctionnement sont ainsi déterminants dans le choix des solutions techniques à envisager. La

conception d'un système en présence d'une exigence de fiabilité donnée, nécessite l'étude des défaillances possibles ; c'est une étude sur les scénarios redoutés qui permet d'orienter la recherche des solutions techniques (reconfiguration, redondance, réduction des taux de défaillances…), autrement dit c'est le respect de l'exigence en sûreté tout au long de la démarche Ingénierie Système.

1.2.2.1 Notions de base

Les notions de sûreté de fonctionnement des systèmes (ensemble de matériel et de logiciel) se basent sur la définition suivante « la Sûreté de Fonctionnement d'un Système (SdF) est la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans le service qu'il leur délivre » [Laprie J.C.,1995].

Nous pouvons tirer un ensemble de définition de ses attributs dans le sens large:

- Etre en mesure de réaliser instantanément une fonction, c'est la «Disponibilité».
- Capable d'assurer la fonction trop longue temps, c'est la «Fiabilité».
- Non-occurrence de conséquences catastrophiques, c'est la «Sécurité-innocuité».
- Non-occurrence de divulgations non-autorisées de l'information, c,,est la « Confidentialité ».
- Non-occurrence d'altérations inappropriées du système, c'est « l'intégrité ».
- Aptitude aux opérations et aux évolutions, c'est la «Maintenabilité».

Dans la terminologie de la sûreté de fonctionnement, nous distinguons entre faute, erreur et défaillance [Laprie J.C.,2004], généralement selon la succession suivante :

Faute → Erreur → défaillance

- Faute est une cause adjugée ou supposée d'une erreur.
- Erreur est la partie de l'état du système qui est susceptible d'entraîner une défaillance.
- Défaillance survient lorsque le service offert par le système ne correspond pas à la fonction qui lui en est demandée.

Dans la pratique, les fautes et leurs sources sont naturellement diverses. Nous pouvons les partager selon plusieurs points de vue :

- Cause phénoménologique (fautes physiques ou fautes dues à l'homme).
- Nature (fautes accidentelles, fautes intentionnelles avec ou sans volonté de nuire).
- Phase de création ou d'occurrence (fautes de développement, fautes opérationnelles).
- Situation par rapport à l'environnement du système (fautes internes, fautes externes).
- Persistance (fautes permanentes, fautes temporaires).

La distinction entre différentes classes de fautes permet de penser à des contres mesures appropriées. Nous pouvons citer quatre catégories d'évaluation pour la sûreté de fonctionnement :

- Prévention aux fautes : comment empêcher l'occurrence ou l'introduction de fautes, c'est le choix et la mise en œuvre d'un ensemble de processus visant à maîtriser la conception, la réalisation et la validation du système, et à assurer le bon fonctionnement du système durant sa vie opérationnelle.

- Tolérance aux fautes : comment fournir un service et remplir la fonction du système même en cas de faute. Elle vise à équiper le système de certains mécanismes de traitement d'erreurs (supprimer les erreurs avant leur propagation) ainsi que le traitement de fautes (éviter qu'une faute ne soit activée de nouveau).

- Elimination des fautes : comment réduire la présence (nombre, sévérité) des fautes.

- Prévision des fautes : comment estimer la présence, la création et les conséquences des fautes. La prévision des fautes est conduite en effectuant des évaluations du comportement du système par rapport à l'occurrence des fautes et à leur activation.

Les méthodes de tolérance aux fautes, sont implantées tant au niveau matériel, que logiciel, sont destinées à rendre le système robuste aux fautes provoquées par l'environnement (radiations, électromagnétisme, défauts électriques, perturbation mécanique, vibration…). Ces méthodes font partie intégrante de l'implémentation, leur fonctionnement est parfois testé par l'utilisation de techniques formelles. Les méthodes d'injection de fautes sont très utilisées pour la simulation des mécanismes de tolérance qui apportent un "durcissement" de l'application. Elles se traduisent en pratique par l'ajout d'instructions, de vérification, de signature aux différents blocs du code. Ces techniques permettent de détecter les flux de commande erronés. Lors d'une détection de faute, selon les méthodes, l'exécution du programme est alors arrêtée, ou dirigée vers une routine de correction qui lui permet de reprendre le flux d'exécution escompté.
L'objectif de notre travail étant la prévision des fautes, nous distinguerons deux types de prévision :

- Evaluations ordinales qui consistent à identifier, classer et ordonner les défaillances, et envisager les méthodes et techniques pour éviter ces défaillances (une étude préliminaire de défaillance).

- Evaluations probabilistes, sont destinées à évaluer en termes de probabilités le degré de satisfaction de certains attributs de la sûreté de fonctionnement. Nous classons notre méthode d'évaluation quantitative nommée approche flux informationnel [Hami et al.,2005] dans ce deuxième type d'évaluation.

Les systèmes multi-composants tolérants aux fautes et hautement fiables sont très présents dans les technologies modernes. Il est important d'être en capacité de déterminer des mesures telles que la fiabilité, le temps moyen d'atteinte d'un état de défaillance ou la disponibilité pour ces systèmes. Pour cela nous construisons un modèle stochastique permettant l'analyse (généralement par une chaîne de Markov à temps continu). Cependant, l'espace d'états s'avère trop grand de tel sorte qu'en pratique nous ne pouvons pas espérer de calculer directement les mesures [Chen et al.,2008].

1.3. Méthode d'évaluation des logiciels

1.3.1 Généralités sur le processus de conception

Les systèmes automatisés complexes requièrent des méthodes de modélisation et de quantification de métriques fiabilistes (Fiabilité, Maintenabilité, Disponibilité, Sécurité), ou de performance (notamment temps réel), pour leur conception, leur analyse et leur validation. Le développement de ces systèmes complexes (matériel/logiciel) robustes repose sur un cycle

établi au cours duquel les concepteurs ont à leur disposition de nombreux environnements de développement [Harel et al., 2002], [Sutherland et al., 2003], [Yalamanchili S., 2001], et de vérification [Dabny et al., 2008], [Kopf T., 1999], [Berar B et al., 2001]. Malgré l'existence de tels environnements, la conception matériel/logiciel est actuellement à la frontière d'une crise importante [Zhan J. et G. Xiong 2006], [ITRS 2004]. Cette crise est directement liée à la taille et la complexité des systèmes, ce qui a des conséquences sur la phase de vérification. Les techniques formelles se heurtent à la barrière des explosions combinatoires, tandis que les procédures de simulation nécessitent des temps de calcul trop grands [Wu Y-F.2007], [Kropf T. 1999]. Une partie de la communauté scientifique semble s'accorder sur le fait que la solution réside en partie dans l'élévation du niveau d'abstraction des modèles, basée sur des méthodes de modélisation et vérification agissant à des niveaux d'abstraction plus élevés [Monstand et al., 2006], [ITRS 2004].

Le cycle de développement est une approche descendante qui consiste en un amalgame des méthodes de conception matérielles et logicielles [Gorse et al., 2004].

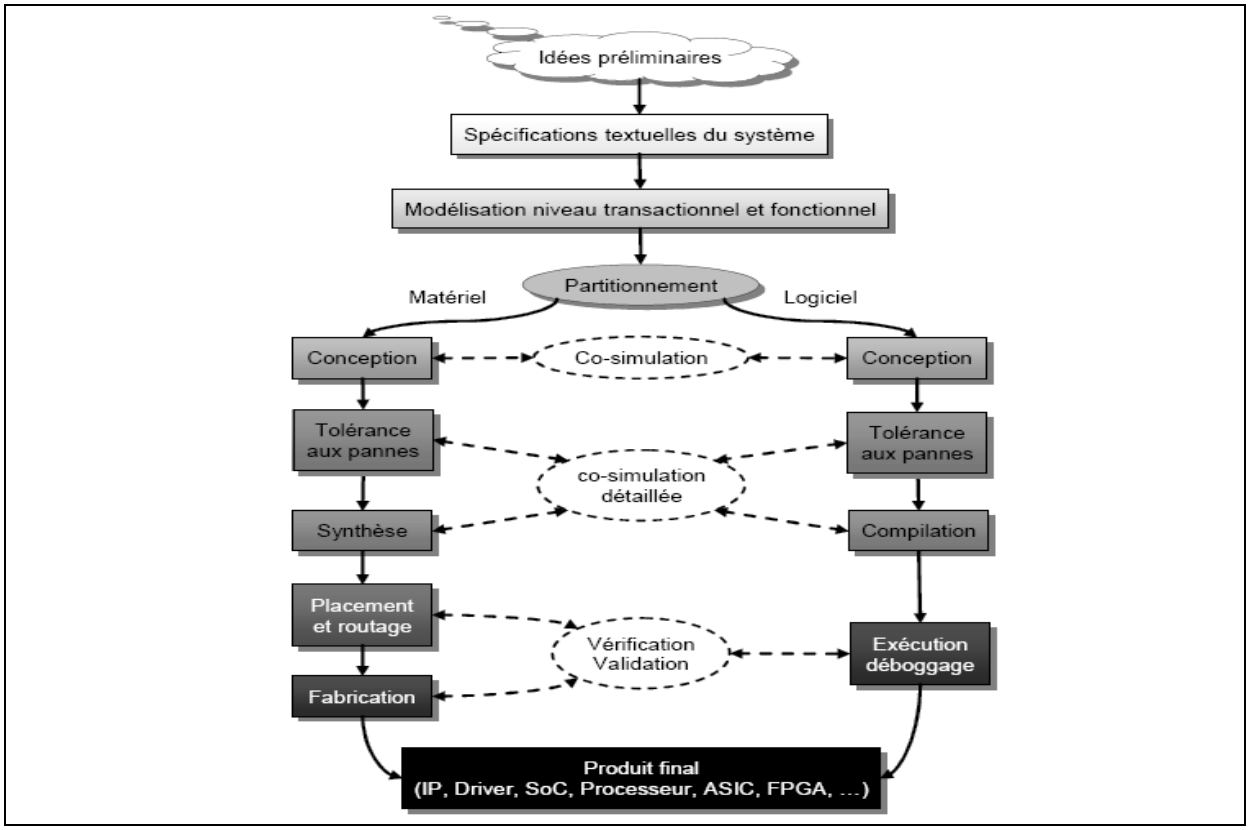


Figure 1.4 : Cycle de développement d'applications Matériel/logiciel.

Le cycle commence avec les idées préliminaires relatives au système à développer, à partir desquelles sont écrits les documents énonçant la description du système et les exigences requises par rapport à ses fonctionnalités. Les documents, aussi appelés "documents informels" consistent en la description, en langue naturelle, du système et de ses fonctionnalités. Ces descriptions sont la plupart du temps accompagnées de figures, tableaux [Belhadaoui et al., 2007-a], automates, algorithmes représentatifs, et parfois même, de détails particuliers concernant l'implémentation. Ce sont ces documents qui servent de point de départ de la phase de modélisation transactionnelle et fonctionnelle [Donlin A. 2004]. Durant cette phase est délivré un premier modèle haut niveau du système. Les divers composants du modèle sont ensuite affectés à une implémentation matérielle ou logicielle en fonction des

contraintes de modularité ou contraintes temporelle. Le reste du cycle consiste en deux branches parallèles au cours desquelles les implémentations matérielles et logicielles sont raffinées et vérifiées par simulation, ou par des techniques formelles [Givargis T. et Vahid F., 2002].

1.3.2. Modélisation et évaluation

La modélisation consiste en une représentation abstraite d'un système réel, dans le but de l'analyser mathématiquement. Le processus de modélisation doit être basé sur des hypothèses motivées par deux considérations contradictoires [Mercier et al., 2006] :

- Simplicité : il faut veiller à éliminer les détails „sans intérêt" (expression volontairement vague, car tout détail a son importance). La simplification permet d'utiliser des méthodes d'analyse beaucoup plus simples et plus rapides.

- Adéquation des résultats avec le système réel : les résultats obtenus avec le modèle doivent être très proches des valeurs réelles.

L'objectif est d'obtenir un compromis entre ces deux notions. Les modèles mathématiques utilisés sont des modèles à événements discrets, des processus stochastiques, et bien souvent des chaînes de Markov en raison de leurs „bonnes" propriétés (exploitables). Afin de diminuer la complexité des modèles manipulés, il est souhaitable de commencer par une description de plus haut niveau du système, à partir de laquelle un modèle mathématique peut être algorithmiquement construit. Le modèle mathématique obtenu, se pose la question de son évaluation. Les mesures qui nous intéressent sont typiquement la fiabilité du système, sa disponibilité, le temps de réponse, et la capacité au recouvrement dans le cadre de l'évaluation des performances. Dans le cas idéal, l'évaluation peut être réalisée analytiquement, c'est à dire résolue exactement [Belhadaoui et al., 2008-a]. Cependant ceci ne peut être obtenu que dans le cas de modèles relativement simples, avec des contraintes fortes (Markov...). De plus, il est important de noter que même quand ces hypothèses sont respectées, l'évaluation n'est pas toujours possible. La technique nécessitant le moins d'hypothèses est la simulation (technique d'évaluation utilisant des nombres aléatoires, souvent définie alors par simulation Monte Carlo).

1.3.3. Méthode de sûreté du logiciel

Nous visons dans ce travail à quantifier la fiabilité d'une architecture matérielle exécutant un logiciel d'une manière analytique tout en évitant toute sorte de simulation lente. Il est nécessaire de poser la question qu'est-ce que la fiabilité des logiciels ?

Comme repose à cette question est dans le sens large, La fiabilité du logiciel est définie comme étant : « La probabilité d'un logiciel à accomplir l'ensemble des fonctions spécifiées dans son document de référence, dans un environnement donné et pour un temps de fonctionnement donné ». C'est une définition de fiabiliste, qui peut être évaluée par des modèles mathématiques. Cette définition est plus restrictive que la définition de la fiabilité du logiciel donnée dans la norme ISO/IEC 9126 comme « l'aptitude du logiciel à maintenir un niveau de performance requis lorsqu'il est utilisé dans les conditions spécifiées ». Cette définition soulève dans le monde informatique principalement deux difficultés selon [Vallée F.et Vernos D., 2000].

Un logiciel ne peut fonctionner qu'avec un support matériel muni de ses interfaces ad-hoc. Cette constatation a déclenché le besoin de maîtriser la conception d'un système composé de logiciel et de matériel. En terme de fiabilité, nous examinons les différences qui existent entre un produit „matériel" et un produit „logiciel" donné par rapport à :

- La durée d'utilisation :
  - ✓ Pour le matériel, la vie utile décroît avec le temps si nous ne le „réparons„ pas ;
  - ✓ La fiabilité logicielle croît avec le temps si nous arrivons à corriger les défauts de conception ou si nous arrivons à corriger le matériel sur lequel s'exécute le logiciel.

- L'origine de la défaillance :
  - ✓ Pour le matériel comme pour un logiciel, les causes de défaillances prennent naissance pendant leur conception et leur production,
  - ✓ Pour le matériel comme pour un logiciel, l'obsolescence peut être la conséquence des effets des défaillances (non réparées), de ceux de la mode ou bien du progrès technologique.

Si le logiciel n'est pas soumis aux contraintes physico-chimiques, le phénomène de vieillissement ne se traduit pas par un changement de ses performances intrinsèques, mais par un changement de son environnement. Il s'agit en l'occurrence de la contrainte «durée de vie» du logiciel. Il apparaît que l'application de la «courbe en baignoire» du matériel au cas du logiciel est non réaliste, la courbe en baignoire ne s'applique pas strictement au logiciel. Cependant, le cycle de développement du logiciel peut être comparé avec le cycle de vie du matériel. Ainsi, pendant les phases du cycle de développement, le taux de défaillance logiciel peut être présenté par la courbe en baignoire (sans partie vieillissement, vue que le logiciel ne veiller pas).

Rappelons que cette forme est due à la période de jeunesse, suivie de la période où le taux de défaillance est constant, et la période de fin de vie. Si nous considérons les différentes modifications apportées au logiciel pendant l'exploitation, nous constatons des courbes en «baignoires» qui se succèdent. Il est nécessaire de prouver que ces dites baignoires sont bien l'effet de phénomènes intrinsèques à la phase de développement du logiciel.

- La première phase commence avec les tests et est considérée comme la phase de correction. Les erreurs de programmation ou les opérations non conformes aux spécifications sont identifiées et corrigées [MIL-HDBK-338B 1998] ;

- La deuxième phase représente la période de vie utile du logiciel dans laquelle le taux de défaillance est constant. La distribution utilisée lors de cette phase est la loi exponentielle [MIL-HDBK-338B 1998] ;

- La dernière phase commence juste après la fin de la vie utile. La plupart des erreurs observées pendant cette période résultent de l'incapacité du logiciel à satisfaire des nouveaux besoins sans modification des spécifications initiales. Nous pouvons considérer ce phénomène comme l'"usure" du logiciel. Les "défauts" observés pendant cette période peuvent servir comme base pour un nouveau logiciel. En plus, il y a le phénomène de vieillissement du logiciel, comme notamment les problèmes liés aux systèmes d'exploitation [MIL-HDBK-338B 1998].

Les risques induits par une défaillance peuvent être prédits, prévus, évalués. Un nombre de méthodes prédictives d'évaluation, ont été étudiées. Avant de détailler les principes de ces méthodes, il est nécessaire d'entreprendre une réflexion et de lister les éléments sur lesquels vont s'appuyer ces différentes méthodes. Au cours de la conception et la réalisation d'un produit logiciel, certaines tâches sont automatisées, d'autres font appel à l'intervention humaine. Les études de fiabilité d'un système doivent tenir compte des différents éléments qui sont en interaction avec lui. La fiabilité du système est la résultante de la fiabilité du matériel, du logiciel, des interfaces matériel-logiciel, de la fiabilité humaine, des interfaces homme-matériel-logiciel.

Des modèles de croissance de la fiabilité logicielle [Rook 1990], [Mihalache et al., 2005]ont été développés. Ces modèles restent peu utilisés dans les applications industrielles [Everett et al., 1998], ils ne permettent pas d'avoir une idée claire sur le niveau de risque d'utilisation d'un tel logiciel ce qui est le but principal de la quantification de la fiabilité.

La fiabilité d'un système, est liée à son aptitude à assurer une mission dans des conditions d'environnement données et pendant une durée donnée. En d'autres termes, la fiabilité caractérise la confiance que l'utilisateur peut placer dans le service rendu par ce système.

L'apparition d'une défaillance système est le fruit d'un processus de propagations, combinaisons et interactions de fautes, selon Yasuura H. Si le code programme (ensemble d'objet et de routine), apparaît parfaitement déterministe, le processus d'apparition d'une défaillance, généralement récursif vis-à-vis de l'architecture hiérarchique du système, peut être vu comme probabiliste en combinant aux fautes dites de premier type (de spécification par incomplétude ou mauvaise compréhension, de conception, de réalisation), des fautes de deuxième type (activation d'une erreur lors de sa révélation au cours de l'exécution du programme).

Il existe une autre constatation dans le domaine du logiciel qui consiste à dire qu'il suffit de corriger tous les erreurs du code pour éliminer toute possibilité de défaillance du logiciel associé et rendre le taux de défaillance nul. C'est une confusion entre la fiabilité et le comptage des erreurs. Les concepteurs font souvent appel à des métriques qui se basent notamment sur le comptage des erreurs dans le code, ils ne s'intéressent pas aux sources de ces erreurs. Cette confusion fait partie de la problématique de quantification de la fiabilité du logiciel.

La quantification de fiabilité du logiciel, nécessite de remonter à l'origine du besoin de l'utilisateur. En effet le logiciel n'est pas une vision abstraite mais il est matérialisé par un système dans lequel le code binaire s'exécute afin d'achever l'objet d'une mission de service.

Selon les besoins et spécifications du cahier des charges, la fiabilité s'exprime par exemple en termes de MTTF (Mean Time To Failure), de taux de défaillance, de la probabilité de pouvoir exécuter la mission avec succès. Le système est constitué du logiciel et d'une architecture matérielle, chacun pouvant être à l'origine d'une défaillance. Cependant, pour pouvoir évaluer la fiabilité globale de ce système, il est naturel de prendre en compte la part de chacun de ses composants. Il s'ensuit assez clairement qu'il est indispensable de trouver une approche globale de modélisation de tout le système.

La démarche SdF dans la phase de conception commence par une analyse préliminaire de risques du système, afin d'identifier les fonctions et les composants à risque [Bel 07-a]. Cette

analyse permettra de prendre en compte les nécessités d'isoler les parties à risques dans le choix de l'architecture.

La démarche repose essentiellement sur une analyse prévisionnelle des risques. Le but de l'analyse prévisionnelle des risques est d'identifier les parties critiques du système complexe et les actions pour réduire les risques associés.

L'analyse de risques peut être réalisée au moyen d'une analyse inductive ou d'une analyse déductive. L'analyse inductive correspond à une approche montante, où l'on identifie toutes les combinaisons d'événements élémentaires possibles qui entraînent la réalisation d'un événement unique indésirable. La démarche de l'analyse déductive est inversée, puisque nous partons de l'événement indésirable et nous recherchons par une approche descendante toutes les causes possibles. Ces analyses qui s'appuient sur la description du système sont dites techniques d'analyse statique [Darricau et al., 1999]. Des techniques d'analyse dynamique sont également utilisées en complément. Ces analyses se déroulent en parallèle et en liaison étroite avec les activités d'analyse/spécification et de conception, de manière continue et itérative. L'analyse statique nécessite de disposer en entrée :

- ✓ des événements redoutés pour le système ;
- ✓ d'une description du système (au niveau de spécifications, puis au niveau de l'architecture, afin de comprendre les interactions entre les différents sous-systèmes).

Ces analyses mettent en évidence les scénarios susceptibles de conduire à la défaillance. A partir de ces scénarios, des actions de réduction des risques sont ensuite identifiées, telles que :

- ✓ Spécification des modes de fonctionnement dégradé pour supprimer ou diminuer la gravité des conséquences du dysfonctionnement considéré. La spécification de ces modes de fonctionnement dégradé doit être cohérente avec les exigences de tolérance aux fautes ;
- ✓ Etudes complémentaires spécifiques visant à démontrer l'improbabilité du risque (modélisation, simulation) ;
- ✓ Identification d'essais de validation spécifiques visant à démontrer que le scénario ne va pas se produire ;
- ✓ Contraintes sur l'architecture du matériel et du logiciel ;
- ✓ Contraintes sur la testabilité et l'observabilité en opération ;
- ✓ Implantation de mécanismes de détection et de traitement de défauts ;
- ✓ Choix de conception minimisant les risques (choix d'architecture, de structures de données, ...) ;
- ✓ Identification de tests permettant de démontrer l'efficacité des mécanismes implémentés.

L'analyse dynamique du comportement d'un système ainsi que sa modélisation dynamique permettent de vérifier des propriétés supplémentaires de cohérence, de complétude, ... Elle permet aussi de tester les modes dégradés du système et l'efficacité des techniques de tolérance aux fautes. Les analyses dynamiques ou de performances permettent de mettre en évidence des problèmes de :

- ✓ Définition incorrecte ou incomplète des modes de fonctionnement ou des transitions entre modes et donc des spécifications ;

- ✓ Dimensionnement et de partage de ressources ;
- ✓ Synchronisation des traitements et des entrées/sorties ;
- ✓ Ordonnancement des tâches ;
- ✓ Protocole de communication.

Ce type de modélisation est surtout utile pour aider ou valider le choix entre plusieurs développements possibles.

Les techniques de modélisation pour la quantification de la fiabilité sont qualifiées de prévisionnelles ou d'expérimentales pour le matériel ou le logiciel :

- ➢ Techniques prévisionnelles : se situent en amont dans le cycle de vie. L'objectif de ces techniques est de valider la conception du système par rapport aux spécifications FMDS (fiabilité, maintenabilité, disponibilité et sécurité) de sûreté de fonctionnement. Dans le cas du logiciel, il n'existe actuellement pas de technique suffisamment mature pour évaluer la fiabilité prévisionnelle. Ce qui évoque un vaste domaine de recherche.

- ➢ Techniques expérimentales : consistent à évaluer le niveau de fiabilité atteint par le système à partir des essais réalisés sur ce système dans sa phase d'exécution. Ces techniques se situent en aval dans les phases du cycle de vie. Elles ont pour objectif principal de vérifier l'obtention du niveau de fiabilité requis en analysant les faits techniques observés.

Les méthodes et modèles d'évaluation de logiciels commencent à être reconnus et permettent de mener à bien des études de fiabilité expérimentale. Par extrapolation du comportement déjà observé du logiciel, les modèles de fiabilité du logiciel apportent des éléments quantifiés de la qualité de service tout en reposant sur des tests [Vallé .F et Vernos .D, 2002].

La norme ISO/CEI 61508 [CEI 61508] qui fait référence dans le domaine dans le domaine de la sécurité fonctionnelle, donne des taux de défaillance quantifiés aux niveaux d'intégrité des systèmes, avec prise en compte du logiciel. Cela invite les industriels et les concepteurs particulièrement à justifier ce niveau de défaillance accepté, à l'aide d'évaluation par modélisation ou à l'aide d'essais de qualification employant les techniques actuelles.

1.3.4. Méthode en sûreté des architectures matérielles

La notion de sûreté de fonctionnement dans le sens large selon [Villemeur A., 1997] est la science des défaillances. Elle inclut leur identification d'une manière exhaustive, leur évaluation probabiliste, leur prévision par des observations et proposition des méthodes, leur mesure statistique ainsi que leur maîtrise par réduction, prévision et tolérance.

La sûreté de fonctionnement dans le sens strict est l'aptitude d'une entité à assumer une ou plusieurs fonctions requises dans des conditions données. L'étude de la sûreté de fonctionnement dans les deux sens est l'un des meilleurs moyens pour assurer la qualité de service.

Une défaillance dans le domaine de la sûreté de fonctionnement est un événement qui indique la cession de l'aptitude d'une entité à accomplir une fonction requise. L'entité signifie

tout élément, composant, sous système, unité fonctionnelle, équipement ou système que l'on peut considérer individuellement. Elle peut être constituée par du matériel ou de logiciel.

Une classification dans la terminologie des architectures des systèmes peut être :

Pièce $\subset$ Composant $\subset$ Sous-système $\subset$ Système élémentaire $\subset$ Système

Les défaillances peuvent être classées selon plusieurs critères, en fonction de leur importance, de la rapidité de leur apparition, de l'instant de leur occurrence, de leurs causes et de leurs effets [CEI 50 191] [Villemeur A., 1997] [Laprie J.C.,1995], ces critères sont donnés par :

- Rapidité de manifestation :
  - défaillance progressive : dûe à l'évolution dans le temps (on peut la prévoir).
  - défaillance soudaine : non prévisible.
- Importance :
  - défaillance partielle : signifie la disparition de quelques fonctions du système.
  - défaillance complète : disparition totale de toutes les fonctions du système.
  - défaillance pertinente : elle a une conséquence directe sur les calculs.
  - défaillance non pertinente : n'entraîne pas d'interprétation du calculs.
- Rapidité et importance :
  - défaillance catalectique : soudaine et complète.
  - défaillance par dégradation : progressive et partielle
- Date d'apparition :
  - défaillance de la jeunesse : taux d'apparition baisse.
  - défaillance par vieillissement : taux d'apparition augmente.
- Par leurs effets :
  - défaillance mineurs : dommage négligeable au système, pas de risque humain.
  - défaillance significative : dommages significatives au système, risque humain.

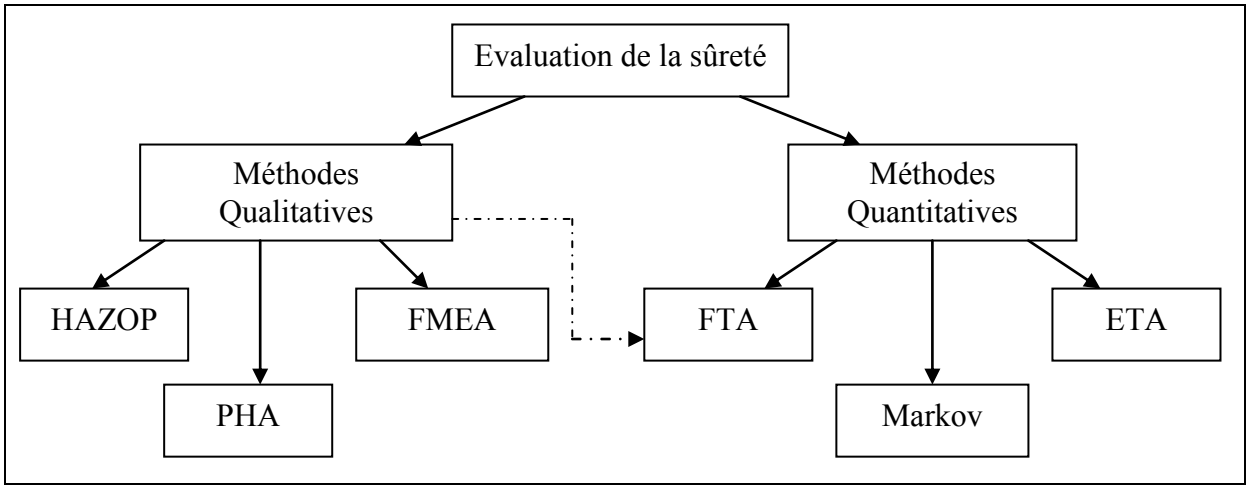


Figure 1.5 : Exemples de méthodes qualitatives et quantitatives de la sûreté de fonctionnement.

Les méthodes d'analyse de la sûreté de fonctionnement des architectures matérielles sont nombreuses. Parmi les méthodes souvent utilisées dans ce domaine, nous distinguons des méthodes qualitatives et quantitatives pour l'évaluation de la sûreté de fonctionnement (en premier lieu nous nous intéressons à l'étude qualitative). Nous pouvons les classer selon la figure 1.5.

Les méthodes qualitatives souvent utilisées pour les systèmes simples sont le PHA, HAZOP. Elles permettent un examen relativement rapide à des situations dangereuses sur l'architecture du système, leur simplicité se termine par la rencontre des systèmes complexes. Nous trouvons les méthodes AAD (Analyse par Arbre de Défaillances) et AMDEC (Analyse des modes de défaillance de leurs criticité) qui consistent à déterminer l'enchaînement des événements pouvant conduire ou non à un accident à partir d'un événement initiateur, dans ces méthodes il faut définir avec discernement l'événement initiateur ce qui rend lourde leur mise en œuvre surtout pour des systèmes complexes.

La méthode qualitative traitée, dans ce travail, est la méthode AMDEC. Cette méthode est précédée par deux étapes fondamentales. Après la définition des spécifications standards du cahier des charges et les spécifications des paramètres FMDS (fiabilité, Maintenabilité, Disponibilité et Sécurité). Nous réalisons une étude d'analyse fonctionnelle qui consiste à définir avec précision les limites matérielles du système, les différentes fonctions et opérations réalisées par ce système et les diverses configurations d'exploitation. Cette étude se situe en amont, elle présente l'étude d'analyse préliminaire de risque qui a pour but à court terme d'établir une liste exhaustive des incidents ou accidents pouvant avoir des conséquences sur la sécurité de personnel et du matériel [Villemeur A., 1997] et à long terme d'identifier les dangers de chaque composant du système, les modes de défaillance les plus rencontrés, leurs causes possibles, et d'évaluer leurs effets sur le reste de l'architecture du système par calcul de leurs gravités.

1.3.5. Sûreté de fonctionnement des systèmes de commande programmable

Le contexte de notre travail est la conception d'une architecture sûre et structurée de haut niveau (au sens de fiabilité et de robustesse) d'un capteur. Ce capteur est spécifique aux systèmes mécatronique en interaction avec un environnement physique ayant sa dynamique propre. Ces systèmes se caractérisent par leur hybridité (continu/discret), leur complexité (diversité des aspects à prendre en compte), et la criticité de leur sûreté. Ils se retrouvent dans des contextes applicatifs divers, et leur modélisation doit prendre en compte ce caractère mixte. Dans le domaine du contrôle de procédés physiques le problème est de trouver le compromis entre la régulation et les lois de commande.

Les systèmes mécatroniques sont en extension et se sont développés avec une cadence très élevée dans ces dernières décennies. L'intelligence de ces éléments est liée à l'existence d'un élément de calcul interne (processeur), qui requiert l'usage de techniques particulières pour assurer leur programmation, d'une part du fait de leur complexité, et d'autre part à la criticité de leur sûreté. En particulier, un modèle global est nécessaire pour fournir un support à des outils d'analyse qui offrent une assistance à la validation.

Les motivations se présentent à la fois dans la conception et le développement des systèmes multi-formalismes et dans la prise en compte des exigences spécifiques en sûreté de fonctionnement de ces systèmes complexes.

- Systèmes complexes : Un système dont la conception nécessite des techniques et des compétences diverses pour leur évaluation, leur mise en œuvre, et la définition de leurs missions, en termes de modes de fonctionnement, et de confrontation et commutation entre ces modes. Ceci suggère une structuration, distinguant différents niveaux d'intervention.

- Systèmes à sécurité critique : Un système dont la conséquence de son dysfonctionnement met souvent en danger les personnes (transportées dans le cas d'un avion ou d'un train par exemple), ou les biens (coûteux, intervention difficile). Ceci induit un besoin d'assistance à la conception ou à l'opération de ces systèmes par des méthodes de modélisation, d'analyse, de validation et de mise en œuvre correcte avec le support d'outils efficaces.

Devant l'incapacité des méthodes classiques d'évaluation de la sûreté de fonctionnement vis-à-vis des systèmes complexes, l'approche flux informationnel [Hami et al.,2005] donne une ouverture et une piste pour répondre à ce besoin, elle fournit un ensemble de scénarios de dysfonctionnement, fondé sur un modèle dynamique des automates d'états finis, qui sert à l'analyse de cohérence des spécifications et à leur vérification. L'intégration de tels outils, peut être une méthodologie de conception.

Dans la suite de ce travail, nous allons montrer l'extension de l'approche flux informationnel pour construire un modèle global qui prend en compte la mixité continu/discret du système de cas d'étude (capteur intelligant). Il permet de structurer la spécification de façon à découpler ces aspects (continu/discret). Il repose sur un modèle dynamique qui offre des outils d'assistance à la spécification, à l'analyse et à la mise en œuvre du prototype final. Nous avons considéré l'approche flux informationnel comme étant une piste par laquelle nous posons la problématique différemment à ce qui existe dans l'état de l'art et nous apportons une contribution vers une solution.

1.3.5.1. Validation de la conception d'une architecture sûre de fonctionnement

Les spécifications du système ont pour but d'établir une première description du futur système. Pour la réalisation des spécifications, le concepteur système doit disposer comme données d'entrées des résultats de l'analyse des besoins et des considérations techniques et de faisabilité. En phase de spécification, il est nécessaire de recenser toutes les exigences liées à la fiabilité du système :

- **Exigences qualitatives :**

✓ Listes d'événements redoutés, classification des défaillances ;
✓ Définition des modes dégradés et des conditions de passage entre modes, comportement dans chacun des modes ;
✓ Types de fautes à considérer et stratégie de tolérances aux fautes ;
✓ Méthodes à employer et normes à respecter ;

- **Exigences quantitatives :**

✓ Probabilité de bon fonctionnement ;

- ✓ Durée de fonctionnement ;
- ✓ MTTF (ou MTBF);
- ✓ Taux de défaillance ;
- ✓ Taux de réparation.

L'étude d'un système ne se limite pas à la phase de spécification et réalisation, mais nécessite une phase en aval de vérification et validation, c'est une activité que nous pouvons mettre en place même dès le début du projet, afin de détecter au plus tôt les risques de non fiabilité et s'assurer que les dispositions prises et les analyses effectuées répondent aux exigences définies.

Un manque de fiabilité constaté en phase finale du développement peut être irrémédiable pour le système. Pendant les activités de vérification et de validation d'un système, effectuées dans les étapes montantes du cycle en V, la fiabilité a un impact sur :

- ✓ Confirmation des choix effectués lors des activités de construction dans les étapes analyse/spécification et conception ;
- ✓ Vérification de l'efficacité des dispositions prises, par des actions spécifiques, notamment lors des étapes de validation du système complexe ;
- ✓ Finalisation des principes liés à l'exploitation opérationnelle du système en s'assurant de l'efficacité des moyens et procédures mis en place pour le diagnostic et/ou les reconfigurations.

La vérification exhaustive du comportement du système est souvent impossible, car elle se heurte aux limites des outils existants. La vérification consiste en des essais et des tests au niveau module (unitaire) et au niveau système (intégration).

- ➢ La vérification au niveau module (unitaire) entraîne :

- ✓ Exécution de l'ensemble des tests unitaires définis et la vérification de la conformité des résultats obtenus aux objectifs ;
- ✓ Exécution des tests des mécanismes relatifs à la fiabilité ;
- ✓ Evaluation de la robustesse des modules système.

- ➢ La vérification au niveau système (intégration) entraîne :

- ✓ Exécution de l'ensemble des tests d'intégration définis et la vérification de la conformité des résultats obtenus aux résultats attendus ;
- ✓ Exécution des tests des mécanismes inter-modules relatifs à la fiabilité ;
- ✓ Evaluation de la robustesse des interfaces entre modules système.

La validation du système est prononcée à la suite de :

- ✓ Exécution de l'ensemble de tests de validation définis et la vérification de la conformité des résultats obtenus aux résultats attendus ;
- ✓ Evaluation du taux de couverture fonctionnelle ;
- ✓ Exécution des tests de robustesse du système (incluant les cas de fonctionnement dégradé du système) ;
- ✓ Conformité aux exigences de fiabilité du système ;

- ✓ Evaluation de la robustesse du système et du niveau de fiabilité atteint.

L'évaluation de la fiabilité d'un système consiste en la détermination des défaillances affectant ses composants matériels en phase opérationnelle. Cette évaluation repose sur une analyse basée sur les structures du système, de l'influence des probabilités de défaillance de ses composants sur la probabilité de défaillance globale.

La démarche consiste alors à observer le comportement du système considéré et à effectuer les calculs statistiques sur les données relatives aux défaillances observées. La collecte des données de défaillance est une démarche qui doit être intégrée dès le début du projet. Pour que ces relevés de défaillance puissent être utiles, il est important que l'utilisation du système soit la plus représentative possible des conditions de sollicitation réelle du système dans son environnement opérationnel. L'objectif principal de cette observation est d'évaluer le niveau de fiabilité du système dans les conditions d'utilisation prévues.

Pour obtenir une bonne étude statistique, il est nécessaire de recueillir un nombre suffisant de données, afin d'en déduire la ou les lois de modélisation les plus proches de ce que l'on a pu constater pendant la période de temps considérée. Une vue globale sur les méthodes utilisées sur un cycle de développement montre que dans les premières phases du cycle les méthodes AMDE/AMDEC (Analyse de Mode de Défaillance de leurs Effets et de leurs Criticité), AdD (Analyse de Défaillance) et APR (Analyse Préliminaire de Risque) sont préférées pour déterminer les éléments redoutés et que dans les dernières phases du cycle les méthodes RdP et MEE sont utilisées pour maîtriser le comportement fonctionnel/dysfonctionnel (RdP et MEE) ou l'aspect dynamique du système (RdP).

1.3.5.1.1. Vérification formelle

Les méthodes formelles sont très prisées, en ingénierie système, pour le développement d'applications critiques quant à la sûreté. Leur utilisation pour la spécification et la vérification des propriétés des systèmes est de plus en plus recommandée par les normes et directives internationales. Il s'agit de techniques mathématiques utilisées pour la modélisation, l'analyse et la conception. Leur intégration dans des environnements de développement devrait aider à leur diffusion. Plusieurs approches coexistent : de la simple description des spécifications, à la génération de code automatisée et la vérification, en association avec des techniques de validation plus conventionnelles. Leur acceptation industrielle est non seulement liée à la maturité des ateliers logiciels disponibles et de leur facilité d'appréhension, et donc de leur coût de possession (acquisition, formation, gain sur les temps de développement et de mise au point). En dépit de leurs qualités réelles [Liu S.et Wang H., 2007]. Ces techniques et méthodes formelles sont plus souvent utilisées qu'en phase de validation et non sur l'ensemble du processus de développement.

Les méthodes et outils de vérification formels peuvent être classés en deux catégories : les approches par modèle et les approches par preuve.

- ➢ Selon l'approche par modèle, le comportement du système est traduit par une représentation d'état finie. La vérification est effectuée par des algorithmes de „model checking", ou des relations d'équivalence. Les conditions de sécurité fonctionnelle peuvent, par exemple, être exprimées sous forme d'équations de logique temporelle et être vérifiées à partir du modèle. Les approches par modèles permettent certes une vérification automatique directe des propriétés des systèmes. Malheureusement, elles souffrent de

l'explosion combinatoire de leur espace d'état. Les systèmes composés de plusieurs sous-systèmes associés à un modèle d'état fini voient leur nombre d'états augmenter exponentiellement. Ces techniques de vérification [Clarke D-W., 2000] sont donc longtemps restées moins utilisées par l'industrie. L'utilisation récente de techniques de manipulation symbolique des modèles [Burch E.et H-J.Kung, 1999], où les relations de transition permettent le traitement de systèmes de tailles importantes. Ces techniques sont représentées implicitement au moyen d'équations booléennes ou décrites au moyen de Diagrammes Binaires de Décision (BDDs) [Bryant et al., 2008].

- Selon l'approche par preuve, le système est modélisé par l'ensemble des structures sur lesquelles sont définis des invariants, et les opérations par des indications de pré et post conditions. Les propriétés à respecter sont décrites par les invariants, et on doit démontrer qu'au cours de son fonctionnement le système respecte les théorèmes édictés. Ces techniques sont généralisables et ne sont pas affectées par le problème de l'explosion d'état. Elles exigent par contre une grande compétence dans leur utilisation, car le processus ne peut être entièrement automatisé.

La tendance actuelle semble être au rapprochement de ces deux approches.

Analyse de l'ordonnancement des tâches

La défaillance d'une application temps réel peut être provoquée, non seulement par une faute fonctionnelle ou matérielle, mais également par une incapacité à tenir les délais imposés. C'est en général le résultat d'une charge de traitement trop importante au regard des capacités de la ressource. La validation des contraintes temporelles (Scheduling) consiste généralement en la recherche d'une charge de traitement des tâches, pour laquelle il peut être prouvé que chaque tâche sera complètement exécutée avant une date limite. Il existe des approches statiques et dynamiques, pour des garanties déterministes ou probabilistes. Les hypothèses faites lors de la modélisation se révèlent souvent peu réalistes : nombre de tâches, temps d'exécution maximum, contraintes de priorité, conflits de ressources… Les études sur les systèmes temps réel tolérants aux fautes [Gergeleit et al ., 2005], [Wang Y., 2004] se sont développées au cours de la dernière décennie. Le besoin d'un environnement de développement intégré global, pour un processus de conception cohérent, permet d'évaluer l'impact des différents algorithmes et mécanismes.

Validation par injection de fautes

Les approches analytiques, basées sur des modèles, montrent parfois leurs limites. Il faut étudier des comportements expérimentaux spécifiques, en présence de mécanismes de détection et de tolérance aux fautes. L'injection de fautes permet d'évaluer ces mécanismes. Indépendamment du niveau d'abstraction, les campagnes d'injection de fautes contribuent à l'évaluation des taux de défaillance, par l'étude de la propagation d'erreurs [Chillarege et al., 1994], [Steininger A., 2000], ou d'erreurs latentes [Chillarege et al., 1994], [Geist et al., 1984], ainsi que des performances des mécanismes de recouvrement [Patton et al., 2006]. De nombreuses techniques et outils ont été développés [Arlat et al., 2004], [Kanawati et al., 1995]. Il convient de considérer la dimension dynamique des processus de traitement et de mener l'évaluation des mécanismes de recouvrement en fonction du temps [Kim K. et T. Lawrence 1992]. La raison d'un recouvrement imparfait est soit une mauvaise connaissance des mécanismes initiateurs de faute soit une mauvaise stratégie de recouvrement.

La validation expérimentale des systèmes tolérants aux fautes requiert une réflexion sur le niveau d'abstraction du modèle utilisé et le type d'injection de fautes appliqué. Le système peut être représenté par un modèle de simulation décrivant sa structure et/ou son comportement. Le type d'injection appliqué peut être physique (les fautes correspondent à des altérations mécaniques ou électromagnétiques sur les composants physiques), ou logique (altération de variables booléennes ou de données). La première méthode (injection de fautes au niveau des broches des circuits intégrés) est confrontée au problème de l'extrême niveau d'intégration des composants actuels, ainsi que de leur fréquence de travail. Un compromis a été développé : l'injection hybride, les fautes sont injectées au niveau logique des composants physiques. Le même principe est appliqué à des niveaux plus abstraits [Bernard V. 2004] pour la validation formelle des mécanismes de tolérance aux fautes.

1.3.5.1.2. Analyse quantitative de la FMDS d'une architecture sûre

La modélisation et l'analyse des performances fiabilistes (Fiabilité – Maintenabilité – Disponibilité – Sécurité) des systèmes sont réalisées par de multiples méthodes et outils. Deux grandes familles coexistent : les approches combinatoire et l'espace d'état, cette dernière incluant les modèles markoviens [Malhotra M. et K. S. Trivedi, 1994].

- Modèles combinatoires : ils incluent des schémas fonctionnels de fiabilité (RBD), des arbres de fautes et les graphes de fiabilité... Des développements associent arbre de fautes et événements répétés (FTRE) : cette méthode combine à la simplicité des arbres de fautes, la puissance de l'approche d'état [Karacal S, 1998].

- Chaînes de Markov : ce sont des processus markoviens à état discret et temps continu. Ce sont des modèles largement admis par la communauté fiabiliste en raison de leur puissance et de leur facilité d'utilisation. Les taux de défaillance sont considérés constants (non vieillissement des composants), sur chacun des pas de calcul. Cette approche largement acceptée pour des composants électroniques (matériel), a également été appliquée aux défaillances de logiciel. Le taux de défaillance pour le logiciel peut être calculé comme le produit d'un taux d'exécution du logiciel constant (λi) et de sa probabilité de défaillance (Pj). Une telle approche a été adoptée pour modéliser des architectures logicielles tolérantes aux fautes [Arlat et al., 2004].

- Modèles de haut niveau (d'abstraction) : c'est une déclinaison de l'approche markovienne pour la modélisation et l'évaluation fiabiliste des systèmes complexes. L'explosion combinatoire de l'espace d'état est inévitable par une approche markovienne classique. Elle engendre d'importantes difficultés de traitement. Les approches de types : Réseaux de file d'attente, Algèbres des Processus Stochastiques, Réseaux de Petri Stochastiques… permettent d'obtenir des modèles très compacts et peuvent être associées à des algorithmes de réduction de l'espace d'état.

Les modèles à base de Réseaux de Petri sont très populaires pour leur approche graphique, ils permettent d'appréhender facilement les représentations de la simultanéité, de la concurrence et de la synchronisation. De nombreuses classes de Réseaux de Petri existent : des RdP autonomes, des RdP temporisés et des RdP Stochastiques [Molloy M. K. 1981]. Ils sont limités par la distribution exponentielle de l'ensemble de leurs variables aléatoires. Pour tenir compte de la présence d'activités stochastiques et instantanées, les Réseaux de Petri Stochastiques Généralisés [Ajmone et al., 1991] ont été introduits. Ces modèles sous-tendent

un processus markovien homogène à temps continu et espace d"état discret. Le formalisme RdPSG est intégré à un grand nombre d"outils destinés à l'évaluation fiabiliste comme UltraSAN, SPNP et TimeNET...Les réseaux de Petri permettent d"avoir d"autres extensions telles les Réseaux d'Activités Stochastiques [Sanders et al., 1995] et les Réseaux Stochastiques de Récompense [Azgomi M.A. et Movaghar A., 2005] qui ont abouti aux outils UltraSAN et SPNP. Ces formalismes engendrent encore des processus markoviens. Il existe des classes de Réseaux de Petri qui ne reposent pas sur le formalisme tel que le RdPSG : Réseaux de Petri Stochastiques Généralisés [Ajmone et al., 1995], l"outil dédié est TimeNET [German et al., 1995] et RdPSMR Réseaux de Petri Stochastiques Markoviens Régénératifs [German et al., 1995].

L'analyse quantitative des paramètres de la sûreté à partir d"un modèle stochastique est un processus qui exige la compétence et l"expérience. La sensibilité aux divers paramètres ne pouvant être connue a priori, les exigences en simplifications et abstractions sont susceptibles d"engendrer des erreurs importantes sur les résultats finaux. Lors de la modélisation de systèmes complexes [Kanoun et al., 1996], [Nelli et al., 2008], plusieurs problèmes apparaissent : interactions entre matériel et logiciel, rigidité de modèles (taux de transitions non équilibrés), explosion de l"espace d'état. En dépit de la modularité du modèle, le maintien des propriétés markoviennes exige une résolution globale. Une hiérarchisation des modèles (où les modèles secondaires peuvent être résolus séparément et les résultats obtenus utilisés par un modèle de plus haut niveau) n"est pas simple d"obtention. Par conséquent ces modèles tendent à être très abstraits, voire simplistes, faisant l"impasse sur des détails parfois importants. L"obligation de choisir soigneusement les caractéristiques à représenter est probablement la cause de la difficulté à l"automatisation du processus de modélisation.

1.3.5.1.3. Intégration des approches

L"évaluation fiabiliste des composants d"automatisme intelligent, consiste, dans l"objectif à élaborer un modèle du système en fonction des besoins et des contraintes fonctionnelles du cahier des charges. Le client exprime ces contraintes entant qu"utilisateur direct du système, l"environnement de l"application de ce système joue aussi un rôle dans l"évaluation fiabiliste.

Toutes ces techniques exigent la définition d"un modèle adapté particulier. L"ingénierie système ne peut se satisfaire de cette situation : juxtaposition de techniques indépendantes. La validation d"Architectures de systèmes temps réel, sûrs de fonctionnement, ne doit pas engendrer des coûts de validation et de certification déraisonnables. L"industrie souhaiterait disposer de modèles génériques déclinables rapidement selon les exigences clients. Cela conduit à la réutilisation de composants déjà validés, la coexistence de composants logiciels de différents niveaux de criticité, la limitation à la validation d"un nombre restreint de sous-systèmes. L'environnement de validation préconisé (projet européen GUARDS) illustre les rapports entre les composants et leurs interactions avec l'architecture de l"environnement de développement.

Le projet européen GUARDS [Bonda et al., 2000] a abouti à la création d"une „trousse à outils" (modèles de fiabilité, modèles formels, diagrammes HRT, code) (Figure 1.6), plutôt qu'à un environnement réellement intégré. Le projet HIDE portait, quant à lui, sur le développement d'un environnement intégré pour la conception de systèmes sûrs par l'intermédiaire d"un langage de modélisation unifié (UML). Ce langage [Fowler et al., 2007], [Rumbaugh et al., 1991] permet de décrire les aspects statique et dynamique (Statechart) du

système. Il est possible à partir de ce langage semi-formel de traduire automatiquement ou semi-automatiquement les spécifications du système en un modèle mathématique pour valider sa fiabilité. Il s'agit de traduire des diagrammes structurels UML en Réseaux de Petri stochastiques généralisés synchronisés pour l'évaluation de la sûreté [Bonda et al., 2002].

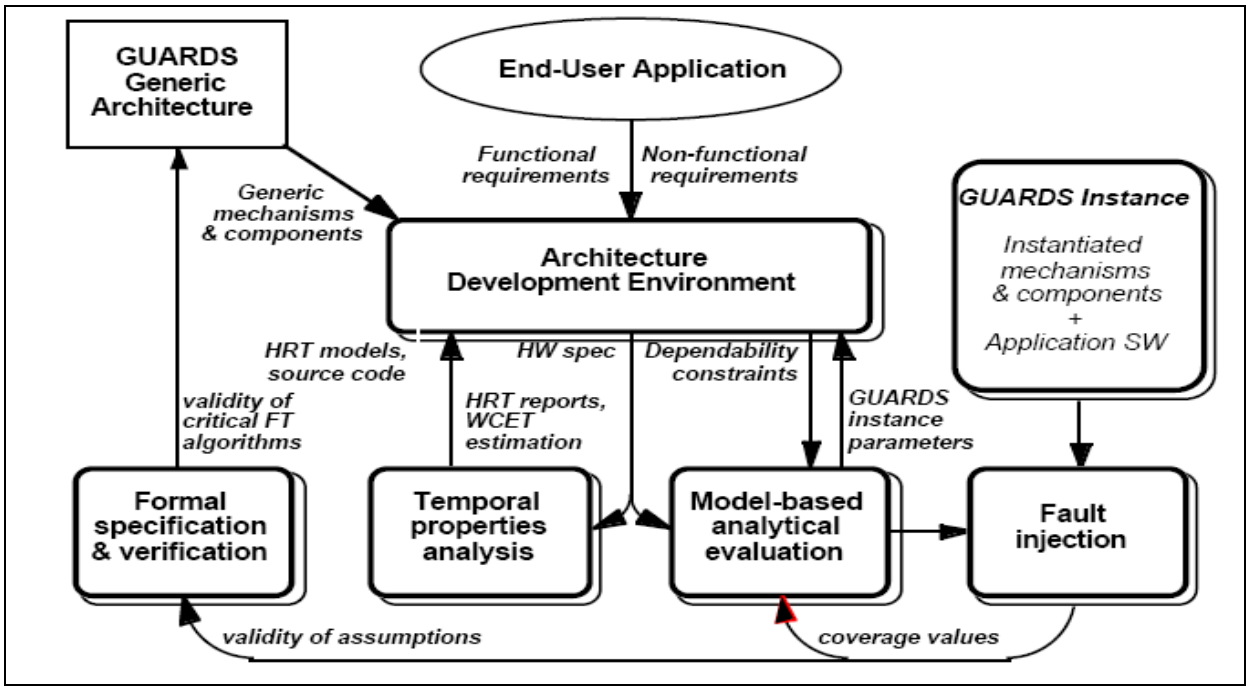


Figure 1.6 : Interactions entre conception et validation des architectures

1.4. Conclusion

La complexité importante des systèmes mécatroniques et la réduction des coûts de conception et d'exploitation incitent les industriels à maîtriser davantage la sûreté de fonctionnement.

Afin d'optimiser le développement des systèmes complexes, des méthodes sont utilisées dans chaque étape du cycle de développement pour analyser la fiabilité. L'ensemble de ces méthodes constitue un processus à part entière : le processus de fiabilité.

Les notions citées dans ce chapitre montrent que la sûreté de fonctionnement est l'un des piliers importants permettent de réduire les coûts des systèmes en cours de conception et contribue de façon remarquable à des gains de productivité. La conception sûre est une démarche rationnelle et structurée et exige de la part des concepteurs une vision globale et systémique des méthodologies existantes pour inclure tous les facteurs contribuant à la sûreté de fonctionnement. L'étude de sûreté de fonctionnement rend obligatoire une approche transverse qui fédère toutes les compétences technologiques.

Ce chapitre montre l'intérêt d'une étude fiabiliste quantitative pour les systèmes complexes programmables. Il clarifie le rôle de la sûreté de fonctionnement comme une obligation dans le domaine de l'ingénierie système et notamment dans la phase de conception.

La complexité croissante de ces systèmes est étroitement liée à des propriétés d'intelligence, tel que la détection et la tolérance aux fautes, voire des procédures de réparation et de

recouvrement. Ce qui implique une vérification de l'exactitude des modèles et méthodes d'évaluations quantitatives appliquées pour ces systèmes.

Cette introduction sur les systèmes complexes dans le présent chapitre, sera utile dans le chapitre suivant pour détailler les problèmes liés à notre système (capteur intelligent). Certains problèmes tels que l'interaction matériel-logiciel, signaux mixtes dans un système hybride, restaient encore mal traités ou mal résolus par les méthodes conventionnelles, malgré les efforts fournis par la technologie actuelle.

Nous détaillons dans le chapitre suivant l'architecture du système en question et les tâches globalement destinées pour chaque partenaire du projet CETIM. L'approche flux informationnel sera largement détaillée pour comprendre son intérêt d'application et voir comment la généraliser pour ce genre de système complexe.

Chapitre 2 :

Problématique liée à la sûreté de fonctionnement d'un sous-système mécatronique

2.1 Introduction

Ce travail propose une méthodologie de conception d'un capteur intelligent dédié à des applications mécatroniques, capable d'assurer une reconfiguration matériel et/ou logiciel en cas de besoin (défaillance). Une des originalités de ce travail réside dans l'approche transdisciplinaire de la conception de ce type de composant. La présence de ressources programmables, capables de traiter des fonctions différentes, multiplie les risques de causes communes de défaillance et d'interactions entre fonctions logicielles et entités matérielles. Garder la qualité du service pour ce type de composant nécessite une étude en parallèle des performances fiabilistes (évaluation de la fiabilité fonctionnelle) et performances temporelles (capacité temporelle au recouvrement de fautes). L'aspect hétérogène de leur architecture (mécanique, électronique analogique et numérique), impose la coexistence de signaux continus et événementiels.

Un système mécatronique est un système combinant des technologies qui relèvent des domaines de la mécanique, de l'hydraulique, de la thermique, de l'électronique, et des technologies de l'infomation [Jang et al., 2007]. D'une manière générale, les systèmes mécatronique est de compléter des parties des systèmes mécaniques ou hydrauliques par des commandes électroniques programmables. Il peut être décomposé de trois entités en interaction figure 2.1. Les capteurs intelligents mesurent des grandeurs physiques continues caractéristiques de la partie opérative. Le système de commande et de reconfiguration établit en fonction de ces mesures les actions à réaliser de façon à garantir le service avec un niveau de performance donné. Les actionneurs agissent sur la partie opérative.

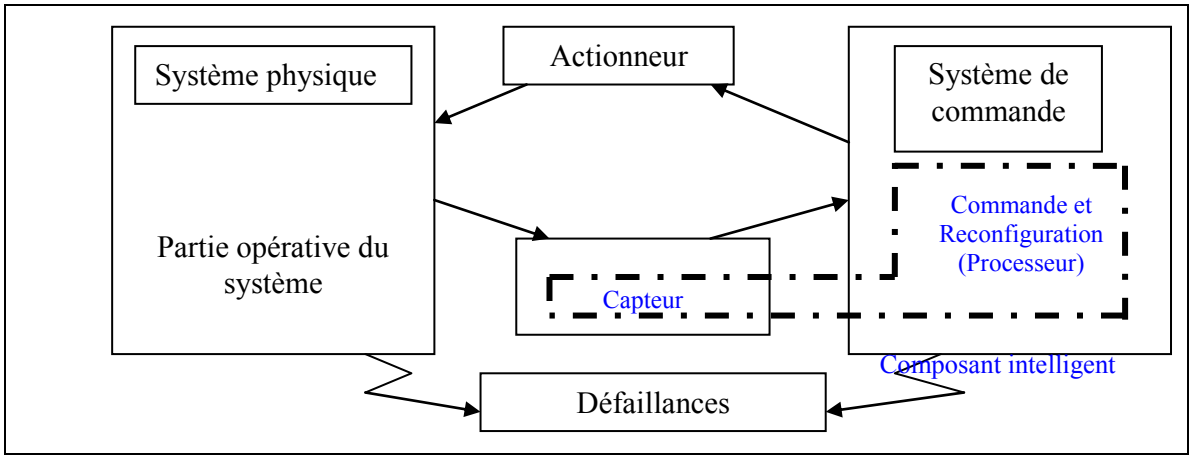


Figure 2.1 : Architecture générale d'un système mécatronique.

La disponibilité de ce type de système est traduite par l'absence de l'interruption de la partie intelligente (processeur). Les systèmes mécatroniques sont des systèmes à configurations dynamiques. La prise en compte des interactions entre le processus continu (partie opérative) et le processus discret (système de reconfiguration) (figure 2.1) nécessite une modélisation hybride. Nous parlerons dans ce cas de la fiabilité dynamique.

Les reconfigurations matérielles sont souvent basées sur le choix d'une nouvelle architecture physique pour réaliser la fonction considérée, tandis que et les reconfigurations logicielles sont assurées par des redondances analytiques pour la reconstruction de données plus accessibles. La vitesse d'exécution dans le cas de reconfiguration matérielle dépond du changement du matériel (partiel ou total). En revanche, la reconfiguration logicielle est plus

au moins rapide, puisqu'elle ne demande pas de changement au niveau de l'architecture matérielle. L'efficacité du recouvrement de défaillance(s) dépend de l'efficacité de la méthode du diagnostic.

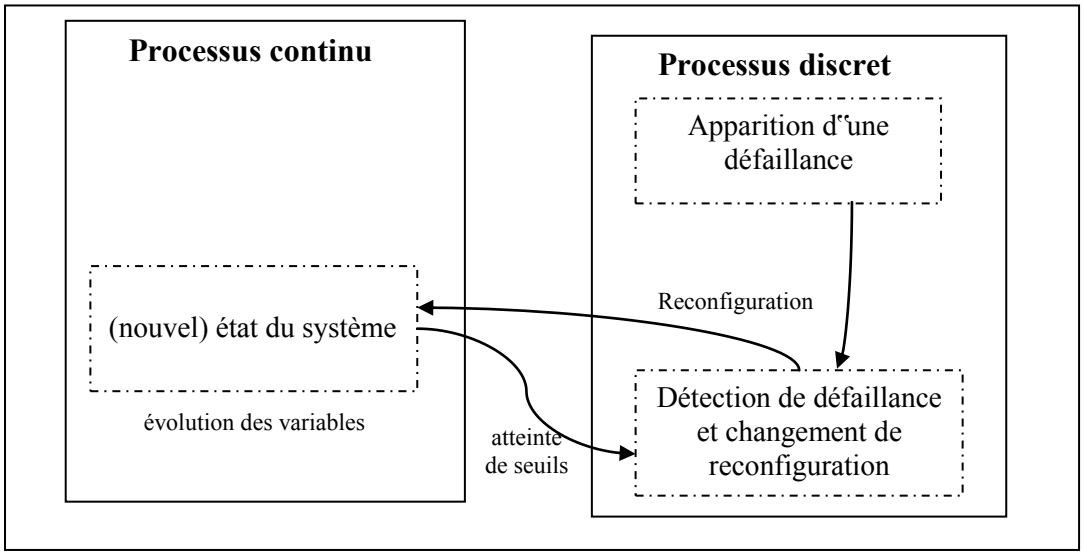


Figure2.2 : Processus de reconfiguration dynamique des systèmes mécatroniques.

L'évolution des technologies et des fonctionnalités des capteurs a suivi l'évolution des contraintes des industriels relatives aux coûts de conception, d'installation et de mise au point des équipements dans un contexte fortement concurrentiel. Du transducteur associé à un amplificateur, au capteur intelligent « plug and play », l'intégration des technologies a permis de doter, même les plus miniaturisés d'entre eux, d'un microprocesseur en une vingtaine d'année. Cependant, malgré leur haut niveau d'intégration, ces capteurs reprennent pour l'essentiel la chaîne de traitement classique : partie sensible, interface de traitement analogique et partie numérique programmable. Dans la partie programmable, nous citons la présence des fonctionnalités telles le diagnostic, la communication…

L'apparition de capteurs intelligents dans la chaîne de conditionnement du signal numérique permet une gestion plus aisée des redondances (figure 2.3 et figure 2.4). Les capteurs intelligents sont capables de s'auto diagnostiquer, de se reconfigurer, de mémoriser et de reconnaitre le contexte (fonction indispensable dans le domaine de la sûreté). Ainsi, l'intelligence distribuée dans le réseau de terrain a réduit sensiblement les coûts de câblage, et donne une meilleure maîtrise de la charge réseau.

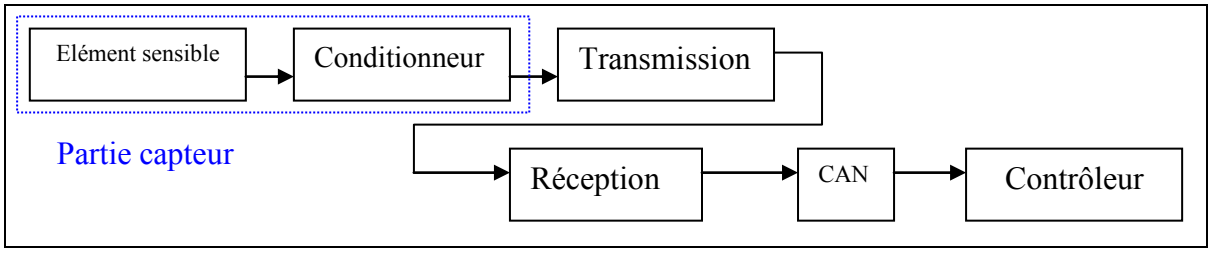


Figure2.3 : Chaîne d'acquisition sans élément de numérisation du signal.

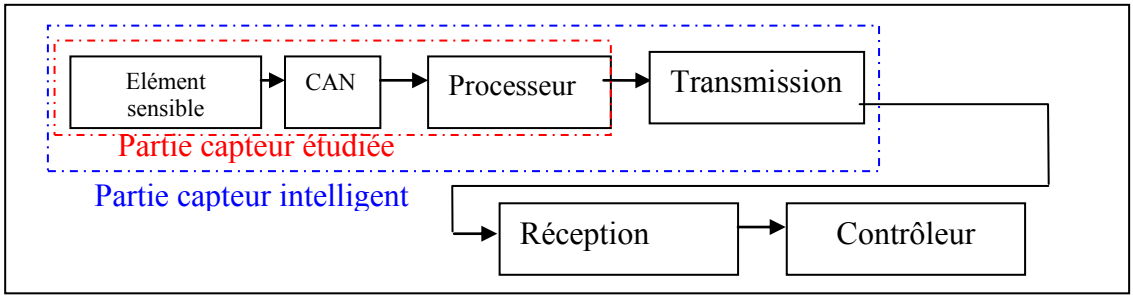


Figure2.4 : Chaîne d'acquisition d'un capteur muni d'un élément intelligent (processeur).

L'évolution de l'électronique et de la microinformatique permet d'exploiter les caractéristiques fréquentielles autant que temporelles des signaux issus des capteurs (transformée de Fourier rapide). Dans le même contexte, nous tenons à signaler aussi l'utilisation des méthodes probabilistes et les variétés de la théorie des croyances basée sur la modélisation bayesienne [Weber et al., 2007].

La prise en considération des notions de sûreté de fonctionnement est un phénomène assez récent dans le cadre de la conception des composants d'automatisme intelligents. Les composants conventionnels bénéficiaient d'une étude rudimentaire de la fiabilité et du retour d'expérience (REX). Les composants intelligents, qui sont des systèmes complexes, leurs sûretés de fonctionnement n'est pas encore traité. Leur complexité liée à l'intégration de puissants processeurs (souvent surdimensionnés) rend en effet difficile leurs analyse fonctionnelle et dysfonctionnelle. Néanmoins, ces composants réalisent des fonctions de diagnostic et de validation de données supposées améliorer la sûreté du système. Cependant, certains modes de fonctionnement non soupçonnés ou sous évalués du composant intelligent peuvent se révéler dangereux pour l'installation instrumentée.

Le point fort d'un capteur intelligent est sa capacité à mémoriser et à traiter des fonctions de conduite, de diagnostic et de sécurité. Sa versatilité et sa facilité de configuration/reconfiguration le rendent un composant économiquement intéressant. Cependant, les normes actuelles d'évaluation des systèmes au niveau de la confiance fonctionnelle, ne permettent pas toujours la cohabitation de n'importe quel type de fonction sur un même cœur de processeur. Un tel capteur intelligent présente les avantages suivants:

- Métrologique : accroissement de la précision (fusion de données, auto-calibrage…).
- Fonctionnel : aide à la maintenance par autotest intégré susceptible de déterminer automatiquement quel est l'élément défaillant, de transmettre des indications d'erreurs, mémorisation des évènements redoutés, configuration à distance.
- Economique : réduction des durées d'étalonnage et de calibration, fiabilité accrue, allègement de la charge du travail du calculateur central qui n'a plus à effectuer de calculs de corrections diverses.

Dans le cas d'un capteur intelligent cela signifie :

- Définition d'une relation bijective entre l'ensemble des valeurs prises par le mesurable pour cette application et l'ensemble des valeurs que peut prendre la mesure fournie par le capteur intelligent, associé à un système d'unités (définition des bornes de l'intervalle de mesure).

- Définition des actions à effectuer en cas de dépassement.
- Définition et activation de la relation caractérisant la relation entre grandeur et mesure.
- Validation du calibrage du capteur.

Toutes ces précisions montrent pourquoi un capteur intelligent va se révéler plus intéressant en matière de crédibilité et de sûreté de fonctionnement.

2.2 Systèmes à microprocesseurs

2.2.1 Une omniprésence mal maîtrisée

Vue que la dépendance de notre société aux systèmes à processeurs est quasi-totale (*Figure 2.5*), nous somme obligé d'accordé un intérêt particulier aux systèmes qui l'intègrent. Le degré de confiance accordable à ces systèmes demeure fort limité. L'origine des fautes y est d'évidence diffuse et variable, elles peuvent être :

- Physiques : bruits (électromagnétique, vibration, électronique de puissance, température, particules), défaut de composants (vieillissement, migration électronique), variation du processus.

- Humaines involontaires : incomplétude des spécifications, erreurs de spécification et de conception, erreurs dans les processus de fabrication et de test, erreurs logicielles, erreurs dans les outils de conception/test/système d'exploitation ; ou volontaires : modification de l'environnement (augmentation des stress), modification du type de missions, attaques en phase de conception/fonderie/test et exploitation (virale).

- D'interactions : de système à système/système à humain/humain à humain, parallélismes (architectures multiprocesseurs, distribuées).

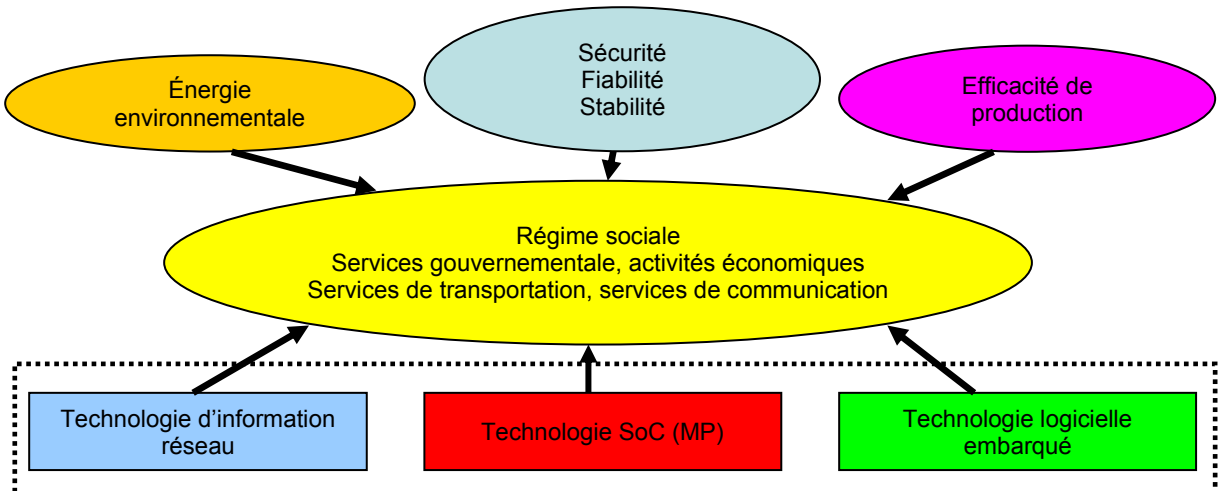


Figure 2.5 Infrastructure de technologie d'information

On peut raisonnablement attendre d'un système (limites de responsabilité à définir) une certaine capacité à fournir les services attendus, même en présence d'événements imprévus. Cette capacité se doit d'être quasi-totale (impératifs économiques ou éthiques) dans un certain nombre de domaines : installations nucléaires, production et transport d'énergie, ferroviaire, aéronautique, systèmes gouvernementaux. Cependant, un gros problème se pose, il n'existe pas de mesures génériques ou d'indicateurs pertinents de la fiabilité pour ce type de système, mais une profusion de métriques : MTBF, MTTR, BER (*Bit Error Rate*)… malgré une très forte demande sociétale et juridique. La science se trouve influencer par des impératifs purement économiques! L'accroissement de la complexité des systèmes et services, du nombre de services fournis par un système, de la diversité des missions des systèmes (authentification, calculs numériques, navigation / Internet…), s'est accompagnée d'une augmentation de la complexité des architectures informatiques, de la puissance et de la complexité d'architecture des processeurs, d'une répartition des services sur plusieurs calculateurs communicants par des bus locaux, réseaux locaux, réseaux sans fil…

La complexité de conception est aggravée par l'implication d'un nombre important d'acteurs dans la phase de développement, la production artisanale des logiciels et prototypes matériels, les pressions du Business model (Réutilisation, Composants, Middleware, Standardisation) avec des cycles de renouvellement des produits de 6 mois pour l'électronique Grand Public. Mais, dans un pareil contexte, que dire de l'expression des besoins, de la propriété intellectuelle, de la responsabilité, de la validation et de l'imbrication entre sûreté de fonctionnement et qualité (FMDS vis-à-vis qualité perçue et performances dynamiques et opérationnelles) et paradigmes de conception (*design for cost*, *performance*, *safety*).

Les études de sûreté de fonctionnement doivent aujourd'hui prendre en considération les problèmes de diversité des fautes (physiques, humaines…), de diversité des relations entre fautes et défaillances (sauts de couches et de limites de sous-systèmes, interactions entre fautes), de définition des fautes (changement dynamique des spécifications).

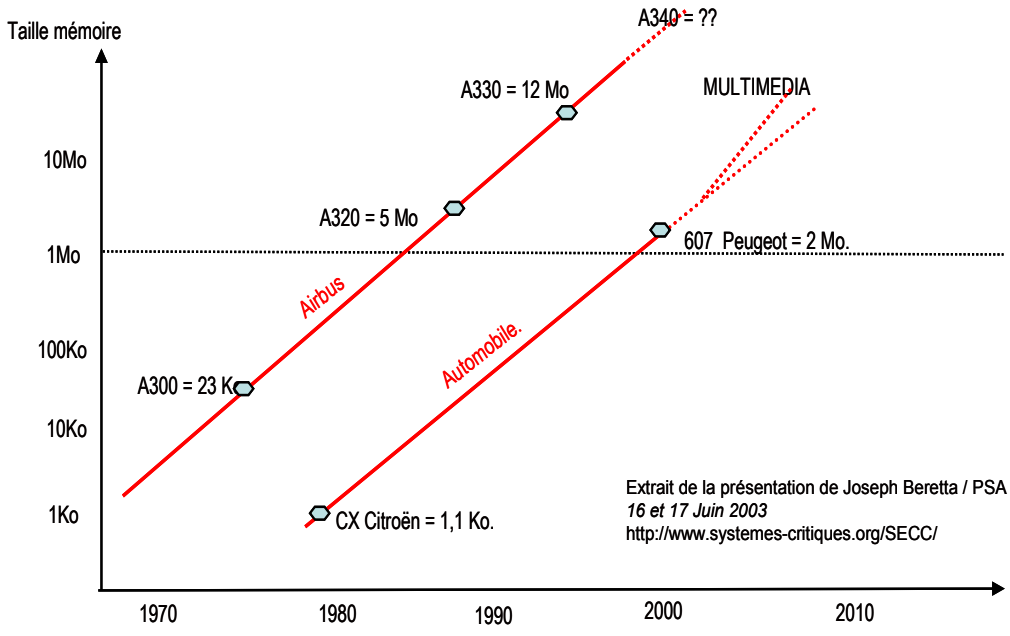


Figure 2.6 : Distribution de la mémoire en fonction de la taille du logiciel

La vérification des systèmes (garantie déterministe, garantie probabiliste, certification…) est un challenge difficile, les dysfonctionnements pouvant résulter de problèmes de conception ou de choix de conception (risque assumé / problème de coût, impossibilité de gestion du risque). Concevoir, programmer et vérifier le parallélisme (intrinsèque des activités des systèmes embarqués), évaluer la sûreté de fonctionnement d'ensembles mêlant intimement matériel et logiciel reste des problématiques ouvertes. Et si les défaillances peuvent être caractérisées par leur domaine (en valeur, temporelles : avance ou retard, par arrêt : plus de sorties perceptibles par l'utilisateur ou retard ∞, défaillances erratiques), leur détectabilité (signalées / non signalées), leur perception par les utilisateurs (cohérentes / incohérentes /byzantines) et leur conséquences (notion de sévérité ou gravité), les problèmes d'interactions, de causes communes de défaillance et autres rendent obsolètes de nombreuses méthodes.

2.2.2. Composants intelligents (SMART components)

Le terme capteur désigne en réalité un ensemble constitué d'un capteur proprement dit, qui transforme une grandeur physique observée en une grandeur utilisable (température → intensité électrique), et de conditionneurs, transmetteurs de signaux, alimentation… Le développement des MEMs (mémoire), dans les années 1980, et de l'intelligence embarquée, a permis l'émergence de composants intelligents (Figure 2.7)capables d'autodiagnostic, d'auto-ajustage, de corriger les erreurs de mesure, de reconfiguration, d'une communication numérique bidirectionnelle [NF 60770-3].

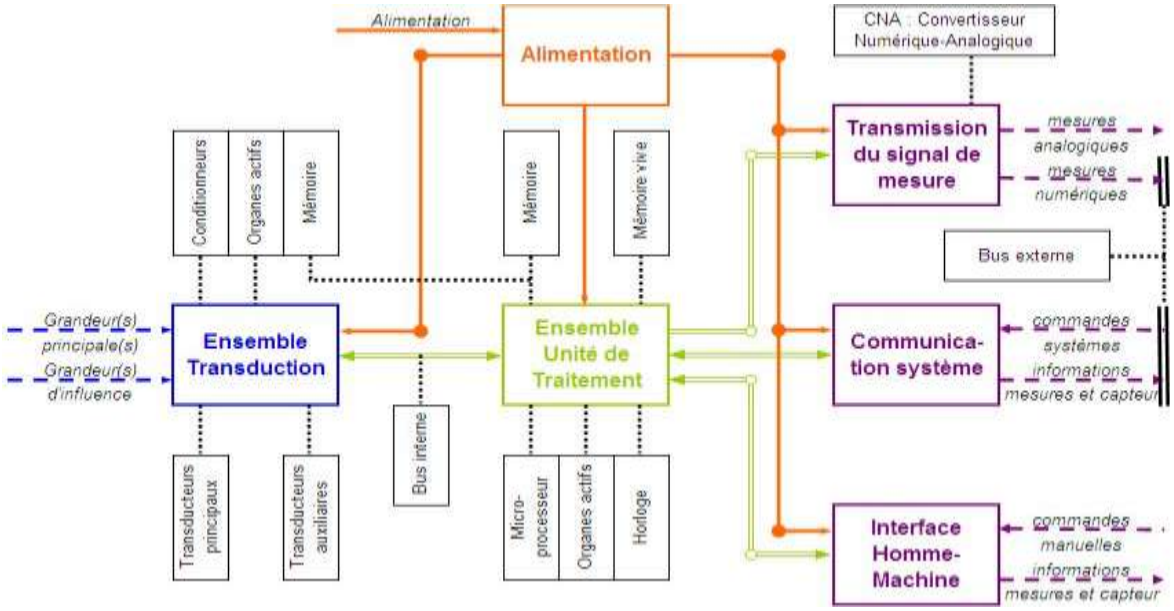


Figure 2.7 : Vision global d'un élément de mesure intelligent

Il en résulte de nombreuses interactions matérielles et fonctionnelles, des réponses de composants difficiles à appréhender en cas de défaillance, un flux informationnel important, avec peu de retour d'expérience. D'où des difficultés d'évaluation de la Sûreté de Fonctionnement, dont les méthodes se révèlent peu appropriées : problème d'exhaustivité et de définition des interactions (AMDEC), limitation en nombre et type d'information des outils booléens (AdD, BDF), définition des états pour les modèles états-transition (Markov, RdP). De nombreuses questions restent en suspend quant aux bénéfices vis-à-vis de la sûreté de fonctionnement. La Fiabilité pâtit des nombreux éléments additionnels (composants

électroniques, unités programmables, aspects logiciels), des sources d'erreurs, des causes et modes de défaillance supplémentaires, de la compensation par des procédures de tolérance aux anomalies, de la présence de réseaux de terrain. La Maintenabilité est réalisables dans des meilleurs conditions de : autodiagnostics, stockage et traitements des données, communication numérique. Quant à la Sécurité, elle pourrait bénéficier d'une meilleure couverture des défaillances détectées, d'une meilleure définition des positions de repli, d'un management plus réactif…Mais quant est-il réellement ?

2.2.3. Typologie d'erreurs et mécanismes de protection

En général, une architecture informatique est équivalente à une série de couches et niveaux d'abstraction : matériel, firmware, assembleur, noyau, système d'exploitation et applications (Figure 2.8)…

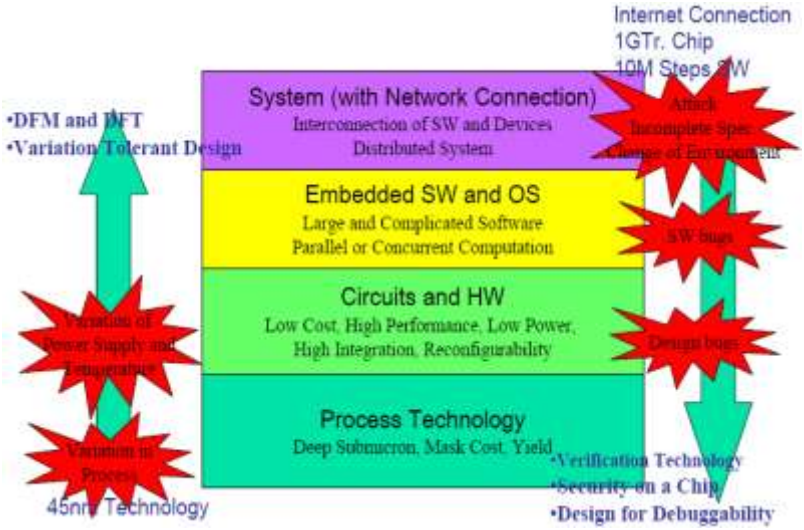


Figure 2.8 : Différentes couches d'une architecture informatique

Dans la suite de l'étude, nous nous concentrerons sur l'architecture matérielle. Les niveaux d'intégration atteints (gravure < 0,10 micron) et l'importante complexité des systèmes actuels rendent utopique l'éradication des erreurs systématiques (de spécification, de conception, de réalisation). Leur nombre résiduel peut cependant être limité par une plus grande précision de développement. Dans le cas de l'électronique (plusieurs centaines de millions de transistors par CPU) et du logiciel (plusieurs millions de lignes de code par application), la vérification (analyse statique, preuves mathématiques, analyse de comportement, exécution symbolique, tests structurels et fonctionnels) et les tests de non régression. Quant aux conditions d'exploitation et d'environnements opérationnels (technologiques et naturels), toujours plus agressifs, ils multiplient les risques d'erreurs aléatoires (matérielles ou informationnelles). L'augmentation des stress (intensité et durée d'exposition) accélère la dégradation des propriétés physiques des composants, provoquant courts-circuits, ouvertures intempestives de jonctions, modifications des délais de propagation des signaux, ou des niveaux de seuil, entraînant des erreurs permanentes à l'origine de défaillances. Les fautes transitoires sont plus difficiles à cerner et peuvent être locales ou globales. L'électronique de puissance à fréquence variable (fortement génératrice de rayonnements électromagnétiques issus de multiples sources, de formes d'onde, d'amplitude et de fréquences très différents) est aujourd'hui omniprésente. Même l'application de contraintes hors bande passante (de par les nombreuses non linéarités toujours présentes : diodes de protections ESD/ElectroStatic Discharge…) peut perturber un composant (cf. bruit d'un ordinateur lorsque l'on allume un portable à 0,9 ou 1,8

Ghz) et les télécommunications (modulation dans les amplificateurs RF) perturbent également les logiques de traitement, voire peuvent les détruire par effet thermique. De plus, en 20 ans, on est passé en haute altitude, d'une particule stellaire/avion/an à 2/avion/vol en moyenne altitude. Ces particules (ions lourds, protons énergétiques, neutrons) notamment, d'essence naturelle ou technologique, peuvent déclencher des phénomènes du type SEE (Single Event Effect). Ce sont des effets localisés soumis à des lois probabilistes (probabilité d'agresser n circuits redondants très faible). Ils se décomposent en :

- **Effets réversibles :** défauts transitoires non destructifs, appelés aléas logiques ou erreurs logicielles, tels les SET (Single Event Transient), les SEU (Single Event Upset) générateurs d'inversion de bit(s) (bit-flip ou upset) et les MBU (Multi Bit Upset) où plusieurs bits sont corrompus par la même particule. Ces derniers peuvent affecter plusieurs structures voisines. Leur probabilité d'occurrence est plus faible, mais devienne importante à cause de miniaturisation.

- **Effets réversibles sur réinitialisation :** tel les SEFI (Single Event Functional Interrupt) caractérisés par un comportement erratique du circuit. La récupération de l'état normal s'obtient par réinitialisation complète, ou après coupure de l'alimentation.

- **Effets irréversibles :** dégradations permanentes destructives, appelées erreurs permanentes ou erreurs matérielles, tels les SEL (Single Event Latchup) générateurs de courts-circuits, les SHE (Single Hard Error) ou collage de bit non reprogrammable, les SEGR (Single Event Gate Rupture) ou destruction d'une structure MOS par claquage de l'oxyde de grille et les SEB (Single Event Burn-out) ou destruction d'un composant par effet thermique.

Un marché spécifique s'est développé pour les composants fortement immunes (LET_{th}/Linear Energy Transfer threshold > 100 MeV.cm²/mg à 125 °C), tel l'AT697 RF d'Atmel Corporation, entièrement durci, destiné aux missions spatiales, il dispose d'une puissance de calcul de 90 Mips/23 MFlops à 100 MHz (0,7 W) et reste extrêmement fiable avec un taux d'erreur par SEU/SET < 10^{-5} erreurs/composant/jour jusqu'à 300 Krad. Il intègre les dernières techniques de durcissement (Full Triple Modular Redundancy, EDAC/Error Detection And Correction, bit de parité). L'agression par rayonnement électromagnétique diffère de celle par SEU : le circuit peut être agressé dans son ensemble (tous les circuits redondants et de contrôle peuvent être perturbés). L'amélioration de l'immunité électromagnétique des systèmes embarqués exige l'association de mesures matérielles et logicielles (prévention, tolérance, ou prévision de fautes) [Alaoui R. 2007]. Quant aux fautes de délai transitoires (origine électromagnétique, SET, vieillissement), elles correspondent à des perturbations temporaires (fluctuation de l'alimentation, interférence électromagnétique, radiations, température…) capables de produire des mémorisations de valeurs erronées par dépassement du délai du chemin critique (temps de cycle d'horloge) : une baisse de la tension d'alimentation diminue les tensions internes du circuit ce qui augmente les temps de transitions des composants. Lorsqu'un délai est dépassé et qu'une valeur erronée est mémorisée sur un bit, son effet peut être assimilé à un bit-flip à l'entrée de la mémorisation, au moment du front d'horloge.

Les fautes engendrées peuvent ainsi se révéler permanentes (persistante jusqu'à réparation), intermittentes (sporadique), ou encore transitoire (apparition isolée). On aboutit ainsi à une multiplicité de type de défaillances : défaillance d'exécution (arrêt d'exécution des

traitements, bouclage infini, exécution erronée, défaillance sur interruption), défaillance fonctionnelle (résultats incohérents, pannes arbitraires), défaillance opérationnelle (temps de réponse inadapté, utilisation abusive de ressources/saturations, indisponibilité). Pour garantir la robustesse d'une fonction, il apparaît nécessaire de mettre en œuvre un ensemble approprié de dispositions préétablies et systématiques (prévention, élimination, inhibition de fautes) destinées à assurer une qualité de service digne de confiance. L'objectif est d'éviter la propagation d'erreurs sur activation de fautes internes ou externes pouvant mener à une défaillance systémique. La tolérance aux fautes permet au système de continuer à fournir un service conforme à la spécification malgré l'apparition de fautes. C'est la redondance de certaines fonctions qui prévient la transformation de la faute en erreur. Lorsque des fautes apparaissent dans un système, on peut minimiser leur nombre et réduire leur impact : c'est l'élimination de fautes (cloisonnement des fonctions pour limiter la propagation d'une erreur). Lorsque l'on connaît le type, le nombre ou les conséquences des fautes, on tente de faire de la prévision de fautes, pour adapter les protections du système (duplication de code par expression régulière, gestion par signature ou par identifiant). La sûreté de fonctionnement exige ainsi le recours à des mécanismes de type contrôle de défaillances, détection d'erreurs, par contrôle temporel, contrôle d'exécution, diversité, redondance d'exécution, redondance matérielle, redondance informationnelle et recouvrement :

- **Détection :** destinée à sécuriser l'exécution d'une application par construction d'un système Fail Silent, à la base des mécanismes de tolérances aux fautes. Caractérisée par sa latence (délai de détection de l'erreur) et son taux de couverture (% d'erreurs détectées), elle peut être obtenue par contrôle temporel (Watchdog rafraîchi ou réarmé périodiquement par le processeur), autotests périodiques ou ponctuels (tests spécifiques, Checksum, CRC, comparaison de signaux), contrôle de cohérence (comparaison de données, vérification des traces d'exécution), contrôle d'intégrité, contrôle de signature. Elle peut être géographique, spatiale (duplication de composants), temporelle (traitements multiples, contrôle de l'évolution des délais associés aux évènements, du paramétrage ou de l'application), informationnelles (codes et signatures), ou modale (multiplicité de moyens).

- **Diversité logicielle :** algorithmes, logique et architecture de programme, séquences et ordonnancement d'exécution, langages de programmation, environnement de programmation.

- **Redondance d'exécution :** utilisation de ressources différentes et autotests (Self Testing Checkers), dissymétrie de codage ou de données.

- **Redondances matérielles (homogène/hétérogène) :** architecture maître/esclave, calcul de plausibilité, Lock-Step Dual Processor Architecture, par contrôle d'autotests, architecture nom froide/chaude/tiède.

- **Redondance informationnelle :** contrôle de parité, Checksum et compteur de processus, CRC/Cyclic Redundancy Check , Code de Hamming, codes arithmétiques.

Dans le cas d'un processeur sécuritaire codé, nous parlons de compensation et de recouvrement de la manière suivante :

- **Compensation (error masking) :** détection et correction des erreurs (par EDAC/Error Detection And Correction code, ECC/Error Correction Code) de manière à maintenir la qualité de service.

- **Recouvrement (error recovery) :** par reprise (par blocs depuis un état antérieur réputé correct), poursuite (en avant sur exception avec reconstruction totale ou partielle de l'état à atteindre), ou par programmation défensive (vérification de consistance du changement d'état).

2.3 Place de l'évaluation dans le processus de conception

La complexité des systèmes et l'intégration de la vérification dans le processus de conception pour répondre aux spécifications fonctionnelles exigent d'importants efforts qui obèrent les problématiques de la sûreté de fonctionnement (FMDS). Ces paramètres sont introduits trop tardivement, par conséquence leur intégration dans le processus de conception est médiocre, bien qu'elles influent au même titre que la vérification des propriétés fonctionnelles sur la phase de conception (Figure 2.9).

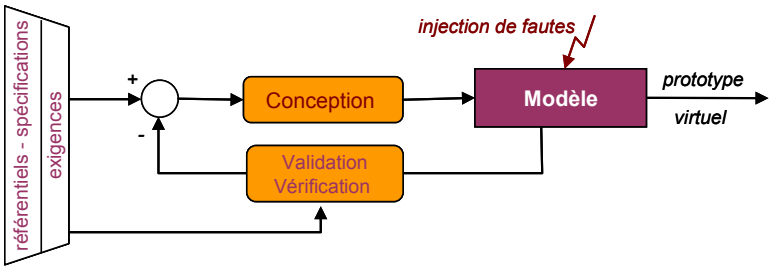


Figure 2.9: Positionnement de l'évaluation fiabiliste dans le cycle de conception

L'objectif de la vérification/validation est de révéler les fautes qui ont pu être introduites au cours de n'importe quelle phase du cycle de développement (spécification, conception, réalisation, fabrication, mise en service). Elle doit accompagner le processus de développement afin de diminuer le coût de leur élimination [Laprie J.C.,2004]. Minimiser le nombre d'erreurs résiduelles, l'éradication des erreurs systématiques n'étant qu'une utopie, est l'un des objectifs majeur de la validation. La présence d'erreurs aléatoires en phase opérationnelle exige la validation des architectures, tant matérielle que logicielle. Les attendus sont autant d'ordre qualitatif (longueur des séquences défaillantes, localisation des points structurellement faibles), que quantitatif (évaluation de métriques telles la FMDS). En fonction du niveau d'abstraction concerné, du niveau de détail des modèles (granularité) et du type de traces recherchées, de multiples approches sont disponibles.

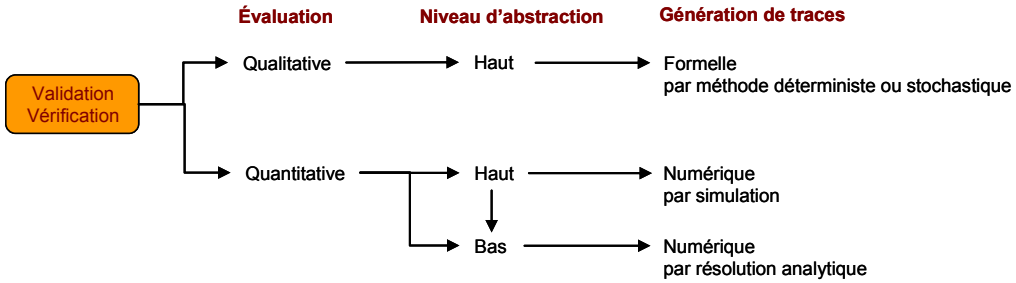


Figure 2.10: Positionnement modèles / traces

Le niveau d'abstraction courant conduit à l'utilisation d'une famille de modèles et à l'obtention d'un type de trace (Figure 2.10). Que ces traces soit qualitatives (formelles), ou quantitatives (numériques), que la solution soit analytique, ou obtenue par simulation, d'où la nécessité d'un modèle. De son niveau d'abstraction dépendra sa capacité à manipuler des objets complexes. Dans le domaine de l'évaluation probabiliste prévisionnelle de la sûreté de fonctionnement, l'émergence du logiciel dans les systèmes de contrôle industriels a favorisé l'application des méthodes formelles lors de la conception logicielle et le recours à la vérification (model checking). Certaines approches consistent à ne s'intéresser qu'aux séquences d'événements menant aux états redoutés pour la mise en place des barrières de défense. Les approches combinatoires peuvent être synthétisées par des arbres de défaillances, des diagrammes ou graphes de fiabilité… Leurs extensions (cf. logiques temporelles) ont donné naissance par exemple aux arbres de défaillance dynamiques [Bechta et al., 2000] qui intègrent certains aspects temporels, mais ne permettent plus de générer simplement la fonction de structure. L'approche markovien, malgré l'hypothèse forte sur le vieillissement, reste fortement usitée dans le cas de composant matériel et logiciel [Laprie et al., 1995]. La modélisation sous forme d'automates et de langages associés, les réseaux de Petri, les réseaux de files d'attente, l'algèbre des processus stochastiques … permettent de manipuler aisément des modèles de grande taille (adaptés à la modélisation des systèmes complexes). Ces modèles compacts peuvent être associés à des méthodes de réduction de la dimension de l'espace d'état.

On doit regretter cependant dans toutes ces approches le peu de préoccupation à considérer les interactions entre le logiciel et le matériel.

2.4 Évaluation conjointe logiciel /matériel

Au sein de la Partie Commande, la conception de l'ensemble logiciel/matériel exige plusieurs niveaux de granularité et d'abstraction. Dans le cas des microsystèmes, à partir du modèle *TLM* (transactionnel) sont élaborés par niveau de granularité décroissant le *PV* (voir figure 2.11) destiné au développement logiciel, puis le *PV+T* (Timing) orientant les choix architecturaux et l'estimation des performances. Côté architectural, la synthèse matérielle s'exprimera au travers du modèle *RTL* (différent de forme/*PV*). Chaque modèle est en capacité de générer une trace permettant sa validation. Les propriétés fonctionnelles peuvent être vérifiées pour chacun des niveaux d'abstraction, par méthode formelle, ou par simulation (rapidité dépendante du niveau de granularité). La comparaison entièrement formelle de traces de niveaux d'abstraction différents, en l'occurrence *TLM* (*PV*, *PV+T*) et *RTL*, devient possible [Hami et al.,2005] à partir de traces *RTL* issues d'automates synchrones

déterministes et dont l'exécution produit (via un traducteur) une trace dans la forme *PV+T*. Cependant, l'analyse de la robustesse des propriétés fonctionnelles et non fonctionnelles a encore recours aux techniques d'injection de fautes. Notre proposition est d'utiliser le modèle *RTL* et le code généré pour, parallèlement à la phase de validation fonctionnelle, valider l'ensemble logiciel/matériel quant aux aspects dysfonctionnels qualitatifs et quantitatifs (Figure 2.11).

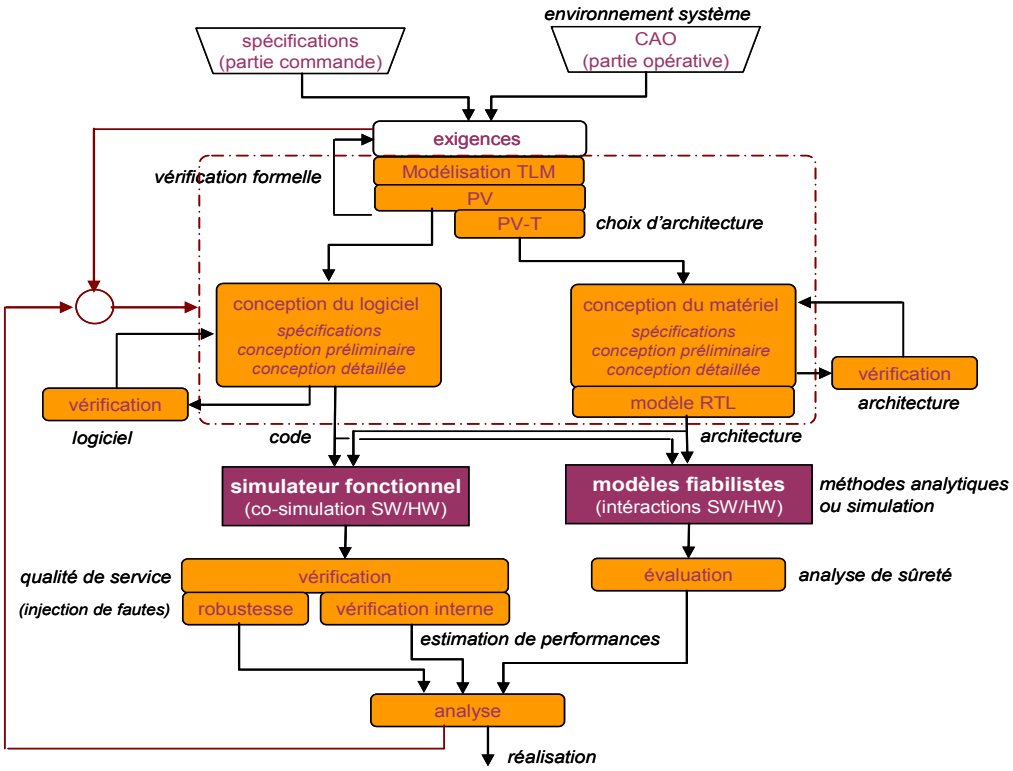


Figure 2.11: Positionnement de l'évaluation fiabiliste dans le cycle de conception logiciel/matériel

La vérification des propriétés, qualitative et quantitative, dysfonctionnelles de l'ensemble logiciel/matériel sera basée sur l'utilisation de modèles formels et d'une solution analytique.

2.5 Approche flux informationnel [Hami et al., 2005]

Dans cette approche, la phase de modélisation se réalise en 2 étapes : un modèle de haut niveau permettant la construction d'un automate à états finis. Les séquences dysfonctionnelles menant à des défaillances systémiques sont obtenues à partir de langages générés à partir de l'automate à états finis. Ces séquences sont à la base de l'analyse qualitative : longueur des séquences, localisation de points faibles… Pour palier à l'incontournable explosion combinatoire de l'espace d'états, une variante de BDD (z-BED) est progressivement construite à partir des états finaux du modèle de haut niveau.

A partir de la défaillance du système, des séquences ou des coupes minimales sont obtenues, une approche markovienne multi phase permet de calculer les probabilités de défaillance de chaque ressource matérielle, permettant ainsi de quantifier les probabilités d'apparition de chacune des listes (combinaison d'événement) correspondant aux différents modes de

fonctionnement du système (productif, improductif, dangereux ...). Le processus est décrit dans la figure 2.12 :

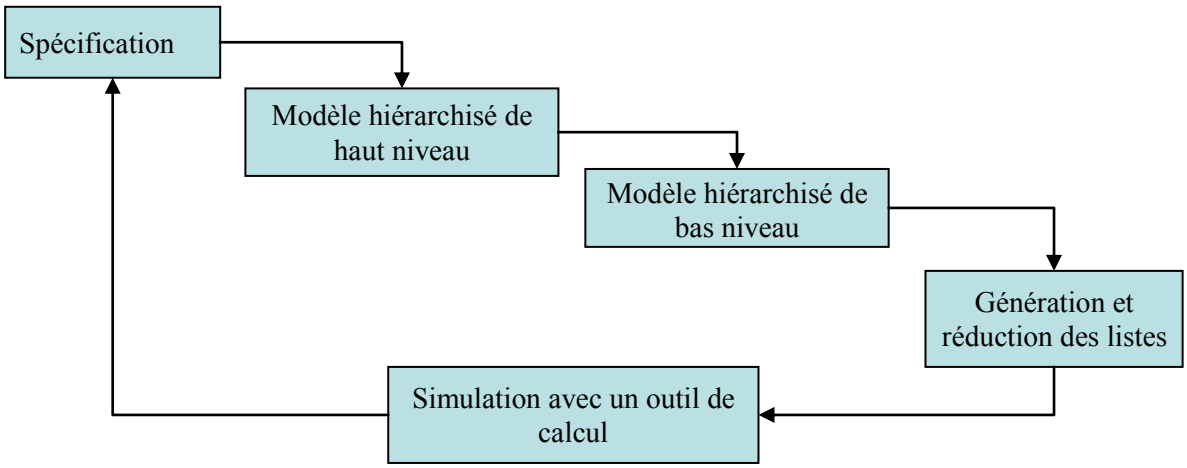


Figure 2.12: Processus d‟évaluation générale

2.5.1 Le modèle flux informationnel :

Nous nous sommes surtout intéressés à deux situations de défaillance différentes du système:
- Les incidents dangereux, qui pourraient conduire à des accidents
- Les incidents non dangereux intempestifs.

Nous voulons générer tous les scénarios menant à l'un de ces deux événements indésirables de tout le système. Particulièrement les points uniques de défaillance, les défaillances de cause commune et à la propagation de défaillance. Ces types de défaillances ne sont pas souvent évidents, ils peuvent être facilement oubliés dans une tentative directe visant à créer un arbre de faute. Ce problème sera résolu par une approche hiérarchique. Nous utilisons un schéma bloc orienté représentant le flux d'information par l'intermédiaire du modèle de haut niveau d'automates d‟états finis et les règles pour la modélisation de bas niveau. Ce diagramme de flux d'information (IFD) et les automates sont des versions généralisées des diagrammes et des automates présentés dans [Arnold et al., 2000].

2.5.2 Diagramme de Flux Informationnel :

Pour les diagrammes de flux informationnel, nous utilisons différents types de blocs qui représentent différentes entités fonctionnelles. Nous distinguons:

- WD-blocks pour les fonctions de contrôle (par exemple chiens de garde)
- SRC-blocks comme source d‟information
- DEC-blocks pour les décisions logiques
- ST-blocks pour toute autre fonction (stockage d‟information, transformation d‟information, self-tests...)

Le bloc du type WD (watch dog) est spécialement utilisé pour les unités de contrôle, avec un chien de garde. Le bloc WD a une entrée et une sortie et permettent de détecter l'absence

d'informations sensibles, afin de réagir par conséquence à la transmission par défaut des valeurs d'erreur particulière.

Les SRC-blocs créent l'information qui circule dans le diagramme. Ils représentent par exemple les capteurs du système et ont une seule sortie.

Les ST-blocks (blocs standard) sont les éléments les plus polyvalents, ils ont une entrée et une sortie, et ils sont utilisés pour toutes les entités fonctionnelles qui ne peuvent pas être représentés par les autres blocs, par exemple le stockage ou la transformation de l'information.

Le dernier type de blocs, DEC-blocs, représentent des entités de décision logique, ils ont plusieurs entrées et une sortie, permettant de décrire le comportement de multiples sources interconnectées d'information, ils ne décrivent pas des entités physiques.

Les éléments qui contribuent à cette décision doivent être matérialisés par l'ajout de ST-block successifs. Un des blocs dans le diagramme, normalement un ST ou DEC-bloc, peut être marqué comme dernier bloc, ce bloc n'a pas de sortie et est utilisé pour générer les scénarios de défaillance comme il le sera décrit dans le paragraphe suivant. Un exemple d'un IDF présenté dans [Hami et al.,2005] est montré dans la Figure 2.13, on peut y voir des blocs pour les différents modules du système, quelques blocs de décision supplémentaire et les flux d'information à partir des blocs source jusqu'au bloc final, en un seul pas le temps T. Pour les blocs source, les capteurs génèrent les informations qui se propagent à travers le schéma, l'information produite par les blocs successifs est traitée et propagée. L'échange entre les blocs est toujours exempt de fautes. Bien que le traitement des données se fasse dans les blocs, des erreurs peuvent apparaître. Cela signifie que l'état du signal peut changer dans un bloc.

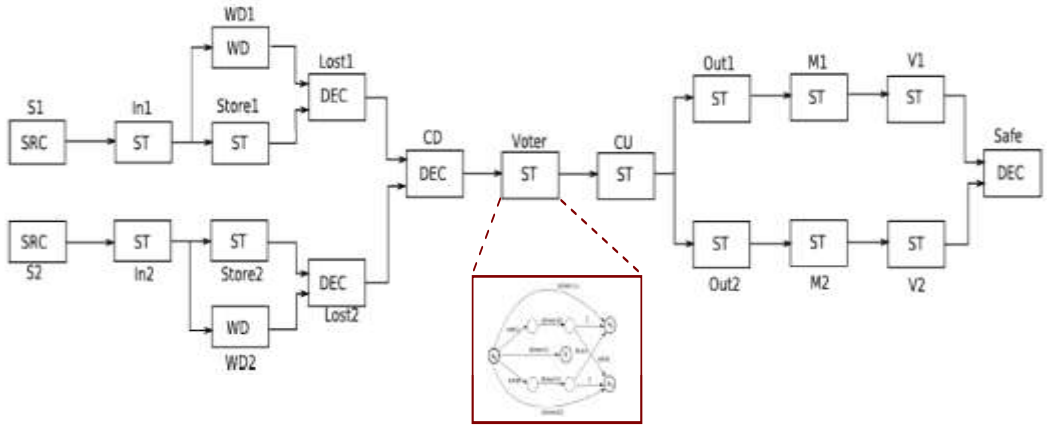


Figure 2.13: Diagramme de flux informationnel pour un arrêt d'urgence

Nous distinguons trois différents états erronés du signal :

- Pas de défaillance détectée (état de défaillance sur S)
- Défaillance non détectée (état de défaillance dangereux D)
- Perte de signal (état d'inhibition I)

2.5.3 Automates d'états finis

La théorie des automates à états finis (à nombre d'états finis) a été associée à avec la théorie des langages. Ces modèles reviennent à spécifier des ensembles d'états et des transitions entre ces états [Hami et al.,2005]. Les langages et les automates permettent de traiter mathématiquement les problèmes relatifs aux Systèmes à Événements Discrets (SED), essentiellement d'un point de vue logique (analyse qualitative).

Chaque SED a un ensemble d'événements qui lui est associé. Ces événements font évaluer le SED. Cet ensemble peut être vu comme un alphabet d'un langage et les séquences d'événements sont des mots (aussi appelés chaînes) de ce langage. Un automate est alors un dispositif qui engendre un langage en manipulant l'alphabet (les événements).

Après avoir modélisé le flux d'information, il est nécessaire de préciser les changements d'état de l'information. Pour les non-DEC-blocs finis, des automates d'états finis acycliques sont utilisés pour représenter une fonction de liaison entre entités. Un automate déterministe fini est un quintuple $(S,\sum, \delta, x_0, F)$ avec :

- Ensemble des états finis S
- Alphabet d'entrée $\sum$
- Fonction de transition $\delta : S \times \sum \rightarrow S$
- Etat initial $x_0 \in S$
- Ensemble des états finaux $F \subset S$

Pour notre exemple (figure 2.14), nous avons un état initial prédéfinis x0 et trois états finaux prédéfinis XS, XD, et XI, qui représentent les trois états défectueux de l'information. Les alphabets utilisés dans ces automates sont les symboles qui représentent différents types d'échecs. Ils sont désignés comme suit:

- input(i) avec $i \in \{S,D, I\}$ (pour ST- et WD-blocks)
- d(x, y) avec x comme ressource matériel et $y \in \{0, S, D, I\}$
- bf(e) avec „e" représente source de faute de type bit flip
- tf (f) avec f représente le test de ressource matériel
- mot vide ε

L'avantage des automates pour notre proposition, c'est qu'ils sont en mesure d'inclure différents types d'anomalies et plusieurs modes de défaillance. En outre, ils sont très intuitifs et on peut les déduire assez facilement pour les sous-systèmes.

Nous prendrons l'exemple, d'automate du bloc Store1, qui est montré dans la figure 2.14. Tout en stockant les résultats, différentes défaillances peuvent se produire. Dans un premier temps, le composant mémoire peut être endommagé physiquement, de sorte qu'il y ait plusieurs bits immobilisés à un (S), enfermé à zéro (S) ou que la mémoire ne soit plus disponible du tout (I).

Il y a aussi la possibilité d'un bit flip dans la mémoire changeant la valeur des données stockées ; input(i) est utilisée pour la propagation d'une faute du bloc prédécesseur, il se produit uniquement à l'état initial x0.

Les valeurs possibles pour i dépendent du type du bloc courant. Dans ST-blocs, i ne peux être que S ou D tel que ST-blocs ne sont pas capables de traiter les informations perdues. En revanche, WD-blocs ne contiennent que des input(I).

Pour les défaillances matérielles, le symbole d(x, y) est utilisé. y indique l'état de la ressource matérielle x : fonctionnement correct (0) ; défaillance non dangereuse (S) ; défaillance dangereuse potentielle (D) ; absence de sortie (I). bf(e) représente les erreurs

environnementales comme bit flips d'une ressource e. ft(f) indique qu'il y a pas d"erreur détectée dans une ressource durant un test.

Pour simplifier le calcul, nous supposons que l'effet du bit flips ou faute de tester d"une ressources est toujours le même dans un échantillon de temps. Notez que tous les trois états finaux ne sont pas toujours nécessaires. Cela sera expliqué plus en détail dans le paragraphe suivant.

Les automates définissent un langage qui décrit tous les scénarios conduisant à un échec de différents blocs (réponse inattendu du bloc). On peut distinguer trois sous langages du langage généré différents correspondants aux trois états finaux, pour les défaillances non-dangereuses, les défaillances dangereuses potentielles, et les défaillances sans aucune sortie. Ces langages peuvent être extraits et sont enregistrés dans les trois listes LS (B), LD (B), et LI (B) pour le bloc B. L'automate de la figure 2.14 définit les langages suivants:

- L_S(Store1) = {d(mem, S); input(S)d(mem, 0); input(D)d(mem, 0)bf(m)}
- L_D(Store1) = {d(mem,D); input(D)d(mem, 0); input(S)d(mem, 0)bf(m)}
- L_I(Store1) = {d(mem, I)}

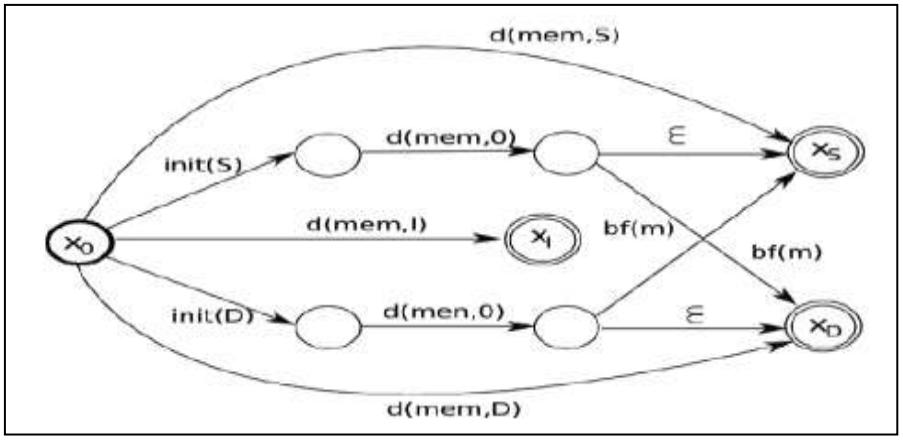


Figure 2.14: Modèle d"un automate d"une entité fonctionnelle

Règles spécifique aux DEC-blocs

Les blocs de décisions utilisent un autre modèle de bas niveau pour décrire leur comportement. À cet effet, des règles booléennes sont introduites. Ces règles utilisent les états des signaux (soit S, I ou D) de chaque entrée. Il y a trois règles, ces règles seront représentés les listes LS, LD et LI. Si on prend le dernier bloc de l'IFD indiqué en figure 2.13, les règles suivantes sont choisies:

- S : V 1 = S $\vee$ V 2 = S
- D : V 1 = D $\wedge$ V 2 = D
- I : false

Remarque : Pour V1et V2 voir la figure 2.13

Ainsi, le bloc final propagera une erreur si au moins l'une des deux vannes injuste une. Une défaillance dangereuse ne se produira que si les deux vannes défaillent. Comme nous sommes

à la recherche d'une liste complète, nous devons définir la manière de l'extraire. Les disjonctions dans les règles seront traitées par l'unification de deux listes, les conjonctions seront traitées par le produit des deux listes. Pour notre exemple, nous pouvons conclure:

- L_S(Safe) = L_S (V 1) $\cup$ L_S (V 2)
- L_D(Safe) = L_D (V 1) $\times$ L_D (V 2)
- L_I(Safe) = {}

2.5.4 Génération des listes globales :

L'intérêt principal est de générer tous les scénarios de défaillances dangereuses et la propagation de ces dernières vers le bloc final. Elles seront stockées dans les listes de LD et LS. Pour obtenir ces listes, les listes de LD(Bf) et LS(Bf) du dernier bloc Bf sont créés afin de les connecter avec les listes locales des autres blocs. Il est nécessaire de distinguer deux cas: DEC-blocs et non-DEC-blocs. Pour DEC-blocs, la méthode présentée dans le paragraphe précédent pour relier les blocs est utilisée.

Pour les non-DEC-blocs, tous les init(i) dans la liste sont substitués. La séquence après une entrée input(i) est combinée avec toutes les séquences d'une liste locale L_I(B) du bloc B par une concaténation. Pour illustrer cela, les trois listes suivantes sont utilisées:

- L_S (Bf) = {input(S)d(x, S)d(y,D); input(D)d(y,0)}
- L_S (B) = {input(S)d(v,0); input(D)d(v,S)d(w,D)}
- L_D (B) = {input(D)d(v,D); input(S)d(w,D)}

Si input(S) et input(D) sont substitués avec L_S (B) et L_D (B), nous obtenons:

L_S(Bf) = {input(S)d(v,0)d(x,S)d(y,D); input(D)d(v,S)d(w,D)d(x,S)d(y,D);
input(D)d(v,D)d(y,0); input(S)d(w,D)d(y,0)}

2.5.5 Evaluation quantitative

L'estimation de métriques tels les PFD (probabilité de défaillance sur demande) et PFS (probabilité de défaillance sure) nécessite 3 phases :

- modélisation de l'évolution des stress environnementaux auxquels est soumise l'architecture (température, pression, radiations…) et de maintenance des composants. L'impact opérationnel des paramètres environnementaux sur les composants sont disponibles dans de nombreux domaines tels l'aéronautique [FIDES 2004], l'automobile, les industries verrière et pétrolière… La prise en compte de contraintes humaines (opérations de maintenance…) n'est pas considérée dans les cas présentés ici.

- modélisation stochastique de l'évolution des probabilités d'erreurs matérielles, des fautes environnementales et d'échec aux tests.

- génération des listes caractéristiques des séquences (combinaisons d'erreurs) menant aux états recherchés. Par exemple, une liste conduisant de manière nécessaire et suffisante à une absence de réaction de la fonction de sécurité sous l'hypothèse d'une présence de demande au

moment d'acquisition, une autre conduisant à un déclenchement intempestif des actions de réaction de la fonction dédiée sécurité (réaction en absence de demande).

Après évaluation des stress environnementaux, les probabilités conditionnelles des processus markoviens associés à l'évolution de l'état des ressources matérielles, et des évènements internes et externes sont actualisées. Les probabilités d'erreurs pour chaque ressource matérielle sont alors actualisées par itération de processus markovien. On utilise finalement ces probabilités d'erreurs évaluées précédemment pour calculer les probabilités d'occurrences pour chacune des listes au temps t=kT. La probabilité d'occurrence de chacune des listes est donnée par la somme des probabilités de chaque combinaison des erreurs simultanées qu'elle contient. Par intégration par morceaux jusqu'à $t=T_{life}$, on obtiendra les probabilités recherchées.

A partir de la construction d'automate à états finis, si le système a une taille raisonnable, les listes (séquences d'événements menant à l'état de défaillance du système) sont construites et simplifiée. Dans le cas des modèles de grande taille, des modèles z-BDD sont construits progressivement à partir de l'état final (états du système global), pour permettre l'obtention de coupes minimales (séquences d'événements non ordonnés). Cette approche, permet le calcul analytique des probabilités de défaillance du système. Le processus de génération de listes est toutefois maintenu pour l'analyse qualitative des défaillances systémiques.

Le 2ème niveau de la représentation décrit le comportement fonctionnel et dysfonctionnel de chaque sous-entité fonctionnelle. Les automates à états finis basés sur la théorie de langage exprimant des séquences d'événements menant à ces deux modes opérationnels : la réaction est activée en l'absence de situation dangereuse (PFS) et la non activation ou bien absence de la réaction en présence d'une situation dangereuse (PFD).

Les indicateurs de performance de sécurité dédiée (PFDavg et PFSavg) sont liés à la notion de risque résiduel par [Hami et al.,2005]:

$$R_{res} = \frac{1}{T_{life}} \sum_{k=0..N} fT.PFD(kT).I_{dem} + \frac{1}{T_{life}} \sum_{k=0..N} (1 - f.T)PFS(kT).I_{abs} = f.PFD_{avg}.I_{dem} + (\frac{T_{life}}{T} - f).PFS_{avg}.I_{abs}$$

Avec f, probabilité moyenne d'occurrence de demande
I_{dem}: l'impact d'accident
I_{abs}: l'impact de déclenchement de défaillance

Si la situation dangereuse détectée par la fonction de sécurité (présence de la demande) mène à certains accidents : f sera égal à la fréquence moyenne d'occurrence d'un accident avant l'ajout d'une fonction dédiée à la sécurité. Une fois la relation entre le risque résiduel et les métriques PFDavg et PFSavg défini, il devient possible de juger la qualité de service d'une fonction de sécurité en définissant une méthode pour estimer les pertes économiques causées par son déclenchement. On note cette métrique:

$$C_{ind} = \sum_{k=1..E\left(\frac{T_{life}}{T}\right)} 1 - p(\ demand, kT).\bar{P}FS(\ kT)\ .c^*(\ kT) + (\frac{T_{life}}{T} - f)\ .PFS_{avg}.c^*$$

c*: induced average cost (in terms of economic loss per unit of time T) associated to an unavailability

Compte tenu de niveau de risque résiduel acceptable et un niveau de perte économique maximale acceptée, il est possible d'établir les exigences relatives aux deux seuils des indicateurs de performance PFDavg et PFSavg afin d'assurer la conformité avec les objectifs énoncés dans les hypothèses mentionnées et de constater que les propagations ne provoque pas un accident.

2.6 Capteur intelligent - Description du prototype

Dans les paragraphes précédents, nous avons montré une vue globale sur les capteurs intelligents, tel que l'intelligence de ce dernier réside dans sa capacité de vérification du bon déroulement d'un algorithme [Belhadaoui et al., 2008-b], test du résultat d'une opération arithmétique. Les avantages de ces capteurs sont :

- **Métrologique :** accroissement de la précision (fusion de données, auto-calibrage…)
- **Fonctionnel :** aide à la maintenance par autotest intégré susceptible de déterminer automatiquement quel est l'élément défaillant, de transmettre des indications d'erreurs, mémorisation des évènements redoutés, configuration à distance.
- **Economique :** réduction des durées d'étalonnage et de calibration, fiabilité accrue, allègement de la charge du travail du calculateur central…

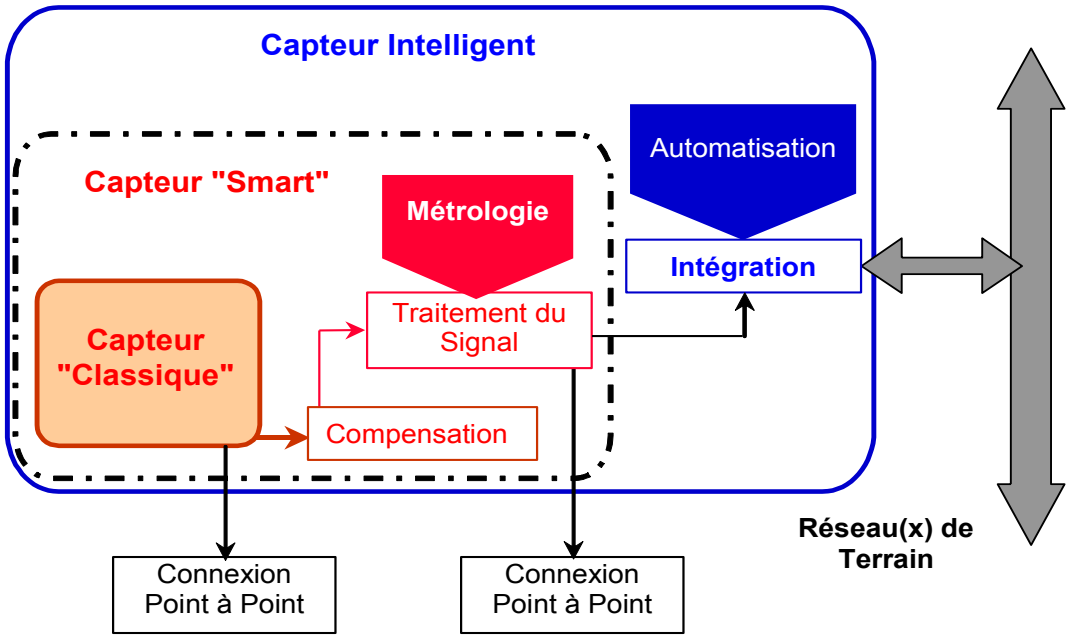


Figure 2.15 : Architecture globale de capteur intelligent [mkhida 08].

La figure ci-dessus (figure 2.15) représente le modèle fonctionnel d'un tel capteur. Les fonctions surveiller, élaborer et décider sont essentielles. La fonction surveiller associée à la fonction élaborer va permettre de transformer des informations brutes en informations validées auxquelles sera associée une crédibilité maximale compte tenu de l'état du capteur à un instant donné. Cette fonctionnalité de validation est inséparable de la fonctionnalité acquérir. Elle est à la base du concept d'instrument intelligent.

2.6.1 Partie sensible du capteur

Le laboratoire de physique des matériaux de Nancy (LPM), spécialiste dans l'étude des propriétés électromagnétiques des matériaux et des capteurs fortement miniaturisés, a développé des compétences très avancées en électronique du spin et dans l'exploitation des propriétés magnétiques. Ce laboratoire, titulaire de plusieurs brevets, est aujourd'hui capable de réaliser des têtes sensibles de quelques micromètres carrés. C'est donc une équipe de ce laboratoire qui a développé la jonction magnétique à effet tunnel utilisée pour le démonstrateur. La photo suivante (Figure 2.16) montre l'exploitation la technologie des couches minces et l'approche LPM dans la construction des capteurs sensibles.

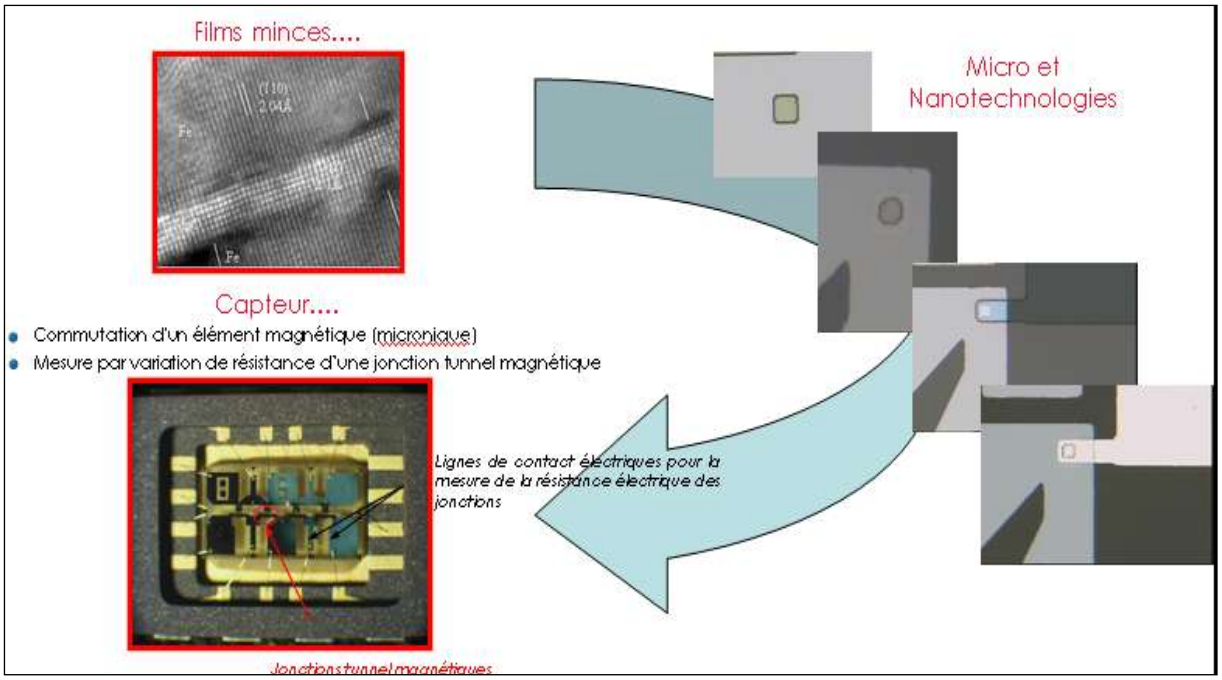


Figure 2.16 : Evolution technologique de l'élément sensible d'un capteur.

2.6.2 Unité de traitement analogique pour le conditionnement du signal

L'équipe d'InESS (Institut d'Electronique et de Solide de Strasbourg) possède une solide expérience dans l'exploitation des propriétés (linéaires et non linéaires) des capteurs magnétiques. Ses compétences en physique des composants ont permis d'élaborer une architecture répondant aux critères fiabilistes recherchés dans cette étude en collaboration avec le LPM. L'architecture retenue, hors redondances, sera étudiée en détail dans le chapitre 4 de ce mémoire.

Le signal issu de la MTJ (la tête magnétique sensible développé par l'équipe LPM) est filtré avant d'être amplifié et codé puis transmis à l'unité de traitement numérique conçue par le LICM. L'analyse du circuit de la partie analogique du capteur développée (Figure 2.17) permet de définir 4 grands blocs fonctionnels :

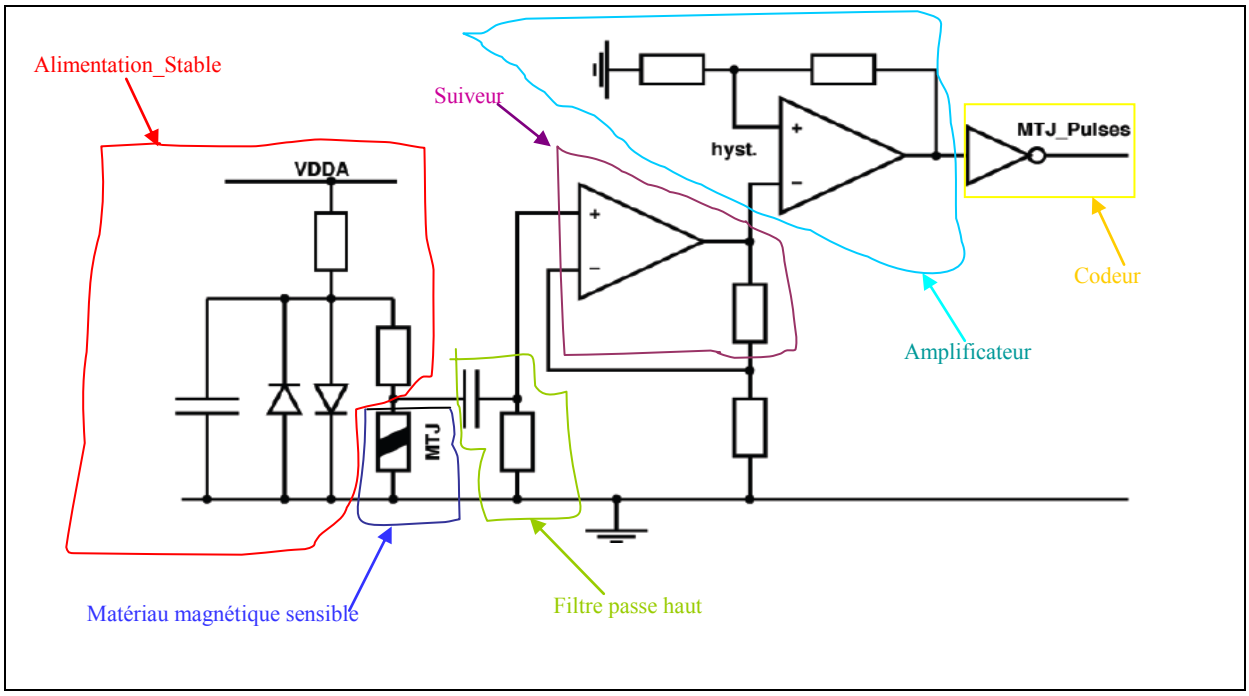


Figure 2.17: Schéma électronique de la partie analogique du capteur.

- Bloc alimentation.
- Le bloc MTJ : on exploite la propriété physique de l'effet hall des matériaux pour faire varier un rapport cyclique en fonction du champ magnétique à mesurer.
- L'étage de filtrage est constitué d'un filtre de 1er ordre.
- Les deux étage suiveur et amplificateur se réalise sur des circuits intégrés.

2.6.3 Unité de traitement numérique

L'ensemble des traitements numériques est réalisé par un cœur de processeur répondant aux spécifications des systèmes mécatroniques. Avec des objectifs et des propriétés complémentaires, on distingue diverses architectures de processeurs à la complexité de leur jeu d'instructions :

- **CISC** (*Complex Instruction Set Computer*) : grand nombre d'instructions (longueur variable), plusieurs modes d'adressage, surface importante, compilateur complexe.
- **RISC** (*Reduced Instruction Set Computer*) : limitation du nombre d'instructions (longueur fixe) et des modes d'adressage. L'accélération des accès mémoire et la gestion des sauts de fonctions imposent l'utilisation de mémoire cache et de registres supplémentaires.
- **MISC** (*Minimal Instruction Set Computer*) : jeu d'instructions réduit, limitation du nombre de ressources matérielles. Inclut la famille des processeurs à Piles (partagées entre le cœur de processeur et la mémoire).

La présence de processeur dans notre capteur nous a poussés à chercher une architecture de processeur qui réponde correctement à nos besoins. La simplicité du jeu d'instructions et de l'architecture matérielle ont constitué les critères de sélection pour l'application retenue (processeur réalisant des fonctions à haute exigence de sûreté de fonctionnement, intégré dans un capteur et donc à faible coût).

Les compétences du LICM ont permis de développer une architecture de type processeur à pile, gage de robustesse et de moindre complexité. D'autres exigences ont présidé aux choix d'un processeur à pile, parmi ces exigences nous citons :

- Economie de la ressource matérielle (utilisation minimale des composants) et rapidité d'exécution.
- Déterminisme de la machine: le microprocesseur à pile est basé sur le principe de la machine de TURING (séquence d'instructions déterminées), donc facile à modéliser par un processus markovien.
- Simplicité du codage.
- Aisance du développement d'un compilateur.
- Réduction de la taille du programme (minimisation du code par sub-routines).
- Limitation du nombre d'exceptions à traiter.
- Réduction de la taille architecturale par rapport à d'autres architectures (exemple : Von Neumann).
- Bonne adaptation de la robustesse et de la simplicité aux systèmes temps réel embarqués.

En raison de ces avantages, l'étude que nous présentons dans ce projet portera sur la conception d'un processeur MISC à Piles disposant d'un jeu de quatre groupes d'instructions : transfert de données, manipulation de mémoires, manipulation de registres, arithmétiques et logiques. Avec ses besoins en ressources mémoire minimaux (bonne immunité au SEU : Single Event Upset), son contexte fortement synthétique (peu de ressources mémoire à utiliser pour une sauvegarde de contexte), sa facilité et sa rapidité de développement (seule la démonstration de la méthode d'évaluation nous intéresse), le processeur à double piles (une pour les adresses de retour, une seconde pour le passage de paramètres ou l'évaluation d'expressions) convient parfaitement à notre projet (Figure 2.18). Les piles sont des mémoires avec une gestion de type LIFO (Last In First Out), elles sont complétées par une mémoire programme et une mémoire explicite (extension de la pile de données). Ce processeur disposera d'une pile de données (Data Stack) associée à une mémoire externe (Data Stack Memory) et une d'adresses (Return Stack) pour les interruptions, les appels à fonctions… associée à une seconde mémoire (Return Stack Memory). Ces piles sont gérées via des pointeurs (Data Stack Pointer et Return Stack Pointer) désignant les sommets (Top Of Stack et Top Of Return Stack) et sous-sommet (Next Of Stack et Next Of Return Stack).

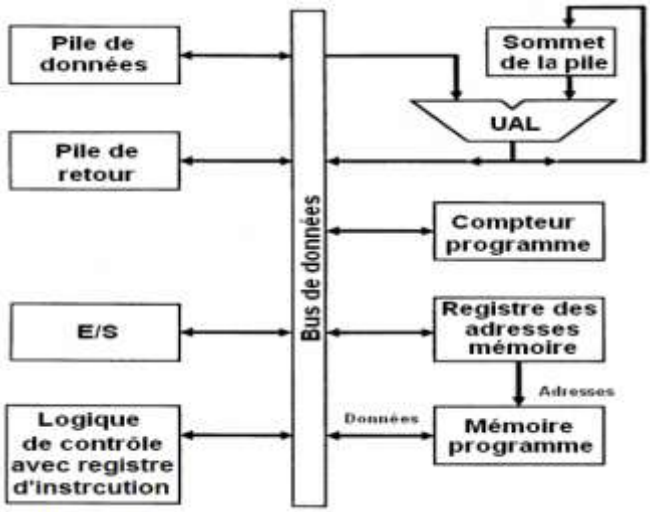


Figure 2.18 : Architecture de processeur à pile

S'il est acquis que la conception intégrée de composants mécatroniques requière une approche systémique pluridisciplinaire, l'intégration de critères de sûreté de fonctionnement dans leurs spécifications impose de nouvelles approches et interactions métiers. Ces systèmes intelligents et communicants, sont constitués de sous-systèmes interconnectés, dotés de circuits de contrôle et de boucles de rétroactions, associant composants matériels et logiciels. La recherche d'une augmentation de la disponibilité de ces systèmes, du moins de leur capacité à terminer une mission, a entraîné l'utilisation d'architectures de commande à électronique programmable dont les processus de décision sont distribués, engendrant des structures fiabilistes variables difficilement évaluables par les méthodes conventionnelles. L'établissement d'un modèle de comportement intégrant les modes de défaillance fonctionnelle et leur propagation, les modes de reconfiguration, d'activation des fonctions tests et d'élaboration du diagnostic, ne peut qu'être le fruit d'une collaboration interdisciplinaire. Cette obligation découle du besoin d'une définition de mécanismes de détection des défaillances efficaces et robustes, et d'une meilleure compréhension des mécanismes de vieillissement, nécessaires pour accorder un niveau de confiance suffisant aux modèles devant déboucher sur une meilleure maîtrise de la fiabilité des sous-systèmes analogiques.

Les choix architecturaux (matériel et logiciel) ont une influence sur les performances dynamiques et la robustesse des systèmes embarqués. Les temps de cycle de l'application et des processus de détection et recouvrement, éventuellement multiples en fonction des taux d'erreurs temporel, sont des paramètres notoirement critiques dans le cas des systèmes de contrôle-commande. La robustesse peut être évaluée via des techniques d'injection d'erreurs. On supposera un taux de couverture à la détection (matérielle) de 100% et une interruption sera générée à chaque occurrence d'apparition d'une erreur. Pour une sauvegarde périodique des sommets et pointeurs de piles, du compteur ordinal et des données traitées, l'exécution d'une procédure de recouvrement sur interruption (toutes les N instructions) à réaliser intensivement, le jeu d'instructions doit être adapté en conséquence. Une machine virtuelle est réalisée, elle supportera plusieurs types de Benchmark (programmes de référence), dont :

- Application de contrôle-commande : traitements logiques et arithmétiques avec entrées et résultats stockés en mémoire. Soit une centaine d'instructions.

- Tri à bulle (de dix variables stockées en mémoire) : utilisation de la mémoire (Figure 2.19).

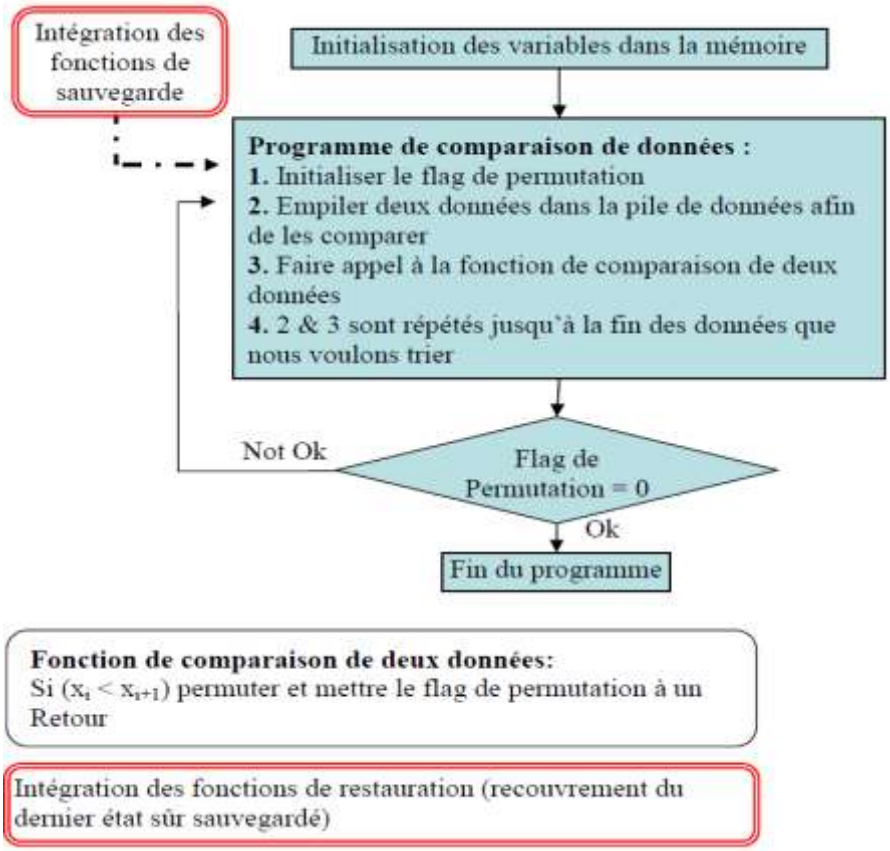


Figure 2.19 : Organigramme d'exécution de programme de Tri.

Un exemple de choix architectural entrant dans le cadre de la recherche d'un optimum dans le compromis coût/performances et pouvant être guidé par la simulation d'une machine virtuelle est le nombre de bus d'adressage. Une architecture à 2 bus (1 bus dédié mémoire programme et 1 bus mémoires piles et explicite) engendrera des durées d'exécution supérieures en moyenne de 30% à une architecture à 3 bus (1 bus dédié mémoire programme, 1 bus mémoire pile de données et 1 bus mémoires pile de retour et explicite), quelque soit le Benchmark.

L'analyse de la robustesse requiert plusieurs scenarii d'apparition de fautes : périodiques (1 erreur toute les N instructions) ou aléatoires (fréquence variable), isolée ou par salves (cas d'un démarrage moteur par exemple). Si le résultat n'est pas original, on peut vérifier qu'une salve d'erreurs de fréquence élevée ne permet à la routine de sauvegarde et de recouvrement de ne s'exécuter qu'après sa fin : la perte de temps n'est pas proportionnelle au nombre d'erreurs, mais au nombre de tentatives de recouvrement (Figure 2.20). Le cas le plus défavorable étant celui où la routine de recouvrement peut s'exécuter entièrement après chaque erreur. Une salve intervenant tardivement, lorsque les traitements logiques et arithmétiques sont réalisés, se révèle moins pénalisante que dans le cas d'une apparition précoce. Il demeure cependant impératif de pouvoir réaliser une première sauvegarde en début de cycle de traitement pour espérer un recouvrement consistant.

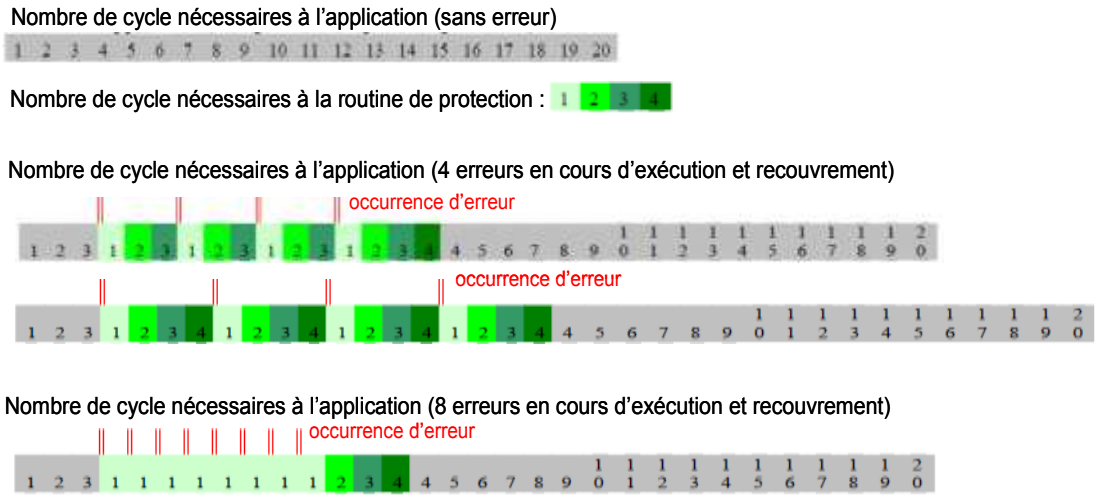
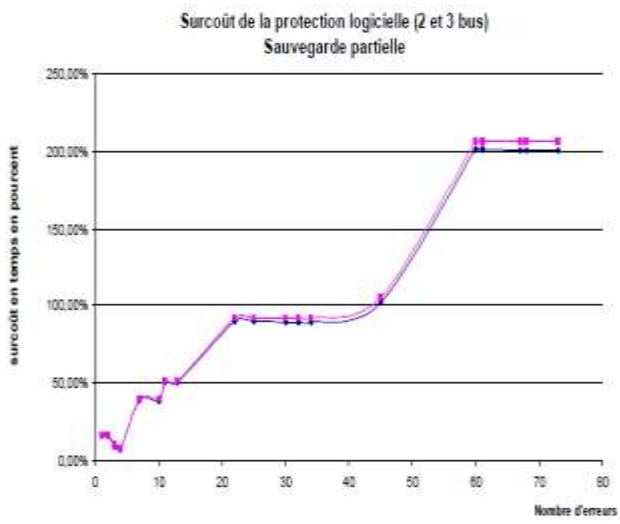


Figure 2.20 : Impact du nombre de recouvrement sur le temps d'exécution

Dans le cas du Benchmark Tri à bulle, 4 scénarii ont été traités : fréquence fixe avec 10 erreurs par cycle de traitement du programme (1833 cycles de traitement de l'application sans occurrence d'erreur), fréquence variable avec 73 erreurs par cycle de traitement du programme, par salves de 40 erreurs sur un intervalle de 200 cycles d'horloge, et par salves aléatoires jusqu'à 818 erreurs sur un intervalle de 818 cycles d'horloge.

Pour le second scénario (le plus défavorable), le surcoût temporel de l'intégration d'un mécanisme de sauvegarde périodique (sans procédure de recouvrement suite à apparition d'erreurs) est de 29%. Et dans l'hypothèse d'un recouvrement d'erreur toutes les 250 instructions, on atteint 47% (Figure. 2.21-a). Nonobstant la sauvegarde des données triées, la correction de 73 erreurs périodiques sur 1 cycle programme imposera une division par 3 de la fréquence utilisable (surcoût de 200%). En tenant compte des délais de sauvegarde des 10 données du Benchmark, le surcoût peut atteindre 1000% pour 249 erreurs ! Dans le cas d'une salve continue choisit sur 800 cycles d'horloge aléatoirement (scénario 4), le surcoût atteint 70%, sans sauvegarde des données triées, 103% avec sauvegarde (Figure 2.21-b).



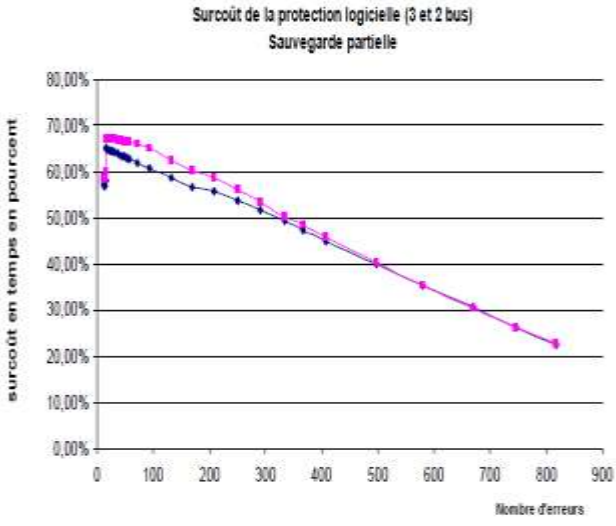


Figure 2.21a : Surcoût protection logicielle scénario 2 **Figure 2.21b :** Surcoût protection logicielle scénario 4

L'évaluation de ces surcoûts temporels permet de vérifier la compatibilité des capacités de traitement du système (impératifs de temps réel nécessaire à l'application vs nombre d'erreurs pouvant apparaître sur un cycle de traitement) et d'optimiser son durcissement en conséquence.

2.7 Conclusion

Dans ce chapitre, nous avons parlé de la tâche de chaque membre du projet de la Fondation CETIM. Nous avons commencé par la partie sensible jusqu'au traitement numérique de l'information.

Vers la fin du chapitre, nous avons montré le choix d'une architecture de la partie électronique programmé et de la partie traitement analogique de notre capteur intelligent.

Nous avons proposé également, une méthode d'évaluation quantitative qui permet une prise en compte accrue des phénomènes de défaillance simultanée, les causes communes de défaillance, les modes latents, l'arrêt intempestif. Cette méthodologie garde une place particulière à la représentation temporelle de l'évolution des probabilités calculées en fonction des contraintes externes.

Cette méthodologie permet à l'évaluation de probabilité d'avoir un mode de dysfonctionnement sous l'hypothèse :

- Description réaliste des profils de stress environnementaux.
- Modélisation correcte des processus stochastiques.
- Exactitude dans les règles de corrélation entre lois d'évolutions des modèles stochastiques et stress environnementaux.
- Détermination exhaustive et précise de probabilité de chaque mode parmi les six modes de défaillance définie par la suite dans le prochain chapitre.

La construction des listes caractéristiques par l'approche flux informationnel est la phase la plus importante. Elle permet de prendre en considération les mesures de stress environnemental, les contraintes architecturales en termes de dépendance temporelle entre les composants et le partage de ressources entre les fonctions de programme. Une application de cette méthode sur une architecture de processeur à pile sera le but du prochain chapitre. Nous montrons dans ce chapitre comment nous aboutissons à ces listes d'une façon systématique à partir des deux modèles, de haut niveau et de bas niveau.

La construction du processus stochastique permet de justifier l'intérêt de l'utilisation des chaînes de Markov non-homogènes et des règles d'évolution de leurs attributs vis-à-vis de l'évolution du stress de l'environnement au cours du temps.

Chapitre 3 :

Application de l‟approche flux informationnelle à l‟évaluation d‟un processeur

3.1. Introduction

La généralisation d'une approche originale basée sur le flux d'informations [Hami et al., 2005], nous permet de combler en partie le hiatus entre les évaluations des parties matérielle et logicielle. Cette méthode permet une description de la réalisation de fonctions logicielles sur une architecture matérielle reconfigurable. Les méthodes conventionnelles d'évaluation de la sûreté (RBD [Villemeur A., 1997], Réseaux de fiabilité [Hout M.et Apostolakis G., 2002], FTA [Vesely et al., 1981] se révèlent revêches à décrire aisément certains types de défaillances (défaillance de cause commune [CHARPENTIER P. 2002], modes latents de défaillance [Bier V. 1988], autotests, prise en compte des arrêts intempestifs…). Une partie de la difficulté d'évaluation de la sûreté de fonctionnement des systèmes complexes est liée aux difficultés de compréhension et de modélisation des interactions entre parties matérielle et logicielle [Belhadaoui et al., 2008-a]. La présente étude intègre aspects fonctionnels, dysfonctionnels et définition des modes de défaillance possibles d'être adoptés par le système.

Dans ce chapitre, nous appliquons, après adaptation, une méthode d'évaluation fiabiliste de systèmes critiques programmés [Camar et al., 2001], pendant sa phase de conception [Wirk N. 2003]. L'objectif principal consiste à évaluer la probabilité qu'une architecture matérielle du processeur exécute correctement ou incorrectement des instructions assembleur. Cette probabilité est calculée à partir des taux de défaillance d'exécution des instructions logicielles sur une architecture matérielle [Belhadaoui et al., 2008-a]. La méthode utilisée, hiérarchisable et modulaire, peut ainsi être utilisée pour l'évaluation de l'architecture (HW et SW) de processeurs pouvant inclure des mécanismes de tolérance aux fautes. Des mécanismes de tolérance aux fautes et de reconfiguration pouvant également être implantés au niveau de l'application et du système complexe étudié, on peut ainsi imaginer évaluer globalement des mécanismes de détection/recouvrement. Les aspects de propagation de l'information, inhibition de l'information, et les causes communes de défaillance devenant incontournables à l'échelle d'un système de systèmes (intégrant de plus des réseaux de communication). Les applications mécatroniques, associant composants numériques et analogiques, peuvent aussi être traités par cette méthode. Il est ainsi possible d'évaluer l'impact de mécanismes de tolérance aux fautes sur la fiabilité d'un microprocesseur [Lei et al., 2007]. Chaque instruction assembleur est évaluée individuellement selon la méthode décrite. Ensuite, nous supposons que l'ensemble d'instructions s'exécute séquentiellement, dans ce cas de figure nous calculons directement les probabilités pour des événements linéaires. En outre des mécanismes de détection/tolérance aux fautes sont également présent à ce niveau, nous appliquons les mêmes démarches de calcul sur un niveau hiérarchiquement supérieur afin de prendre en compte ces mécanismes.

Dans un premier temps, nous considérerons une architecture basique, sans dispositif de détection et de recouvrement de fautes. Le travail réalisé permettra d'analyser l'impact d'un mécanisme de tolérance aux fautes. Enfin, la méthode sera étendue à l'évaluation de la partie intelligente, l'objectif étant la validation, en termes de sûreté, de la conception d'un processeur. Cette validation consiste à quantifier des paramètres liés à la sûreté par une modélisation dynamique de propagation de l'information. Ce modèle dynamique peut être généré automatiquement à partir d'un modèle VHDL-RTL qui permet l'obtention du modèle de haut niveau de la méthode flux informationnel. La modélisation des ressources matérielles pendant l'exécution des instructions assembleur, permet une évaluation de l'architecture matérielle vis-à-vis du code de programme en termes de fiabilité, disponibilité et crédibilité de l'information.

Notre cas d'étude est un capteur se composé d'éléments sensibles électromagnétiques multiples associé à une électronique de traitement réalisant l'interface avec un processeur. Le choix de la simplicité de conception comme gage de sûreté [Robert C. 1992], a fait privilégier une architecture MISC (Minimal Instruction Set Computer) [Jallou et al., 2007]. Ceci permet d'avoir une compréhension assez fine des mécanismes d'échanges et de traitement d'informations internes. Sa structure interne inclut un processeur à deux piles gérées en mémoire externe pour ne pas avoir de restriction sur leurs profondeurs, un adressage par des pointeurs internes au cœur, des instructions à opérande de 8 bits, et un bus de données de 16 bits. L'interface entre la tête de mesure et le cœur du processeur est réalisée par un étage analogique. Les circuits numériques sont essentiellement sensibles aux particules ionisantes, généralement responsables de la modification inopinée de la valeur d'une information (bit-flip) stockée dans un point mémoire (bascules, RAM…). Les défaillances matérielles, notamment celle d'une porte logique (effet d'augmentation de courant) ou celle du à une sur-cadence (altération par augmentation de la température pouvant conduire à une défaillance totale) sont génératrices de fautes qui seront propagées lors du traitement ultérieur d'information. La conception de systèmes programmables implique une évaluation globale de l'ensemble matériel/logiciel.

3.2. Analyse fiabiliste d'un microsystème numérique

Toutes démarches de sûreté de fonctionnement s'entament par une analyse qualitative préliminaire des spécifications. Elle permet de circonscrire les contraintes dues au cahier des charges (analyse préliminaire des risques). Elle détermine ainsi des scénarii aboutissant à l'occurrence de défaillance. Cette étude se base sur différentes méthodes telles HAZOP, FMEA, ou d'outils de spécifications [Henley E.J.K. 1992], et permet la détermination et la sélection des modes de défaillance, ainsi que leur criticité [Belhadaoui et al., 2007]. Le but de cette phase est de classifier les modes de défaillance en catégories (modes de défaillance) selon leurs natures, leurs degrés de criticité, et leurs impacts sur l'architecture globale. Elle nous permet de caractériser les transitions entre les états (modèle de bas niveau sous forme d'automate d'état fini [Bolch et al. 2000]) par l'ajout d'attributs significatifs sur l'information propagée, qui figurera sur les séquences menant à défaillance. L'information correcte ou erronée se propage d'une entité à l'autre suivant l'ordre logique d'exécution d'une instruction. La défaillance d'un composant de l'architecture peut induire la propagation d'une information erronée. La défaillance de l'instruction et de la fonction peut être transitoire ou permanente, selon la nature de la défaillance signalée. Ce travail préliminaire permet dans un premier temps l'analyse qualitative, et dans un deuxième temps de quantifier, évaluer la probabilité d'apparition des modes (dys)fonctionnels.

Suite à l'analyse qualitative, la quantification des métriques de performance fiabiliste de l'architecture est réalisée à partir de l'identification des séquences (dys)fonctionnelles : mode de fonctionnement correct, fonctionnement erroné toléré, non toléré, pires cas… L'analyse préliminaire de notre architecture a permis de définir six modes de fonctionnement. Ces modes correspondent à un niveau de crédibilité de l'infomation délivrée par le capteur. En s'inspirant de la norme [IEC61508 1999], les modes suivants ont été définis :

Modes disponibles :

- **Mode 1**: mesure correcte, pas de défaillance détectée.
- **Mode 2**: mesure incorrecte, défaillance détectée mais tolérée (recherche de disponibilité).

Modes indisponibles :

- **Mode 3**: mesure incorrecte, défaillance détectée mais non tolérée (recherche de sécurité).
- **Mode 5**: mesure correcte et défaillance détectée (arrêt intempestif).
- **Mode 6**: absence de mesure (arrêt du système).

Modes dangereux :

- **Mode 4** : mesure incorrecte, défaillance non détectée (mode dangereux).

Les séquences de propagation d'erreurs générées à partir du modèle de bas niveau (langage marqué des automates d'états finis) permettent, à l'aide d'un modèle markovien non homogène (outil de calcul basé sur les arbres de défaillance) de quantifier et d'obtenir de manière analytique [Laprie J.C.,2004] quelques paramètres de sûreté de fonctionnement du système telle que la fiabilité ou la disponibilité. Elles permettent de caractériser l'état de fonctionnement du capteur selon les six modes définis précédemment.

3.2.1. Etude d'une instruction simple

a - Exemple de l'instruction DUP

Une modélisation préliminaire RTL se fait pour l'instruction DUP pour rendre facile sa modélisation selon l'approche flux informationnel [Belhadaoui et al., 2008]. Avant d'expliquer cette instruction, il convient de mentionner qu'il s'agit de processeur à deux piles. La pile utilisée pour le traitement de données est appelée pile de données (DS).Le haut de la pile (TOS) et l'élément suivant (NOS) correspondent respectivement au premier et au deuxième élément de cette pile. La deuxième pile est utilisée pour les adresses de retour des sous-routines, l'adresse d'interruption et les copies temporaires de données, elle est appelée retour pile (RS). Le haut de la pile de retour (TORS) correspond au premier élément de cette pile.

Les piles sont gérées dans une mémoire externe du processeur afin de ne disposer d'aucune restriction de profondeur de la pile. Elle est adressée par les pointeurs internes (pointeur de pile de données DSP et pointeur de pile de retour RSP). Étant donné que certaines caractéristiques d'architecture sont expliquées, nous pouvons clairement détailler l'instruction DUP.

L'instruction DUP copie le contenue de la pile (TOS) dans le sous-sommet de la pile (NOS). Le contenue de NOS, avant qu'il soit écrasé, il est empilé dans le troisième élément de la pile de données qui se trouve dans la mémoire pile de données (Mem_DS). Le pointeur de pile de données (DSP) est incrémenté a travers l'entrée de sélection du multiplexeur se trouvant en amont du DSP (Mux_DSP) afin d'allouer une nouvelle zone mémoire dans Mem_DS. Sous couvert de la réalisation correcte des fonctions de contrôle (pile mémoire de

données en mode écriture, MUX_DSP en position incrément, MUX_NOS en position lecture et connexion de TOS sélectionnée), l'exécution de la fonction DUP nécessite :

- DSP←DSP +1 „incrémentation du pointeur de pile de données pour empiler un nouveau élément"
- 3rd DS Elément (Memory value addressed by the new DSP) ← NOS „deuxième élément devient le 3rd de la pile des données "
- NOS ← TOS „duplication de l'élément sommet"

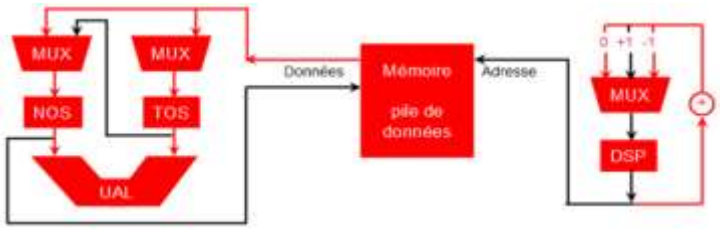


Figure 3.1 Chemin de données de l'instruction DUP

b- Exemple de l'instruction STORE

L'instruction STORE permet le stockage de la valeur de NOS à l'adresse mémoire stockée en TOS. L'instruction STORE consomme 2 cycles d'horloge en raison de la double décrémentation du pointeur de pile DSP. Ainsi, son modèle RTL est présenté en deux parties. Le 1er cycle de contrôle logique de l'exécution de STORE nécessite l'exécution de microcodes selon la séquence suivante :

- Pile de mémoire de données en mode lecture ;
- Mémoire explicite en mode écriture ;
- Signaux de contrôle de MUX_DSP dans la position telle que l'entrée (-1) est sélectionnée ;
- Signaux de contrôle de MUX_TOS dans la position telle que la connexion de bus de données de mémoire DS est sélectionnée.

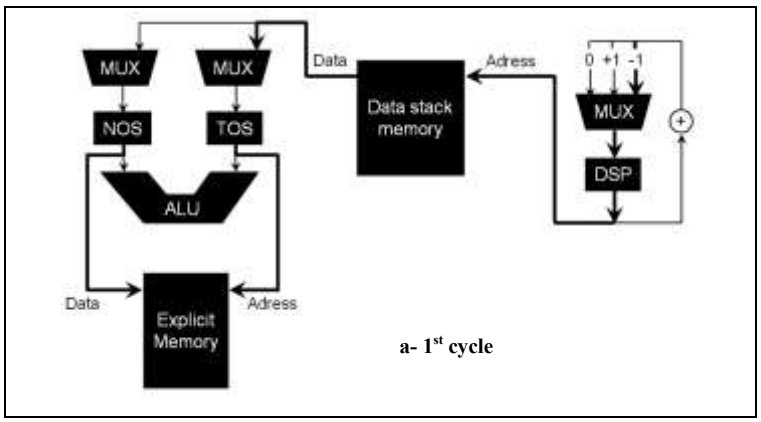


Figure 3.2.a : Modèle RTL de l'instruction STORE -1er front d'horloge

Au prochain front montant de l'horloge, la valeur de NOS est stockée dans la mémoire adressée à la position donnée par la valeur de TOS, le 3ème élément de la pile est stocké dans

le TOS. Le 2ème cycle de contrôle logique de l'exécution de STORE (lignes gras Figure 3.2 b) est caractérisé par :

- Mémoire explicite non valide;
- Signaux de contrôle de MUX_DSP tels que l'entrée (-1) est sélectionnée;
- Signaux de contrôle de MUX_NOS tels que la connexion du bus de données de la mémoire DS est sélectionnée.

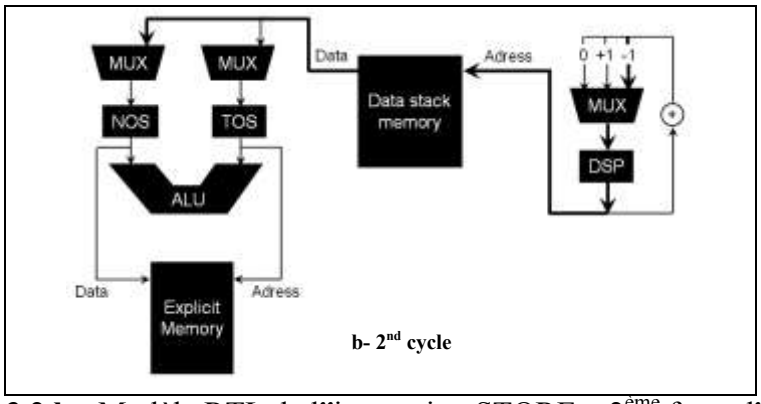


Figure 3.2.b : Modèle RTL de l'instruction STORE – 2ème front d'horloge

Au prochain front montant de l'horloge, le 4ème élément de la pile est stocké dans l'élément NOS. Ainsi, après deux cycles, toutes les opérations sont effectuées et l'instruction STORE est entièrement exécutée. La description du mécanisme d'exécution de chaque instruction permet de connaître les composants critiques mis-en jeux. Ainsi, on établit les transitions possibles entre les états internes du système pendant de l'exécution.

3.2.2 Modèle de haut niveau de l'approche flux informationnel appliqué sur l'instruction DUP

A partir des modèles RTL (*Register Transfer Level*) et des microcodes de chaque instruction, une description du mécanisme d'exécution est obtenue, permettant de déterminer les chemins critiques, ainsi que les transitions possibles entre états internes du système. Des mécanismes de détection de fautes peuvent être ajoutés (*Figure 3.3*).

Les exigences de sécurité et de qualité de service souvent requises en mécatronique nécessitent la mise en œuvre de techniques de détection et de recouvrement de fautes. L'implémentation (matérielle ou logicielle) de ces mécanismes nécessite une analyse de l'impact des fautes sur le comportement de l'architecture. La détection de fautes, dans l'état de l'art actuel, se répartit selon les contraintes de l'application, entre des techniques basées sur des solutions matérielles (Checker) ou logicielles (algorithme de vérification).

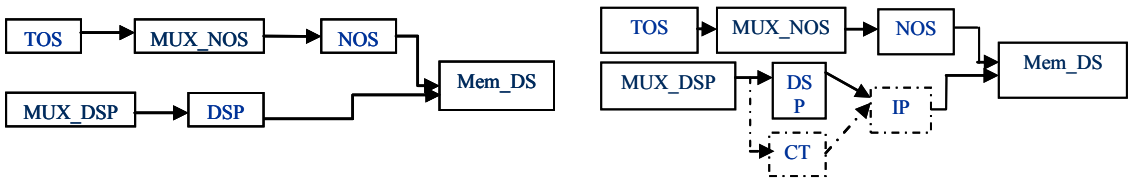


Figure 3.3 : Modèle informationnel de l'instruction DUP sans et avec Checker

Les problèmes les plus critiques au niveau de l'architecture RTL sont les problèmes d'aiguillage de Multiplexeur (élément MUX) et les problèmes d'incrémentation de pointeurs. La technique de *Checker* est capable de détecter les erreurs tant transitoires, que permanentes. Elle requière l'introduction de blocs fonctionnels de codage de l'information et de contrôle pour la surveillance [Carter W.C et Schneider P.R., 1968] [Anderson D.A. 1971]. L'incrémentation du pointeur est une opération critique, un test d'incrément est implémenté à la sortie du pointeur par *Checker*.

3.2.3 Génération des séquences « Exemple de l'instruction DUP »

La deuxième phase de modélisation consiste à traduire chaque entité sous-fonctionnelle du modèle de haut niveau en un automate d'état fini approprié décrivant son comportement dysfonctionnel. Cette phase permet de décrire la propagation des fautes au niveau de l'architecture du système et de définir les modes de fonctionnement résultants. On obtient, suite à cette étape, un ensemble de listes caractéristiques. Ces listes constituent un langage constitué de mots. Les mots de ces listes représentent les transitions entre les états des automates. Les transitions entre états représentent soit la perturbation de l'information, l'inhibition de l'information, ou l'absence totale de l'information. Les listes ainsi générées représentent de manière exhaustive tous les cheminements vers les modes de défaillance possibles. Les défaillances traitées sont les conséquences d'erreurs sélectionnées à partir de cas types souvent rencontrés et de stress environnementaux (voir paragraphe 2.4 du deuxième chapitre).

Les défaillances transitoires (bit-flip), qui peuvent changer la crédibilité de l'information, sont principalement modélisées par référence à des événements aléatoires. Ces événements peuvent être couplés, si nécessaire par un événement externe supplémentaire. En général, dans ce travail, nous traitons les trois types de bit-flip pour les composants numériques : SEU (Single Event Up-Set), MBU (Multiple-Bit Up-Set), SET (Single Event Transient). Les modes de défaillance matérielle (dM) relatives aux sous entités fonctionnelles sont également modélisés, dans ce dernier nous parlons du mode de défaillance permanent. Ainsi, il est possible de modéliser les modes suivants : état non réalisables, comportement correct, la situation de dead-lock, boucle infinie, les types interdits de communication entre les composants.

La probabilité de défaillance de chaque élément de base est calculée à partir des valeurs de la matrice de transition du processus markovien non-homogène de chaque entité matérielle. Après la traduction de l'entité fonctionnelle du modèle de haut niveau par l'automate d'états finis, nous utilisons le processus markovien pour calculer le taux de transition vers chaque état final. Cette méthode peut prendre en compte des défaillances à cause commune, les états de défaillance multiples, et les taux de défaillance variable. Les arcs entre les états d'un automate représentent les valeurs de taux de défaillance de l'information propagée entre ces états. Ces valeurs représentent également les éléments de la matrice du processus markovien. Les valeurs calculées dans le processus markovien seront remplacées par des lois de probabilité pour chaque élément de base dans un arbre de défaillance afin de chiffrer la probabilité de son élément sommet.

La figure 3.4 représente le modèle de bas niveau de l'instruction DUP. A partir de cet automate à états fini (graphe), est généré un ensemble de langages, image de séquences ordonnées. Les listes caractéristiques de ce langage décrivent tous les chemins possibles

menant à un état défaillant. Les listes générées vont permettre de quantifier la probabilité de défaillance des différents modes dysfonctionnels résultant d'un événement externe (exemple : bit-flip), ou interne (exemple : erreur matérielle, erreur logicielle…).

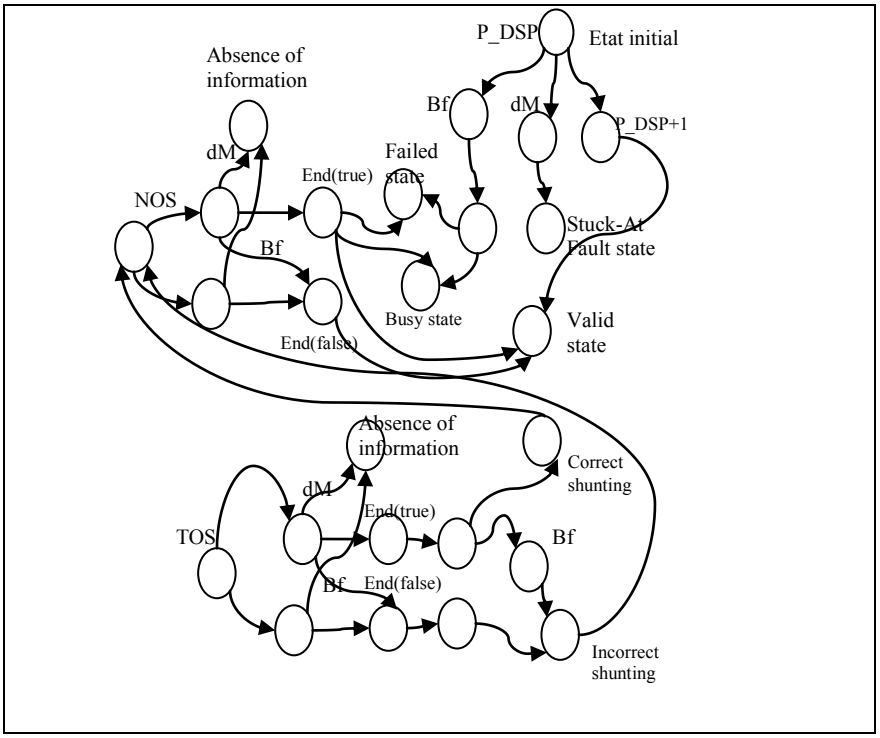


Figure 3.4 : Modèle de bas niveau de l'instruction DUP.

a- <u>Exemple de liste de fonctionnement nominale :</u>

$L_{T\_va\_Bien}$={P_DSP.P_DSP+1.TOS.End(true).Correct_schunting.NOS.End(True).Valid_State}

b- <u>Exemple d'une liste de la combinaison de défaillance :</u>

Ldef={TOS.Bf.End(false).Incorrect_Shunting.NOS.End(true).P_DSP.dM.Stuck_At_Fault_State.Busy_State }

Cette liste donne une combinaison de défaillance. TOS.Bf.End(false) signifie que l'information issue de TOS est incorrecte. Cette information erronée se propage le long de MUX (élément aval dans le modèle de haut niveau). Le mot Incorrect_Shunting traduit l'erreur d'aiguillage provoquée par un Bf (Bit-flip). Les mêmes explications peuvent être appliquées au niveau de NOS qui est une zone de mémoire de même nature que le TOS. Finalement, le mot P_DSP.dM.Stuck_At_Fault_State.Busy_State signifie qu'une défaillance de type matériel provoque une erreur et se traduit par un bit collé et par conséquence par un état de mémoire occupé.

La figure 3.5 montre le modèle de bas niveau obtenu pour l'instruction DUP dans le cas de la présence d'un Checker. Dans cette configuration, le problème d'incrémentation du pointeur est critique au regard du reste des erreurs. Un test d'incrémentation est implémenté à la sortie du pointeur par un élément du Checker. A partir du modèle de bas niveau on montre, après la génération des listes caractéristiques, que l'ajout d'un tel élément a un effet sur la probabilité d'occurrence de l'événement „erreur d'incrémentation".

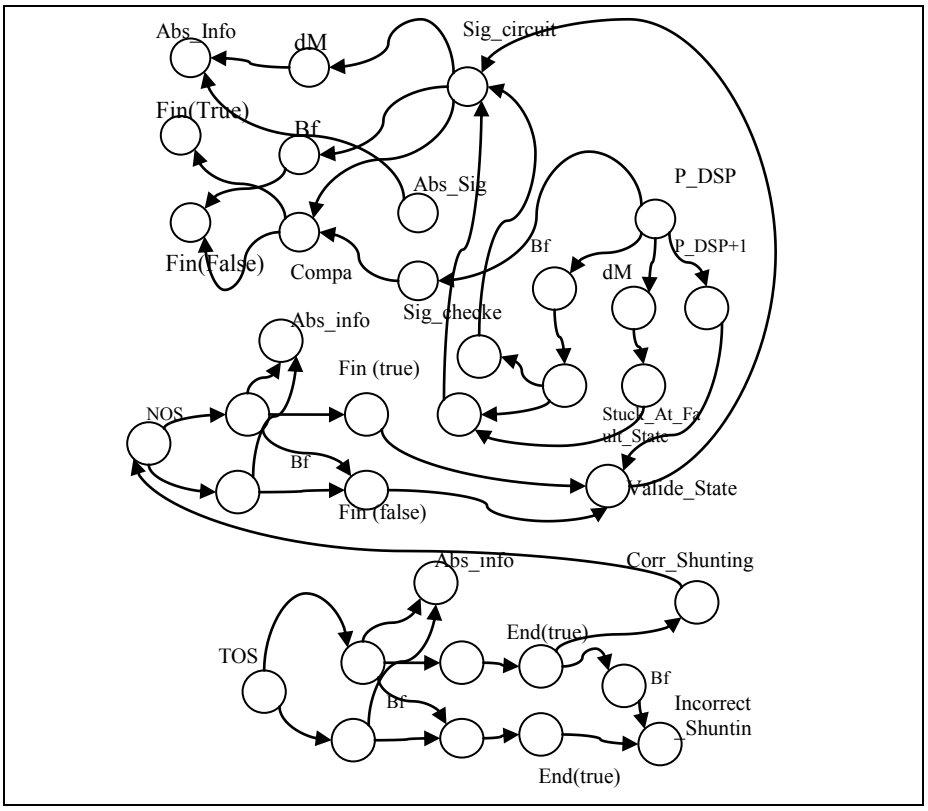


Figure 3.5 : Modèle de bas niveau de l‟instruction DUP en présence d‟un Checker.

L‟analyse qualitative fait apparaître l‟effet de sa présence sur la séquence de propagation de l‟information. Dans le cas de l‟élaboration de la séquence menant à l‟état de Défaillance (exemple :Etat_occup), nous obtenons :

c- <u>Mot d‟une liste sans moyen de contrôle « Checker »:</u>

$L_{défaillant}$= {TOS.Fin(true).Aig_corr.NOS.Fin(true).P_DSP.dM.Val.collé.Etat_occup}

d- <u>Mot d‟une liste avec moyen de contrôle « Checker »:</u>

$L_{défaillant}$ = {
TOS.Fin(true).Aig_corr.NOS.Fin(true).P_DSP.dM.Val.collé.Etat_occup.Compa.Fin(False);
TOS.Fin(true).Aig_corr.NOS.Fin(true).P_DSP.dM.Val.collé.Etat_occup.Compa.Fin(True)}

Dans ces deux extraits de listes, nous constatons que la probabilité d‟avoir le mot „TOS.Fin(true).Aig_corr.NOS.Fin(true).P_DSP.dM.Val.collé.Etat_occup‟ dépend de la réponse du Checker. Cette réponse a un impact sur la probabilité d‟occurrence puisqu‟elle est présente dans ces deux cas.

3.2.4 Génération des séquences « Exemple de l‟instruction STORE »

La même démarche est appliquée pour l‟instruction STORE. Les ressources matérielles utilisées dans cette opération sont explicitées dans le modèle de haut niveau à la figure 3.6.

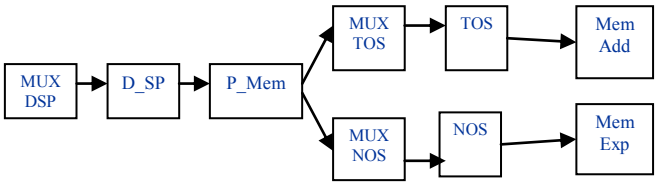


Figure 3.6. Modèle de haut niveau de STORE.

La figure 3.8 montre les chemins de données entre ces ressources matérielles au niveau inférieur du modèle. Ce modèle de bas niveau décrit la propagation de l'infomation erronée et l'état du système après chaque défaillance probable.

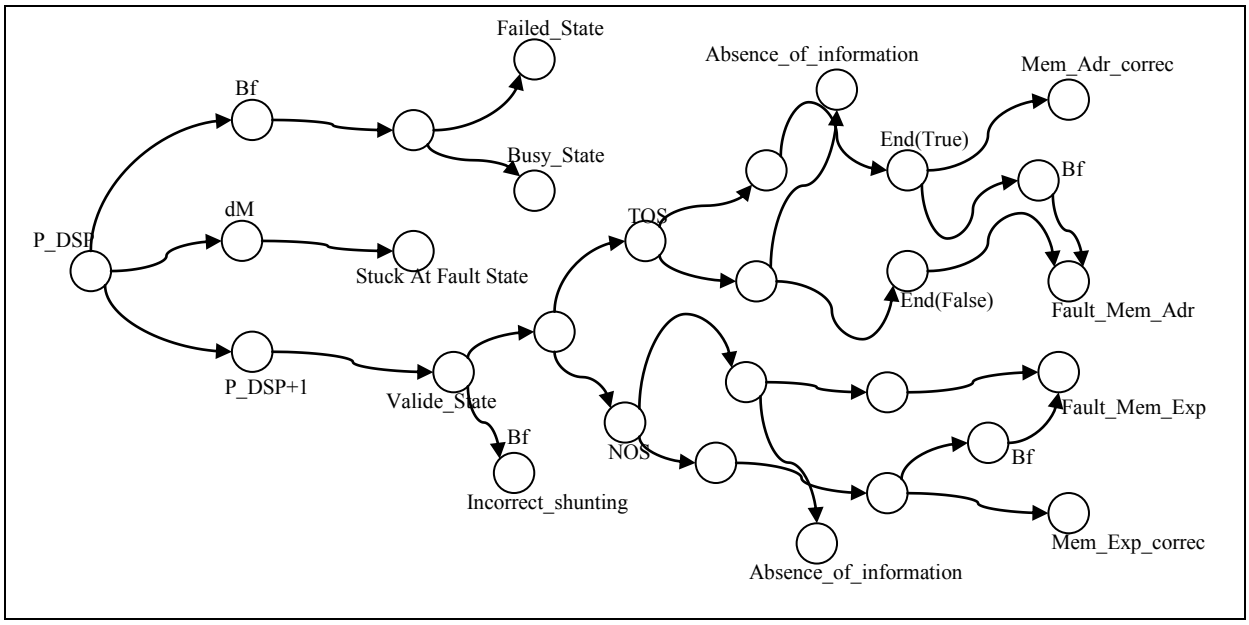


Figure 3.7 : Modèle de bas niveau de l'instruction STORE.

a- <u>Exemple de liste de fonctionnement nominale :</u>

$L_{Tous\_va\_bien}$ ={TOS.End(true).Mem_Adr_correc.NOS.End(true).Mem_Exp_correc.P_DSP.P_DSP+1.ValideState.DSMem1.TOS.DSM2.NOS}

b- <u>Exemple d'une liste de la combinaison de défaillance :</u>

$L_{Defaillance}$ = { TOS.End(true).Mem_Adr_correc.NOS.End(true).Mem_Exp_correc.P_DSP.Bf.Busy_State ou Failed_State}

La défaillance peut avoir lieu au niveau de n'importe quelle entité du système, à savoir les zones mémoires NOS et TOS, la mémoire explicite, l'adressage de la mémoire, l'incrémentation du pointeur de la pile, et éventuellement la mémoire de données. Dans l'exemple suivant, vue la complexité du système (processeur), on se limite au cas d'une erreur sur l'incrémentation du pointeur de pile à cause d'un événement de type bit-flip due à une particule chargée externe. Cette erreur va altérer le fonctionnement du pointeur et se traduit sous forme d'un changement de valeur du bit.

3.3. Descriptive d'évaluation quantitative du jeu d'instructions

Un des objectifs de notre travail est de quantifier les métriques fiabilistes de performance de micro-systèmes. La quantification de ces métriques de sûreté de fonctionnement (fiabilité, maintenabilité, disponibilité, sécurité) permet de caractériser le fonctionnement du capteur selon les six modes définis précédemment. Nous utilisons pour l'évaluation, une approche markovienne non homogène prenant en considération la dégradation due aux stress environnementaux. Une chaîne de Markov est dite homogène si sa matrice de transition associée A(t) est constante au cours du temps. Dans le cas inverse le processus markovien sera dit non homogène.

La possibilité de modéliser l'évolution temporelle de probabilité d'erreur pour une ressource matérielle non réparable par une chaîne de Markov, de pas d'itération t=T et de nature non homogène est indispensable. Dans ce cas de figure la matrice de transition varie en fonction de contraintes environnementales. Cette approche n'est pas aberrante, puisque nous avons précédemment supposé qu'une ressource matérielle conservait le même état sur une période de référence T (pas de calcul T, cf. chapitre 2). La probabilité de ces états est, par conséquent, constante sur ces périodes. La matrice de transitions entre les états du processus markovien [Jing et al., 2005] est constituée de valeurs des taux de défaillance des composants étudiés. Les taux de défaillance sont les transitions entre les états de l'automate (état d'entrée Ei, et de sortie Si) du modèle de bas niveau. La variation en fonction du temps des taux de défaillance permet de prendre en compte l'existence des stress environnementaux. L'utilisation correcte de processus markoviens, consiste en l'utilisation de quelques hypothèses de base :

- Les transitions correspondant à la défaillance d'une séquence de composants, dépendent du niveau d'usure qu'on suppose ici nul, et des contraintes environnementales auxquelles ils sont soumis.
- Le cycle présent ne tient compte que des erreurs matérielles du cycle précédent.
- Les contraintes environnementales doivent évoluer assez lentement sur un cycle, ce qui revient à dire qu'elles sont quasiment constantes.
- Chaque ressource matérielle ne peut présenter au plus qu'un mode de défaillance sur un cycle.

Il est néanmoins tout à fait possible de prendre en compte la dégradation progressive du système modélisé par l'ajout d'états supplémentaires. Nous considérons que le taux de défaillance d'un composant (taux de panne) résulte à la fois d'un niveau d'usure du composant et de l'exposition aux stress environnementaux. Nous proposons l'équation suivante portant sur le taux d'apparition de panne d'un composant, noté $\lambda i(t)$, comme étant égal, sous l'hypothèse d'une sollicitation effective du composant sur un cycle d'horloge T (un pas de calcul), à la probabilité d'apparition de panne sur l'intervalle [t, t+T]. Nous supposons sur un cycle d'horloge, l'existence d'une fonction Ω_c portant sur les stress environnementaux (température, pression, niveau de rayonnement, susceptibilité électromagnétique, présence de neutrons et de particules alpha) auxquels le composant est soumis. Ainsi le taux de défaillance a pour expression générale:

$$\lambda_i (t) = \alpha.\lambda_i(t+T) + \Omega_c (stress, T) \qquad [1]$$

Où la fonction α est liée au mode de sollicitation de la ressource (elle permet la prise en compte d'une usure progressive du composant; on ne la prendra pas en compte dans ce chapitre,α = 1). Certaines bases de données de taux de défaillances des composants comme [EADS/Thales 2004], Siemens [SN29500 2005], [MIL-HDBK-217F 1991] donnent les fonctions Ωc sous forme d'un produit de facteurs correctifs liés aux différentes contraintes environnementales :

$$\Omega c(stress, T) = F(T).F(P).F(R)\ldots\ldots \lambda ref \qquad [2]$$

Exemple d'application pour un composant électronique ; famille de semi-conducteur :

$$\Omega c(stress, T) = \lambda_{ref}. e^{(\frac{NT}{TJ})} e^{[(\frac{TJ}{TO})^{P} (\frac{S}{NS})]^{H}} \qquad [3]$$

Nous illustrons notre démarche par une étude du cas détaillé dans l'annexe B. Dans ce cas d'étude nous aboutissons à un taux de 8.18 10^{-5} h^{-1} pour la famille des circuits intégrés utilisés dans notre application. Les systèmes électroniques numériques imposent de s'intéresser aux causes des défaillances transitoires, et notamment aux changements de valeur des bits (bit-flip). Les causes principales sont souvent liées à l'effet électromagnétique, à la présence de particules Alpha, à des rayonnements cosmiques, et à des particules chargées plus généralement. L'état de l'art nous permet de constater que les neutrons atmosphériques ont des niveaux d'énergie suffisants pour faire basculer le contenu de cellules mémoires, ou même pour perturber les opérations logiques dans des circuits. En général, il y a trois types d'erreurs de type bit-flip pour un circuit numérique : Single Event Up-Set (SEU), Multiple-Bit Up-Set (MBU), Single Event Transient (SET). D'après l'état de l'art présenté en annexe B, sous certaines conditions environnementales de stress, nous obtenons le résultat suivant dans le cas d'une zone mémoire : 64.28 10^{-12} bit-flip/Cycle.

3.3.1 Exemple de deux instructions DUP et STORE

A partir du modèle de bas niveau (ensemble d'automates d'états finis) montré dans la figure 3.7, nous citons quelques listes caractéristiques pour chaque mode de fonctionnement (six modes définies). Cela signifie que pour chaque mode correspond une liste formée de plusieurs mots.

$L_{mode1} = L_{Tous\_va\_bien}$
={TOS.End(true).Mem_Adr_correc.NOS.End(true).Mem_Exp_correc.P_DSP.P_DSP+1.ValideState.DSMem1.TOS.DSM2.NOS}

De même, nous aurions L_{mode2}, L_{mode3}, L_{mode4}, L_{mode5} et L_{mode6}. Chacune est constituée de plusieurs mots de type :

L_{mode2} = {
TOS.End(true).Mem_Adr_correc.NOS.End(true).Mem_Exp_correc.P_DSP.Bf.Busy_State ou Failed_State, …mot_i ….}

Pour avoir la probabilité de chaque liste (probabilité d'être dans un mode parmi les six définies), nous utilisons un outil logiciel comme Arlia ou Gagrif de la manière suivante :

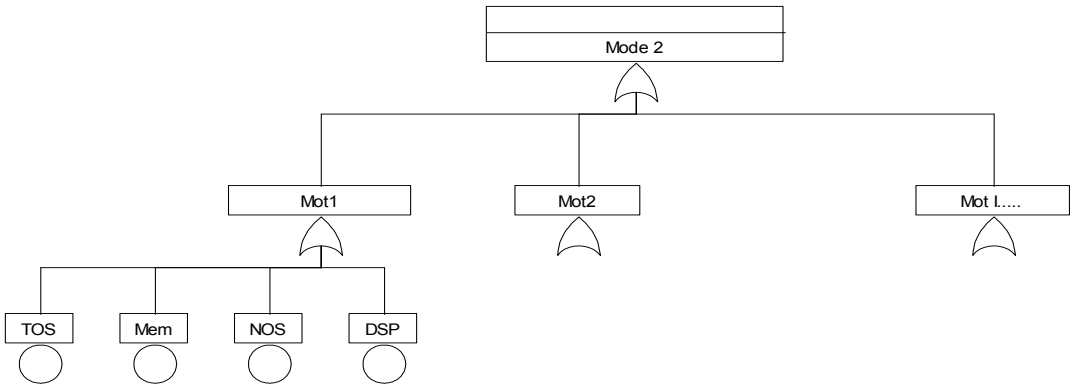


Figure 3.8 : Arbre de mode de défaillance 2 (recherche de disponibilité).

La défaillance (mode non correcte), c'est à cause au moins d'un événement ou une combinaison de défaillance représenté par un mot_i, cela traduit par des portes logiques {OU} entre ces mots dans l'arbre de défaillance. La défaillance d'une liste c'est à cause de la défaillance traduite par le mot1 ou bien le mot2 ou bien le mot_i…

Dans la défaillance d'un mot, si les composants sont en série, nous traduisons la relation entre les composant dans l'arbre de ce mot par une porte logique {OU}. Dans le cas où ces composants sont câblés dans une configuration parallèle, nous devons traduire la relation par une porte logique {ET}.

Chaque sous arbre de chaque mots qui constituent la liste d'un mode quelconque, est constitué des éléments de base. Ces éléments de base donnent une idée sur les ressources matérielles participant à cette combinaison de défaillance.

L'outil logiciel utilisé demande, avant de lancer les calculs, le taux de défaillance de chaque élément de base.

Dans le cas où le composant suit une loi exponentielle (figure suivante), l'outil demande le taux (paramètre λ). Puis il injecte ce taux pour calculer la fiabilité de l'élément sommet de l'arbre. Sachant que la fiabilité est : $R(t) = \exp(-\lambda t)$

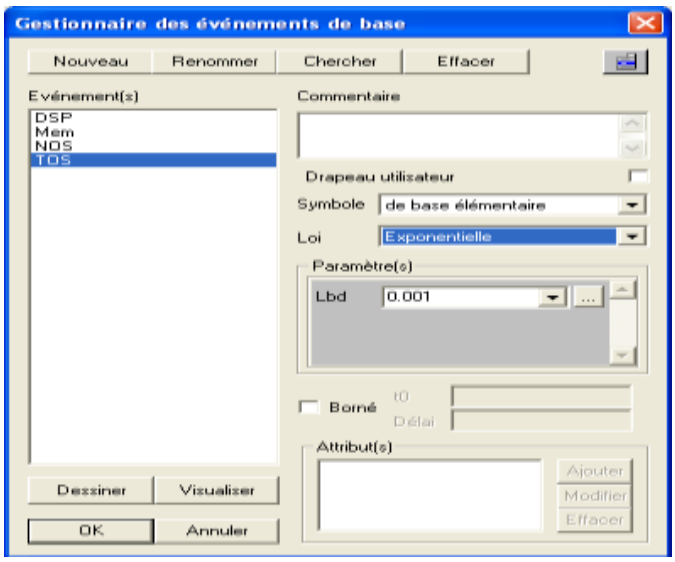


Figure 3.9 : Interface de création des événements de base

Dans le cas où les composants sont câblés en parallèle nous utilisons le principe suivant :

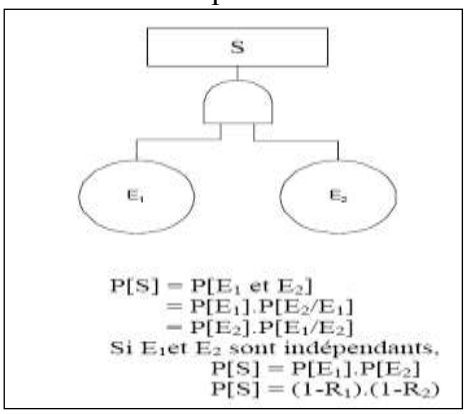


Figure 3.10 : Calcul de probabilité d'un événement en cas de deux composants en parallèle.

Dans le cas où les composants sont câblés en série, nous utilisons le principe suivant :

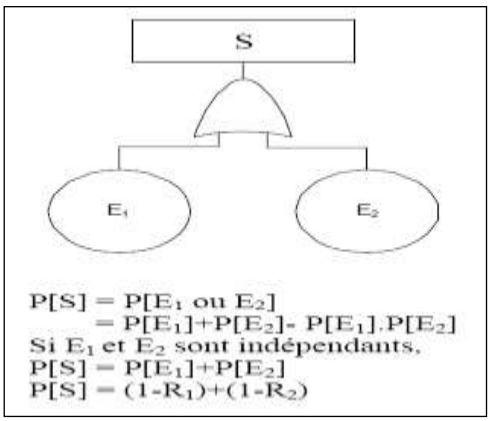


Figure 3.11 : Calcul de probabilité d'un événement en cas de deux composants en série.

Dans notre cas d'étude, nous estimons que nos composants électroniques suivent une loi de Weibull, en tenant compte du vieillissement de ces derniers.

Dans ce cas de figure, l'outil demande les paramètres (α, β et λ), comme le montre la figure suivante :

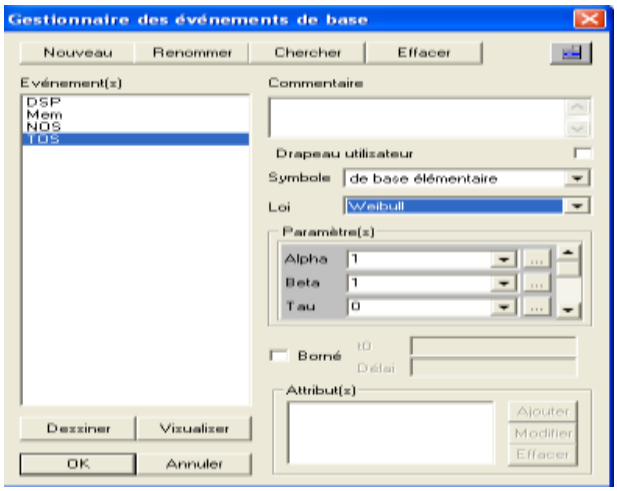


Figure 3.12 : Interface du logiciel Aralia en choisissant la loi de Weibull.

Les tableaux suivants présentent les probabilités du système de se retrouver dans un des modes de fonctionnement, en présence, ou absence d'un dispositif de détection et de recouvrement.

	Sans Checker			
Probabilité / Instruction	Mode 2	Mode 3	Mode 4	Mode 5
DUP	--	--	$38\ 10^{-5}$	--
STORE	--	--	$45.7\ 10^{-3}$	--

Tableau 3.1 : Résultat sans dispositif de détection

	Avec Checker			
Probabilité / Instruction	Mode 2	Mode 3	Mode 4	Mode 5
DUP	$72\ 10^{-3}$	$72\ 10^{-5}$	$8.99\ 10^{-6}$	$4.9\ 10^{-11}$
STORE	$11.3\ 10^{-4}$	$11.3\ 10^{-3}$	$4.49\ 10^{-5}$	$9.72\ 10^{-9}$

Tableau 3.2 : Résultat avec dispositif de détection et de recouvrement

A l'analyse de ces deux tableaux, on constate que cette technique de détection est plus efficace dans le cas de l'instruction DUP puisque la probabilité de non détection diminue lors de l'ajout de „Checker". De fait, l'instruction DUP utilise moins de ressources matérielles, d'où sa probabilité d'être dans l'un des modes 3, 4 et 5 est inférieure à celle de STORE [Belhadaoui et al., 2008-a]. L'intérêt d'implémenter les dispositifs de détection est de diminuer le risque d'avoir un système en „Mode 4" (existence de défaillance avec non détection), ce qui est favorable à l'augmentation de la crédibilité de l'information. La probabilité d'être dans le mode de fonctionnement „Mode 2" (défaillance détectée et tolérée) pour l'instruction DUP est plus importante que pour l'instruction STORE. C'est un résultat logique en rapport avec la complexité de l'instruction STORE face à l'instruction DUP, qui nécessite plus de ressources matérielles et de nombre de cycles d'horloge pour l'exécuter.

Chaque instruction assembleur est ainsi évaluée à partir du micro code et de l'architecture utilisée. L'évaluation d'une application logicielle revient à traiter séquentiellement une suite d'instructions assembleurs. Il est alors possible d'évaluer la fiabilité, la disponibilité, ou la sécurité d'une fonction de contrôle/commande et de mesurer l'impact de l'introduction de mécanismes de tolérance aux fautes sur ces métriques. Les conclusions de l'étude permettent, en association avec l'analyse des performances dynamiques, une correction au sein du processus de conception, si nécessaire. Le choix du type de mécanisme de tolérance aux fautes et les taux de défaillance utilisés a été fait de manière relativement arbitraire car notre seul but a été de montrer la pertinence de notre démarche. Dans tous les calculs réalisés dans ce chapitre, nous supposons que le moyen de détection de fautes du système est parfait.

La méthode est appliquée à l'ensemble du jeu d'instruction créé pour le processeur, les résultats d'une partie de ce jeu est présenté ci-dessous :

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
PUSH_DSP	$4.199\ 10^{-1}$	$4.552\ 10^{-3}$	$1.99\ 10^{-4}$	$1.11\ 10^{-6}$	$14.34\ 10^{-11}$	$75\ 10^{-12}$
PUSH_RSP	$3.882\ 10^{-1}$	$3.366\ 10^{-3}$	$0.14\ 10^{-4}$	$0.99\ 10^{-6}$	$12.23\ 10^{-11}$	$69\ 10^{-12}$
DLIT	$3.779\ 10^{-2}$	$1.002\ 10^{-2}$	$1.004\ 10^{-5}$	$3.2\ 10^{-5}$	$11\ 10^{-10}$	$155\ 10^{-11}$
STORE	$9.069\ 10^{-3}$	$11.3\ 10^{-3}$	$1.13\ 10^{-4}$	$4.49\ 10^{-5}$	$9.72\ 10^{-9}$	$30.4\ 10^{-11}$
DUP	$4.190\ 10^{-2}$	$72\ 10^{-3}$	$72\ 10^{-5}$	$8.99\ 10^{-6}$	$4.9\ 10^{-11}$	$55.7\ 10^{-11}$
FETCH	$1.944\ 10^{-1}$	$1.028\ 10^{-2}$	$1.003\ 10^{-4}$	$15\ 10^{-5}$	$132\ 10^{-9}$	$11.578\ 10^{-11}$
POP_DSP	$9.12\ 10^{-1}$	$0.85\ 10^{-3}$	$8.96\ 10^{-5}$	$2.1\ 10^{-6}$	$14\ 10^{-11}$	$34.4\ 10^{-12}$
POP_RSP	$1.89\ 10^{-1}$	$0.79\ 10^{-3}$	$11.2\ 10^{-5}$	$1.99\ 10^{-6}$	$17.9\ 10^{-11}$	$54.3\ 10^{-12}$
SWAP	$4.665\ 10^{-1}$	$5.01\ 10^{-3}$	$1.999\ 10^{-3}$	$1.2\ 10^{-6}$	$120\ 10^{-11}$	$277\ 10^{-11}$

Tableau 3.3 : Probabilités des instructions simples.

Le calcul de probabilité des modes de défaillances, notamment des modes 2 et 3 de chaque instruction, prends en compte la stratégie de tolérance sur les données sensibles de l'architecture de processeur à pile (DSP, TOS, NOS, RSP, TORS ….). Pour chaque élément sensible nous avons prévus deux fonctions de sauvegarde et de restauration. L'évaluation des probabilités des mode 2 et mode 3 de ces fonctions de sauvegarde et de restauration nécessite la connaissance des probabilités des éléments de base, car ces fonctions sont constituées d'instructions assembleur que nous devons évaluer aussi. C'est une relation de récurrence entre instruction-fonction.

3.4. Évaluation d'une application pour le mécanisme de recouvrement de fautes

La prise en compte des défaillances envisageables permet, vis-à-vis des contraintes de la mission, de prévoir les actions de recouvrement jugées pertinentes. En général, le développement (tenant compte des contraintes environnementales de la mission) devra fournir des moyens permettant d'intégrer les réactions aux défaillances (mécanismes de tolérance et de recouvrement). Le développement d'un système programmé est souvent entaché de fautes, lors de sa phase de conception et/ou de son implémentation, sans qu'elles soient corrigées lors de la phase de validation. Ces erreurs systématiques restent latentes lors de la phase d'exploitation jusqu'à ce qu'elles soient activées [Colwell R.P. et Lethin R.A., 1996]. L'origine d'une erreur est une faute, qui peut être dans un certain nombre de cas d'ordre conceptuelle ou physique (défaillance matérielle). L'origine d'une erreur est, en général, une faute humaine au niveau de la conception, ou une faute physique d'un composant matériel de ce système. L'activation d'une erreur d'un composant dans le système, peut affecter le service que rend le système. Le composant est dans ce cas déclaré défaillant. Par ailleurs, la défaillance d'un composant représente une faute auprès des autres composants avec lesquels il interagit. Les défaillances, les erreurs et les fautes, en présence de mécanismes de tolérance aux fautes, ne doivent plus entraîner la défaillance du système.

La détection de faute est une fonctionnalité importante embarquée dans les capteurs intelligents. La tolérance aux fautes, indispensable, est utile en cas d'impossibilité de recouvrement. La stratégie de recouvrement logiciel développée dans l'émulateur du processeur à pile, se base sur la sauvegarde du contexte de l'exécution du programme, d'une façon périodique (après une dizaine d'instructions). L'arrivée d'un événement porteur d'erreur, provoque une interruption de l'exécution, dans ce cas la méthode de sauvegarde prend le relais. La modélisation selon l'approche flux informationnel permet d'évaluer l'efficacité et la pertinence de ce type de stratégie.

La tolérance aux fautes représente la capacité d'un système à remplir sa fonction en dépit des fautes [Aiguo L. et Hong B., 2007]. Plusieurs moyens de la sûreté de fonctionnement peuvent être envisagés pour bâtir des systèmes sûrs de fonctionnement (i.e. l'élimination, la prévention ou encore la prévision des fautes). L'étude de la tolérance aux fautes englobe l'ensemble des techniques de conception et de fabrication des architectures qui peuvent continuer à fonctionner, même en présence de la panne de l'un de leurs composants. L'ensemble de l'architecture considérée continue de rendre le service attendu en présence de types de fautes prévues, grâce à des stratégies de recouvrement spécifiques. Un exemple concret est le cas de l'exécution d'un jeu d'instructions respectant l'intégrité des données en mémoire et le temps de traitement conforme aux spécifications [Qin X. et Jiang H., 2005].

Le Hardware/Software Co-conception [Wayne W. 2007] [Sosnowski J. 2006]est un domaine de recherche récent et très actif, générant des méthodologies de conception efficaces. Les premières tentatives datent du début des années 1990. Les méthodes de conception conventionnelles faisaient la distinction entre la conception de la partie matérielle (HW) et de la partie logicielle (SW). Le HW est la partie physique (UAL, transistors et connecteur…), tandis que le SW est la partie abstraite (sous forme de séquences d'instructions). La conception RTL nécessite une étape amont de validation nécessaire à la description du HW par le modèle VHDL. Suivant une modélisation hiérarchisée, l'approche flux informationnel participe à la mouvance HW/SW Co-conception. L'objectif de ces démarches est de mettre en évidence les points faibles de l'architecture et de détecter au plus tôt les chemins défaillants menant aux six modes de dysfonctionnement décrits précédemment.

3.4.1 Modélisation des fonctions de sauvegarde

La stratégie de tolérance fait appel à un ensemble de fonctions de sauvegarde des données jugées sensibles dans l'architecture du processeur à pile. Ces bouts de programme montrent que les blocs du programme exécutés sont des fonctions de plusieurs instructions assembleurs. Le nombre de ces instructions varie selon la boucle de programme considérée.

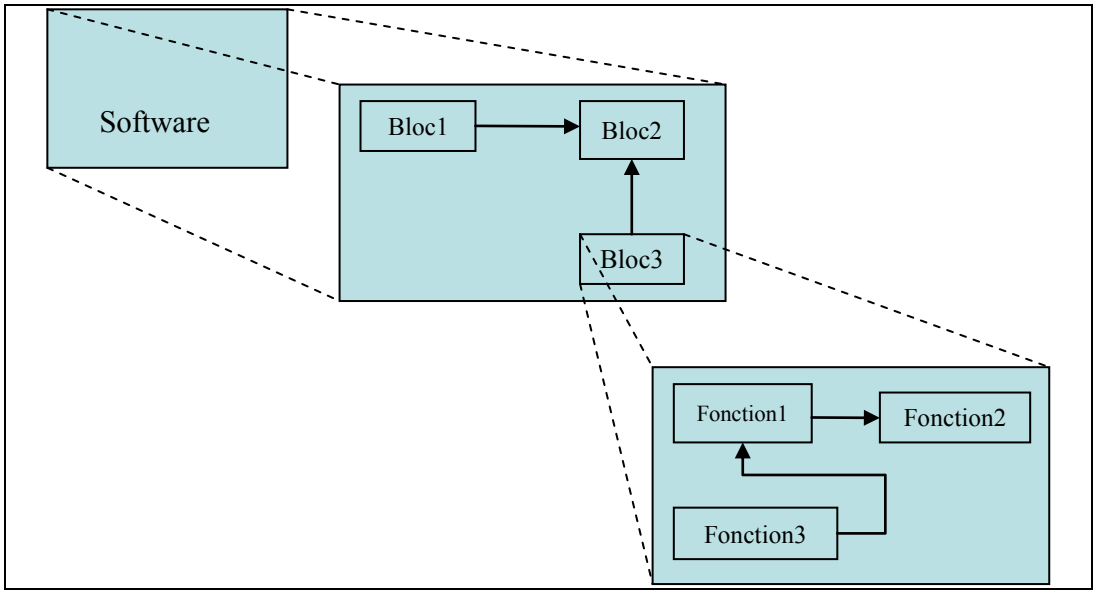


Figure 3.13 : Vue Globale d'une application logicielle.

Ci-dessous, nous présentons quelques fonctions de sauvegarde des éléments sensibles de l'architecture de processeur à piles. La stratégie de tolérance aux fautes consiste à enregistrer les éléments (DSP, RSP, TOS...) périodiquement via des fonctions de sauvegarde :

```
Sauvegarde_DSP :  Push_DSP
                  DLIT @_Sauv_DSP
                  STORE
Sauvegarde_RSP :  Push_RSP
                  DLIT @_Sauv_RSP
                  STORE
Sauvegarde_TOS :  DUP
                  DLIT
                  STORE
Sauvegarde_NOS :  SWAP
                  DUP
                  DLIT
                  STORE
                  SWAP
```

3.4.2 Modélisation des fonctions de restauration

L'événement redouté pendant l'exécution du programme se traduit par l'interruption de ce dernier et nécessite l'appel des données sauvegardées périodiquement (appel du contexte du programme) pour son recouvrement. La restauration des données est réalisée par les fonctions de récupération des données. Nous décrivons quelques fonctions de récupération utilisées dans la stratégie de recouvrement, il s'agit des fonctions de récupération des données les plus sensibles de l'application.

```
Récupérer_DSP :  DLIT @_Sauv_DSP
                 FETCH
                 STORE
Récupérer_RSP :  DLIT @_Sauv_RSP
                 FETCH
                 STORE
Récupérer_TOS :  DLIT @_Sauv_TOS
                 FETCH
Récupérer_NOS :  DLIT @_Sauv_NOS
                 FETCH
```

Les instructions utilisées dans ces boucles de sauvegarde et de restauration sont des instructions assembleur spécifiques au processeur à piles. Les plus usitées parmi ces instructions pour les fonctions de sauvegarde et de tolérance sont : PUSH_DSP, PUSH_RSP, DLIT, STORE, DUP, FETCH, POP_DSP, POP_RSP, SWAP.

Nous considérons la fonction de sauvegarde de la DSP «Sauvegarde_DSP» qui fournit un exemple intéressant de relation de récurrence fonction-instruction. Nous trouvons l'instruction DLIT (supprimer la valeur qui se trouve dans TOS) parmi les instructions de cette fonction, dont l'arbre de défaillance est donné par la figure 3.14. Nous remarquons l'existence de l'élément DSP dans cet arbre. La tolérance sur cet élément est assurée par la bonne exécution de la fonction de sauvegarde «Sauvegarde_DSP» et la fonction de restauration

„Récupérer_DSP". La relation de récursivité fonction-instruction entre l'instruction DLIT et la fonction de sauvegarde de DSP « Sauvegarde_DSP » est donnée par le logigramme suivant :

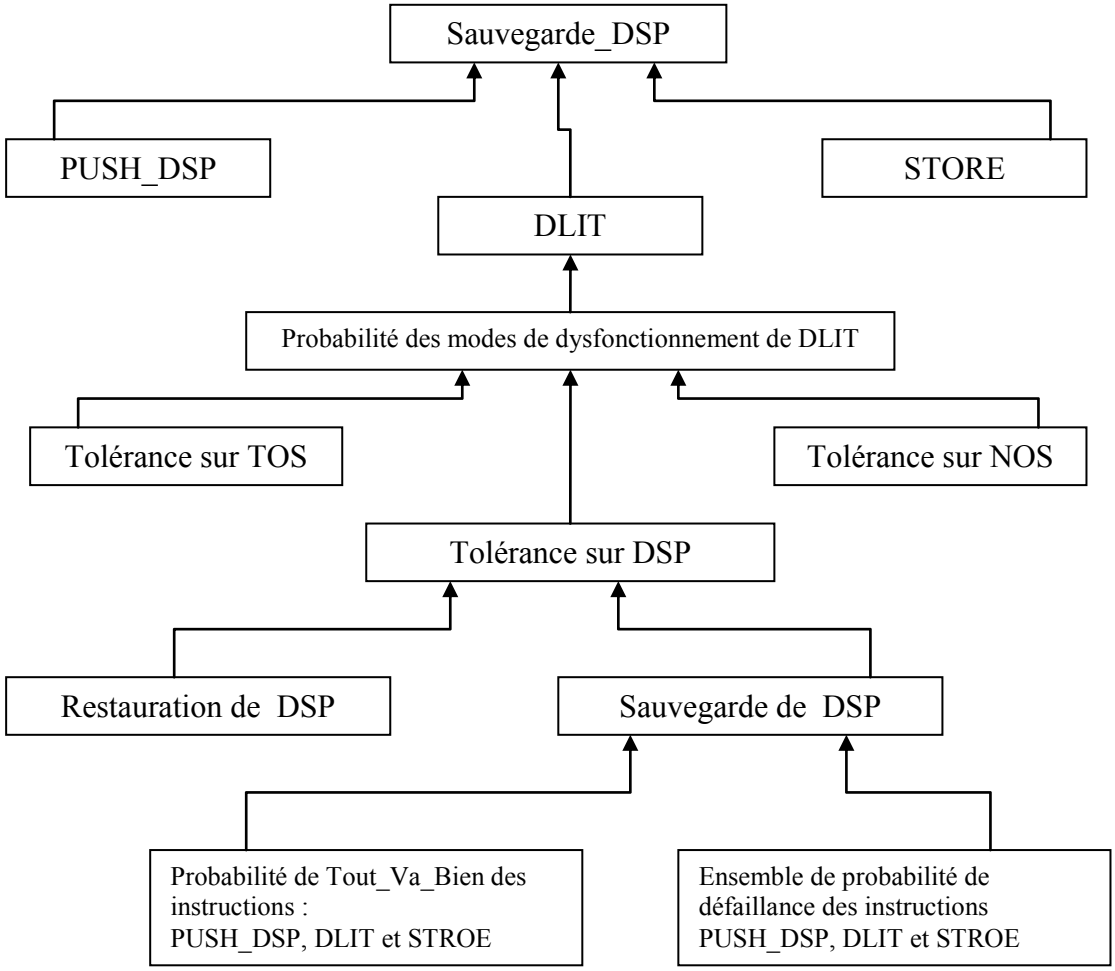


Figure 3.14 : Organigramme de calcul des probabilités de défaillance des zones 2 et 3.

A partir de la fonction „Sauvegarde_DSP", nous construisons le modèle d'instruction, suivi par le modèle de haut niveau de chaque instruction et finalement le modèle de bas niveau (figure 3.15). Pour chaque instruction, nous avons déjà établi le modèle de haut niveau, ce modèle caractérise les ressources matérielles du processeur à pile consommées pendant son exécution. Pareillement, le modèle de bas niveau substitue à chaque ressource ou composant matériel, un automate d'état fini permettant à son tour de caractériser les flux d'information entre ces différents composants. La construction de ce modèle dans sa globalité montre la corrélation entre ces instructions, et notamment au niveau de la génération des listes de défaillances de l'ensemble (trois instructions). Il reste compliqué dans le cas de plusieurs instructions, ce qui conduit au problème de l'explosion combinatoire et à la non maîtrise du modèle (non fondamentalement gênant grâce à l'automatisation possible du processus d'établissement des modèles, mais plus handicapant pour sa résolution analytique).

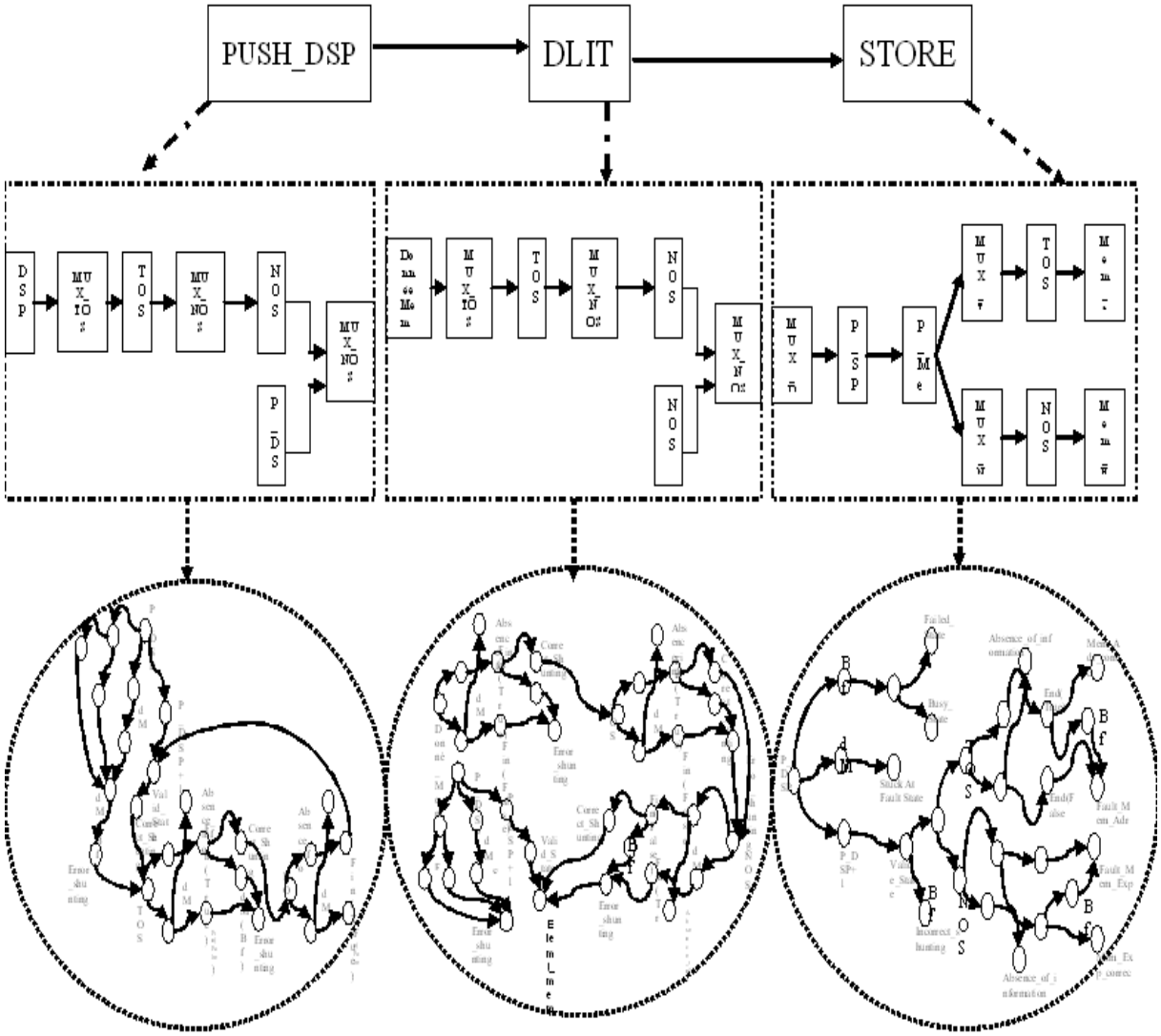


Figure 3.15 : Modèle de boucle de programme pour la sauvegarde de DSP selon les trois niveaux

Après cette nouvelle phase de modélisation, hiérarchisée, partant du modèle d'instructions, au modèle de bas niveau, via le modèle de haut niveau, se pose le problème du calcul de la probabilité d'échouer dans l'un des six modes de fonctionnement par l'utilisation des probabilités propres à chaque instruction.

La probabilité du mode 1 est la probabilité la plus simple à calculer. Il existe un seul chemin menant à ce mode de fonctionnement. La probabilité du mode 2, quand à elle, privilégiant la recherche de la disponibilité, caractérise le cas d'une mesure incorrecte, avec une défaillance détectée, mais tolérée. La valeur de cette probabilité pour la fonction «Sauvegarde_DSP» se calcule via la concaténation des listes caractéristiques d'une défaillance détectée et tolérée des trois instructions (PUSH_DSP, DLIT, et STROE). Cette concaténation des mots dans une liste permet de construire la combinaison finale dysfonctionnelle. La probabilité d'avoir cette liste se calcule à partir de sa traduction en un arbre de défaillance, de telle manière que chaque mot de cette liste compte pour un élément de base pour cet arbre. La subdivision de l'arbre global en sous-arbres de chaque instruction, permet de remplacer chaque sous-arbre par sa valeur de probabilité pour un mode de dysfonctionnement. La valeur globale de la probabilité

pour la fonction «Sauvegarde_DSP» est donnée par le calcul de la probabilité du sommet de l'arbre global.

Résultats des probabilités de défaillance des fonctions de recouvrement :

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
Sauvegarde_DSP	$1.439\ 10^{-1}$	$1.594\ 10^{-2}$	$1.23\ 10^{-3}$	$1.15\ 10^{-6}$	$55\ 10^{-11}$	$75\ 10^{-12}$
Sauvegarde_RSP	$1.173\ 10^{-2}$	$3.09\ 10^{-2}$	$0.06\ 10^{-3}$	$1.99\ 10^{-6}$	$32\ 10^{-11}$	$89.6\ 10^{-12}$
Sauvegarde_TOS	$1.44\ 10^{-2}$	$7.308\ 10^{-2}$	$2.193\ 10^{-3}$	$17\ 10^{-6}$	$12\ 10^{-11}$	$123\ 10^{-12}$
Sauvegarde_NOS	$6.829\ 10^{-2}$	$55.2\ 10^{-2}$	$11.24\ 10^{-5}$	$180\ 10^{-6}$	$19\ 10^{-11}$	$12\ 10^{-12}$
Récupérer_DSP	$6.662\ 10^{-1}$	$1.332\ 10^{-2}$	$2.107\ 10^{-3}$	$85\ 10^{-6}$	$45.6\ 10^{-11}$	$167.9\ 10^{-12}$
Récupérer_RSP	$7.662\ 10^{-1}$	$1.332\ 10^{-2}$	$2.107\ 10^{-3}$	$94\ 10^{-6}$	$67.4\ 10^{-11}$	$12.5\ 10^{-12}$
Récupérer_TOS	$1.58\ 10^{-3}$	$1.128\ 10^{-2}$	$1.99\ 10^{-4}$	$720\ 10^{-6}$	$12.4\ 10^{-11}$	$11.23\ 10^{-12}$
Récupérer_NOS	$1.58\ 10^{-3}$	$1.128\ 10^{-2}$	$1.99\ 10^{-4}$	$167\ 10^{-6}$	$45.5\ 10^{-11}$	$23.6\ 10^{-12}$

Tableau 3.4 : Résultats des instructions propres aux fonctions de tolérance.

Après l'évaluation des probabilités de toutes les fonctions de sauvegarde et de récupération, on peut tenter d'évaluer leur impact dans le cas d'une application réelle. Cette évaluation permet de valider la stratégie de recouvrement proposée.

3.5 Modélisation d'une application logicielle «Programme de Tri »

3.5.1 Programme sans prise en compte de tolérance

L'exemple de programme étudié dans cette partie est celui du tri de dix variables. À partir d'un tableau contenant dix variables, l'algorithme de ce programme consiste à lire les données du tableau deux par deux, à « swapper » en cas de désordre et remplir le tableau avec des valeurs ordonnées. L'initialisation des variables à comparer, entraîne le déclenchement du programme. Le programme initialise d'abord les variables dans la mémoire, via les instructions suivantes :

```
/*initialisation des pointeurs*/
DLIT 0x0090
D2R
DLIT 0x00B1
DLIT 0x00A1

/*initialisation des mesures à l'entrée*/
DLIT 0x0001     --donnée i
DLIT 0xe000     --adresse i
STORE
```

Après l'initialisation des variables, le programme de tri appelle la fonction de comparaison, cette fonction de comparaison se résume en 9 instructions assembleur :

- fct_comparaison_2_donnees:

```
//condition initiale : data1 data2 (le TOS est à gauche)
SUB // data2-data1
SIGNE // si data2>=data1 --> 0x00 sinon 0xFF
ZBRA 0x03 // Si TOS=0 --> Saut de 3 octets (vers DLIT 0x0001) sinon pas de saut
SWAP // permuter --> data2 data1
SBRA 0x07 // Saut 7 octets (vers RETURN)
DLIT 0x0001
DLIT 0XC000 // flag_permut<--1

// Ça se traduit par une fonction ou un autre CALL
STORE
RETURN
```

Ce bloc d'instructions se répète selon le nombre d'itérations du programme (fonction du nombre de variables à traiter). Dans notre cas, il y a dix variables à initialiser. Par conséquent il faut considérer ce bloc programme dix fois dans le calcul. Le bloc de base du programme est celui qui traite les variables, après leur initialisation. Pour réaliser la tâche de comparaison, ce bloc appelle la fonction „fct_comparaison_2_données à chaque fois qu'il rencontre deux variables non ordonnées. Ce bloc se décrit comme suit :

```
//flag_permut<--0 condition initiale
DLIT 0x0000
DLIT 0xc000
STORE
DLIT 0xe000
FETCH //mesure0 @mesure1 @mesure0
DUP //mesure0 mesure0 @mesure1 @mesure0
DLIT 0xe002
FETCH //mesure1 mesure0 mesure0 @mesure1 @mesure0
SWAP //mesure0 mesure1 mesure0 @mesure1 @mesure0
DLIT 0xe002
FETCH //mesure1 mesure0 mesure1 mesure0 @mesure1 @mesure0
CALL @_fct_comparaison_2_donnees
//si mesure0>=mesure1 --> rien (mesure1 mesure0 @mesure1 @mesure0) sinon
//permuter (mesure0 mesure1 @mesure1 @mesure0) ; le plus petit se trouve sur //le
sommet.
DLIT 0xe000
STORE
DLIT 0xe002
STORE  //Pour faire l'enregistrement dans les bonnes adresse (le plus petit est //stocké
dans 0xe000 (@mesure0)
```

A la fin de la comparaison de toutes les variables, le programme se clôt avec les instructions suivantes (il s'agit d'un test sur le flag de permutation pour vérifier la fin du programme) :

```
/*
Si (flag != 0) ALORS CALL fct_comparaison_10_donnees
FinSi
*/
```

```
DLIT 0XC000
FETCH
ZBRA 0x0003 //Si TOS=0 --> Saut, sinon pas de saut
LBRA @_fct_comparaison_10_donnees
HALT
```

Nous décomposons le programme global en blocs de sous-programme afin de simplifier le calcul des probabilités. Le programme global de Tri est constitué de plusieurs blocs de programme concaténés (bloc d'initialisation, fonction de comparaison, test de fin de programme). Nous supposons une hypothèse qui consiste en l'exécution d'instructions d'une manière séquentielle, les unes après les autres. Pour calculer la probabilité totale du programme, après le tri des dix variables, on considère le pire cas. C'est le cas le plus défavorable dans lequel les variables sont totalement désordonnées, le programme exécute alors le bloc Bloc_tri un maximum de fois :

	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6
Bloc_initiation	$4.894\ 10^{-1}$	0.0	$3.091\ 10^{-3}$	$77.3\ 10^{-6}$	$95\ 10^{-9}$	12.9910^{-12}
Bloc_tri	$1.27\ 10^{-3}$	0.0	$10.5\ 10^{-3}$	$145\ 10^{-6}$	$591.4\ 10^{-9}$	$123.5\ 10^{-11}$
Test_Flag	$9.17\ 10^{-2}$	0.0	$4.59\ 10^{-3}$	$12.1\ 10^{-5}$	$55\ 10^{-10}$	$55.7\ 10^{-12}$
Fonc_Comparaison	$58.49\ 10^{-2}$	0.0	$7.9\ 10^{-4}$	$123\ 10^{-5}$	$280.1\ 10^{-11}$	$312.12\ 10^{-11}$
Programme_tri	**$7.49\ 10^{-2}$**	**0.0**	**$6.67\ 10^{-2}$**	**$128.45\ 10^{-6}$**	**$17.8\ 10^{-10}$**	**$98.61\ 10^{-11}$**

Tableau 3.5 : Valeurs de probabilités du programme de Tri sans prise en compte de fonctions de recouvrement.

3.5.2 Programme avec prise en compte de stratégie de tolérance

Considérons maintenant l'exemple du programme précédent de Tri des variables. La stratégie de recouvrement se traduit par l'ajout des fonctions de sauvegarde et de recouvrement que nous avons modélisées au début de ce chapitre. Ces fonctions ont pour objectif de minimiser l'erreur sur les données sensibles lors de l'exécution du programme sur l'architecture de processeur à pile. Parmi ces données critiques nous citons DSP, TOS, NOS… Comme le montre l'algorithme du programme (figure 3.16), l'implémentation du bloc de code de ces fonctions se fait avant chaque itération du programme de tri, afin de garder en mémoire toutes les données de pointeurs jugées correctes.

La différence par rapport au programme précédent, c'est l'ajout de la stratégie de recouvrement. Cette stratégie est basée sur la sauvegarde des pointeurs avant chaque début de cycles de comparaison. En cas de problème pendant le déroulement du programme, la stratégie de recouvrement consiste à restaurer le dernier enregistrement de ces pointeurs. Après cette restauration le programme continue à s'exécuter avec des données sûres.

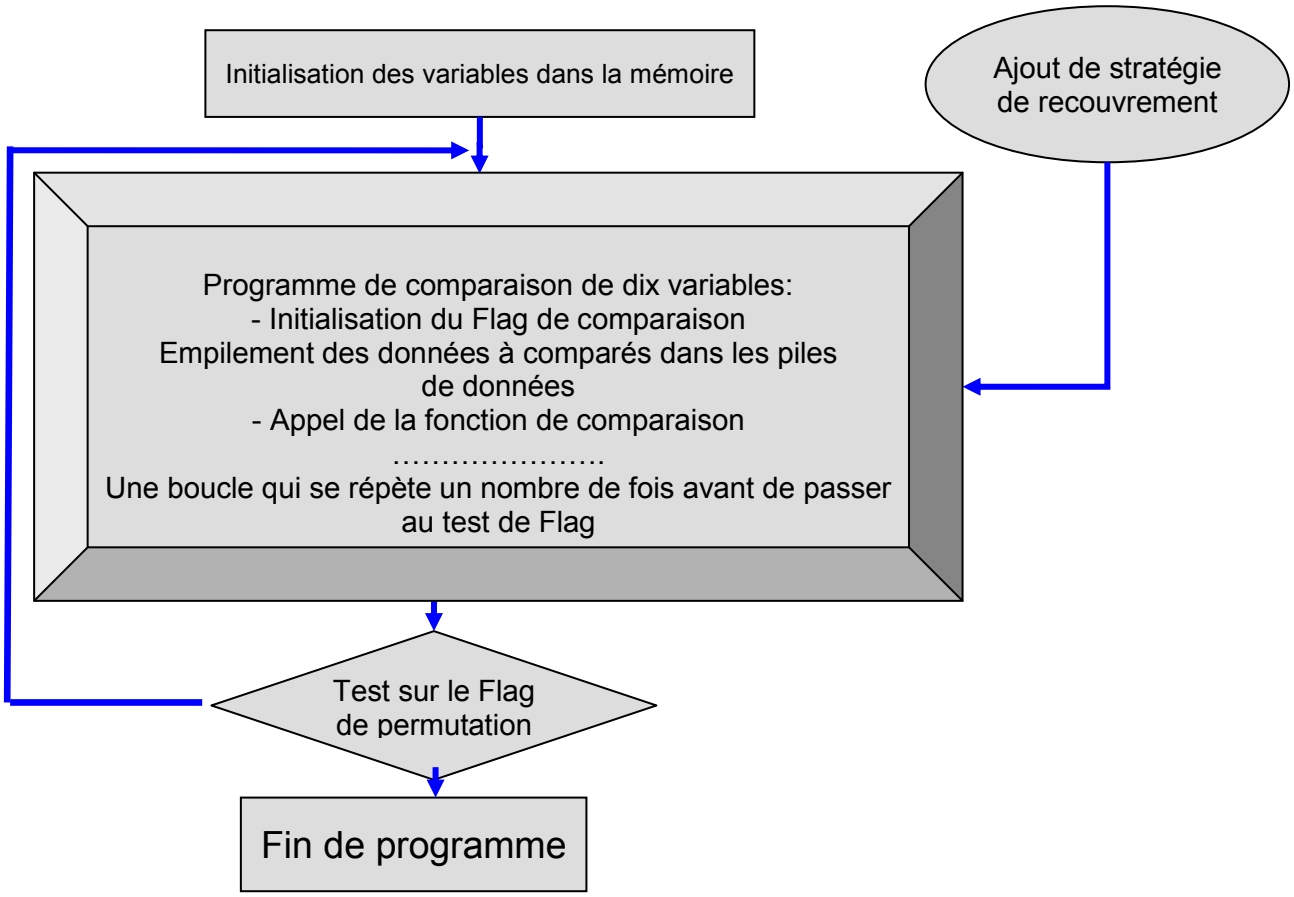


Figure 3.16 : Organigramme de l'algorithme du programme de tri de dix variables avec l'ajout de tolérance aux fautes.

La démarche pour calculer la probabilité de défaillance d'exécution de ce programme (avec recouvrement), est la même qu'au paragraphe précédent. L'ajout de la partie sauvegarde du contexte, après chaque aller-retour sur le tableau de dix valeurs, nécessite la prise en compte des valeurs de probabilité afin de pouvoir récupérer à tout moment les valeurs des pointeurs (TOS, NOS, DSP, TORS…). La probabilité de défaillance du programme sera influencée par la probabilité de défaillance des fonctions de sauvegarde et de récupération comme le montre l'arbre de défaillance figure 3.17 Il est important de connaître à chaque instant les valeurs des probabilités des six modes fonctionnel/dysfonctionnel pour l'ensemble du programme [Belhadaoui et al., 2009]. Ceci peut se réaliser par l'exploitation des résultats relatifs aux instructions simples, ainsi que l'exploitation des résultats relatifs aux blocs du programme. La probabilité de l'élément sommet (figure 3.17) s'obtient facilement par la connaissance des probabilités des éléments de base (sommets des sous-arbres). La subdivision de l'arbre de défaillance global en sous-arbres est une méthode pratique et efficace pour le calcul de la probabilité globale. Pour calculer la probabilité pour que le programme soit dans un mode dysfonctionnel quelconque, il suffit de remplacer les éléments de chaque sous-arbre (figure 3.17) par leur valeur de probabilité du même mode de dysfonctionnement.

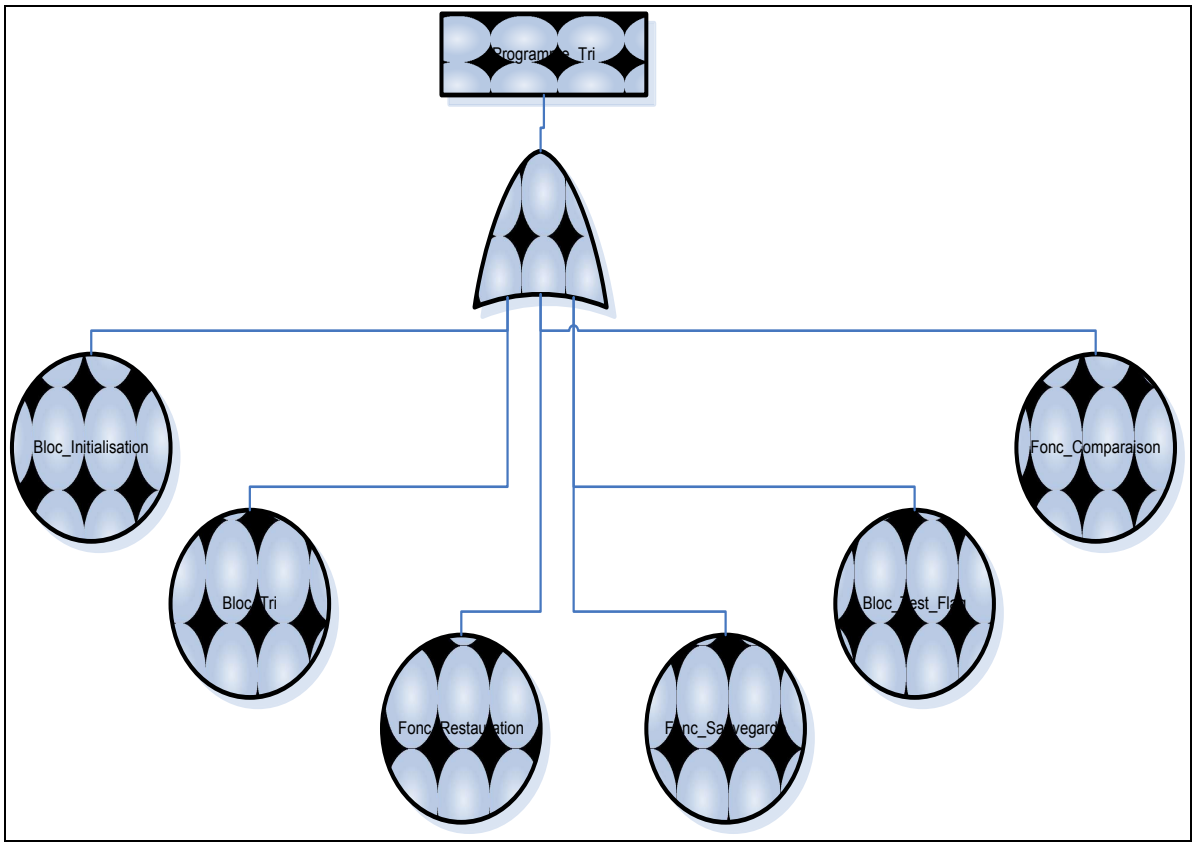


Figure 3.17 : Arbre de défaillance de programme de Tri en présence de recouvrement.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
Bloc_initiation	$4.894\ 10^{-1}$	$1.430\ 10^{-2}$	$3.091\ 10^{-3}$	$77.3\ 10^{-6}$	$95\ 10^{-9}$	12.9910^{-12}
Bloc_tri	$1.27\ 10^{-3}$	$1.57\ 10^{-2}$	$10.5\ 10^{-3}$	$145\ 10^{-6}$	$591.4\ 10^{-9}$	$123.5\ 10^{-11}$
Test_Flag	$9.17\ 10^{-2}$	$3.55\ 10^{-3}$	$4.59\ 10^{-3}$	$12.1\ 10^{-5}$	$55\ 10^{-10}$	$55.7\ 10^{-12}$
Fonc_Comparaison	$58.49\ 10^{-4}$	$3.70\ 10^{-2}$	$7.9\ 10^{-4}$	$123\ 10^{-5}$	$280.1\ 10^{-11}$	$312.12\ 10^{-11}$
Fonc_Sauvegarde	$7.44\ 10^{-2}$	$1.55\ 10^{-2}$	$2.2\ 10^{-4}$	$3.23\ 10^{-6}$	$7.45\ 10^{-9}$	$156\ 10^{-9}$
Fonc_restauration	$11\ 10^{-2}$	$0.33\ 10^{-2}$	$1.9\ 10^{-5}$	$54.3\ 10^{-6}$	$89.1\ 10^{-9}$	$117.5\ 10^{-9}$
Programme_tri	**$6.32\ 10^{-2}$**	**$1.57\ 10^{-2}$**	**$2.39\ 10^{-4}$**	**$55.8\ 10^{-5}$**	**$73\ 10^{-9}$**	**$45.91\ 10^{-11}$**

Tableau 3.6 : Valeurs de probabilités relatives au programme de Tri avec prise en compte de fonctions de recouvrement.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
Programme_tri (Sans prise en compte de toelarance)	**$7.49\ 10^{-2}$**	**0.0**	**$6.67\ 10^{-2}$**	**$128.45\ 10^{-6}$**	**$17.8\ 10^{-10}$**	**$98.61\ 10^{-11}$**
Programme_tri (Avec prise en compte de tolerance)	**$6.32\ 10^{-2}$**	**$1.57\ 10^{-2}$**	**$2.39\ 10^{-4}$**	**$55.8\ 10^{-5}$**	**$73\ 10^{-9}$**	**$45.91\ 10^{-11}$**

Tableau 3.7 : Résultats de comparaison de programme de Tri dans le cas de tolérance et sans tolérance.

Les graphes en général parlent mieux que les tableaux, nous traduisons les résultats, avant de commencer la discussion, au graphe suivant :

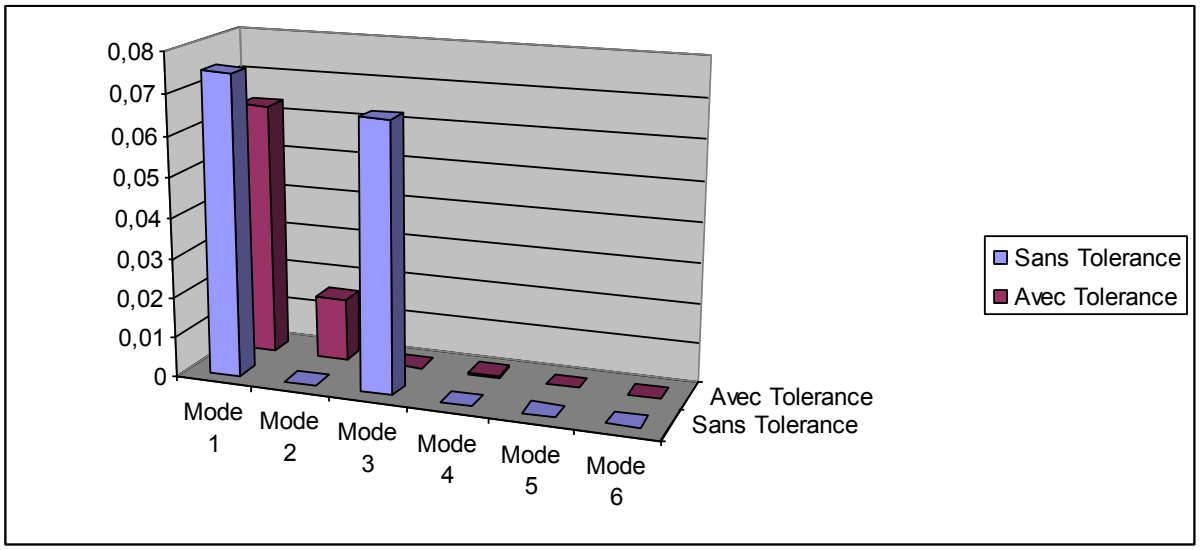


Figure 3.18 : Probabilité d‟être dans un mode de défaillance avec et sans présence de tolérance

Les avantages qu‟on constate après l‟ajout des fonctions de sauvegarde et de restauration (technique de tolérance) :

- Tendance d‟exécution du programme de Tri dans le mode 2 (probabilité nulle au départ avant l‟ajout de la fonction tolérance) et diminution de la probabilité du mode3.

- Les fonctions de recouvrement (sauvegarde et restauration), sont constituées par des instructions qui ont la caractéristique de favoriser le mode 2 par son exécution.

- Ces résultats numériques, permettant la confirmation de l‟intérêt de ce type de fonctions de recouvrement (ses instructions améliorent tendent l‟exécution vers le mode 2 le plus sûr), de cette manière nous avons pu augmenter la probabilité pour que le programme fonctionne dans le mode 2.

- Avant l‟ajout de la stratégie de recouvrement, les parties de programme „Bloc_initialisation‟, „Bloc_Tri‟, Bloc_Test_Flag‟ et „Bloc_Fonct_Comparaison‟ ont des probabilités nulles d‟être dans le mode 2, et une probabilité non négligeable pour que ces erreurs soient non tolérées (signifie le mode 3). Après l‟ajout de fonctions de recouvrement, la probabilité du mode 2 de ces blocs de programme devient non nulle et plus importante devant la probabilité du mode 3. Ceci dit que l‟effet de ces fonctions est incontournable, et les instructions qui constituent ces fonctions sont à l‟origine de l‟augmentation de la probabilité du mode 2 de ces fonctions et par la suite de tout le programme de Tri.

3.6 Conclusion :

L'approche de modélisation et d'évaluation d'une architecture de processeur à pile, présentée dans ce chapitre, met en évidence les points critiques de l'architecture en termes de fiabilité. Il s'agit d'une méthode de modélisation qui entre dans le cadre de l'étude fiabiliste nécessaire à l'évaluation des architectures de systèmes complexes. La génération de listes caractéristiques (séquences ordonnées) permet de mettre en évidence les chemins de données les plus critiques menant à la défaillance du système étudié. Elle permet de discriminer les faiblesses de l'architecture conçue, dès la phase de conception [Schoenig R. et Aubry J-F. 2006] et d'évaluer rapidement la pertinence des modifications et ajouts (mécanismes de détection et de recouvrement) envisagés et de leur impact.

L'objectif dans ce chapitre était de proposer une approche qui permette de vérifier les qualités fiabilistes d'une architecture lors de la phase de conception, complémentairement à la phase de tests et validation. Parmi ses intérêts, elle permet d'évaluer l'impact des mécanismes de détection et de recouvrement, ainsi que la tolérance aux fautes. Cette étude vise à garantir une certaine crédibilité de l'information fournie par le système.

Nous avons détaillé globalement la stratégie de la méthode flux informationnel. En utilisant la modélisation RTL-VHDL. Nous avons démontré qu'il est possible d'évaluer et de quantifier les métriques de la sûreté de fonctionnement. La méthode de modélisation se compose d'un modèle de haut niveau et d'un modèle de bas niveau. Le premier décrit l'ensemble des ressources matérielles utilisées par l'application logicielle. La seconde consiste à créer un automate d'états finis permettant de décrire les cheminements et les états possibles de l'information au sein de cette architecture.

Ainsi, grâce au langage généré par les automates d'état fini pour chaque entité sous-fonctionnelle, il est possible de déterminer toutes les combinaisons de défaillances possibles. Chaque combinaison est une liste qui caractérise l'état de tous les composants de l'architecture et par conséquence le mode de fonctionnement du système (un mode parmi six). Les listes caractéristiques sont transformées en arbres de défaillance dont en nous calculons la probabilité de l'élément sommet. Les matrices de transitions relatives aux automates d'état fini sont constituées des taux de défaillance de chaque composant. Pour calculer ces taux de transition en présence des différentes perturbations, nous utilisons un modèle de Markov non-homogène.

L'étude des cas sur les instructions assembleur, permet d'évaluer le niveau de performances atteint et les faiblesses structurelles inhérentes à ces instructions. Nous remarquons dans l'étude précédente que la probabilité de présence dans le mode 2 (défaillance détectée, mais tolérée) est supérieure à celle de présence dans le mode 3 (défaillance détectée, mais non tolérée), cela clarifie la pertinence d'adapter la technique de détection choisie. Nous pouvons également remarquer que la probabilité de présence dans l'état « défaillance détectée, mais tolérée » pour l'instruction DUP est plus élevée que dans le cas de l'instruction STORE. Cela signifie une meilleure tolérance aux pannes pour l'instruction DUP. C'est un résultat logique, l'instruction STORE étant plus complexe que l'instruction DUP, en plus l'instruction STORE a besoin de plus de ressources matérielles et de cycles d'horloge.

L'utilisation de l'approche flux informationnel sur les instructions assembleurs DUP et STORE est un exemple d'illustration. Les mêmes démarches peuvent être appliquées sur le reste des instructions assembleurs de ce type de processeur. Ces résultats sur les instructions

simples ont un intérêt dans l'évaluation d'une application logicielle constituée de plusieurs instructions (application embarquée).

L'objectif principal est d'intégrer des méthodes de la SdF dès le début du cycle de vie (phase de conception) du système, afin d'améliorer la disponibilité, la fiabilité et la sécurité. La cohérence des outils choisis et la pertinence de leur organisation sont des gages d'applicabilité d'une étude fiabiliste.

Jusqu'à ces dernières années, les solutions proposées pour l'analyse de la sûreté se révélaient souvent incapables de modéliser ce type de systèmes, qui intègrent composants matériels et logiciels, et leurs interactions. En revanche, l'approche flux informationnel montre son adaptation à évaluer ces architectures où la complexité fausse les résultats issus des approches classiques. L'étude de l'implémentation de stratégies de tolérance aux fautes a permis de sélectionner et valider les concepts et méthodes retenues à la conception. Cette démarche permet un développement éclairé de l'intelligence embarquée dans les capteurs et actionneurs en mécatronique.

Le travail présenté dans ce chapitre permet d'améliorer la conception des systèmes à électronique programmable, donc complexes, pour les applications critiques. En associant aspects fonctionnels et dysfonctionnels (travail commun avec le LICM), nous avons montré l'utilité et la possibilité du couplage entre étude dynamique (à partir d'un émulateur) et fiabiliste (en utilisant l'approche flux informationnel), par génération automatique du modèle de haut niveau modélisant le modèle RTL du processeur à piles.

Dans le cas d'une application réelle, le programme embarqué est constitué de différentes routines et son exécution s'effectue sous forme de tâches. Nous intégrons les fonctions de conduite et de sécurité utilisées dans les routines de traitement logiciel. Au niveau le plus bas, le code source est composé de fonctions et de sous-fonctions (niveau structurel). Dans cette hiérarchie, nous avons modélisé un programme de Tri présentatif d'une application réelle. Dans le modèle hiérarchique de la figure 3.17, l'information de données et de mesures, générée par modélisation, est propagée d'un niveau à un autre. Au premier niveau (niveau d'instruction), les entrées sont les valeurs des probabilités déjà calculées relatives aux instructions simples. À ce niveau de modélisation, nous représentons les différents blocs (modules) de programme. Ces blocs sont constitués des fonctions et des routines de plusieurs instructions. La représentation de ressources matérielles consommées par chaque instruction pendant l'exécution par le système est l'objectif du deuxième modèle de haut niveau (premier sous-modèle). Ce modèle est facilement traduit en automates d'états finis (deuxième sous modèle), afin de générer toutes les combinaisons de défaillances nécessaires et suffisantes au dysfonctionnement du système.

La modélisation du système sans stratégie de recouvrement dans un premier temps, et la modélisation avec protections logicielles dans un deuxième temps, a permis de comparer plusieurs approches et de vérifier l'impact et l'utilité de telles protections dans un environnement critique. La répartition des probabilités de chacun des modes de fonctionnement des blocs de programme a une dichotomie très différente après l'ajout des techniques de recouvrement. A un mode de fonctionnement 2 inexistant avant, l'ajout des fonctions de recouvrement (sauvegarde et restauration) change les données. Ces fonctions, constituées d'instructions qui favorisent le mode2, permettent d'entraîner le programme de Tri vers le mode 2, ceci par protection de ses données sensibles.

Grâce à notre collaboration étroite avec les autres équipes de consortium CIM‟tronic, nous avons pu mener des études de fiabilité et de robustesse des capteurs mécatroniques de manière plus rationnelle dès la phase de conception et contribuer au développement de la co-ingénierie dans le domaine des systèmes complexes hautement intégrés.

Chapitre 4 :
Evaluation fiabiliste de l‟architecture d‟un capteur mécatronique

4.1 Introduction :

Du point de vue fonctionnel, le système (capteur mécatronique) est constitué de différentes parties : une tête sensible, une partie analogique de conditionnement de signal, une partie numérique de traitement de l'infomation. Dans ce chapitre, nous allons montrer notre proposition de modélisation du système complet associant partie analogique (amplificateur, alimentation stabilisée, adaptation d'impédance…) et numérique. Grâce à la même approche basée sur le flux d'information, nous avons modélisé ces différents composants analogiques, passant par plusieurs niveaux d'abstraction, nous serons sûrs que l'information finale est générée sous forme de listes caractéristiques et sans perte. La même approche permet également de suivre l'information jusqu'à la partie processeur, et de proposer un modèle englobant à la fois la partie analogique et numérique.

Selon une étude qualitative, nous citons, les types de défaillance qui peuvent être rencontrés dans la partie analogique. La prise en compte de ces types de défaillance est une étape nécessaire dans la construction des scénarii et les combinaisons d'erreur possibles.

4.2 Etude fiabiliste de l'architecture d'un capteur :

Un des problèmes des systèmes industriels complexes est de prendre en compte, d'une façon réaliste dans leurs études de la fiabilité, les interactions dynamiques existant entre les paramètres physiques (comme la pression, la température, le débit d'un liquide, etc…) et le comportement nominal ou dysfonctionnel des composants [Dutuit et al., 1997]. En outre, nous pouvons ajouter le problème de la complexité liée à la présence d'une couche logicielle qui s'exécute sur l'architecture matérielle se traduisant par l'interaction matériel/logiciel.

Notre système mécatronique est un système hybride par nature, sa description prendra en compte l'aspect continue et l'aspect discret. La dynamique continue est représentée par des variables continues, la dynamique discrète est représentée par ses différents états et leur changement lié à l'occurrence d'événements. Ces deux aspects rendent la modélisation hybride indispensable.

Globalement l'étude des systèmes hybrides se fait selon deux approches :

- Approche intégrée prenant en compte au sien d'un même formalisme, les aspects continus et discrets. L'approche intégrée réunit des modèles continus comme les Bond Graph à commutation [Buisson J. 1993] et ceux issus de modèles à événements discrets comme les réseaux de Pétri hybrides [David R et Alla H. 1989].

- Approche séparée qui fait coopérer deux modèles différents : un pour les aspects continus et un pour les aspects discrets. L'approche séparée regroupe les modèles à base d'Automates hybrides, de Statecharts hybrides, de réseaux de Petri mixtes ou de réseaux de Petri Prédicats-Transitions-Différentiels (RdP PTD) [David R et Alla H. 1997] [Medjouji M . 2006].

La vision des automaticiens pour un système dynamique hybride consiste à décrire ce système par deux modèles en interaction : un ensemble d'équations d'états sur des variables continues et un automate à états finis sur un ensemble d'événements discrets.

Pour les fiabilistes, la fiabilité dynamique des systèmes (dynamic reliability) ou encore Probabilistic Dynamics [Devooght J. et Smidts C. 1992], est une nouveauté dans le domaine la sûreté de fonctionnement, définie dans l'état de l'art comme [Labeau et al., 2000] :
« Une partie de sûreté de fonctionnement qui étudie de manière intégrée le comportement des systèmes industriels complexes affecté par une évolution dynamique continue sous-jacente ».

Différentes approches traitant le problème de l'évaluation de la fiabilité dynamique sont proposées[Chabot et al., 2003] [Labeau et al., 2000] [Perez et al., 2007] [Laulheret R. et Cabarbaye A. 2005].

Un système mécatronique selon les automaticiens, est un système dynamique puisque la défaillance d'un composant peut ne pas avoir au cours du temps la même conséquence, comme il peut être considéré comme un problème hybride puisque la défaillance du système dépend de son état continu. Toute approche pour l'étude de la sûreté de fonctionnement de tels systèmes doit donc tenir compte de ces deux aspects (dynamique hybride).

4.2.1 Influences des fautes et critère de performance :

Le but de notre étude est d'analyser l'influence des fautes transitoires et permanentes sur notre système de capteur intelligent en particulier l'influence de la partie analogique à signaux continus et les conséquences de ses fautes (transitoires et permanentes) sur la fiabilité des mesures. Il nous semble judicieux en premier lieu de mettre l'accent sur l'influence des fautes transitoires et quantifier leur probabilité et, en second lieu, sur les fautes permanentes. On appellera faute de capteur tout événement capable d'entraîner une « erreur » sur la mesure. Une mesure étant la délivrance d'une information représentative d'une grandeur physique à un instant donné, nous pouvons distinguer trois catégories de fautes menant à l'absence d'information, à son altération et à son retard

La qualité d'un système capteur se définit en caractérisant l'ensemble des comportements admissibles de ce système : évolution des informations représentatives des différentes grandeurs en cas de défaillance, ou bien réaction face à une perturbation. Ainsi, un système est dit en équilibre quand il n'y a plus de variation des données qu'il délivre autres que celles liées aux grandeurs physiques. La qualité du système peut s'exprimer par sa capacité à atteindre l'équilibre. Plus précisément, dans le cas d'un changement de grandeur, il s'agit de quantifier la capacité à atteindre l'équilibre désiré (régime permanent) au bout d'un temps fini. Généralement trois paramètres sont importants pour un capteur : le dépassement, le temps de réponse à une entrée en échelon et la stabilité. Il s'agit de concevoir un capteur avec un minimum de dépassement et un temps de réponse aussi court que possible et de prouver que le système reste stable.

4.2.2 Modes de défaillance de la partie analogique du capteur :

- **Les défaillances en valeur** : dans ce cas de figure le capteur envoie des données erronées. Par exemple, un capteur de température peut, suite à un défaut de fabrication, délivrer une valeur fixe indépendante de la valeur de la grandeur physique. Ce type de défaillance est constaté dans les REX (retour d'expérience). Il est également imaginable que la mesure de la température soit momentanément modifiée par une perturbation électromagnétique, dans le cas d'un capteur numérique.

- **Les défaillances temporelles :** dans ce cas le capteur envoie les données en dehors des limites temporelles prescrites. Elle peut se produire à chaque sollicitation, mais souvent elle peut se produire aléatoirement.

- **Les défaillances par arrêt :** les capteurs n'envoient plus de données (silence) ou une valeur constante en permanence égale à la dernière valeur valide.

- **Les défaillances incohérentes ou défaillances byzantines :** dans le cas où le capteur est lié à deux régulateurs, les défaillances de capteur peuvent être perçues différemment par les régulateurs. Il est possible par exemple que le capteur envoie deux valeurs différentes aux deux régulateurs [Poledna S. 1995-b].

Il est possible aussi que des combinaisons entre ces différents modes de défaillance se présentent. Par exemple, les valeurs d'un capteur peuvent être à la fois erronées en valeur et décalées dans le temps. De nombreuses techniques de tolérances aux fautes sont employées afin d'assurer le fonctionnement correct du système global même en présence de défaillances des capteurs, comme la réplication du capteur ou la relecture du même capteur qui permet surtout de tolérer des fautes transitoires.

Si on définit la mission du système capteur comme étant la mesure d'une grandeur physique avec une qualité de service (ou qualité de contrôle) définie par un certain nombre de critères : dépassement borné, temps de réponse minimaux, et tendance à la stabilité, ce système doit être capable de détecter une erreur sur la valeur mesurée et capable de corriger cette erreur. Alors tout comportement du système en dehors de ces spécifications doit être considéré comme une défaillance de celui-ci.

Les sources de défaillances que nous avons envisagées étant probabilistes, cette défaillance du système revêt en conséquence un caractère probabiliste que l'on pourrait assimiler au concept de fiabilité. On définit la défaillance par critère comme étant la probabilité qu'un critère ne soit pas vérifié, par exemple la défaillance par dépassement aura lieu si à un moment donné la sortie dépasse la consigne d'un écart spécifié. La défaillance de système entier aura lieu si au moins une défaillance sur un des critères se présente. Il serait alors idéal d'identifier la relation existant entre fiabilité du système et probabilité des défaillances aux multiples causes de défaillances évoquées.

Ces défaillances comme nous l'avons vu font que le calcul des paramètres de performance et à plus forte raison la détermination de non satisfaction des critères (ensemble de critères définies dans le cahier des charges) sont difficilement réalisables d'une manière analytique avec les méthodes conventionnelles. C'est pourquoi nous proposons une approche de calcul de la probabilité de défaillance sur chaque critère basée sur le flux d'information.

4.2.3 Défaillance par dépassement :

La défaillance par dépassement d'un capteur consiste à fixer un écart maximum entre la donnée mesurée et la réponse idéale de ce capteur pour une grandeur physique. Si à un moment donné le dépassement en présence de fautes transitoires excède l'écart considéré, on détecte une défaillance par dépassement. On peut justifier physiquement ce choix car dans de nombreuses applications industrielles le dépassement de certaines variables même transitoirement peut avoir un caractère dangereux. Citons par exemple le courant électrique

dans un composant d'électronique de puissance (phénomène d'avalanche), la tension électrique aux bornes d'une charge (arc électrique), le couple mécanique sur un arbre de transmission (rupture de l'arbre), etc… [Rony et al., 2008]

4.2.4 Défaillance par temps de réponse et de précision :

La défaillance par temps de réponse est étudiée avec la même approche que la défaillance par dépassement : on fixe un écart maximum $T_{réponse}$ entre le temps de réponse en présence de perturbations et le temps de réponse idéal du système. Si à un moment donné l'écart en présence de fautes transitoires excède $T_{réponse}$, on détecte une défaillance par temps de réponse. On peut justifier ce choix puisque le temps de réponse caractérise la durée du phénomène transitoire qui dans de nombreux cas est source de perte énergétique. De telles pertes doivent être systématiquement évitées lorsqu'elles risquent par exemple de réduire dangereusement l'autonomie énergétique d'un système, une problématique non traité dans ce cadre du travail effectué par [Rony et al., 2008].

On cherche ici à identifier des situations où les perturbations ont une tendance à rendre le système infidèle. Pour définir la défaillance de précision, on surveille les réponses successives de mesure en présence de ces fautes transitoires. Si on observe des réponses différentes sur la même mesure, on constate que l'on est dans une situation de tendance à la non précision et on déclare une défaillance de précision.

Dans le cas général, une étude antérieure doit précéder la phase d'évaluation de cette probabilité pour décider du nombre de mesures inacceptables, ce paramètre dépend du contexte de l'application. Dans cette étude on a supposé que ce paramètre est une donnée pour l'étude de la sûreté de fonctionnement.

4.3 Etude préliminaire et analyse des modes de défaillance :

4.3.1 Détermination des contraintes :

Les listes des composants ayant été utilisés, on devra procéder pour chacun d'eux à l'analyse des modes de défaillance possibles. Ces modes ayant été déterminés, on calculera les contraintes auxquelles les composants sont soumis, soit à partir des documents de calcul du projet, soit à partir de mesures si elles sont possibles.

Pour des pièces à usage électrique, les contraintes sont relatives à la tension, au courant, à la puissance, la température, etc …, auxquelles elles sont normalement soumises ; pour des pièces mécaniques, ce sont les forces, vibrations, chocs, frottement et la température.

On devra ensuite déterminer un indice de contrainte par comparaison aux contraintes normales prévues par les constructeurs et auxquelles les données de fiabilité, quand on en a, correspondent, (il est bien évident que si les contraintes changent au cours de l'utilisation normale, on devra calculer plusieurs indices de contraintes.)

Apres détermination, les contraintes et indices seront reportés sur les listes des composants établis préalablement.

4.3.2 Détermination des taux de défaillance :

4.3.2.1 Détermination des taux de défaillance ou de la fiabilité des composants :

Si on dispose de sources de renseignements convenables et suffisamment récentes qui sont généralement donnés sous forme de taux de défaillance constants, il sera relativement facile pour la plupart des composants de calculer le taux de défaillance à prévoir, en tenant compte des indices de contrainte. De telles données ont été réunies dans des mémentos tels que le MIL-HDBK-217 publié aux Etats Unis il y a de nombreuses années déjà. Ces données doivent être tenues à jours et c'est la tâche d'un service de fiabilité que de publier des documents généraux sur les taux de défaillance à utiliser dans les calculs prévisionnels.

On verra dans la suite du chapitre comment modifier les taux de défaillance de référence pour tenir compte des contraintes réelles.

Les données concernant les composants à taux de défaillance variables ou les composants destructibles (fusibles, boulons explosifs), sont généralement fournies sous forme de probabilité de survie et une mention particulière doit en être faite dans les listes.

Si on ne dispose pas de données répertoriées, ou si le dispositif est nouveau, on devra procéder à des essais préalables.

4.3.2.2 Calcul des taux de défaillance ou de la fiabilité des blocs et des systèmes :

Ayant obtenu au moins partiellement, les taux de défaillance ou de la fiabilité prédite des composants des différents blocs (sous-entité fonctionnelle), on devra alors calculer les taux de défaillance de chaque bloc, en tenant compte des redondances si elles existent ; ceci en appliquant les équations établies antérieurement.

On notera soigneusement pour chaque bloc les taux d'utilisation adoptés pour la prévision afin d'en tenir compte dans le calcul de fiabilité du système.

Le calcul de la fiabilité prédite du système termine la tâche de prévision de fiabilité ; suivant la complexité et la configuration du système, les taux d'utilisation des différents composants, etc., on pourra, soit faire un calcul de taux de défaillance global, d'où le M.T.B.F. se déduit immédiatement, soit, dans les cas de redondance ou de taux de défaillance variables, calculer un nombre suffisant de valeurs de la fiabilité à différentes époques afin d'obtenir une détermination du M.T.B.F. par intégration (graphique ou numérique) de la fonction fiabilité.

On doit tenir compte des redondances pour prévoir la fiabilité d'un système, c'est-à-dire pour prévoir la probabilité de défaillance du système. Par contre si on s'intéresse au nombre de défaillances à prévoir en vue de chiffrer les dépenses de maintenance corrective, on ne doit pas tenir compte des redondances au niveau des sous systèmes.

4.3.3 Généralités :

Les données numériques sur les taux de défaillance des composants sont essentielles pour mener à bien les prévisions de fiabilité.

Ces données doivent êtres sûres. Les différents recueils de données sont malheureusement basés sur des critères de défaillance différents et sur des utilisations peu comparables, si bien que les taux minimaux, moyens et maximaux provenant de quatorze sources d'informations différentes : on peut constater des variations de l'ordre de 1000 peuvent exister entre différentes données, publiées avant 1963 aux Etat-Unis.

Il est évident qu'en fonction des applications, on devra disposer, pour effectuer des prévisions convenables, de données mieux adaptées.

Taux de défaillance horaire / Composant	Mini	Moyen	Maxi	Rapport
Condensateur	1.10^{-8}	300.10^{-8}	1000.10^{-8}	1000
Diode silicium	$15.\ 10^{-8}$	$154.\ 10^{-8}$	$385.\ 10^{-8}$	26
Moteur C.A	$15.\ 10^{-8}$	$6700.\ 10^{-8}$	$55600.\ 10^{-8}$	3700
Résistance à couche	$2.\ 10^{-8}$	$40.\ 10^{-8}$	$100.\ 10^{-8}$	50
Relais	$10.\ 10^{-8}$	$2300.\ 10^{-8}$	$12500.\ 10^{-8}$	1250
Transistors	$10.\ 10^{-8}$	$730.\ 10^{-8}$	$1700.\ 10^{-8}$	170

Tableau 4.1 : Taux de défaillance des composants standards.

4.3.4 Sources d'informations disponibles :

Il existe un certain nombre de sources de données sur les taux de défaillance, plus particulièrement en électronique. Parmi les plus importantes, on peut citer le MIL-HDBK-217 préparé par R.C.A. sur un contrat de l'U.S. Air Force et le rapport A.R.I.N.C n°203-1-344.

Les données du MIL-HDBK-217 ont été reprises par le centre de fiabilité du C.N.E.T. et publiées dans la revue « Fiabilité ». Cependant, les données correspondantes doivent être maniées avec précautions et on préconise de diviser par deux les taux de défaillance indiqués dans cet ouvrage si on les applique à des circuits digitaux.

Des renseignements complémentaires plus récents ont été publiés, correspondant à des composants de calculateurs électroniques digitaux dans une ambiance de laboratoire ou d'exploitation normale, par G.E.C. Ltd. Un extrait sera donné plus loin.

4.3.4.1 Ajustement des taux de défaillance :

Les taux de défaillance de base, donnés dans les recueils, correspondent à des conditions de contrainte différentes de celles rencontrées dans les matériels étudiés, et il est nécessaire de les modifier pour tenir compte des contraintes mécaniques, thermiques, électriques et d'ambiance envisagées pour le matériel.

Pour chiffrer les facteurs de modification, on doit fréquemment procéder à une analyse poussée des contraintes et ce travail complexe se montre souvent très utile pour mettre en évidence des zones potentielles de fiabilité réduites ; tout en conservant présent à l'esprit que la fiabilité prédite ne sera pas aussi précise qu'on le souhaiterait.
D'une façon très générale, le taux de défaillance modifié sera donné par une fonction :

- des contraintes propres à son emploi ; par exemple, la puissance dissipée pour un semi-conducteur ou une résistance, la tension appliquée pour un condensateur, les efforts de traction ou de compression pour une pièce mécanique ;

- des contraintes d'ambiance, représentées, par exemple, par un facteur multiplicatif fonction du mode d'utilisation (laboratoire, matériel, militaire sol, sur bateau, sur avion, sur fusée, sur satellite, etc.). On remarque alors que les diverses évaluations de ces facteurs multiplicatifs du taux de défaillance peuvent diverger considérablement, et qu'une prévision correcte ne pourra se faire dans une spécialité que grâce à des informations sur le comportement en exploitation analysées avec toute la rigueur possible. A titre d'exemple, le tableau ci-dessous donne les facteurs multiplicatifs proposés, d'une part dans Reliability Engeneering de A.R.I.C. Research Corporation, d'autre part dans un article de W.P. Cole, « Prediction and Engineering Assessment in Early Design » paru dans The Radio and Electronique Engeneering de janvier 1966 :

Environnement	A.R.I.N.C	W.P.Cole
Laboratoire	--	1
Satellite	1	1
Matériel sol	1	8
Bateau	1	15
Train	--	22
Avion	6,5	50
Fusée	80	900

Tableau 4.2 : Coefficients de correction selon l'environnement de travail.

- de facteurs correctifs, tenant compte du type de matériel. Par exemple, en électronique, certains auteurs distinguent les alimentations avec un coefficient de correction (k=1), les circuits de commande de mémoire avec un coefficient de (k=0.5), les circuits de commutation (k=0.3). En réalité, le facteur k permet de tenir compte grossièrement des contraintes propres à l'emploi, indiquées plus qu'il est souvent impossible de déterminer avec exactitude ;

- de facteurs correctifs divers : par exemple un facteur pour tenir compte du rapport entre le nombre des défaillances aléatoires et celui des défaillances de dérive, un facteur pour tenir compte des procédures de maintenance, un facteur pour tenir compte de la complexité du système, etc.

4.3.4.2 Exemple de données de base extraites du MIL-HDBK-217

- **Semi-conducteur,**

Les taux de défaillance minimaux annoncés pour les diodes sont de 10.10^{-8} défaillances par heure et de 20.10^{-8} pour les transistors. Ces chiffres sont probablement convenables pour des diodes à pointe au germanium de fabrication relativement ancienne et pour des transistors à jonctions alliées au germanium, mais ne sont pas représentatifs des composants actuels.

- **Résistances,**

On trouvera ci-dessous les taux de défaillance indiqués pour une puissance égale à la puissance nominale de la résistance et pour une température ambiante de 40 °c.

Résistance agglomérées :	6.10^{-8} h^{-1}
Résistance à couche de carbone	65.10^{-8} h^{-1}
Résistance bobinées de puissance	220.10^{-8} h^{-1}

La conclusion qui s'impose, au point de vue prédictions de fiabilité, est de ne pas utiliser un recueil de données sans être assuré que les composants utilisés dans le projet sont de même nature que ceux sur lesquels ont été relevés les taux de défaillance.

4.4. Modélisation selon l'approche flux informationnel :

Dans ce paragraphe, nous avons modélisé à la fois les deux parties du capteur : partie analogique constituée du schéma de la figure 4.1, et partie processeur traitée dans le chapitre précédent. Vu la différence radicale entre l'information analogique continue et l'information numérique discrète (des bits), il fallait répondre à certaines questions qui s'imposent, par un modèle adéquat (approche flux informationnel), pour pouvoir unifier ces deux informations de nature différente.

4.4.1 Sources de défaillance :

D'après les spécialistes, les principales sources de fautes en électronique intégrée sont les courts-circuits, les circuits ouverts, les perturbations extérieures et le vieillissement des composants élémentaires.

Les courts-circuits et les circuits ouverts sont la conséquence du phénomène d'électro-migration, c'est-à-dire du déplacement de matière (le métal des pistes d'interconnexion) induit par l'écoulement des charges électriques. Ce phénomène peut être évité en respectant quelques règles simples de dimensionnement des pistes en fonction du courant maximal qu'elles doivent supporter.

Les perturbations électromagnétiques (CEM) et les perturbations engendrées par les particules ionisantes peuvent provoquer l'apparition de signaux parasites indiscernables du signal utile. Leurs effets sur les circuits analogiques peuvent être atténués à l'aide de techniques de traitement du signal au niveau système (filtrage). Les circuits numériques, quant à eux, sont

essentiellement sensibles aux particules ionisantes qui sont généralement responsables de la modification inopinée de la valeur d'une information (bit) stockée dans un point mémoire (bascules, RAM, . . .).

Le vieillissement des transistors sous l'effet des porteurs chauds a des conséquences sur les performances des circuits analogiques. En effet, leurs propriétés essentielles (gain, bande passante, offset, . . .) dérivent lentement au cours du temps.

La nature de l'information portée par un système analogique fait que la sûreté de tels dispositifs ne peut pas être optimisée à l'aide des mêmes méthodes que celles employées pour les systèmes numériques. La dérive dans le temps des caractéristiques d'un circuit analogique se traduit par une erreur sur l'infomation qui est par nature difficile à estimer. On peut cependant définir des intervalles pour les différents paramètres d'un circuit analogique et on considère alors que la fonctionnalité de ce circuit est correcte tant que ces paramètres restent dans ces intervalles (par exemple : gains minimal et maximal pour un amplificateur). Pour maximiser la durée de vie d'un circuit analogique, il faut donc minimiser les dérives des paramètres importants de sorte qu'ils restent le plus longtemps possibles dans les intervalles définis.

Le vieillissement des transistors MOS sous l'effet des porteurs chauds est la principale cause de dérive des paramètres des circuits analogiques. Sur la base d'une modélisation de la dérive des caractéristiques des transistors basée sur des lois physiques [Chen et al.,2008], nous avons développé une méthode permettant au concepteur d'estimer les effets du vieillissement des transistors sur le comportement du circuit complet. A l'aide de ce modèle, le concepteur peut alors évaluer la durée de vie d'un circuit donné ou encore intégrer les contraintes de la sûreté de fonctionnement dès la conception du circuit de manière à maximiser sa durée vie [Dubois et al., 2007].

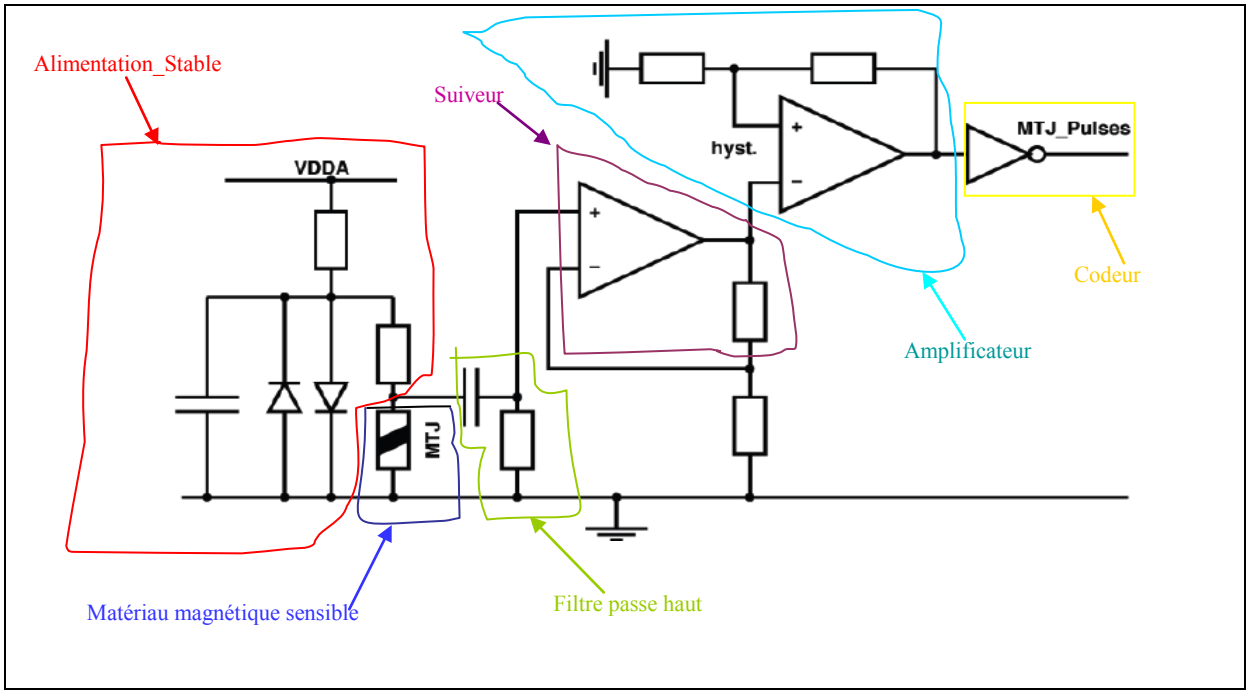


Figure 4.1: Schéma électronique de la partie analogique du capteur.

Nous avons commencé notre étude de fiabilité par une étude qualitative sous forme d'analyse préliminaire de risque, traduit par tableau AMDEC. L'analyse fine de l'étude précédente (AMDEC) de ce circuit permet de classer les problèmes les plus fréquents et qu'on ne peut pas nier, parmi ces problèmes liés à la partie analogique du capteur analogique ci-dessus (figure 4.1):

- Bloc alimentation : la défaillance de l'une des Diodes de ce bloc peut ne pas assurer l'alimentation demandée.

- Le bloc MTJ : on exploite la propriété physique de magnéto-résistance des matériaux pour faire varier un rapport cyclique en fonction du champ. Pour s'assurer de la bonne réponse de la MTJ, il faut imaginer une vérification sous forme de test périodique de sa réponse avant même de commencer les mesures.

- L'étage de filtrage est constitué d'un filtre de 1^{er} ordre. Dans cette partie, dans n'importe quel instant on peut perdre la précision de la bande passante.

- Les deux étages suiveur et amplificateur se réalise sur des circuits intégrés dont on ne peut pas assurer à 100% leur adaptation d'impédance.

Dans le même cadre nous pouvons citer quelques mécanismes de défaillance incontournables dans le monde des composants analogiques dont quelques un sont cités dans le tableau 4.3. :

Mécanisme de défaillance	Mode de défaillance	Zones affectées
Décharge électrostatique	Court-circuit, circuit ouvert	Interconnexion, MOS, bipolaire
Claquage de l'oxyde de grille	Courant de fuite, court circuit	MOS
Effet des porteurs chauds	Décalage de la tension de seuil, réduction du gain en courant	MOS, bipolaire
Electromigration	Court-circuit, circuit ouvert	Interconnexion
Corrosion	Circuit ouvert	Interconnexion
Latchup	Court circuit	CMOS
Radiation	Soft errors	Mémoires

Tableau 4.3 : Mécanismes et leurs modes de défaillance analogique.

Ces mécanismes de défaillance microscopiques sont à l'origine de défaillance macroscopique des composants et par la suite de la défaillance du système.

4.4.2 Modélisation d'une instruction simple :

La première phase à traiter dans cette étude, est la traduction de chaque composant de l'architecture matériel par une entité sous fonctionnelle appropriée pour l'approche flux informationnel, à savoir :

- Transformation du signal (**TF**)
- Stockage du signal (**SB**) (Modélise par exemple les registres internes et temporels)
- Décision (**IP**) (Modélise par exemple le Multiplexeur et l'UAL)

- Contrôle de flux (**CT**) (Modélise par exemple les Bus internes)
- Test en ligne (**ST**) (Modélise par exemple le contrôle de trafic)

Selon la librairie des entités sous fonctionnelles, et par application de l'approche flux informationnel sur l'architecture globale de notre capteur (partie analogique plus partie numérique), nous obtiendrons le modèle global de haut niveau pour une simulation d'addition de deux mesures :

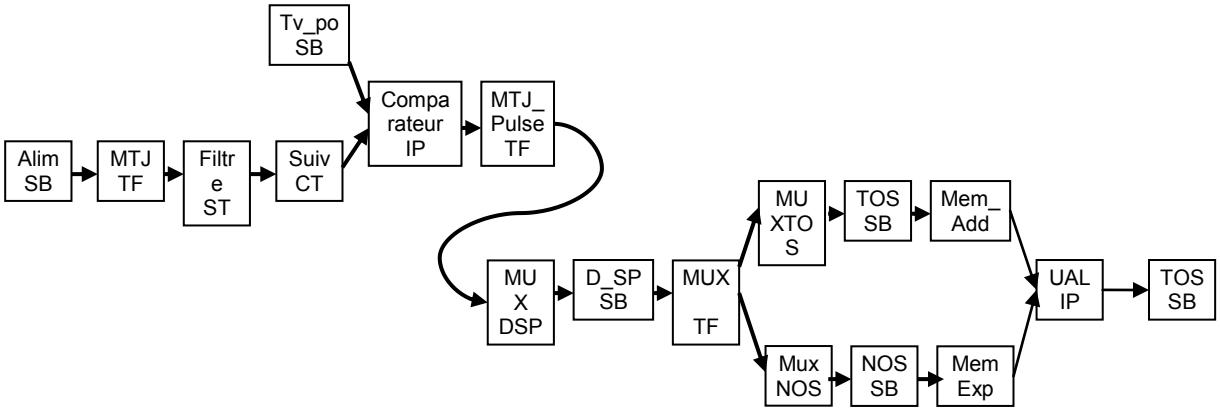


Figure 4.2: Modèle de haut niveau de l'architecture du capteur dans le cas d'une instruction ADD.

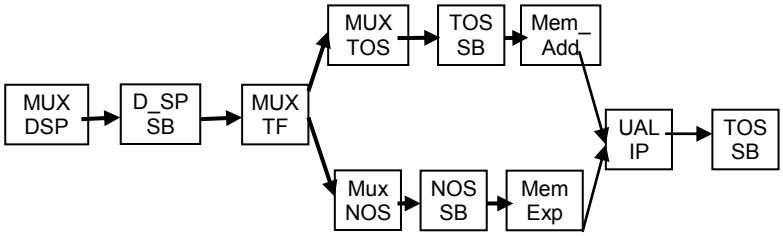


Figure 4.3 : Modèle de haut niveau en absence de la partie analogique.

Naturellement, la prochaine étape de la modélisation selon l'approche flux informationnel, nécessite le passage par le modèle de bas niveau sous forme d'automate d'états finis. Le problème qui se pose est l'interconnexion entre les deux mondes, l'automate qui représente la partie analogique (information continue) et l'autre qui représente l'architecture numérique du processeur (information discrète).

Pour comprendre mieux ce problème, dans la partie analogique, nous rencontrons des types de défaillance différents de ceux de la partie numérique. Dans la partie analogique nous trouvons trois états finaux à savoir :

- **Défaillance en valeur ou de précision.**
- **Défaillance temporelle** : - soit par dépassement
  - soit par temps de réponse
- **Défaillance par arrêt ou perte de mesure**.

En revanche dans la partie numérique, nous avons caractérisé et classé à chaque fois notre architecture selon six modes de fonctionnement, à savoir :

- **Mode 1**: mesure correcte, pas de défaillance détectée.
- **Mode 2**: mesure incorrecte, défaillance détectée mais tolérée (recherche de disponibilité).
- **Mode 3**: mesure incorrecte, défaillance détectée mais non tolérée (recherche de sécurité).
- **Mode 4** : mesure incorrecte, défaillance non détectée (mode dangereux).
- **Mode 5**: mesure correcte et défaillance détectée (arrêt intempestif).
- **Mode 6**: absence de mesure (arrêt du système).

Dans cette logique, il est normal de dire que les types de défaillance dans la partie analogique peuvent provoquer les défaillances de la partie numérique. En effet il y aura une continuité de propagation de l'information (flux d'information) entre les deux parties. Les trois états finaux analogiques seront l'origine de l'un des six modes au dessus.

Par la suite nous considérons la concaténation et le regroupement des automates concernant les deux parties (analogique et numérique), afin de construire un modèle global.

Dans la figure 4.4, nous trouvons plusieurs automates en cascade, chaque automate en amont alimente un autre en aval par une information propagée. Les six premiers automates présentent les différents composants de la partie analogique. A titre d'exemple nous considérons l'automate de bloc alimentation „Alim", il comporte les états d'entrée et de sortie suivants :

- Etat de fonctionnement nominal où il n'y pas d'anomalie. **E-N**
- Etat de défaillance en valeur (peut être une valeur non précise) **E-D-V**
- Etat de défaillance temporel **E-D-T**
- Etat de perte de l'infomation. **E-D-P**

Le passage de l'un à l'autre de ces états peut se faire par l'intermédiaire d'états qui représentent les modes de défaillance à l'origine du changement de l'information, ces états pour l'automate en question sont :

- Etat de décharge électrostatique **D-Elc**
- Etat d'éléctromigration **Elcmi**
- Etat de corrosion **Corro**

Lorsque l'infomation nominale (pas de défaillance) au niveau du composant **„Alim"** passe par l'un de ces états intermédiaires, cette information se dirige vers l'un des états de défaillance cités ci dessus comme il est montré dans l'automate en jaune de la figure 4.4.

Les autres automates se comportent de la même manière, sauf qu'ils ne comportent que deux états intermédiaires:

- Etat de défaillance par décharge électrostatique **D-Elc**
- Etat de défaillance par effet des porteurs chauds. **D-P-CH**

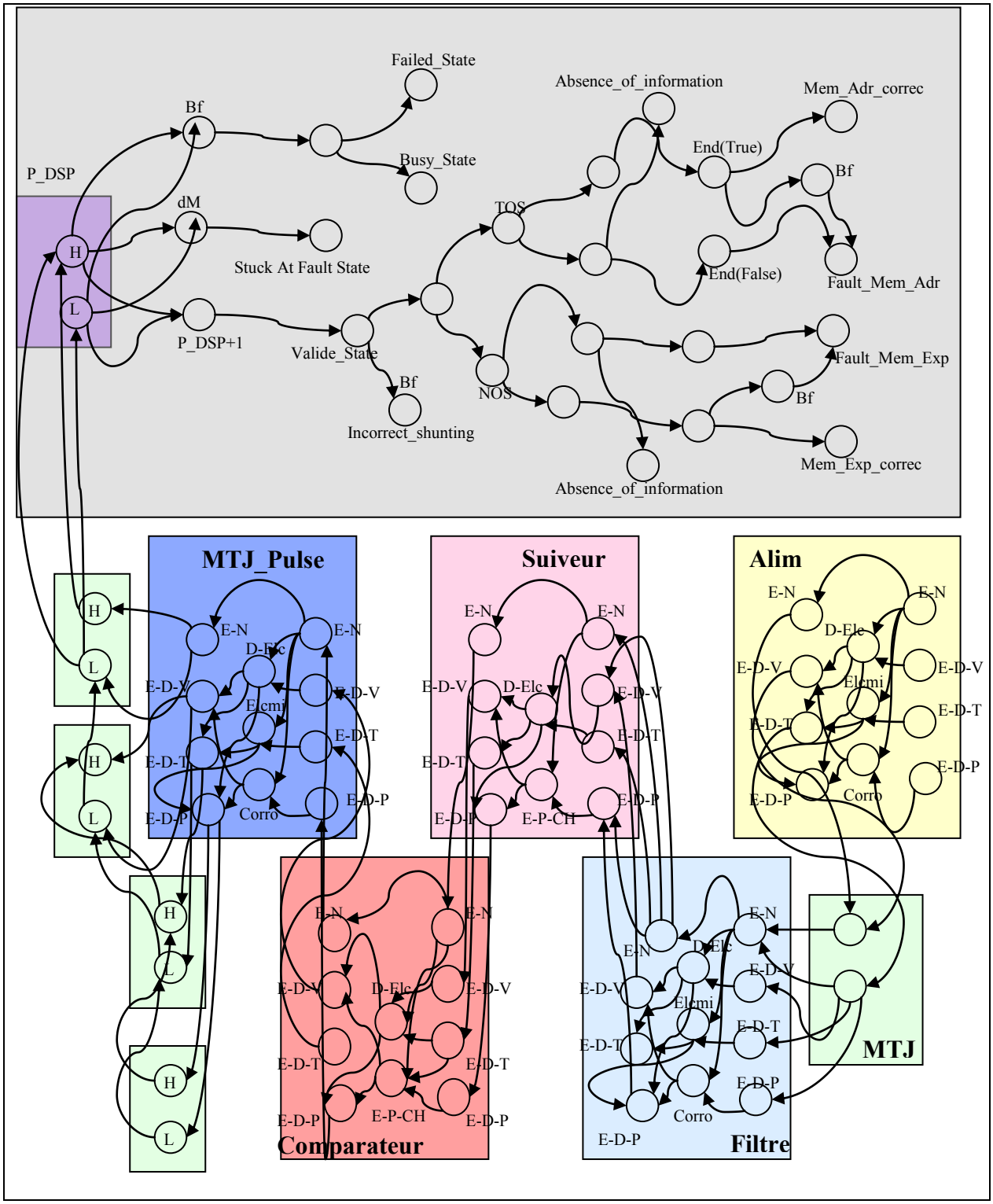


Figure 4.4: Modèle de bas niveau correspond à une instruction assembleur.

Nous combinons les deux mondes d'automates, de la partie numérique (en haut, couleur mauve), et de la partie analogique (en bas, les autres couleurs). Le fruit de cette unification est d'avoir la possibilité de générer des listes globales contenant à la fois des informations sur la

partie analogique et la partie numérique pour chaque instruction lorsqu'elle s'exécute sur l'architecture proposée du capteur.

Le suivi de l'information commence dès le premier composant de la partie analogique jusqu'au dernier élément de l'architecture numérique. La défaillance de l'infomation peut être due à la défaillance de différentes parties ou composants. Les listes globales générées sont capables de localiser la partie défaillante et l'effet de cette partie sur le reste de l'architecture. La quantification des probabilités de chaque mode de fonctionnement se fait de la même manière qu'expliqué en détail dans le chapitre précèdent. Nous trouvons les valeurs numériques cités dans le tableau 4.4 ci-dessous :

Mode de fonctionnemt / **Instruction**	**Mode 1**	**Mode 2**	**Mode 3**	**Mode 4**	**Mode 5**	**Mode 6**
ADD (Sans partie analogique)	**$3.3\ 10^{-2}$**	**$0.7\ 10^{-2}$**	**$2.9\ 10^{-4}$**	**$1.1\ 10^{-5}$**	**$0.99\ 10^{-9}$**	**$1.11\ 10^{-11}$**
ADD (Avec partie analogique)	$72\ 10^{-5}$	$32\ 10^{-3}$	$893\ 10^{-3}$	$87\ 10^{-4}$	$4.99\ 10^{-4}$	$45.7\ 10^{-3}$

Tableau 4.4 : Résultats relatifs à une instruction ADD.

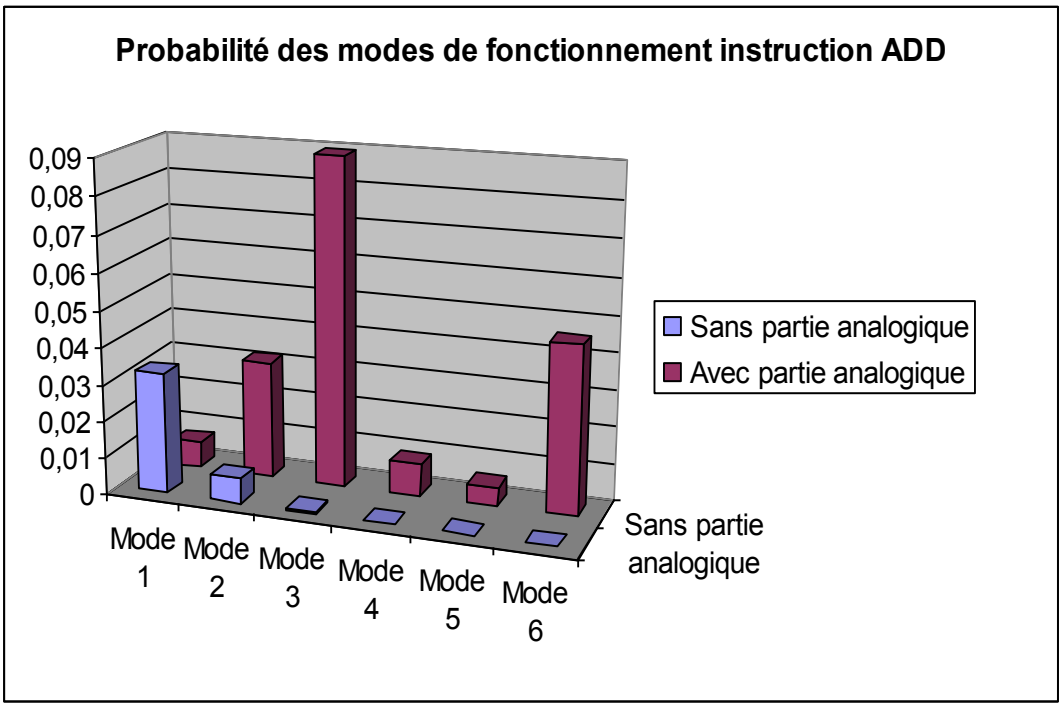


Figure 4.5 : Représentation graphique des résultats du tableau 4.4

La différence entre ces résultats et les résultats du modèle figure 4.3, est l'ajout de la partie analogique, cette partie à pour effet d'augmenter en quelque sorte la probabilité d'arrêt du système (mode 3), ainsi que diminuer la probabilité du mode nominal (mode 1).

Les résultats obtenus à ce niveau sont relatifs à une instruction assembleur simple, sans oublier que notre objectif est d'évaluer des fonctions de programme, sachant que chaque fonction est constituée de plusieurs instructions.

Pour illustrer cet effet, nous reprenons le programme de Tri déjà traité dans le chapitre 3, afin de constater l'ajout de la partie analogique dans le calcul des probabilités.

4.4.3 Cas d'étude: Programme de Tri :

Du point de vue fonctionnel d'un système numérique hiérarchique, le logiciel est conçu pour remplir des fonctions que le système numérique effectue au niveau 0. Le logiciel est composé de modules. Ces modules logiciels de niveau 1 réalisent leurs tâches grâce à la combinaison de jeux d'instructions fournies par le microprocesseur. Des parties des composants matériels du niveau 3 (des parties de processeur comme exemple la mémoire) sont utilisées pour le traitement d'une instruction de niveau 2. Afin que le système numérique complète sa fonction requise, le logiciel détermine l'ordre correct dans lequel les ressources du matériel devraient être utilisées. La défaillance du système, ainsi, se produit lorsque le logiciel ne peut pas organiser correctement la séquence des ressources matérielles utilisées ou lorsque l'une ou plusieurs des ressources matérielles utilisées ont des défauts bien que le logiciel ait déterminé les séquences correctes des ressources matérielles.

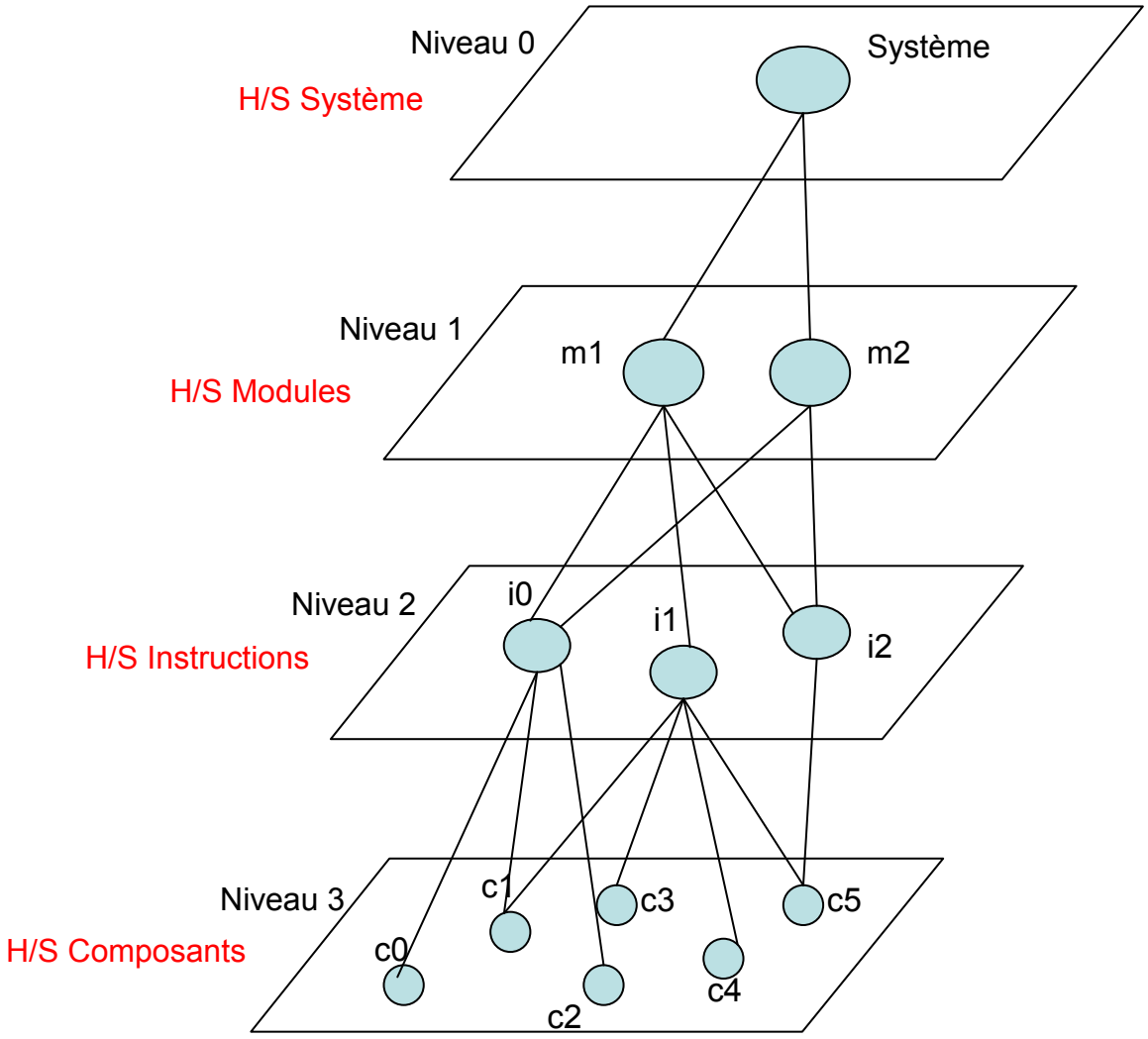


Figure 4.6: Modèle hiérarchique d'une application logiciel.

Dans ce travail, nous étudions un modèle combinatoire pour estimer la fiabilité du système numérique embarqué au moyen d'une description multifonctionnelle. Ce modèle ne considère pas seulement la manipulation des fautes imputables à la partie numérique, mais aussi l'interaction entre le logiciel et le matériel avec l'examen du profil de fonctionnement du logiciel. Le profil opérationnel du logiciel détermine la fréquence de l'utilisation de chacun des modules logiciels qui contrôlent la fréquence d'utilisation des composants matériels. Le profil opérationnel du logiciel est modélisé par l'adaptation de contrôle de flux logiciel multifonctionnelle.

Le modèle fonctionnel (dysfonctionnel) de bas niveau montré dans la figure 4.4, permet de générer l'ensemble des listes relatives au niveau 3 (H/S composants) de la figure 4.6. Cet ensemble remonte dans le niveau 2 (H/S instructions) pour donner une idée sur l'exécution d'une instruction assembleur.

$$\mathbf{L_{Instruction} = L_{Alim}.L_{MTJ}.L_{Filtre}.L_{Suiveur}.L_{Comparateur}.L_{MTJPULSE}.L_{Processeur}}$$

L_{Alim} : liste caractéristique de bloc alimentation.
L_{MTJ} : liste caractéristique de la tête sensible.
L_{Filtre} : liste caractéristique de bloc Filtre.
$L_{Suiveur}$: liste caractéristique de bloc Suiveur.
$L_{Comparateur}$: liste caractéristique de bloc Comparateur.
$L_{MTJ\text{-}PULSE}$: liste caractéristique de bloc MTJ-PULSE.
$L_{Processeur}$: liste caractéristique de la partie processeur à pile, déjà étudié dans le chapitre 3.

$$\mathbf{L_{Fonction} = L_{Instruction1}, L_{Instruction2}, L_{Instruction3}, L_{Instruction4}\ldots. L_{Instruction\ i}\ \ldots}$$

$$\mathbf{L_{module} = L_{Fonction1}, L_{Fonction2}, L_{Fonction3}, L_{Fonction4}\ldots. L_{Fonction\ i}\ldots.}$$

$$\mathbf{L_{application} = L_{module1}, L_{module2}, L_{module3}, L_{module4}, L_{module5}\ \ldots\ L_{module\ i}\ \ldots}$$

Appliquant la méthode de calcul et de quantification de chaque mode de fonctionnement (voir détail dans le chapitre 3), nous avons abouti à ces résultats numériques cités dans le tableau suivant :

	Mode 1	**Mode 2**	**Mode 3**	**Mode 4**	**Mode 5**	**Mode 6**
Programme_tri (Sans prise en compte de la partie analogique)	**$6.32\ 10^{-2}$**	**$1.57\ 10^{-2}$**	**$2.39\ 10^{-4}$**	**$55.8\ 10^{-5}$**	**$73\ 10^{-9}$**	**$45.91\ 10^{-11}$**
Programme_tri (Avec prise en compte de la partie analogique et sans tolérance)	**$5.22\ 10^{-4}$**	**$1.1\ 10^{-3}$**	**$5.5\ 10^{-4}$**	**$67.3\ 10^{-3}$**	**$12.2\ 10^{-6}$**	**$45.2\ 10^{-9}$**
Programme_tri (Avec prise en compte de la partie analogique et avec tolérance)	**$11.49\ 10^{-2}$**	**$1.2\ 10^{-2}$**	**$9.67\ 10^{-4}$**	**$99.3\ 10^{-5}$**	**$27.8\ 10^{-9}$**	**$98.61\ 10^{-11}$**

Tableau 4.5 : Effet de l'ajout de la partie analogique sur l'étude de la fiabilité du capteur.

Les résultats numériques montrent combien l'ajout de la partie analogique diminue globalement la fiabilité du système. Nous remarquons la diminution de cette fiabilité dans les indices mesurés dans le tableau (six modes de fonctionnement).

La partie analogique connait certains modes de défaillance spécifiques comme la dérive de l'information qui est la conséquence directe du phénomène de vieillissement.

4.5 Exploitation des résultats pour l'amélioration de la fiabilité :

4.5.1 Redondance et vote majoritaire :

Dans la partie numérique, nous avons proposé en collaboration avec l'équipe LICM une redondance pour augmenter la fiabilité de cette partie, cette stratégie de redondance est de type logiciel pur. Nous avons montré l'intérêt de l'implantation de cette redondance pour tolérer les fautes de famille numérique en cas de besoin.

En général, améliorer la sûreté de fonctionnement d'un système peut être fait par 2 moyens essentiels:

i- Améliorer la sûreté de fonctionnement des constituants, c'est à dire réduire la probabilité d'apparition de leurs défaillances :

- Par le choix d'éléments plus fiable (testé selon la méthode proposée avant).
- Par l'étude poussée des conditions de fonctionnement, définir toutes les conditions et événements possibles, et comment faire des barrières de protection contre l'influence des ces événements.
- Par la maintenance préventive des constituants

ii- Améliorer la structure du système en introduisant des redondances dont l'objectif est d'augmenter le nombre et la probabilité de bon fonctionnement par rapport aux états de défaillance.

De la même manière que la redondance logicielle de la partie processeur, nous proposons dans la partie analogique une redondance matérielle, l'idée se base sur l'implémentation d'un capteur témoin. Dans cette étude, nous ne nous intéressons pas à l'étude du coup supplémentaire de son installation, mais nous nous focalisons sur la valeur ajoutée après l'implémentation du capteur témoin en terme de disponibilité, fiabilité ainsi que les interventions de maintenances prévues afin d'augmenter la disponibilité de l'ensemble.

4.5.2 Capteur témoin et sa stratégie de maintenance [Annexe D]:

En collaboration avec l'équipe InESS, après une réflexion justifiée, nous avons décidé d'ajouter un élément témoin pour le capteur analogique. Ce type de redondance ne permet pas d'assurer le fonctionnement en cas de panne du capteur principal, mais plutôt, il va donner une information sur la dérive du capteur et notamment en cas de vieillissement de

ce dernier. Par conséquence, nous devons proposer une stratégie de maintenance de ce capteur témoin pour assurer son fonctionnement en permanence.

Les prévisions de fiabilité étudiées jusqu'ici reposent sur l'hypothèse du taux de défaillance constant.

On a vu au chapitre précèdent que la fiabilité d'un système dépendait de l'âge des pièces lorsque celles-ci étaient sujettes à la dérive ou à l'usure ; en pratique les échanges de telles pièces entraîneront assez tôt une grande diversité dans les âges des constituants de telle sorte qu'on peut se demander s'il n'est pas possible, pour un système comportant beaucoup d'éléments d'âges différents et sujets à l'usure, de définir un taux de défaillance constant équivalent.

4.6 Conclusion :

Les prévisions de fiabilité ne contribuent pas essentiellement à la fiabilité d'un système, mais elles doivent constituer un élément de jugement pour apprécier les actions à mener pour l'améliorer et elles doivent également servir à évaluer l'état actuel de la fiabilité d'un système vis-à-vis des objectifs qui ont été fixés. Les objectifs essentiels de ces prévisions sont : l'évaluation de la possibilité de réalisation de ce système, la comparaison de solutions et méthodes concurrentes, la détermination des objectifs requis, la mise en lumière des problèmes de fiabilité, la recherche des données numériques insuffisantes, l'étude de compromis avec d'autres éléments de coût dans le système, l'appréciation des progrès accomplis et les prévisions concernant la maintenance de ce système. L'ensemble de ces points se résume comme suit :

- Evaluation des possibilités : vérifier la compatibilité des principes de fonctionnement avec les objectifs de fiabilité qui ont été proposés.

- Comparaison des solutions et méthode concurrentes : état de l'art.

- Répartition des objectifs de fiabilité en phase de conception sur les partenaires du projet.

- La mise en évidence du problème de fiabilité.

- Exploitation des données de fiabilité : au cours de l'analyse nécessitée par la prévision ; on mettra en évidence les points pour lesquels les données de fiabilité ne sont pas parfaitement convenables à cause de conditions d'utilisation très différente ou bien suite à un changement d'un composant différant du composant initial.

- Suivi de progrès : vérification de la situation du projet vis-à-vis de l'objectif fixé.

- Prévision de maintenance : finalement, les prévisions de la fiabilité permettent d'évaluer la charge de maintenance de dépannage pour faire face aux défaillances aléatoires du système.

Dans ce chapitre, nous avons répondu aux objectifs mentionnés dans le cahier des charges de ce projet, concernant l'étude des signaux de nature différente (signaux mixtes). Grâce à l'approche proposée basée sur le flux d'information, nous avons tiré tous les scénarios de combinaisons des possibilités d'être dans un mode de fonctionnement quelconque parmi ceux qui sont définis le long de ce travail. L'étude de fiabilité menée par l'équipe SACSS a permis de montrer l'utilité de l'ajout stratégie de tolérance et son apport sur la fiabilité de l'ensemble, comme elle a montré l'intérêt de la prise en compte de la dégradation et du vieillissement du capteur.

Dans le chapitre précédent, l'étude de la partie numérique (processeur), nous avons constaté au départ que la fiabilité nécessite une amélioration, d'où l'intérêt d'implanter une redondance de type logiciel (stratégie de recouvrement). L'approche basée sur le flux d'information a montré sa capacité d'évaluer à la fois les deux modes de fonctionnement (avec et sans tolérance aux fautes).

Dans ce chapitre, l'étude de la partie analogique, l'étude montre que les modes de défaillance sont radicalement différents par rapport à la partie numérique, tenant compte de la nature de l'information (signal continu). Nous avons montré de même l'effet du vieillissement des composants sur la fiabilité. Heureusement que l'approche flux informationnelle est valable et nous a permis d'évaluer les deux parties du capteur (processeur, partie analogique) ensemble dans un modèle qui unifie les deux

La présence de la partie analogique a évidement changé les valeurs de probabilité, dans ce cas de figure le capteur a tendance être dans l'état 3 et 4 avec une probabilité faible d'être dans l'état 1. La prise en compte de la tolérance a permis de remettre le capteur dans un état normal (une probabilité importante de l'état 1) et deux avec une probabilité négligeable de l'état 3 et 4.

Conclusions générales et perspectives

Les systèmes programmables complexes suscitent un intérêt croissant dans l'industrie du fait des enjeux considérables qu'ils représentent par la capacité de progrès technologiques, par l'importance des marchés concernés et par la nécessité d'expertises multi-métiers. Ils traduisent un besoin de services élaborés afin d'accroître l'efficacité des systèmes opérationnels.

Ce travail de recherche fait apparaître la nécessité d'une analyse spécifique des systèmes complexes critiques répondant à des exigences de qualité et de sûreté de fonctionnement très strictes, et propose des méthodes et outils de conception pour y parvenir. Cependant leur large déploiement au sein de la société va de manière croissante impliquer la prise en compte effective de différents types de risques qui leur sont associés :

• la sûreté de fonctionnement : il s'agit de s'assurer que des systèmes complexes remplissant notamment des tâches critiques à divers égards, font bien ce pourquoi ils ont été conçus, ne font que cela et sont capables de survivre à des pannes de leurs composants.

• la sécurité : face à la multitude des individus qui peuvent avoir accès à ce système, il va s'agir de préserver la disponibilité de ces systèmes contre des agressions volontaires et la confidentialité et l'intégrité des informations qu'ils traitent.

• le respect de la vie privée: l'individu lui-même se doit d'être protégé contre toute atteinte aux données qui concernent sa vie privée et au-delà il importe de pouvoir garantir les droits de propriété d'informations numériques.

Pour répondre à ces besoins, les travaux de recherche présentés dans ce mémoire porte sur la problématique de la conception des systèmes complexes à électronique programmable, en prenant en compte les interactions entre matériel et logiciel spécifiques à ce type d'architecture. Une démarche méthodologique, de la modélisation à l'évaluation a été proposée. Elle est basée sur l'approche flux informationnel.

Les points globalement traités dans ce travail se résument en :

➢ Intégration de la sûreté dans les processus d'ingénierie système

Devant la complexité de ces systèmes, il nous est apparu nécessaire de proposer une approche globale d'analyse de sûreté de fonctionnement qui permette de prendre en compte les risques liés à l'intégration de plusieurs technologies. Nous proposons une approche permettant d'intégrer la sûreté de fonctionnement dans les processus de conception du système. Les normes dans ce domaine décrivent le processus du cycle de vie, du développement des systèmes depuis la formulation des exigences, jusqu'à leur démantèlement. Ceci permet une traçabilité des exigences de sûreté de fonctionnement et permet de s'assurer de la prise en compte de ces exigences tout au long du cycle de vie du système.

Pour mettre en évidence les scénarios redoutés, [Hami et al.,2005] dans sa thèse, a proposé une méthode d'extraction de scénarii à partir d'une modélisation basée sur le flux d'information le long d'une architecture de sécurité fonctionnelle. Ses travaux ont donné naissance à un premier outil permettant d'automatiser la méthode de génération de scénarios redoutés.

Nous avons généralisé les travaux précédents dans le but d'améliorer l'approche et d'aborder certains aspects qui n'ont pas été traités lors de ce premier travail [Belhadaoui et al., 2008-a] et [Belhadaoui et al., 2008-b].

➢ Bilan

Ce travail concerne la conception de systèmes programmables, complexes, dédiés à des applications de type mécatronique, contraintes par des impératifs de sûreté, voire de sécurité. Nous allons revenir sur les problèmes posés en proposant et en mettant en évidence quelques perspectives d'extension des résultats obtenus.

Au départ, nous avons distingué conventionnellement les fautes de conception de celles induites par l'environnement, ces dernières étant subdivisées en deux classes : les fautes fonctionnelles (mauvaises utilisations du système) et non-fonctionnelles (induites par l'environnement physique : température, rayonnements, etc.). De cet ensemble de fautes, nos études abordent les mauvaises conceptions et ne considèrent qu'une classe de fautes non-fonctionnelles environnementales : celles induites par les champs électromagnétiques (bit-flip).

Nous avons ainsi consacré un effort particulier à :

La prévention et la protection des dysfonctionnements liés aux champs électromagnétiques sous forme de perturbation de l'information et de la susceptibilité des composants à ces champs,

La prévention des fautes relevant du processus de conception (analyse de défaillance par approche informationnelle) ou de la technologie de programmation et de modélisation (automate à états finis).

Nous avons développé, dans cette thèse, une méthodologie d'évaluation de la fiabilité prévisionnelle, en préservant une solution analytique afin de permettre aux concepteurs de mieux appréhender les variations des métriques fiabilistes. La méthodologie proposée dépend des choix techniques matériels et logiciels, dans le cas de capteurs, ou d'actionneurs intelligents intégrables dans des applications mécatroniques. Selon la nature dynamique complexe de ces systèmes, nous avons proposé une étude comparative sur les principales méthodes classiques d'évaluation de la fiabilité à travers le cycle de développement, de retenir l'approche flux informationnel. Quatre raisons principales justifient ce choix. En effet, l'approche flux informationnel permet :

- ✓ Modélisation d'un système mécatronique intégrant différentes technologies
- ✓ Utilisation dans chaque étape du cycle de développement
- ✓ Analyse du comportement fonctionnel et dysfonctionnel
- ✓ Prise en compte de l'aspect dynamique du système

La méthodologie d'évaluation fiabiliste prévisionnelle s'appuie sur une :

- ✓ Modélisation fonctionnelle (permettant de donner les temps de fonctionnement)
- ✓ Modélisation dysfonctionnelle (donnant les instants de défaillance) du système mécatronique

- ✓ Recueils des données pour chaque composant

Nous avons appliqué la méthodologie proposée pour l'évaluation de fiabilité prévisionnelle de la partie programmable de sous-systèmes mécatroniques. C'est le cœur du capteur intelligent. Nous voulons valider l'applicabilité de cette méthodologie sur un cas d'étude réelle, et démontrer son intérêt par rapport aux autres méthodes conventionnelles. Sur l'exemple retenu (partie de traitement programmable), nous avons identifié les instructions les plus susceptibles d'entraîner un échec de l'application logicielle. Ceci par évaluation de sa probabilité d'exécution sur une architecture de processeur à pile [Belhadaoui et al., 2008-b].

➢ Perspectives

Beaucoup de points restent à améliorer. Une particularité des modes de défaillances des systèmes mécatroniques concerne les défaillances liées aux interactions entre les différentes technologies. Cet aspect est peu étudié et constitue un verrou technologique. En effet, l'analyse de la fiabilité de chaque composant du système n'est pas suffisante. Il devient nécessaire d'étudier le système dans son ensemble, ses composants, les interactions entre les différents composants et d'intégrer ces aspects dans une méthodologie globale d'estimation de la fiabilité.

L'algorithmique de recherche de scénarii proposé dans ce travail pour un système hybride, nécessite encore quelques adaptations en vue de l'appliquer dans un contexte industriel. Il est nécessaire de le valider sur des exemples réels. Ceci nous permettra de valider la méthodologie proposée et les outils associés à sa mise en œuvre, voir l'automatisation de toutes les démarches d'application de cette méthode comme c'était réalisé dans [Belhadaoui et al., 2006].

Un autre point important pour lequel il reste beaucoup de travail est l'approche de l'intégration de la sûreté de fonctionnement dans le processus d'ingénierie système. L'approche proposée n'est qu'un travail préliminaire qui a besoin d'être développé. Pour faciliter l'utilisation future de notre méthodologie dans un contexte industriel, il est nécessaire de concevoir un outil informatique automatisant les traitements.

L'analyse quantitative est un point important à explorer pour estimer la probabilité d'apparition des états redoutés. Car même si l'étude qualitative présente un intérêt réel, elle ne peut qu'être complémentaire d'une étude quantitative.

➢ Proposition d'une méthodologie globale

Pour une maîtrise fiabiliste entière d'un système mécatronique, trois phases doivent être intégrées dans un processus méthodologique global. Cette démarche doit être initiée le plus en amont possible du cycle de développement. Elle permet de consolider toutes les informations collectées au cours du processus de fiabilisation. Lors de l'étude de la fiabilité prévisionnelle, l'évaluation de la fiabilité du système est généralement déduite de celle des composants. La

fiabilité expérimentale et la fiabilité opérationnelle sont estimées de manière quantitative, après une phase de modélisation imposant de nombreuses hypothèses simplificatrices. Les connaissances acquises dans une phase sont exploités dans la suivante. A chaque phase, si les objectifs de fiabilité du système ne sont pas atteints, une étude d'évaluation détecte les composants responsables de la défaillance et permet de rétroagir sur la conception.

La figure C.G.1 donne le synoptique de la méthodologie globale proposée.

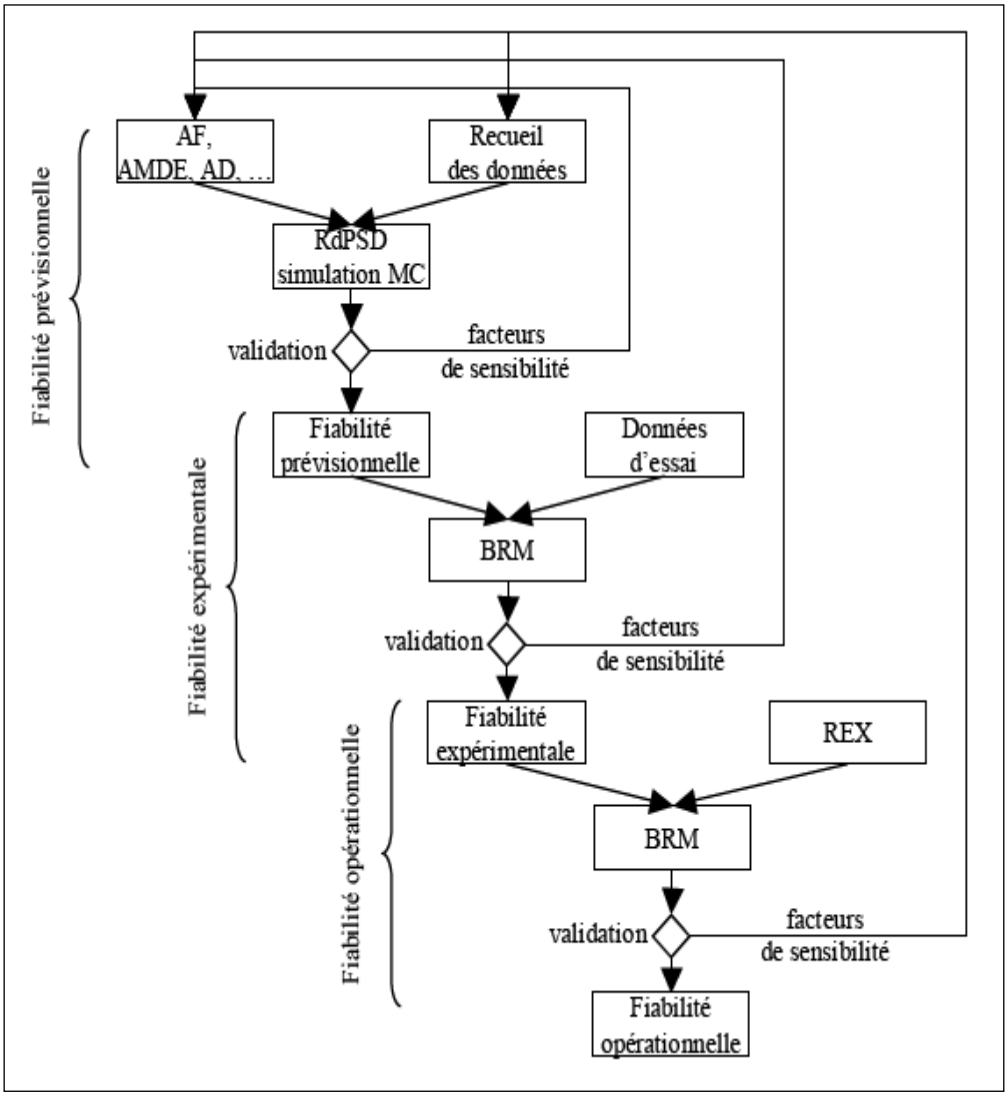


Figure C.G.1 : Démarche de la méthode globale proposée.

Notre contribution propose une partie de cette démarche de conception, nous cherchons par la suite de compléter cette méthodologie de conception jusqu'à un niveau de mise en exploitation du système.

Nous avons abordé les problématiques liées à la partie logique programmée (processeur à pile). La prochaine étape est l'intégration des problématiques propres à la conception de la partie traitement analogique (ceci en collaboration étroite avec l'InESS). Afin de généraliser l'étude sur des systèmes à signaux mixtes (booléens/numériques/analogiques).

Au sein du consortium pluridisciplinaire financé par la fondation CETIM, notre rôle a consisté à aider les laboratoires partenaires à comprendre les problématiques de la sûreté de

fonctionnement et de les intégrer dans leurs problématiques propres. Il a fallu s'entendre sur les concepts, les contraintes et objectifs des uns et des autres, afin de définir les travaux préliminaires nécessaires à la mise en œuvre d'une véritable méthodologie de co-conception des systèmes intelligents hautement intégrés, appelés couramment mécatroniques. Notre contribution s'est focalisée sur la conception de la partie numérique, l'approche utilisée semble adaptée à la modélisation de la complexité des systèmes actuels. Ce travail va être poursuivi afin d'y intégrer la partie analogique, et permettre l'analyse fiabiliste d'un système (ici un capteur mécatronique) complet.

➢ Vers une méthodologie de conception :

Les concepteurs de systèmes modernes doivent gérer des projets associant plusieurs disciplines et plusieurs technologies. Depuis des années, les systèmes électroniques ne sont plus conçus isolément : Ils intègrent des préoccupations de systèmes et de microsystèmes multidisciplinaires, dans divers secteurs d'applications scientifiques et industrielles. En raison de la complexité et de l'hétérogénéité de ces sous-systèmes, il est indispensable de mettre en place des méthodes et des outils facilitant l'intégration de solutions analogiques, numériques, mixtes (analogique-numérique), matérielles et logicielles dans des approches pluridisciplinaires. Ce problème pluridisciplinaire plus le besoin de maîtriser le processus de sa conception afin d'augmenter sa fiabilité, a conduit au développement de techniques telles que la modélisation et la validation à haut niveau, la modélisation fonctionnelle, la réutilisation et la génération de modules… Ces composantes doivent être considérées dès les premières étapes de la conception. Nos propositions auront à tenir compte de nombreuses exigences, et donc à s'appuyer sur des modélisations et des procédures standardisées.

Les méthodes actuelles doivent aussi prendre en compte la conception coopérative et la réutilisation des acquis REX (retour d'expérience). Elles doivent donc être basées sur des procédures et des langages normalisés ou standardisés facilitant les échanges. Le but est alors de proposer des outils généraux et des méthodes capables de soutenir le travail coopératif entre divers participants d'un projet de conception. Dans une première étape, les méthodes utilisées doivent s'appuyer sur des modèles de haut niveau, fonctionnels, exécutables, que nous pouvons appeler des Prototypes Virtuels. Ces prototypes permettent de vérifier, par simulation, leur conformité fondamentale avec le cahier des charges, avant d'entamer les démarches de matérialisation et de réalisation technologique.

Dans le contexte de la conception des systèmes et de la réutilisation des savoir-faire, le Co-design définit un domaine nouveau et prometteur dans le développement d'outils de Conception Assistée par Ordinateur (CAO). Ses objectifs principaux sont de faire, de façon simultanée, la conception du matériel et du logiciel afin de réduire les temps de développement et d'élaboration, de faciliter la détection des fautes et des erreurs, et de permettre le test des prototypes virtuels avant la mise en fabrication des produits. Nous devrons trouver et définir quelle est la place précise du «Co-design» dans notre démarche méthodologique.

L'intégration pluridisciplinaire autour de projets nécessite d'adopter des normes et des méthodes dans la définition des spécifications aussi bien que dans la description et la mise en place des modèles créés à partir de telles spécifications. Les concepteurs sont également obligés de trouver des solutions aux défis techniques posés par les cahiers des charges dans des créneaux de temps de plus en plus serrés. En résumé, la complexité croissante des

systèmes et les contraintes liées aux délais imposent la définition de nouvelles méthodes et de nouveaux outils capables de répondre simultanément aux besoins des concepteurs et à la coordination de tous les acteurs du problème.

Les motivations de recherche et d'industrie, et l'influence de la concurrence créent des intérêts forts chez les chercheurs, pour le développement, l'utilisation et l'optimisation de technologies de la conception système. Le développement des prototypes virtuels et la nécessité de valider leur cohérence avec les spécifications, demande l'appui d'outils informatiques permettant de modéliser, dès les niveaux d'abstraction les plus élevés, les aspects suivants:

- Les interactions du système avec son environnement opérationnel,
- L'évaluation et la définition des entrées/sorties,
- L'étude et le développement des modèles comportementaux des constituants,
- La représentation graphique des relations fonctionnelles proposées,
- L'exploration architecturale,
- L'estimation des performances et les états critiques de fonctionnement.

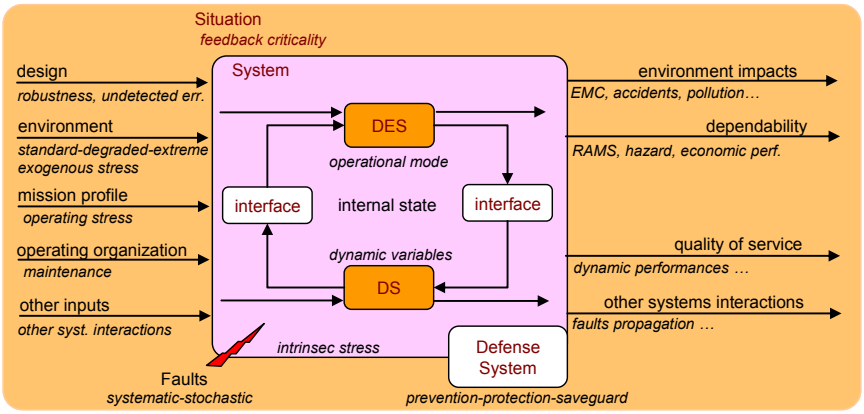
Les exigences générales se situent au niveau de la gestion de ces aspects et de la recherche de techniques permettant de réduire le temps de développement des produits et d'accroître les performances de la conception sur des points essentiels comme la robustesse, la sûreté de fonctionnement (avec ses quatre paramètres FMDS) et la vérification. Plusieurs axes de réflexion et de développement devront être explorés :

- La réutilisation, autant que possible, des acquis et des modèles précédemment validés. Dans ce contexte, l'extension de l'utilisation des outils de CAO pour l'électronique vers d'autres domaines peut représenter une source de progrès important.

- L'utilisation de techniques telles que le Co-design, la Co-simulation, la création et la gestion de modules de propriété intellectuelle ont amené le monde de l'électronique numérique au sommet de ce qu'il est convenu d'appeler l'EDA (Electronic Design Automation) jusque dans les applications commerciales.

- La partie analogique qui pourtant reste en retard par rapport à ces techniques à cause de la complexité du problème et de la difficulté à mettre en place une vraie politique de standardisation des méthodes, et des langages : Il n'est pas facile d'y établir une base commune de ressources informatiques et méthodologiques, à l'image de la modélisation VHDL ou de la gestion des IPs dans le domaine de l'électronique numérique.

La plus grande difficulté de l'approche est de généraliser les méthodes et de considérer la conception système comme un tout. De cette manière, en partant des spécifications ou des cahiers des charges, les concepteurs pourraient établir et valider des modèles fonctionnels et proposer des solutions architecturales. Une étape essentielle de cette problématique est la traduction ou l'interprétation des spécifications sous forme de modèles fonctionnels.
Nous pouvons illustrer globalement cette problématique et le contexte de nos activités par le biais de la figure suivante :



Dans ce contexte, notre approche concerne la mise au point d'une méthodologie de «Haut-niveau» de développement de modèles fonctionnels exécutables permettant aux concepteurs d'aborder les problèmes du choix des technologies et des architectures, des partitionnements analogique/numérique d'un part et logiciel/matériel d'autre part, sur la base sûre d'une proposition structurellement fiable et cohérente avec les spécifications. Cette méthodologie intervient avant l'établissement des choix technologiques, il n'y a pas donc lieu, à ce stade, de distinguer les micros systèmes et les systèmes. Nous parlerons, pour simplifier le discours, uniquement des systèmes dans les développements suivants. La proposition d'une méthodologie pour la conception de haut-niveau devra rechercher et prendre en compte les langages standards utilisables aussi bien pour la description système que pour les descriptions de niveaux inférieurs d'abstraction. L'utilisation de normes et de standards facilitera l'application d'une méthodologie dans les cas réels, en évitant qu'elle reste isolée comme une proposition seulement théorique.

L'objectif de notre travail est de concevoir, de développer et de mettre en œuvre les fondements d'un outil de Co-design pris au sens général : Matériel/logiciel/pluridisciplinaire, permettant d'appliquer une méthodologie de conception descendante dans le processus de conception système. Nous aurons à tester cet outil par le biais d'exemples concrets de gestion du «Co-design» dans le processus de conception : intégration de la partie analogique et arbitrages matériels / logiciels.

Une fois cette méthodologie et cet outil établis, nous analyserons quel est l'impact de leur mise en place sur la vérification « système » que nous pouvons envisager à partir du prototype virtuel. Pour cela, nous nous appuierons sur la gestion standardisée de l'information vis-à-vis de la réutilisation des savoir-faire et da la génération de modules de propriété intellectuelle IP. Finalement, nous réfléchirons à une première approche vers la vérification et la validation des modèles et des prototypes, notamment dans le domaine de l'avionique.

Références bibliographiques

[AFIS] Association Française pour l"Information Scientifique.

[Aiguo L. et Hong B., 2007] Aiguo Li and Bingrong Hong, „Software implemented transient fault detection in space computer". Aerospace Science and Technology, Volume 11, Issues 2-3, March-April 2007, Pages 245-252

[Aït-Kadi et al., 2000] Aït-Kadi, D., El Khair El Idrissi, A. & Gouget, N. (2000). "Prise en compte de la fiabilité et de la maintenabilité au stade de la conception". IDMME'2000, 3eme Conférence internationale sur la conception et la fabrication intégrées en mécanique -Actes, Montréal.

[Ajmone et al., 1991] M. Ajmone Marsan, G. Balbo, G. Chiola, G. Conte, S. Donatelli and G. Franceschinis, „An introduction to generalized stochastic Petri nets". Microelectronics and Reliability, Volume 31, Issue 4, 1991, Pages 699-725

[Ajmone et al., 1995] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli et G. Franceschini, Modelling with generalized stochastic Petri nets. Wiley series in parallel computing. 1995.

[Al-Dhaher A. H. G., 2001] A. H. G. Al-Dhaher, 'Integrating hardware and software for the development of microcontroller-based systems' Microprocessors and Microsystems, Volume 25, Issue 7, 15 October 2001, Pages 317-328

[Alaoui R. 2007] Rim MRANI ALAOUI, « conception d"un module de diagnostic a base des suites de bandes temporelles en vue de la supervision des procèdes énergétique ». Application en ligne à un générateur de vapeur, thèse doctorat, Université des Sciences et Technologies de Lille, 2007.

[Angelini A.M., 1996] A. M. Angelini, 'A note on the role of system engineering in the industry of a changing society' Technology in Society, Volume 18, Issue 4, 1996, Pages 461- 466

[Anderson D.A. 1971] D.A. Anderson, Design of self-checking digital networks using coding techniques, Coordinates, Sciences Laboratory, Report R/527, University of Illinois, Urbana, September 1971.

[Arlat et al., 2004] Mohamed Kaâniche, Yannick Le Guédart, Jean Arlat and Thierry Boyer, „An investigation on mutation strategies for fault injection into RDD-100 models". Safety Science, Volume 42, Issue 5, June 2004, Pages 385-403

[Arnold et al., 2000] Arnold, A., A. Griffault, G. Point, et A. Rauzy (2000). The AltaRica formalism for describing concurrent systems. Fundamenta Informaticae 40, 109–124.

[Azgomi M.A. et Movaghar A., 2005] Mohammad Abdollahi Azgomi and Ali Movaghar, 'A modelling tool for hierarchical stochastic activity networks'. Simulation Modelling Practice and Theory, Volume 13, Issue 6, September 2005, Pages 505-524

[Barana et al., 2004] O. Barana, A. Luchetta, G. Manduchi and C. Taliercio, 'Progress in real-time feedback control systems in RFX'. Fusion Engineering and Design, Volume 71, Issues 1-4, June 2004, Pages 35-40

[Bechta et al., 2000] Bechta Dugan J., Sullivan K.J., Coppit D., Developing a low-cost high-quality software tool for dynamic fault-tree analysis, IEEE Transactions on Reliability, vol. 49(1), pp. 49–59, 2000.

[Belhadaoui et al., 2006] H.Belhadaoui, O.Malasse, H.Medromi. „Développement d"un outil logiciel pour l"évaluation du Niveau de sécurité de fonctionnement de contrôle Relatif a la sécurité. (Calcul des paramètre de sûreté de Fonctionnement des systèmes de commande)". MOSIM Syrie, 2006

[Belhadaoui et al., 2007-a] H.Belhadaoui, C.Cassier, O.Malassé, J.F. Aubry, „Outil d"aide à la conception des systèmes mécatroniques". Qualita-Tanger Maroc, Mars 2007.

[Belhadaoui et al., 2007-b] H.Belhadaoui; B.Dubois; M.Jallouli; O.Malassé; J.F.Aubry; H.Medromi. 'Instrumentation sûre de fonctionnement, Une synergie multidisciplinaire'. 4ème Colloque Interdisciplinaire en Instrumentation Nancy France. October 2007.

[Belhadaoui et al., 2008-a] M.Jalouli, H.Belhadaoui, C.Diou, F.Monteiro, O.Malasse, J-F.AUBRY, A.DANDACHE, G.BUCHHEIT, H.MEDROMI. 'Dependability Consequences of Fault-Tolerant Technique Integrated in Stack Processor Emulator using Information Flow Approach'. Design & Technology of Integrated Systems, Tozeur Tunisia, March 2008.

[Belhadaoui et al., 2008-b] M.Jalouli, H.BELHADAOUI, C.DIOU, F.MONTEIRO, O.MALASSE, J-F.AUBRY, A.DANDACHE, G.BUCHHEIT, H.MEDROMI. „Evaluation of Important Reliability Parameters using VHDL-RTL modelling and Information Flow Approach". ESREL Valencia Spain, September 2008.

[Belhadaoui et al., 2009] H.BELHADAUOI, G. BUCHHEIT, O.MALASSE, J-F.AUBRY, H.MEDROMI. „Méthode quantitative d"Evaluation de la Fiabilité des Systèmes Critiques Programmés" Lamda_Mu16, Avignon, Septembre 2009.

[Bergeron et al., 2010] Bergeron S., Imbeau D., Giraud L., Infante Campos C. Sécurité des travaux électriques in Au-delà de nos perceptions... une culture de prévention : 32e Congrès de l'Association québécoise pour l'hygiène, la santé et la sécurité du travail / AQHSST, (32e 12-14 mai, 2010 : Lévis, Canada), 2010.

[Bernard V. 2004] V. Bernard, Evaluation de la sûreté de fonctionnement des systèmes complexes, basée sur un modèle fonctionnel dynamique : la méthode SAFE-SADT, Thèse de doctorat de l"Université de Valenciennes et du Hainut Cambresis, décembre 2004.

[Bérar et al., 2001] Bérar B et al. „Systems and software validation, model-checking techniques and tools". Springer; 2001.

[Bier V. 1988] BIER, Vicky, “Illusions of Safety”, Nuclear Safety- Engineering, Dpt of Industrial Engineering, University of Wisconsin.

[Bonda et al., 2000] A. Burns, D. Prasad, A. Bondavalli, F. Di Giandomenico, K. Ramamritham, J. Stankovic and L. Strigini, „The meaning and role of value in scheduling flexible real-time systems“. Journal of Systems Architecture, Volume 46, Issue 4, January 2000, Pages 305-325

[Bonda et al., 2002] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico and J. Xu, „An adaptive approach to achieving hardware and software fault tolerance in a distributed computing environment“. Journal of Systems Architecture, Volume 47, Issue 9, March 2002, Pages 763-781

[Bolch et al. 2000] C.Bolchini, R.Montandon, F.Salice and D.Sciuto, Finite State Machine and Data-Path Description. IEEE Transactions on very large scale integration (VLSI) systems, Vol.8, No.1, February 2000 pp. 98-103.

[Bryant et al., 2008] Rune M. Jensen, Manuela M. Veloso and Randal E. Bryant, „State-set branching: Leveraging BDDs for heuristic search“. Artificial Intelligence, Volume 172, Issues 2-3, February 2008, Pages 103-139

[Buisson J. 1993] Buisson J., « Analysis of switching devices with Bond Graphs », journal of the Franklin Institute, vol. 330, n°6, p. 1165-175, 1993]

[Burch E.et H-J.Kung, 1999] Evangeline Burch and Hsiang-Jui Kung, „Modeling software maintenance requests: A case study“. Computer Standards & Interfaces, Volume 21, Issue 2, 15 June 1999, Page 174

[Cabarbaye A. 1998] A.Cabarbaye - Apports, difficultés et prospectives dans le domaine de l'évaluation quantitative en Sûreté de Fonctionnement, Qualité Espace, supplément au N°33, septembre 1998.

[Camargo et al., 2001] J.B. Camargo, E. Canzian, J.R. Almeida, S.M. Paz, B.A. Basseto, Quantitative analysis methodology in safety-critical microprocessor application, Reliability Engineering and Systems Safety 74 (2001) 53- 62.

[Carter W.C et Schneider P.R., 1968] W.C., Carter, P.R. Schneider, Design of dynamically checked computers, Proc. 4th Congress IFIP, vol.2, Edinburgh, Scotland, 5Ð10 Aug. 1968, pp. 878Ð883.

[CEI 9126] Norme CEI 9126. – Génie du logiciel – Qualité des produits – Partie 1 : Modèle de qualité, Genève 2001.

[CEI 61058] Norme CEI 61508. – Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité, Genève 2001.

[CEI 50 191] International Electro-technical Vocabulary, Chapter 191: Dependability and quality of service. CEI. (1990)

[CEI 9126 01] Norme CEI 9126. – Génie du logiciel – Qualité des produits – Partie 1 : Modèle de qualité, Genève 2001.

[Chabot et al., 2003] J.-L. Chabot, Y. Dutuit, A. Rauzy & J.-P. Signoret. An engineering approach to optimize system design or spare parts inventory. Risk Decision and Policy, 8:1-11, 2003.

[CHARPENTIER P. 2002] CHARPENTIER Philippe, "Architecture d"automatisme en sécurité des machines : Etudes des conditions de conception liées aux défaillances de mode commun", Thèse de doctorat, Institut National Polytechnique de Loraine (CRAN), octobre 2002, 129p.

[Chen et al.,2008] Kun-Yue Chen, Sheng-Hung Wu, Yih-Wen Wang and Chi-Min Shu, „Runaway reaction and thermal hazards simulation of cumene hydroperoxide by DSC". Journal of Loss Prevention in the Process Industries, Volume 21, Issue 1, January 2008, Pages 101-109

[Chillarege et al., 1994] Michael Halliday, Inderpal Bhandari, Jarir Chaar and Ram Chillarege, 'Experiences in transferring a software process improvement methodology to production laboratories'. Journal of Systems and Software, Volume 26, Issue 1, July 1994, Pages 61-68

[Clarke D-W., 2000] D. W. Clarke, Intelligent Instrumentation, Transactions of the Institute of Measurement and Control 22,1 pp. 3-27, 2000.

[Colwell R.P. et Lethin R.A., 1996] ROBERT P. COLWELL and RICHARD A LETHIN, „Latent design faults in the development of the multiflow TRACE" . IEEE Transactions on Reliability, 34(4), 557 (December 1994). Microelectronics and Reliability, Volume 36, Issue 4, April 1996, Page 537

[Dabny et al., 2008] Richard W. Dabney, Letha Etzkorn and Glenn W. Cox, „A fault-tolerant approach to test control utilizing dual-redundant processors". Advances in Engineering Software, Volume 39, Issue 5, May 2008, Pages 371-383

[David R et Alla H. 1989] DAVID, R. et ALLA, H. Du grafcet aux réseaux de Petri. Hermes.1989. 500 p. ISBN : 2-86601-195-3.

[David R et Alla H. 1997] Alla, H. and David, R. 1998. Continuous and Hybrid Petri Nets. Journal of Circuits, Systems & Computers 8(1): 159-188. 3.

[Devooght J. et Smidts C. 1992] Devooght J., Smidts C. Probabilistic reactor dynamics – I: The theory of continuous event trees. Nuclear science and engineering, 111, 1992, p. 229 – 240.

[Donlin A. 2004] Donlin A. „Transaction level modeling: flows and use models". In: IEEE international conference on hardware/software codesign and system synthesis, September; 2004.

[Dubois et al., 2007] Brini A., Boughanem M., Dubois D., « Réseaux possibilistes pour un modèle de recherche d‟information, », Conférence francophone en Recherche d‟Information et Applications, Lyon, 15/03/2006-17/03/2006, Association Francophone de Recherche d‟Information et Applications (ARIA), http ://www.irit.fr/ARIA, p. 143-154, mars, 2006.

[Dutuit et al., 1997] Dutuit Y., Rauzy A., Signoret J.-P., Thomas P. Dependability modelling and evaluation by using stochastic Petri nets: application to two test cases. Reliability Engineering and System Safety 55, 1997, p. 117-124.

[EADS/Thales 2004] EADS/Thales, "FIDES et Ingénierie fiabilité en électronique", Actes de conférence, Lambda Mu 04, 12-19/10/2004. Bourges, France.

[EIA 1999] EIA STANDARD, Processes for engineering a system, January 1999, Electronic industries alliance.

[Everett et al., 1998] EVERETT .W, KEENE .S et NIKORA .A, „Applying Software Reliability‟ in the 1990s IEEE Transactions on Reliability, September 1998.

[Fowler et al., 2007] Amit Gadkari, Michele Pfund, Yan Chen and John W. Fowler, „Scheduling jobs on parallel machines with setup times and ready times‟. Computers & Industrial Engineering, In Press, Accepted Manuscript, Available online 4 November 2007

[FIDES, 2004] FIDES (2004). Méthodologie de fiabilité pour les systèmes électroniques. DGA-DM/STTC/CO/477.

[Geist et al., 1984] Kishor Trivedi, Joanne Bechta Dugan, Robert Geist and Mark Smotherman, „Hybrid reliability modeling of fault-tolerant computer systems‟. Computers & Electrical Engineering, Volume 11, Issues 2-3, 1984, Pages 87-108

[Gergeleit et al., 2005] L.B. Becker, E. Nett, S. Schemmer and M. Gergeleit Robust scheduling in team-robotics Journal of Systems and Software, Volume 77, Issue 1, July 2005, Pages3-16

[German et al., 1995] Reinhard German, Christian Kelling, Armin Zimmermann and Günter Hommel, 'TimeNET: a toolkit for evaluating non-Markovian stochastic Petri nets'. Performance Evaluation, Volume 24, Issues 1-2, November 1995, Pages 69-87

[Givargis T. et Vahid F., 2002] Vahid F, Givargis T. Embedded system design: a unified hardware/software introduction. Wiley; 2002.

[Goldenfeld et al., 2006] Akio Wakabayashi, Simon Baron-Cohen, Sally Wheelwright, Nigel Goldenfeld, Joe Delaney, Debra Fine, Richard Smith and Leonora Weil, „Development of short forms of the Empathy Quotient (EQ-Short) and the Systemizing Quotient (SQ-Short)‟. Personality and Individual Differences, Volume 41, Issue 5, October 2006, Pages 929-940

[Gorse et al., 2004] Gorse N, Be´langer P, Aboulhamid EM, and Savaria Y, „Mixing linguistic and formal techniques for high-level requirements‟. engineering. In: IEEE international conference on microelectronics, December; 2004.

[Hami et al.,2005] Hamidi K., Malassé O., Aubry J-F., Coupling of information flowregation method and dynamical model for a more accurate evaluation of reliability, ESREL, Gansk, 2005.

[Harel et al., 2002] David Harel, Hagi Lachover, Amnon Naamad, Amir Pnueli, Michal Politi, Rivi Sherman, Aharon Shtull-Trauring and Mark Trakhtenbrot, „Statemate: A Working Environment for the Development of Complex Reactive Systems‟. Readings in Hardware/Software Co-Design, 2002, Pages 135-146

[Henley E.J.K. 1992] Henley, E.J, K., 1992 Probabilistic Risk Assessment, Reliability Engineering, Design, and Analysis. IEEE PRESS, Piscataway, New Jersey.

[Houtermans M.et Apostolakis G., 2002] M. Houtermans, G. Apostolakis, The dynamic flowgraph methodology as a safety analysis tool: programmable electronic system design and verification, Safety Science 40 (2002) 813-833.

[IEC61508 99] IEC 61508, 1999 Functional Safety of Electrical/ Electronic/ Programmabel Electronic Safety-Related Systems, Part 1-7. International Electrotechnical Committee.

[ITRS 2003] International Technology Roadmap for Semiconductors Design; 2003.

[Jallou et al., 2007] M. Jallouli, C. Diou, F. Monteiro and A. Dandache “Stack processor architecture and development methods suitable for dependable applications”, in Proc. 3rd International Workshop on Reconfigurable Communication Centric System-On-Chips (ReCoSoC‟07), June 2007.

[Jing et al., 2005] Jing-An Li, Yue Wu, King Keung, Ke Liu, Reliability estimation and prediction of multi-state components and coherent systems, Reliability Engineering and System Safety 88 (2005) 93-98.

[Kanawati et al., 1995] Roland Baiter, Slim Ben Atallah and Rushed Kanawati, „Architecture for synchronous groupware application development‟. Advances in Human Factors/Ergonomics, Volume 20, Part 1, 1995, Pages 371-376

[Kanoun et al., 1996] Nicolae Fota, Mohamed Kaâniche and Karama Kanoun, „Dependability evaluation of an air traffic control computing system‟. Performance Evaluation, Volume 35, Issues 3-4, May 1999, Pages 253-273

[Karacal S. 1998] S. Cem Karacal, „A novel approach to simulation modeling‟. Computers & Industrial Engineering, Volume 34, Issue 3, July 1998, Pages 573-587

[Khobare et al., 1998] S. K. Khobare, S. V. Shrikhande, Umesh Chandra and G. Govindarajan, 'Reliability analysis of microcomputer circuit modules and computer based control systems important to safety of nuclear power plants'. Reliability

Engineering & System Safety, Volume 59, Issue 2, February 1998, Pages 253-258

[Kim K. et T. Lawrence 1992] KH (Kane) Kim and Thomas F Lawrence, 'Adaptive fault-tolerance in complex real-time distributed computer system applications'. Computer Communications, Volume 15, Issue 4, May 1992, Pages 243-251

[Kropf T. 1999] Kropf T. Introduction to formal hardware validation. Springer Verlag; 1999.

[Laprie J.C.,1995] Laprie, J. C. (1995). Dependable computing : concepts, limits, challenges. In 25th International Symposium on Fault-Tolerant Computing (FTCS-25) Special Issue, pages 42–54, Pasadena - USA.

[Laprie J.C.,2004] J.-C.Laprie, „Sûreté de fonctionnement informatique : concepts, défis, directions", ACI Sécurité et Informatique - Toulouse, 15-17 novembre 2004.

[Laprie J.C.,2002] J.C.Laprie, M.Kaâniche and J.P.Blanquart, „A framework for dependability engineering of critical computing systems" Safety Science, Volume 40, Issue 9, December 2002, Pages 731-752.

[Laulheret R. et Cabarbaye A. 2005] Cabarbaye A., Laulheret R. Evaluation de la sûreté de fonctionnement des systèmes dynamiques par modélisation récursive. 6ème Congrès International pluridisciplinaire, qualité et sûreté de fonctionnement, Qualita 2005, Bordeaux, 2005.

[Lei et al., 2007] Z.Lei, H.Yinhe, L.Huawei, L.Xiaowei, "Fault Tolerance Mechanism in Chip Many-Core Processors", Tsinghua Science and Technology, ISSN 1007-0214 30/49, vol. 12, No. S1, July 2007 pp.169-174.

[Liu S.et Wang H., 2007] Shaoying Liu and Hao Wang, „An automated approach to specification animation for validation". Journal of Systems and Software, Volume 80, Issue 8, August 2007, Pages 1271-1285.

[Lyonnet P., 2006] Lyonnet, P. (2006). Ingénierie de la fiabilité. Lavoisier.

[Malhotra M. et K. S. Trivedi, 1994] M. Malhotra, K. S. Trivedi. Power-hierarchy of dependability-model types. IEEE Transactions on Reliability, vol 43. pp, 493-502. 1994.

[Medjouji M . 2006] M. Medjouji, Contribution à l'analyse des systèmes pilotés par calculateurs : Extraction de scénarios redoutés et vérification de contraintes temporelles. Thèse de l'Université Paul Sabatier de Toulouse. Mars 2006.

[Mercier et al., 2006] David Mercier, Benjamin Quost and Thierry Denœux, „Refined modeling of sensor reliability in the belief function framework using contextual discounting". Information Fusion, In Press, Corrected Proof, Available online 12 October 2006

[Mihalache et al., 2005] Mihalache, A., Guerin, F., Barreau, M., Todoskoff, A., et Dumon,

B. (2005). Reliability estimation of embedded mechatronic systems. In IEEE The 6International Workshop on Research and Education in Mechatronics REM 2005, pages 208–213, Annecy, France. ISBN2-9516453-6-8.

[MIL-HDBK-338B 1998] MIL-HDBK-338B, MILITARY HANDBOOK: ELECTRONIC RELIABILITY HANDBOOK (1 OCT 1998).

[MIL-HDBK-217F 1991] MIL-HDBK-217F, "Reliability prediction of electronic equipment", Department of Defense/Reliability Analysis Center and Rome Laboratory at Griffiss AFB, NY, USA, December 1991.

[MIL-HDBK 95] Department of Defense, Washington DC, "MIL-HDBK-217F – "Reliability prediction of electronic equipment" (+ notice 1and 2) (dernière révision en 1995).

[Molloy M. K. 1981] M. K. Molloy, „On the integration of delay and throughput measures in distributed processing models". Th. 1981. Université de Californie, Los Angeles, EU.

[Monstand et al., 2006] Myrto Konstandinidou, Zoe Nivolianitou, Chris Kiranoudis and Nikolaos Markatos, „A fuzzy modeling application of CREAM methodology for human reliability analysis". Reliability Engineering & System Safety, Volume 91, Issue 6, June 2006, Pages 706-716

[Nelli et al., 2008] Alberto Landi, Alberto Mazzoldi, Chiara Andreoni, Matteo Bianchi, Andrea Cavallini, Marco Laurino, Leonardo Ricotti, Rodolfo Iuliano, Barbara Matteoli and Luca Ceccherini-Nelli, „Modelling and control of HIV dynamics". Computer Methods and Programs in Biomedicine, Volume 89, Issue 2, February 2008, Pages 162-168

[NF 60770-3] AFNOR, NF EN 60770-3 Transmetteurs utilisés dans les systèmes de conduite des processus industriels - Partie 3 : Méthodes pour l'évaluation des performances des transmetteurs intelligents. Norme AFNOR, 2006.

[Patton et al., 2006] R.J. Patton, C. Kambhampati, A. Casavola and G. Franzè, 'Fault-tolerance as a key Requirement for the Control of Modern Systems'. Fault Detection, Supervision and Safety of Technical Processes 2006, 2007, Pages 13-22

[Perez et al., 2007] Pérez Castaneda G. A., Aubry J-F., Brinzei N. Modélisation et simulation d"un système dynamique hybride pour calculer sa fiabilité dynamique en utilisant le toolbox Scicos de Scilab. 7ème édition du Congrès International pluridisciplinaire Qualita 2007 – Tanger (Maroc), p. 311 – 318, 2007.

[Poledna S. 1995-b] S. Poledna. Tolerating sensor timing faults in highly responsive hard realtime systems. IEEE Transaction on computers. Vol (44), No 2, p:181-191, February (1995)

[Qin X. et Jiang H., 2005] Xiao Qin and Hong Jiang, „Estimation of software reliability with execution time model using the pattern mapping technique of artificial neural

network". Computers & Operations Research, Volume 32, Issue 1, January 2005, Pages 187-199 Nidhi Gupta and Manu Pratap Singh

[Robert C. 1992] Robert C. (1992), L"analyse statistique bayesienne, Economica, Paris.

[Rony et al., 2008] R.Ghostine, J.M. Thiriet, J.F. Aubry, M. Robert. A framework for the reliability evaluation of networked control systems. To appear in the 17th IFAC World Congress. July 6-11, 2008, Seoul, Korea.

[Rook 1990] Rook, Software Reliability Handbook. Springer. P. (1990).

[Rumbaugh et al., 1991] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, "Object Oriented Modeling and Design", Prentice Hall International, 1991.

[Sadou et al., 2005] Sadou N, H Demmou, J. C Pascal , R Valette. Object oriented approach for deriving feared scenarios in hybrid system. 2005 European Simulation and Modelling Conference, Porto (Portugal), 24-26 October 2005, pp.572-578.

[Sahraoui A.E.K.,2006] A.E.K.SAHRAOUI "Processes for engineering a system. An introduction to processes, methods and tools" 2nd IEEE International Conference on Information & Communication Technologies: from Theory to Applications (ICTTA'06), Damas (Syrie), 24-28 Avril 2006, 6p.

[Schoenig R. et Aubry J-F. 2006] Raphaël Schoenig, Jean-François Aubry, Thierry Cambois and Tony Hutinet,
'An aggregation method of Markov graphs for the reliability analysis of hybrid systems'. Reliability Engineering & System Safety, Volume 91, Issue 2, February 2006, Pages 137-148

[Seung J.R. et Kosuke I., 2003] Seung J. Rhee, Kosuke Ishii. "Using cost based FMEA to enhance reliability and serviceability". Advanced Engineering Informatics 17 (2003) 179–188

[SN29500 2005] "Siemens SN29500" Siemens internal Data Base for failure rate evaluation of electronically components, dernière actualisation mai 2005.

[Sosnowski J. 2006] Janusz Sosnowski, "Software-based self-testing of microprocessors". Journal of Systems Architecture, Volume 52, Issue 5, May 2006, Pages 257-271.

[Steininger A., 2000]Andreas Steininger, 'Testing and built-in self-test – A survey'Journal of Systems Architecture, Volume 46, Issue 9, July 2000, Pages 721-747

[Sutherland et al., 2003] Sutherland S, Davidmann D, Flake P, „SystemVerilog for design: a guide to using SystemVerilog for hardware design and modeling". Kluwer; 2003.

[Vallé .F et Vernos .D, 2000] VALLÉE .F et VERNOS .D, „Le test et la fiabilité du logiciel sont-ils antinomiques ?". 12ème Colloque national de Fiabilité et Maintenabilité, Montpellier, 2000.

[Vallé .F et Vernos .D, 2002] VALLÉE .F et VERNOS .D, „Comment utiliser la fiabilité du logiciel comme critère d'arrêt du test". 13e Colloque national de Fiabilité et Maintenabilité, Lyon, 2002.

[Vesely et al., 1981] W.E. Vesely, F.F. Goldberg, N.H. Roberts, D.F.Haasl, Fault Tree Handbook, Systems and Reliability Research Office of Nuclear Regulatory Research U.S. Nuclear Regulatory Commission Washington, 1981.

[Villemeur A., 1997] VILLEMEUR; Alain, "Sûreté de fonctionnement des systèmes industriels ", édition Eyrolles, 03/1997, ISBN 2-212-01615-8, disponibilité http://www.eyrolles.com/

[Wang Y., 2004] Y. Wang, Development of a computer-aided fault tree synthesis methodology for quantitative risk analysis in the chemical process industry. PhD thesis, Texas A & M University, 2004.

[Wayne W. 2007] Wayne Wolf, „Hardware and Software Co-design". High-Performance Embedded Computing, 2007, Pages 383-432.

[Weber et al., 2007] C. Simon, P. Weber et E. Levrat, Bayesian Networks and Evidence Theory to Model Complex Systems Reliability, Journal of Computers, 2, 33-43, 2007.

[Wen Y.K. 2001] Y.K. Wen, "Reliability and performance-based design". Structural Safety 23 (2001) 407–428

[Whitesides et al., 2007] Xingyu Jiang, Shuichi Takayama, Robert G. Chapman, Ravi S. Kane and George M. Whitesides, „Micro-scale patterning of cells and their environment". Principles of Tissue Engineering (Thir Edition), 2007, Pages 265-278

[Wirk N. 2003] N.Wirk, Hardware/Software co-design then and now, Information Processing Letters 88 (2003) 83-87.

[Wu Y-F.2007] Yun-Fu Wu, „Correlated sampling techniques used in Monte Carlo simulation for risk assessment". International Journal of Pressure Vessels and Piping, In Press, Corrected Proof, Available online 13 November 2007

[Yalamanchili S., 2001] Yalamanchili S. Introductory VHDL, „from simulation to synthesis". Prentice Hall; 2001.

[Zhan J. et G. Xiong 2006] Jinyu Zhan and Guangze Xiong, „Optimal hardware/software co-synthesis for core-based SoC designs". Journal of Systems Engineering Electronics, Volume 17, Issue 2, June 2006, Pages 402-409

Acronymes

AMDEC : Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité
ANSI: (American National Standards Institute)
APD : Analyse préliminaire de dangers
API : Automates Programmables Industriels
APR : Analyse Préliminaire de Risques
BPCS : (Base Process Control System) système de commande de processus de base
CD : couverture de diagnostic
CIM : (Computer Integrated Manufacturing), fabrication intégrée sous le contrôle de l‟informatique
CEI : Commission Electrotechnique Internationale
CRC : (Cyclic Redundancy Check), Contrôle de Redondance Cyclique
E/E/PE : systèmes électriques, électroniques et programmables E/E/EP
EN: Norme Européenne
EUC : (Equipment Under Control),
FPL : (FPL, Fixed program language), langage de programme figé
ISA: (Instrumentation, Systems and Automation Society)
ISO : (International Standardardisation Organization), Organisme international de normalisation
MDT : (Mean Down Time), durée moyenne d‟indisponibilité,
MTBF : (Mean Time Between Failures), ou durée moyenne entre deux défaillances successives
MTTF : (Mean Time To Failure), durée moyenne de bon fonctionnement avant la première défaillance
MTTR : (Mean Time To Repaire), Durée moyenne de réparation
NCS, (Networked Control System), Systèmes commandés par réseau,
NF : Norme française
OMT : (Object Modeling Technique) technique orientée objet
OSI : (Open Systems Interconnection)
PES : (Programmable Electronic System), système électronique programmable
PFD : (Probability of Failure on Demand), probabilité de défaillance à la demande
PFD_{avg} : (Average Probability of Failure on Demand), moyenne de la probabilité de défaillance à la demande
PFH : (Probability of a dangerous Failure per Hour), probabilité de défaillance dangereuse par heure
PFS : (Probability of Failure to Safe), probabilité de défaillances en sécurité,
RdP : Réseaux de Petri
RFF : (Risk Reduction Factor), facteur de réduction de risque,
SADT : (Structured Analysis and Design Technique) ou Analyse Structurée et Technique de Conception
SAID : Systèmes d‟Automatisation à Intelligence Distribuée
SAP : Système Automatisé de Production
SIF : (Safety Instrumented Function), fonction instrumentée de sécurité
SIL : (Safety Integrity Level), niveaux d‟intégrité de sécurité
SIS : Systèmes Instrumentés de Sécurité

SISID : (Systèmes Instrumentés de Sécurité à Intelligence Distribuée)
SFF : (Safe Failure Fraction), taux des défaillances en sécurité
SNCC : Systèmes Numériques de Contrôle Commande
SRS : (Safety related systems) systèmes relatifs à la sécurité
TOR : Tout Ou Rien (capteurs)
USOM : (USer Operating Mode)

Accident : événement résultant d'une dérive de l'installation et se traduisant par des dommages, c'est à dire des blessures physiques ou atteintes à la santé affectant des personnes soit directement soit indirectement comme conséquence à un dégât causé aux biens ou à l'environnement [CEI1050]

[Situation] accidentogène : (ou situation dangereuse (hazard)): situation potentiellement dangereuse car pouvant entraîner un accident, en absence de la mise en œuvre effective d'actions permettant son évitement.

Accessibilité : propriété pour un graphe orienté, correspondant à l'existence d'au moins un chemin, c'est à dire une séquence de transition, permettant le passage d'un état initial fixé à un état but. L'état but est alors dit « accessible ».

Activation d'erreur matérielle : sollicitation d'une ressource matérielle sous un mode pour lequel l'état d'erreur de la ressource induit sa défaillance

Activation d'erreur « environnementale » : perturbation lors de sa transmission ou de son stockage d'un signal du fait de contraintes environnementales (EMI, température…).

Agrégation d'état : méthode de simplification d'un modèle consistant à fusionner plusieurs états, dont une propriété est jugée équivalente. Cette opération aboutit à la réduction de la complexité du modèle de départ (nombre d'état plus faible) et doit permettre d'estimer l'évolution des nouveaux états résultant de cette opération.

Alphabet : ensemble de caractère (ou de lettres) permettant de générer les mots d'un ensemble de langages.

Amont (respect. aval) : se dit d'une entité sous- fonctionnelle dont le signal de sortie (resp. d'entrée) peut être utilisé par (resp. provient de) l'entité sous- fonctionnelle considérée.

Approche dysfonctionnelle : étude des conséquences de défaillances locales sur les lois fonctionnelles d'un système.

Architecture distribuée : Organisation décentralisée des ressources matérielles et logicielles d'une architecture, induisant la répartition des processus de traitement d'information et de décision entre des ressources matérielles distantes.

Architecture matérielle : décomposition de l'architecture suivant les ressources matérielles permettant de détailler les connections physiques existantes entre celles-ci, ainsi que leurs composants (électroniques, électriques, mécaniques…).

Architecture fonctionnelle : décomposition sous- fonctionnelle d'une architecture.

Architecture logicielle : spécification abstraite consistant à décrire un ensemble de fonctions en définissant leurs opérations, leurs sources d'informations, leurs interfaces et leurs interactions [Hayes94]

Automate d'état fini : définition usuelle (alphabet, ensemble fini d'état, fonction de transition, état initial, ensemble marqué) cf. chapitre 3 (notions de base)

Automate hybride déterministe : Un automate hybride est (Hélias01) un 5-uplet {S, X, u, I, E, F} avec :

S : un ensemble fini de sommets,

X : un espace d'état continu de dimension finie,

u un vecteur d'états évoluant sur l'ensemble S * X,

A est l'ensemble des transitions (auxquelles sont associées une condition de passage portant sur u, appelée Gard, et une opération de transformation des valeurs de u, appelée saut

I une fonction qui à tout sommet associe l'ensemble des variables continues invariantes

E un ensemble d'événement lié aux transitions,

F un ensemble d'application qui associe à chaque sommet de l'automate une règle d'évolution des composants continus de u

Un automate hybride est dit déterministe si l'évolution du vecteur d'état u l'est.

Entité sous fonctionnelle : procédé de traitement de signal utilisant un ensemble fini de signaux d'entrées et devant délivrer un signal de sortie unique, il est basé sur l'application d'une règle d'utilisation de ressources matérielles sur l'ensemble des signaux d'entrées. Cette règle est choisie sur un ensemble fini de règles d'utilisation possibles des ressources matérielles, en regard de l'état des signaux d'entrée à traiter

Boucle de sécurité : architecture opérationnelle (matérielle + logicielle) visant à transformer un ensemble d'information issu de l'observation de l'état de demande au niveau des interfaces d'acquisition (capteurs) en une action adéquate (activation des moyens de réaction ou absence d'action) suivant le cahier des charges de la fonction de sécurité.

Capteurs (ou actionneurs) « intelligents » (smart sensor/actuator) : se dit d'un capteur (ou actionneur) possédant intrinsèquement des capacités
· De test permettant la détection de fautes internes
· De décision, permettant son adaptation fonctionnelle
· Et de communication, permettant la transmission ou la réception de données de diagnostic avec d'autres modules distants.

Chemin : séquence de transition, équivalent à la séquence des événements correspondant à chacune de ces transitions (étiquettes) ….

Concaténation : opérateur de manipulation de mots. Si u = u1 … un et v= u = v1 … vp (avec u1,…, un et v1,…,vp lettre d'un alphabet), alors la concaténation de u et v est le mot u.v = u1 … un v1 … vp. Cet opérateur est non commutatif et possède comme élément neutre le caractère e.

Conflit : présence de propositions explicite et directement contradictoire

Combinaisons d'erreurs matérielles minimales : combinaison d'erreurs matérielles révélées sur une période de référence T, se traduisant par une défaillance du système dans un mode de défaillance fixé.

Consommation d'une information : utilisation et destruction d'une information par une sous-entité fonctionnelle, visant à la délivrance d'un signal de sortie.

Contrainte environnementale (stress) : facteurs pouvant dégrader (stress environnementaux) ou restaurer (réparation) l'état d'une ressource matérielle, ou contribuer à l'occurrence d'une faute environnementale.

Contraintes environnementales partagées : exposition de plusieurs parties du système (ressources) à une contrainte environnementale. Cette contrainte est alors « partagée ».

(Evénements) contributeurs : événement influençant les mécanismes de propagation d'un accident (et donc l'impact associé à celui-ci), mais non suffisants à eux seuls à son occurrence.

COTS : de l'anglais « component on the shell », ressource matérielle ou logicielle existante et pouvant être utilisée dans une architecture, fournie par un tierce. Ces composants imposent des justifications en termes de fonctionnement, de qualité et de tests.

Coupes minimales : plus petite combinaison d'entités entraînant l'échec de la mission du système (elle ne contient aucune autre coupe).

Coupure (coupe-circuit = de-energizing process) : procédure visant à la mise en position sûre du système par l'intermédiaire de relais de coupure sur le circuit de contrôle des actionneurs du système.

Coût d'implantation : coût totale de mise en œuvre d'une modification (développement, test et matériel).

Coût de maintenance : coût induits par les opérations de maintenance (coût humain + coût d'inspection + coût de remplacement) portant sur un ensemble fini non vide de ressources matérielles.

Criticité : produit de la fréquence estimée d'un accident et de son impact.

Décision : Capacité de faire des choix en fonction d'informations disponibles

Déclenchement intempestif d'une fonction de sécurité : (spurious trip of a safety- related function) activation des moyens de réaction d'une fonction dédiée sécurité en l'absence de demande. Cette défaillance peut se traduire par une perte d'exploitation, une baisse de la

qualité des missions de l'installation et/ou une mise en danger de l'installation et de son environnement.

Défaillance : cessation de l'aptitude d'une ressource à accomplir une fonction requise (CEI61508-4)

Défaillance systématique : défaillance liée de manière certaine à une cause, qui ne peut être éliminée que par modification de la conception, du procédé de fabrication, du mode d'emploi, de la documentation ou d'autres facteurs appropriés. (IEC61508)

Défaillance de cause commune (CCF) : défaillance du système d'étude dû à plusieurs événements de défaillances simultanées, résultat d'erreurs dont l'apparition simultanée peut être attribuée à une cause unique (partagée).

Délivrance d'une information : émission par une sous- entité fonctionnelle d'un signal de sortie.

Disponibilité : capacité d'un système ou d'une entité de fournir un service correct quand celui-ci est demandé (notion instantanée).

Défectueux (composant matériel) : état d'un composant matériel le rendant inapte à remplir correctement l'ensemble de ses spécifications fonctionnelles.

Données : éléments bruts, de nature mesurable, n'ayant pas été interprété

Durée de test : nombre moyen de cycle (de largeur T) permettant la réalisation complète d'un test (balayage de son spectre de détection).

EMI : de l'anglais, interférences électromagnétiques, type de contraintes environnementales pouvant affecter l'état d'un signal transmis ou stocké. On parle en général d'EMC (compatibilité électromagnétique) quand on considère un support de transmission ou de stockage d'information est robuste vis-à-vis d'un niveau d'interférence magnétique estimée au regard des interférences auxquels il pourrait être soumis du fait de son environnement architectural (matérielle) ou de sources externes.

Environnement : ensemble extérieur au système d'étude, incluant l'installation industrielle et les contraintes qu'elle peut faire porter sur l'évolution temporelles des contraintes environnementales.

Erreur : Ecart ou discordance entre une valeur ou une condition calculée, observée ou mesurée, et la valeur vraie, prescrite ou théoriquement correcte.

Erreur de conception : Ecart entre le cahier des charges fonctionnel ou sous- fonctionnel et le fonctionnement de l'architecture proposée en l'absence de défaillances matérielles.

Erreur humaine : Ecart entre les actions menées et les actions prescrites (par exemple dans les tâches de maintenance).

Information erronée : Existence d'un écart entre l'état d'une information et l'état attendu de celle-ci sous des hypothèses de bon fonctionnement du système, en regard de l'état de demande (présence/absence) observable ayant induit sa délivrance.

Erreur matérielle : Ecart entre les règles de fonctionnement d'une ressource matérielle et celles attendues, dans le cas où aucun de ses composant n'est défectueux. On peut classer les erreurs matérielles suivant les modes de défaillances par lesquelles elles se manifestent.

Erreur environnementale : Conséquence d'une faute environnementale (situation de stress) induisant lors de l'échange ou le stockage la génération d'une erreur sur un signal et la perturbation de l'information lui étant rattachée.

Erreur de type « commande » : Erreur dont l'existence disparaît avec sa cause.

Erreur matérielle d'ordre 1 : Erreur matérielle suffisante à l'apparition d'un mode de défaillance du système d'étude.

Etat d'erreur : présence ou absence d'une erreur (matérielle, environnementale…) sur un cycle T donné.

Etat « puit » : état d'un automate d'état ne possédant aucun arc sortant.

Etude préliminaire de danger (PHA : preliminary hazard analysis) : Approche visant à identifier les situations potentielles d'accidents et à estimer leur probabilité d'occurrence et leurs conséquences en terme d'impact pour une installation existante

Evénement initiateur : événements permettant de figurer la réception par une entité sous fonctionnelle d'un ou plusieurs signaux d'entrées.

Evénement « certain » : événement dont la probabilité est égale à 1.

(Evénements) mitigatifs : événements dont l'occurrence à tendance à réduire la vitesse du processus de destruction des biens ou des personnes, voir à l'arrêter, et réduit de ce fait l'impact d'un accident.

Evénement « échec de test » : événement permettant de figurer, sous l'hypothèse d'une réalisation normale d'un test en ligne la non- détection par celui-ci, sur un cycle T du système d'étude, d'une combinaison d'erreurs appartenant pourtant á son spectre de détection.

Faute : toute cause adjugée ou hypothétique (événement, circonstance) de la présence d'une erreur.

Faute environnemental : passage à un niveau de stress environnemental (interférences électromagnétiques, température) induisant l'altération d'un signal.

Fiabilité : Aptitude d'une entité à accomplir une fonction requise, dans des conditions données, pendant un intervalle de temps donné.

Fiabilité dynamique : discipline permettant de prendre en compte l'interaction entre l'installation, en y incluant le système d'étude, et leur environnement et de simuler l'évolution

de celui-ci par des modèles (automates hybrides, réseau de Pétri) dont les conditions de transition sont simulées par tirs probabilistes, suivant des profils de réaction de l'installation.

Flux élémentaire d'information : transfert d'une information de sortie d'une sous- entité fonctionnelle à l'entrée de la sous- entité fonctionnelle suivante.

Flux d'information : toute séquence de sous- entité fonctionnelles pouvant être reliées par des flux d'information élémentaire (chemin).

(Approche) FMDS (RAMS en anglais) : acronyme correspondant au sigle fiabilité, disponibilité, maintenabilité, sécurité. Approche visant à mettre en évidence les liens entre ces attributs et d'intégrer une approche d'allocation d'objectifs en terme de performances fonctionnelles fiabilistes, basées sur l'estimation des conséquences des modes d'interactions fonctionnels et une évaluation de ceux-ci en fonctions des choix architecturaux (matériels, fonctionnels) et les politiques de maintenance préconisées.

FTA : arbre de défaillance (fault tree analysis).

FME(C)A(D) : failure mode and effect analysis. Analyse de conséquences par exploration des défaillances locales du système, détermination de leur conséquence directe (en absence d'autres défaillances). Elle peut être complétée par une étude exploratoire des moyens de détections attachées aux erreurs en étant la cause (ajout de la lettre D) et d'une analyse de criticité de ces conséquences (ajout de la lettre D).

Fonction temps réelle : fonction devant satisfaire des contraintes de temps de réponse explicites et bornées, le non-respect de ces bornes entraînant l'apparition de dégradations de performances et de mauvais fonctionnement [Pal98].

Hiérarchisation : ordonnancement suivant une relation d'ordre partielle ou complète.

HAZOP : Analyse de risque et d'opérabilité. Analyse exploratoire basée sur l'identification d'événements potentiellement dangereux, la recherche de toutes les causes de ces événements, l'analyse des conséquences liées à ces dérives. L'étape d'identification est basée sur une approche de type « mots-clefs », comme le montre des travaux existants sur l'utilisation d'une telle méthode sur un „process" chimique et aéronautique (Université de York).

(Processus Markovien) homogène : processus de Markov dont les taux de transitions sont supposés comme constants au cours du temps.

Horloge de contrôle (Watch Dog) : dispositif permettant le contrôle de la bonne réception d'une information sous une contrainte de temps, dite d'acceptation, ou de la bonne réalisation d'un ensemble de fonction. Il permet en cas de non- validation de cette condition, de générer un signal de diagnostic pouvant être utilisé par des fonctions de recouvrement fonctionnelles ou d'affectation par défaut de variables.

Indépendance (terminologie probabiliste) : deux événements sont dits « indépendants » si la probabilité de la combinaison de ces deux événements est différente de celles du produit de ces événements pris séparément.

Impact : conséquences en termes de dégâts matériel, humain et/ou environnementaux directe, indirecte ou différée d'un accident, mesuré en fonction d'une échelle de gravité préalablement choisie comme référentiel.

Information : signal auquel a été adjointe une signification.

Initiation d'erreur d'information : délivrance par une entité sous- fonctionnelle d'une information erronée alors que l'ensemble de ses informations d'entrées ne le sont pas.

Inhibition d'erreur d'information : délivrance par une entité sous- fonctionnelle d'une information non erronée alors qu'au moins une de ses informations d'entrée est erronée.

Intégrité : propriété d'absence d'altération pour une entité.

Langages marqués : langage constitué de l'ensemble des séquences d'événements (représentés par leur caractère) permettant le passage de l'état initial d'un automate d'état fini à au moins un de ses états marqués.

(Erreur) latente : erreur non activée sur un cycle T du système d'étude.

Listes caractéristiques : combinaisons d'erreurs dont l'existence est nécessaire et suffisante à l'apparition d'un mode de défaillance du système d'étude et dont l'activation est certaine au vue des choix architecturaux de celui-ci.
La liste caractéristique pour un déclenchement intempestif de la fonction dédiée sécurité.
Ld liste caractéristique pour une défaillance sous demande de la fonction dédiée sécurité.

Logique linéaire : Logique issue de la logique intuitionniste, basée sur les concepts de présence ET de validité d'une proposition, et se distinguant de ce fait de la logique classique, attachées à des conditions de vérité. Elle peut être vue comme un raffinement et une symétrisassions de la logique intuitionniste. Une définition précise de sa sémantique de base est donnée par [Girard87]. Un énoncé s'y interprète comme l'ensemble de ses démonstrations, et une implication A => B est vue comme l'ensemble des fonctions qui transforment les démonstrations de A en démonstrations de B…..

Loi exponentielle : loi de dégradation pour laquelle la fonction de survie d'un composant à un profil exponentiel par rapport au temps.

Loi de Weibull : loi de dégradation, souvent utilisée pour les composants mécaniques, basée sur trois paramètres estimés et permettant de prendre en compte l'usure du composant et donc l'effet de l'accumulation de contrainte dans celui-ci au cours du temps (équation donnée au chapitre 3).

Maintenabilité : Aptitude d'une entité, dans un contexte d'utilisation fixé, à être maintenue ou rétablie dans un état dans lequel elle peut remplir une fonction requise, sous des conditions de respect des procédures et moyens prescrits à cet effet.

Maîtrise des risques : démarche visant à d'estimer le niveau de risque de son installation, prendre les mesures qui s'imposent pour les détecter au plus vite, déclencher des actions d'évitement de l'accident ou de réduction de son impact, ayant pour but de garantir un niveau de risque tolérable à une installation.

Graphes de Markov : processus markovien basé sur une échelle de temps continu.

Chaîne de Markov : processus markovien basé sur une échelle de temps discrète.

Mise en danger : apparition d'une situation accidentogène vu comme la conséquence directe de l'occurrence d'un ou plusieurs événements générés par l'installation, le système d'étude ou une interaction de ces deux systèmes.

Mise en position « sûre » : (abus de langage, traduction anglo-saxonne de « safe state ») activation forcée des moyens de réaction du système d'étude visant à un retour à une situation non dangereuse de l'installation suite à la détection d'une défaillance de celui-ci.

Mode de défaillance : ensemble des effets, dans un contexte de sollicitation fixée, par lequel une défaillance est observée [Zingelstein85].

Mode de défaillance latent : mode de fonctionnement du système se traduisant par un service correct de celui-ci qui induirait suite à l'occurrence d'une défaillance locale supplémentaire une défaillance du système alors que cet événement n'est pas une condition suffisante d'apparition d'une défaillance du système.

Mode de fonctionnement dégradé du système d'étude : mode de fonctionnement incomplet du système, qui en présence de demande ne permet pas d'éviter un accident mais en atténue les conséquences (se traduisant donc par une réduction de son impact)

Opération de réduction de langage : opération visant à réduire la taille d'un langage (par l'exclusion de mots permettant d'éviter les séquences conflictuelles) ou de ses constituants (par suppression de caractères inutiles pour le but recherché)

Opération de croisement de langage : opération sur les mots de deux langages, introduit dans le chapitre 3 permettant de prendre en compte l'utilisation par une entité décisionnelle (bloc IP) d'informations soumise à des sources ou à des influences non indépendantes (partage de ressources éventuelles (de nature informationnelle ou matérielle))

Pas d'itération d'une chaîne de Markov : largeur du pas d'incrémentation temporelle pour une chaîne de Markov

Panne (en anglais fault) : Etat d'un composant matériel (électrique, électronique ou mécanique) le rendant inapte à accomplir sa mission.

Partage de ressource matérielle : utilisation par deux entités sous- fonctionnelles distinctes d'une même ressource matérielle pendant un intervalle de référence T.

Partage d'information : utilisation par deux entités sous- fonctionnelles distinctes d'une même information pendant un intervalle de référence T.

Perturbation d'information passage d'une information d'un état correct à un état erroné ou d'un état erroné à un état correct.

Phase de jeunesse d'un composant : phase précédent l'implantation durant laquelle un composant matériel est amélioré, elle vise à réduire son taux de panne lors de sa durée d'utilisation future, par une maîtrise des politiques de production et de test de celui-ci.

Propagation d'erreurs d'information : transmission d'une information erronée le long d'un flux d'information.

Propagation de langage : utilisation d'un langage généré par une entité sous- fonctionnelle comme événement initiateur de l'automate d'état fini représentant l'entité fonctionnelle suivante (avale le long du flux d'information).

Qualité de service : l'ensemble de ses caractéristiques justifiant de l'aptitude d'un système durant sa durée d'utilisation à satisfaire à la fois les besoins exprimés (i.e. capacité d'évitement d'accident pour notre système d'étude) et implicites (i.e. non perturbation des missions du système en absence de situation dangereuse pour notre système d'étude) ISO84102.

RBD (diagramme de fiabilité) : Modèle d'une entité ou d'un système par une approche diagramme bloc de type « tout ou rien ». Qui se base sur l'indépendance des modes de défaillances attribuées à chaque bloc, et vise à évaluer la fiabilité de l'entité représentée par des opérations simples d'additions et de multiplications des probabilités de défaillances de chaque bloc (supposées constantes pour une approche RBD classique).

RdPs = GSPN : extension des réseaux de Pétri classiques, réseau de Pétri dont les transitions sont conditionnés par des événements déterministes et aléatoires.

Réparation : action visant à la remise en état d'une ressource matérielle ou logicielle (remplacement éventuel de tout ou partie de la ressource). On définit le MTTR (temps moyen de réparation), comme l'espérance mathématique de la somme de la durée d'inspection nécessaire au choix de cette action et de la durée de mise en œuvre effective.

Redondance : doublement d'une voie de traitement d'information dans le but affirmé d'augmenter la fiabilité locale en s'appuyant sur les informations délivrées par ces deux voies (jugées équivalente) de manière comparative ou complémentaire (type de redondance : Redondance active Redondance passive …)

Ressource : chose qui peut être consommées, utilisée et produite

> Ressource matérielle ensemble de composants électronique et programmable, pouvant remplir une fonctionnalité unique (et donc non reconfigurable) durant la durée de vie du système étudiée, dont la réparation est menée globalement et pour lequel ces procédures de réparation (préventive et ou corrective) implique une indisponibilité de l'ensemble des composants durant la période de réparation.
>
> Ressources matérielles critiques ressource matérielle dont la défaillance induit de manière suffisante celle du système.
>
> Ressource informationnelle signal auquel est adjoint une signification et ayant pour but de permettre de manière directe ou indirecte la réalisation de la fonction de sécurité. On distingue les ressources informationnelles de classe 1 (F- et D-informations), des ressource informationnelles de classe 2 (instructions, code).

Risque : mesure d'un danger associant une mesure de l'occurrence d'un événement indésirable et une mesure de ses effets ou conséquences, définie en général comme le produit de la probabilité d'occurrence de cette situation et de son impact.

> Risque résiduel somme des risques possibles suite à l'occurrence de la situation accidentogène i ou au déclenchement intempestif des fonctions de sécurité basées sur sa détection.
>
> Risque tolérable (ou acceptable) niveau d'acceptation du risque (global pour une installation, individuel sur le risque résiduel d'une situation accidentogène précise).

Robustesse : capacité de résistance vis-à-vis de contraintes environnementale d'une ressource matérielle ou d'une entité lui permettant de continuer à offrir un service correct ou suffisant au regard de son cahier des charges fonctionnel.

Séquence d'activation d'erreurs : séquence d'événements d'activations d'erreurs (matérielles, environnementale) et/ou d'événements « échec de test ».

Signal : ensemble structuré de données.

SIL (Safety Integrity Level) : niveau de confiance (introduit par l'IEC61508) vis-à-vis d'une fonction EP dédiée sécurité basée sur l'application de normes de qualité, dans les phases de définition, de conception et de vérification, mais aussi de contrôle de bonne mise en œuvre des politiques de suivi et de maintenance d'une architecture dédiée à la sécurité et à la vérification des objectifs de performances fiabilistes d'une telle fonction par l'évaluation quantitative de son seul PFDavg.

(Défaillances) Simultanées : occurrence de deux événements de défaillance sur un même cycle T.

(Erreurs) Simultanées : existence de deux erreurs (matérielles ou environnementales) sur un même cycle T.

Stratégie : Manière d'organiser et de coordonner une série d'actions, un ensemble de conduites en fonction d'un résultat.

Sûreté de fonctionnement : ensemble des propriétés permettant de placer une confiance justifiée dans le service délivré par un système au vu de ses utilisations.

Système (d'étude) : ensemble des ressources matérielles (éventuellement maintenus correctivement et/ou préventivement) et logicielles visant à assurer correctement la fonction de sécurité et, ce quelque soit l'état de demande.

Taux de panne d'un composant matériel : limite du quotient de la probabilité conditionnelle d'apparition d'une panne d'un composant matériel sur un intervalle de temps, sachant que ce composant est non défectueux au début de l'intervalle, sur la largeur de cet intervalle quant cette valeur tend vers 0 (par valeur positive).

Temps critique de réaction Tc : temps maximal pour lequel, suite à l'occurrence d'une demande, la non- activation de contre-mesures ne débouche pas sur un accident.

Temps caractéristique, T : durée séparant deux observations successives de l'état de demande par le système d'étude.

Temps de cycle : (£ T) temps maximal admis entre l'acquisition d'un état de demande par une boucle de sécurité et la réception par le circuit de contrôle des ses actionneurs en résultant.

Temps de panne latente: Intervalle de temps entre une défaillance et la détection de panne qui en résulte.

Temps inertiel Ti : Temps maximum entre la réception par l'ensemble des moyens de réactions (actionneurs) d'un ordre de contrôle et l'action correspondante en absence de défaillance de celui-ci.

Temps de réaction : temps maximal admis entre l'acquisition d'un état de demande par une boucle de sécurité et la réception par le circuit de contrôle des ses actionneurs en résultant

Temps de réponse : Temps maximal nécessaire à la réaction de la fonction de sécurité pris comme la somme du temps de cycle T et du temps inertiel Ti.

Temps (moyen) de réparation (MTTR, TR) : temps moyen (espérance mathématique) du temps nécessaire à l'inspection hors-ligne et à la réparation d'une ressource lors d'une opération de maintenance. On distingue la durée moyenne pour une opération de maintenance corrective, notée MTTR (mean time to repair) et la durée moyenne d'une opération de maintenance préventive, notée TR.

Tolérance aux fautes : Propriété correspondant à la capacité d'une entité d'accomplir une fonction malgré les pannes (fault) de certains de ses constituants.

UML : (Unified Modeling Language, traduisez "langage de modélisation objet unifié") est une notation permettant de modéliser un problème de façon standard. Elle est née de la fusion des trois méthodes qui ont le plus influencé la modélisation objet au milieu des années 90 : OMT, Booch et OOSE Unified.

Utilisation d'une ressource matérielle : manière de tirer parti d'une ressource matérielle pour mener à bien une opération de traitement de signal.

Annexe A

Résultats de l'étude qualitative

Composant	Fonction	Mode de défaillance	Cause	Méthode de protection des pannes	Impact et criticité de défaillance
CONVERTISSEUR TENTION-COURANT.	Convertisseur de tension en courant.	Absence de courant. Courant trop faible.	Défaillance du composant. Absence de calibrage.	Redondance matériel + technique de détection	Faible
CONVERTISSEUR COURANT-CHAMP MAGNETIQUE.	Convertisseur du courant en champ magnétique.	Absence de champs magnétique, ou champs insuffisant.	Défaillance des bobines.	Checksum Contrôle de parité	Moyen
GENERATEUR DES SIGNAUX.	Générer deux signaux carrés.	Sortie soit collée au niveau soit au niveau bas. Modification de fréquence de cadence.	Bruit dans le matériel. compatibilité électromagnétique	Technique d'étalonnage	Moyen
CONVERTISSEUR DU SIGNAL.	Convertisseur du signal carre en un signal triangulaire.	Collage des données. Saturation de trafic. Bit flip.	Compatibilité électromagnétique. .Défaillance matérielle	Technique de control fonction de transfert.	Grave
ALIMENTATION STABLE.	Tension stable aux bornes de la MTJ.	Absence de courant. Courant trop faible.	Défaillance matériel. Absence de calibrage.	Redondance matériel + technique de détection.	Moyen
MTJ CAPTEUR.	Convertisseur du courant en champ magnétique.	Absence de champs magnétique, ou champs insuffisant.	Défaillance des bobines.	Redondance Matériel	Grave

Composant	Fonction	Mode de défaillance	Cause	Méthode de protection des pannes	Impact et criticité de défaillance
FILTERE AMPLIFICATEUR.	Filtrer les signaux basse fréquence. Amplifier le signal filtré.	Tension incorrect, ne correspond pas à la caractéristique de l‟AO.	Variation climatique (bruit). Variation de la caractéristique de l‟AO (vieillissement)	Redondance matériel + technique de détection	Faible
CONVERTISSEUR DU SIGNAL EN SERIE D‟IMPULSION.	Convertisseur du signal en une série d‟impulsions.	Sortie collé au niveau haut ou bas Passage intempestif haut-bas.	Défaillance du composant + perturbation climatique (température et bruit).	Technique proposé de détection	Grave
CODEUR DU SIGNAL.	Modulateur de largeur d‟impulsion.	Sortie soit collée au niveau soit au niveau bas. Modification de fréquence de cadence.	Bruit dans le matériel. Surtensions. Compatibilité électromagnétique.	duplication matériel, et technique de concurrence.	Moyen

Tableau A.A.1 **:** AMDEC de la partie de traitement analogique.

Partie numérique :

Composant	Fonction	Mode de défaillance	Cause	Méthode de protection des pannes	Impact et Criticité.
DATA STACK	Mémorisation des variables et les données. Garde l'adresse des données à manipulées par un programme.	Bit flip (changement de donnée). Lecture erronée d'une donnée (Valeur erroné de l'adresse d'une valeur = lecture erroné d'une valeur)	Bruit dans le matériel. Décharges électrostatiques. Compatibilité électromagnétique.	Méthode Checksum	Grave
RETURN STACK	Garde la valeur issue de l'UAL comme résultat de calcul.	Bit flip. Lecture erronée d'une adresse.	Bruit dans le matériel. Surtensions. Compatibilité électromagnétique	Contrôle de parité. Méthode Checksum.	
I/O system	Lecture-écriture de données via l'extérieur	Bit flip.	Bruit dans le matériel. Surtensions. Compatibilité électromagnétique		
DATA BUS	Transporte l'infomation entre les différentes parties de microprocesseur.	Collage des données. Saturation de trafic. Bit flip.	Compatibilité électromagnétique Défaillance matérielle.		
CONTROL LOGIC & IR	Supervision de trafic sur le Bus. Supervision et déclanchement des toutes les opérations	Désynchronisation entre l'horloge et les opérations. Perte de fonction ou fonction intempestive.	Compatibilité électromagnétique Défaillance matérielle.	Méthode Checksum Contrôle de parité.	

Composant	Fonction	Mode de défaillance	Cause	Méthode de protection des pannes	Impact et Criticité
BUS	Transfert des données entre les différentes parties de Processeur	Collage des données. Saturation de trafic	Défaillance matérielle. Compatibilité électromagnétique	Méthode Checksum. Contrôle de parité	Moyen
PC: Program Counter	Indication sur l'état de validité de l'opération. Indication de la prochaine instruction	Fonction intempestive (déclenchement d'une fonction sans demande)	Bruit dans le matériel Décharge électrostatique- Compatibilité électromagnétique	Méthode Checksum d'erreur au niveau de communication avec le contrôleur logique.	Grave
UAL	Les opérations arithmétiques et logiques.	Bit flip. Erreur dans le code d'opération.	Bruit dans le matériel. Compatibilité électromagnétique Surtension.		

Tableau A.A.2 **:** AMDEC de la partie logique programmable.

Annexe B

Calcul de taux de défaillance des composants électroniques Nombre de bit-flip par cycle d‟horloge

Dans ce paragraphe nous indiquons les bases de données les plus utilisées pour des composants électroniques, elles sont regroupées dans le tableau suivant (tableau A.B.1).

Source	Titre	Editeur	Dernière version
FIDES	Méthodologie de fiabilité pour les systèmes électroniques	DGA-DM/STTC/CO/477	2004
IEEE STD	IEEE Guide to the Cellection and Presentation of Electrical, Electronic Sensing Component and Mechanical Equipement Reliability Data for Nuclear Power Generating Station.	Institut of Electrical and Electronic Engineers, New York, USA.	IEEE STD500, 1984
MIL-HDBK-217K	Military Hndbook-Reliability Prediction of Electronic Equipment.	United States Department of Defense.	MIL-HDBK-217F, notice 2,28 Février 1995.
BT-HRD	Handbook for Reliability Data.	British Telecommunications.	HRD 5, 1995
EPRD	Electronic Parts Reliability Data.	Reliability Analysis Center, RAC, NEW YORK, USA.	EPRD 97, 1997
GJB	Chinese Military Standard.	Beijing Youtong Forever Sci.-Tech.	GJB/Z229B, 1998.
RDF (CENT)	Recueil de Données de Fiabilité.	Centre National d‟Etudes des Télécommunications, UTE, Paris, France.	RDF 2000-UTE. C80-810, Juillet 2000.
Telcordia/Bellcore	Reliability Prediction Procedure for Electronic Equipment.	Telcordia Technologies, New Jersey, USA.	Telcordia SR-332 Issue1, Mai 2001.

Tableau A.B.1 **:** Bases de données de la fiabilité électronique.

L‟évaluation de la fiabilité prévisionnelle de composants électronique reste incontournable dans les études de la sûreté de fonctionnement et surtout dans les systèmes électroniques

programmables, malgré les efforts fournis pour améliorer ou mettre à jour les bases de données classiques. Devant cet état de l'art, et depuis 2004 un consortium des grosses compagnies a proposé FIDES comme une nouvelle référence pour le calcul de la fiabilité prévisionnelle des composants électroniques agissant sur la précision du calcul final du taux de défaillance.

➢ Modèle de fiabilité prévisionnelle pour les composants électroniques : FIDES

On trouvera dans ce paragraphe un modèle de fiabilité prévisionnelle pour les composants électroniques. Modèle FIDES général.
L'expression du taux de défaillance dépend de plusieurs facteurs dont la technologie de conception, la technologie de fabrication et l'environnement de fonctionnement du composant [FIDES, 2004]. Ainsi, le taux de défaillance dépend d'un taux de défaillance de base, pondéré par des facteurs de technologie, de conception, de fabrication, d'utilisation, d'environnement,...

Actuellement, dans le domaine de l'électronique, ces calculs sont principalement réalisés en se basant sur des standards tels que la MIL-HDBK-217F [MIL-HDBK 95] ou l'UTEC 80-810 [UTEC 80].

De fait, il est courant que les concepteurs appliquent des facteurs correctifs "fait maison" basés sur des retours d'expérience «Rex», des habitudes d'utilisation, des coutumes ou des avis d'experts.

Les différents standards existants (MIL-HDBK- 217F, UTEC 80-810, PRISM, BellCore, British Telecom HRD, …).

De part sa notoriété, le manuel de prédiction de fiabilité MILHDBK-217F, est devenue aujourd'hui une exigence incontournable dans la majorité des spécifications techniques. Il comprend un ensemble de modélisations de taux de défaillances pour de nombreux composants électroniques tels que les circuits intégrés, les transistors, les diodes, les résistances, les condensateurs, les relais, les Switch, les connecteurs, etc.

L'évaluation du taux de défaillance d'un composant (λp) est basée sur le principe suivant :

$$\lambda p = \lambda_b \, \pi T \, \pi Q \, \pi E \, \pi S \, \pi R \, \pi C \quad [1]$$

λb représente le taux de défaillance de base du composant (exprimé à partir d'un modèle classique par exemple la loi exponentiel) tandis que les facteurs $\boldsymbol{\pi}$ traduisent des contraintes telles que la température d'emploi, la qualité de fabrication, des contraintes environnementales, etc. Le calcul du taux de panne d'une carte est basé sur un modèle série des composants considérés.

Cependant, la dernière mise à jour de cette norme date du début des années 90 et à cause de l'évolution des technologies les composants récents ne peuvent être traités sans approximation. De plus, de part l'origine militaire du recueil, certains composants sont mal pris en compte. Enfin, le MIL-HDBK-217F ne permet pas de considérer les phases de non

fonctionnement, obligeant à utiliser soit des coefficients maison, soit d'autres standards tels que RADC-TR-85-91 [RADC-TR 91] et le Reliability Toolkit [RLRT 93].

Une autre approche consiste à utiliser la méthode de prévision de fiabilité RDF 2000 (ou UTEC 80-810 ou IEC 62380) qui contrairement au MIL-HDBK-217F prend en considération les effets des différentes phases du profil de mission sur les composants en fonctionnement ou non. Cette méthode prend en compte les effets des cycles thermiques sur le taux de défaillance des composants en raison des variations de la température ambiante et/ou de l'équipement en marche ou à l'arrêt. L'influence de la tension aux bornes du composant et la complexité technologique sont également considérées. Contrairement au MIL-HDBK-217F, l'usage de l'UTEC 80-810 est peu répandu, car il apparaît peu adapté pour les environnements sévères. En effet les données de fiabilité sont issues ou extrapolées à partir de données de retour d'expérience observées dans des environnements favorables et défavorables.

Relativement à ce constat, FIDES est née en 2004, c'est une méthodologie globale d'ingénierie de la fiabilité en électronique [FIDES 04] et [FIDES-Lamdmu 04], qui porte sur une évaluation prévisionnelle de la fiabilité et sur une maîtrise du processus de fiabilité. Au travers des paramètres technologiques, physiques ou de processus, FIDES permet d'agir sur l'ensemble du cycle de vie des produits pour améliorer et maîtriser leur fiabilité.

L'équation générale de FIDES est :

$$\lambda = \lambda_{Physique} \prod_{Part-manufacturing} \prod_{Process} \quad [2]$$

Où :

- ✓ $\lambda_{physique}$ représente la contribution physique :
- ✓ $\Pi_{Part\_manufacturing}$ traduit la qualité et la maîtrise de fabrication du composant,
- ✓ $\Pi_{Process}$ représente le processus de développement, de fabrication et d'utilisation du produit.

Le facteur Π_{induit} prend en compte les facteurs $\Pi_{Placement}$, $\Pi_{application}$ et $\Pi_{durcissement}$.

L'objectif de FIDES est de permettre une évaluation réaliste de la fiabilité des composants électroniques, pour l'ensemble des environnements, en considérant le profil d'emploi propre de chaque équipement.

➢ Hypothèses prises

Les valeurs par défaut proposées par le guide FIDES ont été utilisées à savoir :

- ✓ $\Pi_{Part\_manufacturing} = 1.6$
- ✓ $\Pi_{Process} = 4$
- ✓ $\Pi_{Durcissement} = 1.7$
- ✓ $\Pi_{Placement} = 1$

Ce choix a été motivé par le souhait de ne pas influencer les résultats obtenus par FIDES (ces paramètres intervenant directement comme des multiplicateurs) et de nous permettre de réaliser des comparaisons objectives dans le même référentiel relatif.

Famille	Sous-famille/λ (en fit)	λ_{OB}PRISM	λ_{REF}RDF	λ_{0TH}FIDES	λ_{OB}PRISM / λ_{0TH}FIDES	λ_{REF}RDF / λ_{0TH}FIDES
circuits intégrés MOS	numérique, hermétique	0,23	2,04	0,056	4	36
	linéaire, hermétique	0,084	9,2	0,27	0,3	34
	SRAM 4Mb	0,008	11,52	0,067	0,12	172
	DRAM 128Mb	0,008	18,4	0,091	0,09	202
	Flash EPROM 128 Mb	0,008	182,4	0,025	0,32	7296
	Flash EPROM 256 Mb	0,008	348,8	0,025	0,32	13952
circuits intégrés bipolaires	Linéaire	-	47,3	0,059	-	802
	mixte basse tension	-	22,7	0,34	-	67
transistors	JFET	0,078	0,75	0,0146	5	51
	LF, bipolaire	0,23	0,75	0,017	14	44
diodes	Régulation Zener	0,41	0,4	0,038	11	11
	LF, usage général	0,062	0,07	0,0315	2	2

Tableau A.B.2**:** Résultats FIDES pour des composants électroniques.

Ce tableau présent des résultats issus de différentes méthodes, c'est une analyse de sensibilité attachée à l'étude des taux de défaillances de base λ_0 de FIDES en les comparants aux autres guides récents RDF2000 [RDF 00] et PRISM [PRISM 00].

On constate qui il est impossible de comparer les taux de défaillance de base directement car la philosophie de construction des modèles est différente et seule la comparaison du modèle complet est réalisable. Toutefois, il est intéressant de regarder les tendances.
Ainsi, on constate que les λ_0 des puces (tableau A.B.2) sont assez différents entre les différentes méthodes.

➢ Données du retour d'expériences (REX)

La méthodologie d'estimation de la fiabilité opérationnelle s'appuie sur les données obtenues par retour d'expériences (REX).

Le REX vise une meilleure connaissance du comportement des composants, de leurs modes de dégradation, de dysfonctionnement ou d'endommagement. Il est basé sur la collecte et la gestion des faits techniques, observés pendant toute la durée de vie du système, de sa mise en service jusqu'à sa désintégration. Dans ce contexte, la constitution d'un échantillon correct de données pour l'estimation d'un modèle de vie d'un système passe par la reconstruction de l'histoire complète d'exploitation de ce système, pour une période d'observation déterminée.
Plusieurs précisions sont à définir sur les données à collecter en exploitation :

- ✓ Systèmes qui font l'objet de cette collecte,
- ✓ Informations à collecter pour ces systèmes,
- ✓ Structure de données adoptée,
- ✓ Niveau de qualité souhaitable pour la collecte.

Exploitation des données du REX suppose trois phases :

- ✓ Sélection, parmi les systèmes suivis, d'un échantillon pertinent,
- ✓ Reconstruction de l'histoire de chaque matériel pris en compte,
- ✓ Calcul des temps jusqu'à la défaillance ou jusqu'à censure pour l'échantillon choisi.

La fonction de vraisemblance nécessite l'accès à des données de défaillances issues des banques de données de REX. Elle permet ainsi de déterminer la loi de fiabilité observée du système à partir des observations de défaillance.

En même temps, l'objectif du REX est l'amélioration de la qualité et de la fiabilité des systèmes et pour cet objectif le traitement de données brutes sera indispensable. La qualification et la validation de données du REX sont nécessaires avant tout calcul. Ces deux étapes représentent la tâche la plus difficile et la plus critique du REX, mais elles sont indispensables pour obtenir des résultats crédibles.

Les données nécessaires pour établir un échantillon pour construire la vraisemblance sont :

- ✓ Date de la mise en service du système,
- ✓ Dates de défaillances du système,
- ✓ Dates de censure, la fin d'observation des essais.

Toutes les banques de données du REX doivent permettre l'extraction de ces données.
La construction d'un échantillon des données collectées par le REX pour estimer le modèle de vie associé au système passe par la valorisation de l'exploitation du système pendant une durée d'observation. Cette durée est plutôt courte afin de permettre le calcul de la fiabilité opérationnelle le plus vite possible. Contrairement aux données d'essais, la taille d'échantillon est souvent importante pour les données issues du REX.

Annexe C

Génération des listes caractéristiques

Pour illustré notre démarche de génération des listes, nous considérons un bout du schéma RTL (Register Transfer Level) d'un processeur à pile, le model de haut niveau d'entités sous fonctionnelles est :

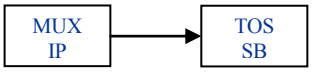


Figure A.C.1 : Modèle de haut niveau de deux composants

- SB : Entité de stockage du signal, elle permet la mémorisation sur un support d'un signal d'entré et sa restitution ultérieure, ces entités peuvent permettre de représenter les procédés de mémorisation nécessaire à la synchronisation des entités d'un processus de décision.

- IP : Entité de décision, elles permettent de délivrer une information de sortie au regard de deux informations d'entrées.

L'étape suivante consiste à modéliser le comportement fonctionnel et dysfonctionnel de chaque entité (IP, SB) du model de haut niveau. Nous traduisons chaque entité par un automate d'état fini. Nous classons les événements de transitions des ces automates, en associant à chacune des transitions entre états un attribue. Nous avons modélisé principalement les défaillances transitoires (bit-flip) liées directement à la qualité de l'information, c'est un mode de défaillance transitoire relatif à la disponibilité du système. Nous avons aussi modélisé la défaillance liée aux problèmes matérielles, ceci est relatif à la fiabilité du système vue sa nature permanente.

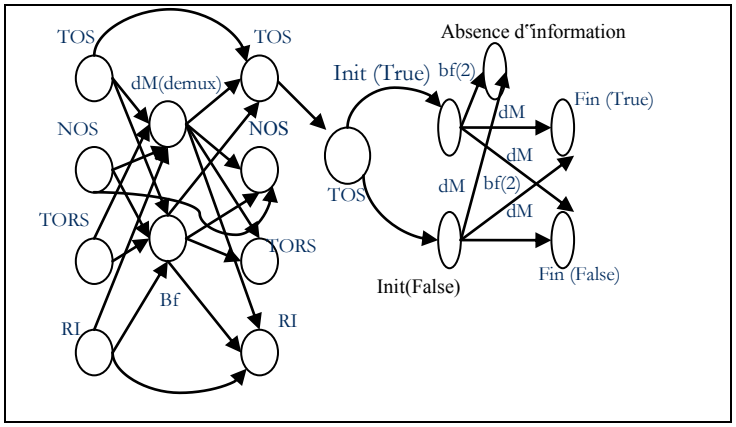


Figure A.C.2 : Modèle de bas niveau de la figure A.C.1

Nous somme devant le cas d'agrégation série, où la procédure de propagation de langage est basé sur l'héritage de la propriété bloc amont à bloc aval, En effet, pour la génération de listes finales sans perte de l'information, nous appliquons les règles de concaténation suivants :

- Les événements de classe intermédiaire de chaque mot m seront remplacés par l'élément neutre pour la concaténation ε, leur trace ne sera donc pas conservée.
- Les événements apparaissant plusieurs fois dans un mot m seront remplacés par l'élément neutre pour la concaténation des mots ε, leur trace ne sera donc pas conservée.

Selon ce model de bas niveau de la figure A.C.2, il était possible de citer tous les chemins probables de défaillance et de bon fonctionnement. Voici ci-après quelques exemples :

- ✓ Liste de bon fonctionnement (chemin non défaillant) :

$L_{correct}$ = {TOS.TOS.Init(True).Fin(True)}

Cette liste représente le seul chemin correct qui existe dans ce cas de figure, le terme TOS.TOS signifie que le MUX (multiplixer) fonctionne correctement, aussi le terme Init(True).Fin(True) que la partie TOS (top of stack) fonctionne aussi correctement.

- ✓ Liste de défaillance (chemin défaillant) :

$L_{incorrect}$ = {TOS.Bf.OR.dM.NOS ; TOS.TOS.Init(True).Bf.Fin(False)}

Dans ce cas, la liste présente juste deux chemins parmi plusieurs, il est probable que le MUX défaille par l'incident de Bf (bit-flip) ou bien dM (défaillance matériel), et donné par conséquence un mauvaise aiguillage de donné, ou le défaut arrive à la fin dans la zone TOS par l'incident de Bf pour donner un résultat faut à la fin.

Apres la spécification détaillé de notre système (étude de l'AMDEC annexe A), nous modélisons l'architecture en question par deux modèles hiérarchisés afin de tirés le maximum d'information concernant le mode fonctionnel et dysfonctionnel de chaque composants. La génération des listes caractéristiques via le modèle de bas niveau est une manière exhaustive d'avoir les scénarios et les combinaisons d'erreur menant à un état de défaillance.

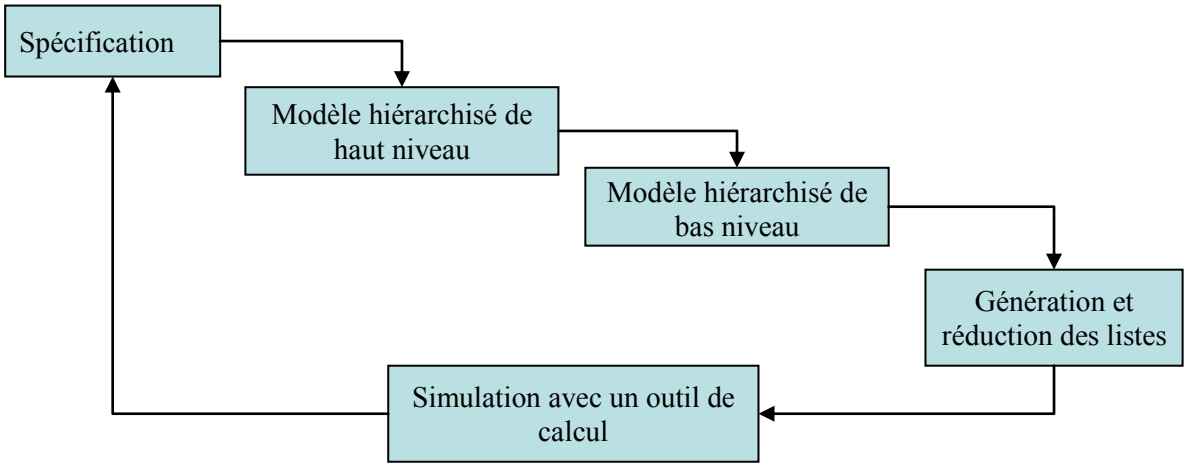


Figure A.C.3 : Démarche de l'évaluation fiabiliste selon l'approche flux informationnel.

Les scénarios sont en réalité des coups minimaux. La transformation de ces coups en arbres de défaillance permet grâce à l'outil logiciel Aralia de quantifier les valeurs de probabilités des tous les modes de défaillance que nous avons proposé le long de ce travail.

La première liste générée de la figure A.C.2 corresponde à la combinaison d'événement d'avoir le mode 1 de fonctionnement (Tout_Va_Bien). La probabilité d'avoir ce mode de fonctionnement se trouve par la transformation de cette lise à un arbre de cause suivante tel que la probabilité d'avoir le bon fonctionnement de l'élément sommet exige que tous les éléments de base soient en mode correct de fonctionnement. D'où l'utilité de la porte logique „ET" sur l'arbre.

La probabilité de taux de défaillance de chaque élément de base se calcul par la prise en compte de vieillissement accéléré par les contraintes de l'environnement tel que les radiations et la compatibilité électromagnétique. Nous utilisons les résultats des bases de données FIDES (Annexe B) dans le calcul de ces taux de défaillance. L'évolution de ces taux de défaillance en fonction des contraintes environnementaux est un processus markovien, sa matrice de transition est constituée par ces taux de défaillance que cherchons leurs évolutions en fonction de stress.

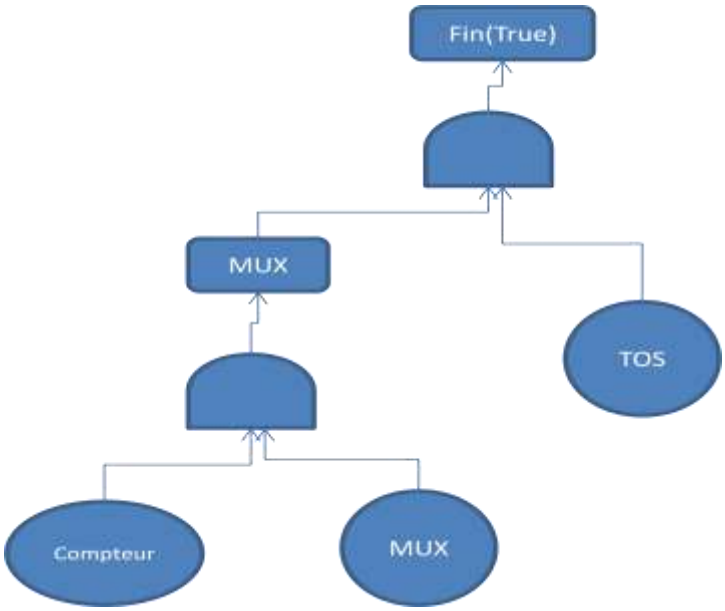


Figure A.C.4 : Arbre de cause correspond à la liste $L_{correct}$.

En occurrence, la deuxième liste de l'exemple traité dans la figure A.C.2, correspond quand à elle à une combinaison des événements menant à un mode de défaillance. Sa probabilité d'occurrence peut se calculer par l'arbre suivant sachant que la défaillance d'un seul composant peut amener à une défaillance de l'ensemble. D'où l'utilité de la porte logique „OU" sur l'arbre.

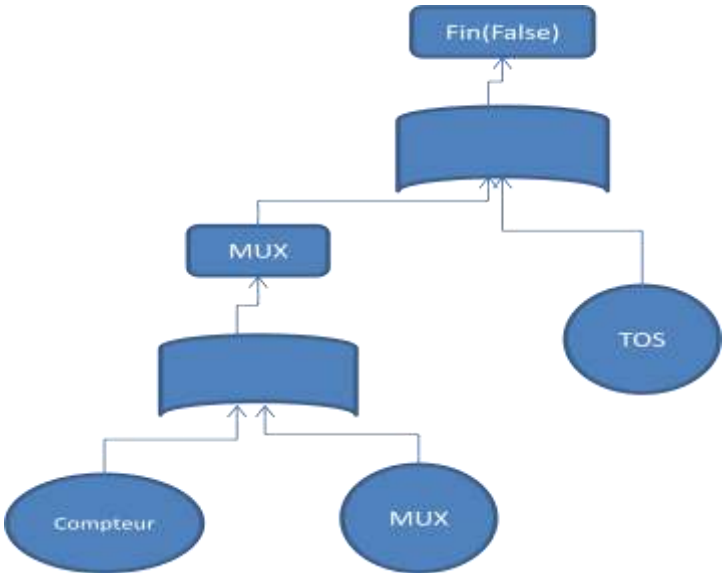


Figure A.C.5 : Arbre de défaillance correspondant à la liste $L_{incorrect}$.

Annexe D

Généralité sur les méthodes de la sûreté de fonctionnement

1. Sûreté de fonctionnement

La sûreté de fonctionnement est la science de défaillances [VIL 88]. Elle inclut ainsi leur connaissance, leur évaluation, leur prévision, leur mesure et leur maîtrise. Au sens strict, c'est la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans la qualité du service délivré [LAP 88].

A partir de ces deux définitions, une ***défaillance*** du système survient lorsque le service délivré dévie de l'accomplissement de la fonction du système. Une ***erreur*** est la partie de l'état du système qui est susceptible d'entraîner une défaillance, c'est-à-dire qu'une défaillance se produit lorsque l'erreur atteint l'interface du service fourni, et le modifie. Une ***faute*** est la cause adjugée ou supposée d'une erreur. Il existe donc une chaîne causale entre faute, erreur et défaillance.

La sûreté de fonctionnement englobe les attributs suivants :

- ✓ La fiabilité qui est l'aptitude d'une entité à accomplir une fonction requise, dans des conditions données, pendant un intervalle de temps donné,
- ✓ La disponibilité qui l'aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données, à un instant donné ou pendant un intervalle de temps donné, en supposant que la fourniture des moyens extérieurs est assurée,
- ✓ La maintenabilité qui est l'aptitude d'une entité à être maintenue ou rétablie, sur un intervalle de temps donné, dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est accomplie dans des conditions données, avec des procédures et des moyens prescrits,
- ✓ L'intégrité qui est la non-occurrence d'altérations inappropriées de l'information,
- ✓ La sécurité confidentialité (ou encore immunité) qui est l'absence de divulgation non autorisée d'informations,
- ✓ La sécurité innocuité, qui est la non-occurrence de conséquences catastrophiques pour l'homme, l'environnement et les biens.

L'association des qualificatives innocuités et *immunité* permet de lever l'ambiguïté associée à *sécurité*. Il est à noter que cette ambiguïté n'existe pas en anglais, qui dispose de ***safety*** pour la sécurité innocuité et de ***security*** pour la sécurité-immunité, sûreté de fonctionnement étant ***dependability***.

Le développement d'un système sûr de fonctionnement passe par l'utilisation combinée d'un ensemble de méthodes, appelées moyens de la sûreté de fonctionnement, qui peuvent être classées en :

- ✓ Prévention des fautes, comment empêcher par construction, l'occurrence ou l'introduction de fautes
- ✓ Tolérance aux fautes, comment fournir par redondance, un service conforme à la spécification en dépit des fautes,
- ✓ Elimination des fautes, comment minimiser par vérification la présence de fautes,
- ✓ Prévision des fautes, comment estimer la présence, la création et les conséquences de fautes.

Le schéma complet de la taxonomie de la sûreté de fonctionnement est donné à la figure A.D.1 :

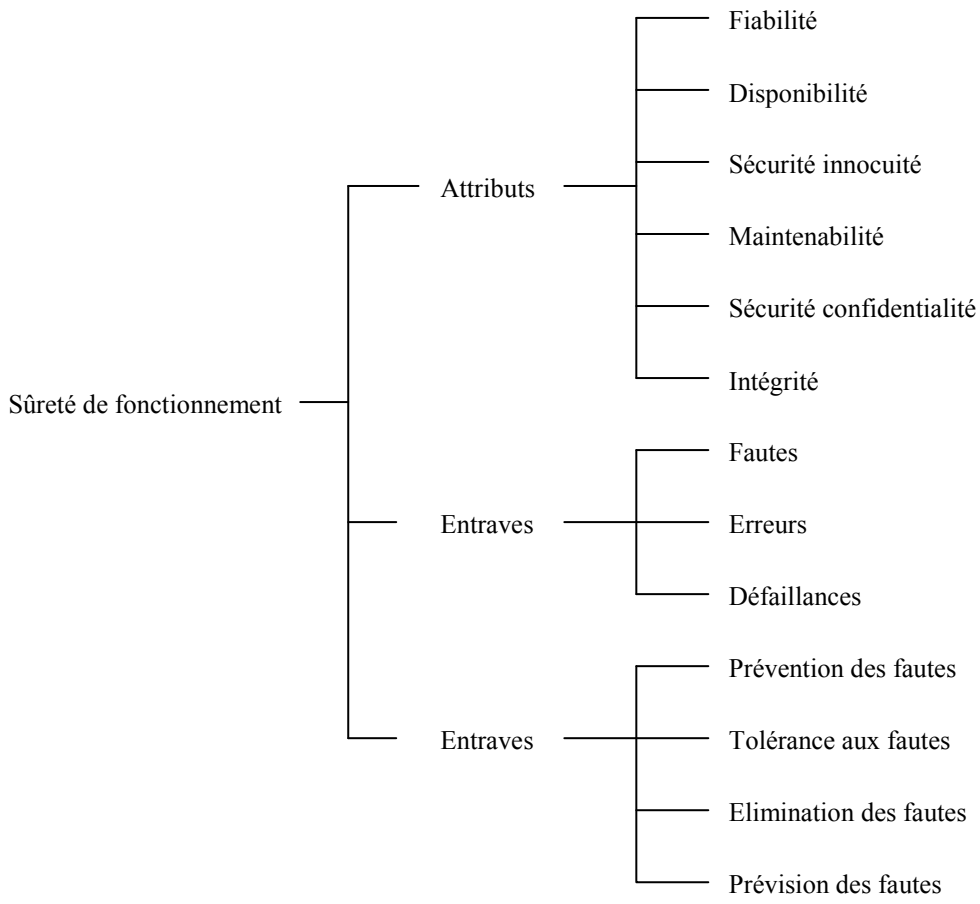


Figure A.D.1 : Taxonomie de la sûreté de fonctionnement [AVI 04]

2. Méthodes de la sûreté de fonctionnement

Les méthodes d'analyse de la sûreté de fonctionnement utilisent souvent une décomposition des systèmes entiers en sous-systèmes ou composants individuels dont les caractéristiques sont supposées connues.

L'évaluation de la sûreté de fonctionnement d'un système consiste à analyser les défaillances des composants pour estimer leurs conséquences sur le service rendu par le système. Les principales méthodes abordées dans ce travail lors d'une analyse de la sûreté de fonctionnement sont décrites ci-dessous.

2.1. Analyse préliminaire de risques

L'Analyse Préliminaire des Risques (APR) est une extension de l'Analyse Préliminaire des Dangers (APD) qui a été utilisée pour la première fois aux Etats-Unis, au début des années soixante [VIL 88]. Elle s'est généralisée à de nombreux domaines tels que l'aéronautique, chimique, nucléaire et automobile.

Les objectifs de cette méthode:

1) Identifier les dangers d'un système et de définir ses causes,

2) Evaluer la gravité des conséquences liées aux situations dangereuses et les accidents potentiels.

L'APR permet de déduire tous les moyens, toutes les actions correctrices permettant d'éliminer ou de maîtriser les situations dangereuses et les accidents potentiels. Il souvent préférable de commencer l'APR dès les premières phases de la conception. Cette analyse sera vérifiée, complétée au fur et à mesure de l'avancement dans la réalisation de système. L'APR permet de mettre en évidence les événements redoutés critiques qui devront être analysés en détail dans la suite de l'étude de sûreté de fonctionnement, en particulier par les méthodes quantitatives qui seront décrites par la suite.

2.2. L'analyse des modes de défaillances, de leurs effets et de leur criticité

L'Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité (AMDEC) est une extension naturelle de l'Analyse des Modes de Défaillance et de leurs Effets (AMDE) utilisée pour la première fois à partir des années soixante pour l'analyse de la sécurité des avions [VIL 88]. L'AMDEC considère la probabilité d'occurrence de chaque mode de défaillance et la classe de gravité de ces défaillances, mais aussi les classes correspondantes de probabilités d'occurrence plus que les probabilités elles-mêmes. On peut ainsi s'assurer que les modes de défaillance ayant d'importants effets ont des probabilités d'occurrence suffisamment faibles, grâce aux méthodes de conception, aux diverses vérifications et aux procédures de test.

2.3. Arbres des causes

La méthode a pour objectifs [VIL 88] la détermination des diverses combinaisons possibles d'événements qui entraînent la réalisation d'un événement indésirable unique, la représentation graphique de ces combinaisons sous forme d'une structure arborescente.

Un arbre des causes est composé de portes et d'événements. Les événements sont combinés par les portes classiques (ET, OU) ou des portes étendues (m sur n…).

L'analyse par arbre des causes est une analyse déductive qui permet de représenter graphiquement les combinaisons d'événements qui conduisent à la réalisation de l'événement redouté.

L'analyse par arbre des causes doit rechercher, à partir d'un événement indésirable (ou redouté), les défaillances de composants dont la combinaison entraîne l'apparition de l'événement indésirable.

Deux aspects d'analyse peuvent être élaborés :

- ✓ l'aspect qualitatif en déterminant l'ensemble des coupes minimales (la coupe minimale est la combinaison d'événements de base entraînant l'événement redouté),

- ✓ l'aspect quantitatif permettant la détermination par un calcul la probabilité d'apparition de l'événement redouté ou indésirable.

L'analyse par arbre des causes est largement utilisée dans les études de sûreté de fonctionnement car elle caractérise de façon claire les liens de dépendance, du point de vue dysfonctionnement, entre les composants d'un système. Bien que cette méthode soit efficace, elle présente des limites. L'une de ces limites est que l'ordre d'occurrence des événements menant vers l'état redouté n'est pas pris en compte.

2.4. Diagrammes de fiabilité

Un diagramme de fiabilité permet le calcul de disponibilité ou la fiabilité du système modélisé, mais avec les mêmes restrictions qu'un Arbre des causes. Tous les chemins entre l'entrée et la sortie décrivent les conditions pour que la fonction soit accomplie. On suppose que les composants n'ont que deux états de fonctionnement (fonctionnement correct ou panne).

2.5. Analyse de Markov

La méthode de graphes de Markov est utilisée pour analyser et évaluer la sûreté de fonctionnement des systèmes réparables. La construction d'un graphe de Markov consiste à identifier les différents états du système (défaillants ou non défaillants) et chercher comment passer d'un état à un autre lors d'un dysfonctionnement ou d'une réparation. A chaque transition, de l'état E_i vers l'état E_j, est associé un taux de transition τ_{ij} défini de telle sorte que $\tau_{ij}.dt$ est égal à la probabilité de passer de E_i vers E_j entre deux instants très proches t et t+dt sachant que l'on est en E_i à l'instant de temps t.

Les états sont classés en deux catégories :

- ✓ Etats de fonctionnement : ce sont les états où la fonction du système est réalisée, des composants du système pouvant être en panne, l'état du bon fonctionnement est l'état où aucun composant n'est en panne,

- ✓ Etats de panne : ce sont des états où la fonction du système n'est plus réalisée, un ou plusieurs composants du système étant en panne.

Le processus d'analyse comprend trois parties :

- ✓ Recensement et le classement de tous les états du système en états de bon fonctionnement ou états de panne.

- ✓ Recensement de toutes les transitions possibles entre ces différents états et l'identification de toutes les causes de ces transitions.

- ✓ Calcul des probabilités de se trouver dans les différents états au cours d'une période de vie de système ou le calcul des caractéristiques de sûreté de fonctionnement.

La modélisation avec les graphes de Markov permet de prendre en compte les dépendances temporelles et stochastiques plus largement que les méthodes classiques. En dépit de leur simplicité conceptuelle et leur aptitude à pallier certains handicaps des méthodes classiques, les graphes de Markov souffrent de l'explosion du nombre des états [MON 98], car le processus de modélisation implique l'énumération de tous les états possibles et de toutes les transitions entre ces états.

2.6 Stratégie de maintenance d'un capteur témoin

- **Cas du remplacement des composants après défaillance :**

Si l'on considère une durée totale très grande, l'intervalle de temps moyen entre défaillances d'un composant est précisément la durée de vie moyenne E(t) du composant considérer, et le nombre moyen de défaillances pendant le temps total sera :

$$E(K) = t / E(t)$$

Si le système était à taux de défaillance constant λ, le nombre moyen de défaillance serait :

$$E(k) = \lambda t$$

On peut alors définir un taux de défaillance moyen équivalent en égalant (1) et (2) :

$$\lambda = 1/ E(t)$$

Si le composant a une distribution normale des durées de vie, on aura $E(t) = \mu$,
D'où :

$$\lambda = 1 / \mu$$

Si le composant a une distribution des durées de vie de type Weibull avec les paramètres β et η ($\gamma = 0$), on a :

$$E(t) = \eta\Gamma(1 + 1/ \beta)$$

D'où : $$\lambda = 1/ \eta\Gamma(1 + 1/ \beta)$$

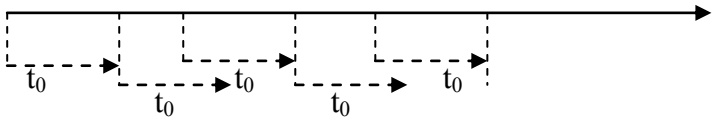
Exemple : on considère comme exemple un composant électronique qui suit la distribution de Weibull :

$\eta = 74{,}5\ 10^{6}$ $\beta = 1$
$\Gamma(1 + 1/ \beta) = \Gamma(2) = 1$ alors : $\lambda = 1/ \eta = 1{,}79\ 10^{-8}$

- **Cas du remplacement selon l'âge du composant :**

On envisage la politique d'entretien consistant à effectuer des remplacements préventifs après t_0 heures de fonctionnement du composant considérée ou après défaillance.

R R D R D R t



R = Remplacement préventif
D= Remplacement suite à une avant t_0 défaillance

Figure 4.6 : répartition de maintenance durant le temps d'utilisation.

La durée de vie moyenne de ce composant sera :

$$m(t_0) = \int_0^{t_0} R(t)dt$$

Le taux global de remplacement de ce composant est alors :

$$\lambda = \frac{1}{m(t0)}$$

Se décomposant en :

taux moyen de remplacement par dépannage :

$$\lambda_D(t_0) = \frac{1-R(t0)}{m(t0)}$$

taux moyen d'échange systématique :

$$\lambda_g(t_0) = \frac{R(t0)}{m(t0)}$$

Si $t_0 \rightarrow \infty$ ces expressions deviennent :

$$m(\infty) = \int_0^{\infty} R(t)dt = \theta$$

$$\lambda_D(\infty) = \frac{1}{m(\infty)} = \frac{1}{\theta}$$

$$\lambda_g(\infty) = \frac{0}{m(\infty)} = 0$$

- Cas d'une distribution exponentielle :

On a : $$m(t_0) = \int_0^{t0} \exp(-\lambda t)dt = \frac{1-\exp(-\lambda t0)}{\lambda}$$

D'où : $$\lambda_D(t_0) = \lambda \frac{1-\exp(-\lambda t0)}{1-\exp(\lambda t0)}$$

$$\lambda_g(t_0) = \lambda \frac{\exp(-\lambda t0)}{1-\exp(-\lambda t0)}$$

Remarque :

Le taux global est $\frac{\lambda}{1-\exp(-\lambda t0)}$; il est donc toujours supérieur à celui qu'on aurait sans remplacement. Comme la fiabilité n'est pas améliorée, on voit qu'il n'y a aucun intérêt à effectuer des remplacements préventifs dans le cas d'une distribution exponentielle.

Dans cette étude, nous avons considéré la distribution de Weibull pour traiter les composants électroniques, ce type de distribution permet de suivre l'évolution de

- Cas d'une distribution de Weibull à 2 paramètres :

Dans ce cas, on a :

$$R(t_0) = e^{-(\frac{t0}{\eta})^{\beta}}$$

Et : $$m(t_0) = \int_0^{t0} e^{[\frac{t}{\eta}]^{\beta}} dt = \eta \int_0^{t0} e^{-[\frac{t}{\eta}]^{\beta}} d[\frac{t}{\eta}]$$

On posera : $$x = \frac{t}{\eta}$$

D'où : $$\lambda_D(t_0) = \frac{1 - e^{-[\frac{t0}{\eta}]^{\beta}}}{\eta \int_0^{x_0} e^{-x^{\beta}} dx}$$

$$\lambda_g(t_0) = \frac{e^{-[\frac{t_0}{\eta}]^{\beta}}}{\eta \int_0^{x_0} e^{-x^{\beta}} dx}$$

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE
DU DOCTORAT DE L'INSTITUT NATIONAL
POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Madame Zineb SIMEU-ABAZI, Maître de Conférences, INPG, G-SCOP, Grenoble

Monsieur Frédéric KRATZ, Professeur, ENSI, PRISME, Bourges

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur BELHADAOUI Hicham

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 54501
VANDŒUVRE CEDEX

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

"Conception sûre des systèmes mécatroniques intelligents pour des applications critiques"

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : « **Automatique, Traitement du Signal et des Images, Génie Informatique** »

Fait à Vandoeuvre, le 26 octobre 2010
Le Président de l'I.N.P.L.,
F. LAURENT

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE - NANCY -

TEL. 33/03.83.59.59.59
FAX. 33/03.83.59.59.55

Résumé

La criticité des systèmes complexes programmables nécessite de garantir un niveau de fiabilité et de sécurité convenable. Des études de sûreté de fonctionnement doivent être menées tout au long du cycle de développement du système. Ces études permettent une meilleure maîtrise des risques et de la fiabilité. Les points faibles sont mis en évidence et permettent aux concepteurs de spécifier des stratégies de reconfiguration avant la phase de prototype réel et les tests réels. Les études de sûreté de fonctionnement doivent être menées au plus tôt dans la phase de conception, afin de réduire les coûts et le nombre de prototypes nécessaires à la validation du système.

Le travail présenté dans ce mémoire de thèse a pour objectif de définir une méthodologie de conception des systèmes complexes programmables dédiés à une application mécatronique [Belhadaoui et al., 2008-a], intégrant dès les premières phases du cycle de développement [Aït-Kadi et al., 2000], les aspects sûreté de fonctionnement. L'apport d'une telle méthodologie doit permettre de faire face à un certain nombre de contraintes propres au domaine des capteurs intelligents (les exigences de cahier des charges, le respect des normes législatives en vigueur).

La méthodologie développée doit permettre de :

- Modéliser et simuler les comportements fonctionnels et dysfonctionnels des systèmes
- Estimer la fiabilité par modélisation
- Réaliser des mesures de sensibilité afin de connaître la contribution de chaque composant à la fiabilité du système
- Capitaliser la connaissance sur le système au cours des différentes phases d'évaluation (prévisionnelle, expérimentale et opérationnelle) pour affiner les estimations de fiabilité

Ce Travail introduit le concept d'information en sûreté de fonctionnement. Nous interprétons la défaillance de celle-ci comme étant le résultat de l'initiation et de la propagation d'informations erronées à travers l'architecture d'un capteur intelligent dédié à une application mécatronique. Cette propagation s'est accompagnée de contraintes (partage de ressources matérielles et informationnelles, modes dégradés d'information…) qui tendent à influencer fortement la crédibilité de cette information. Nous débutons sur un état de l'art pour montrer l'intérêt de l'approche flux informationnel sur un cas d'étude complexe. Ceci est lié à la présence d'une partie programmable (interaction matériel-logiciel) et évidement du système hybride (signaux mixtes analogique-numérique). Cette nouvelle approche distingue, les phénomènes d'apparition et de disparition d'erreurs (matérielles, logicielles et environnementales), ainsi que les séquences de propagation aboutissant à un mode de dysfonctionnement du système. Grâce à cette distinction nous expliquons les concepts mal traités par les méthodes conventionnelles, tels que la défaillance simultanée, la défaillance de cause commune et abordons d'une manière réaliste les problématiques des interactions matériel-logiciel et celle des signaux mixtes.

Les séquences de propagation d'erreurs générées permettent à l'aide d'un modèle markovien non homogène, de quantifier d'une manière analytique les paramètres de la sûreté de fonctionnement du système (fiabilité, disponibilité, sécurité) et de positionner le capteur dans un mode de fonctionnement parmi les six que nous avons définis suivant les spécifications du cahier des charges.

MOTS CLEFS : Sûreté / Mécatronique / Evaluation probabiliste / Modélisation

Abstract

The complexity of critical programmable systems requests the guarantee of high level of reliability and safety. The dependability studies should be conducted throughout the development cycle of the system. These studies provide better risk management and reliability. The weak points are highlighted, and enable designers to specify reconfiguration strategies before the prototype stage and real testing. The dependability studies must be conducted as soon as possible in the design phase, in order to reduce costs and the number of prototypes necessary to validate the system.

The work presented in this thesis aims to define a design methodology of complex systems dedicated to a mechatronic programmable application [Belhadaoui et al., 2008-a], integrating as soon as possible dependability aspects in the development cycle [Aït-Kadi et al., 2000]. The provision of such a methodology must resist face a number of constraints specific to the intelligent sensors field (requirements specifications, compliance with standards legislation).

The methodology developed enable to:

- Modeling and simulate the functional and dysfunctional behavior of systems.
- Estimate the reliability by modelling.
- Achieve measures sensitivity to deduce the contribution of each component in the reliability of the system.
- Capitalize the system knowledge during different phases of evaluation (planning, experimental and operational) to refine estimations of reliability.

This work introduces the concept of information dependability. It interprets the information failure, as a result of the initiation and propagation of failure information through the architecture. This spread has been accompanied by constraints (sharing hardware resource and information, degraded modes of information…), which tend to influence the credibility of this information. We begin on a state of the art to show the interest of the information flow approach in a complex case study. This interest is linked to the presence of programmable part (hardware-software interaction), and obviously to the hybrid character of the system (mixed-signal analog and digital). This new approach distinguishes, the phenomena of appearance and disappearance of errors (hardware, software and environmental), as well as the sequences of propagation resulting to the system failures. With this distinction we explain the concepts badly treated by conventional methods, such as the simultaneous failure, the common cause failure and in a realistic manner convincing the issues of hardware-software interactions, and the mixed signals.

The generated errors propagation sequences allows, with using a non-homogeneous Markov model, to quantify an analytical dependability parameters of the system (reliability, availability, security) and to position the sensor in an operating mode among six that have been defined according to the standards specifications.

KEYWORDS : Safety / Mecatronics / Probability evaluation / Modelling