Prise en compte de la sûreté de fonctionnement lors du choix d'architectures des systèmes complexes

Anis Baklouti

Thèse de doctorat

NNT : 2020UPASC002

Prise en compte de la sûreté de fonctionnement lors du choix d'architectures des systèmes complexes

Thèse de doctorat de l'université de Sousse et de l'université Paris-Saclay

École doctorale n° 573, interfaces : approches interdisciplinaires, fondements, applications et innovation

Spécialité de doctorat: sciences et technologies industrielles

Unité de recherche : ISM Paris, ENSEA, EISTI, Quartz, 93400, Saint-Ouen-sur-Seine

Unité de recherche : Université Paris-Saclay, 91190, Saint-Aubin, France

Référent : CentraleSupélec

Thèse présentée et soutenue à Paris, le 24 Janvier 2020, par

Anis BAKLOUTI

Composition du Jury

Jean-Jacques LESAGE Professeur des Universités, ENS Cachan (LURPA)	Président
Nizar AIFAOUI Professeur des Universités, IPEIM (LGM)	Rapporteur & Examinateur
Frédéric KRATZ Professeur des Universités, INSA-CVL (PRISME)	Rapporteur & Examinateur
Nouredine BENYAHYA Professeur des Universités, ENSIT	Examinateur
Nga NGUYEN Maître de Conférences, EISTI (ETIS)	Examinatrice
Jean-Yves CHOLEY Professeur des Universités, SUPMECA (QUARTZ)	Directeur de thèse
Abdelfattah MLIKA Professeur des Universités, ENISo (LMS)	Co-Directeur de thèse
Faïda MHENNI Maître de Conférences, SUPMECA (QUARTZ)	Co-encadrante & Examinatrice
Vincent IDASIAK Maître de Conférences, INSA-CVL (PRISME)	Invité

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Remerciements

Contrairement à ce qu'on peut imaginer, une thèse, ce n'est pas la contribution d'une mais de plusieurs personnes de manières complémentaires et je souhaiterais à travers ces quelques mots faire part de ma gratitude envers tous ceux et celles qui ont participé de près ou de loin à cette aventure.

Je souhaiterais tout d'abord remercier les membres du jury, qui ont accepté d'examiner et juger ce travail. Je vous remercie pour le temps consacré à mon travail, pour les commentaires et échanges constructifs lors de la soutenance.

J'exprime ma plus grande reconnaissance à mes directeurs de thèse Monsieur MLIKA Abdelfatteh et Monsieur CHOLEY Jean-Yves qui m'ont accueilli dans leurs labos (LMS et Quartz), font confiance durant ces années et qui ont su équilibrer subtilement exigence et tolérance à mon égard.

Je voudrais remercier Nga NGUYEN et Faïda MHENNI pour leur encadrement, la confiance et l'autonomie qu'ils m'ont accordée.

Merci également aux membres du LMS et Quartz pour leur support.

Enfin, mes derniers remerciement et pas des moindres vont à ma famille pour leur soutien sans faille, leur patiente et leur aide logistique…

Je remercie du fond du coeur mes soeurs Khouloud et Maïssa pour leur soutien.

Je ne saurais assez remercier ma femme, Nouha pour sa patience, son soutien inconditionnel, son amour et pour avoir accepté de me suivre dans cette aventure avec tous les sacrifices que ça a impliqué.

Merci à mes beaux-parents pour leur support et leur amour.

Enfin, je pense que les mots ne suffisent pas pour exprimer ma gratitude envers mes parents qui m'ont permis d'en arriver là aujourd'hui, merci d'avoir cru en moi, même lorsque je doutais moi-même et merci pour votre amour. Merci pour tout !

Table des matières

Table des Figures

Liste des tableaux

Glossaire

AADL: Architecture Analysis and Design Language

AFDS: Aircraft Fuel Distribution System

AFIS: Association Française d'Ingénierie système

APD: Analyse Préliminaire des Dangers

ATL: Atlas Transformation Langage

BDD: Block Definition Diagram

CAO: Conception Assisté par Ordinateur

CCA: Common Cause Analysis

CCF: Common Cause Failure

CFT: Component Fault Trees

CR: Central Reservation

CRT: Central Reservation Tank

CS: Cut Set

DDB: Diagrammes de Décision Binaire

DIFTree: Dynamic Innovative Fault Tree

DRBD: Dynamic Reliability Block Diagram

EMA: Electro-Mechanical Actuator

FFA: Functional Failure Analysis

FHA: Functional Hazard Assessment

FLM: Failure Logic Modeling

FMEA: Failure Mode and Effects Analysis

FSAP: Formal Safety Analysis Platform

FTA: Fault Tree analysis

HAZOP: HAZard and OPerability analysis

HiP-HOPS: Hierarchically Performed Hazard Origin and Propagation Studies

IBD: Internal Block Diagram

INCOSE: International Council on Systems Engineering

IS: Ingénierie système

MBSE: Model Based Systems Engineering

MBSA: Model Based Safety Assessment

MCS: Minimal Cut Set

MPS: Minimal Path Set

MTBF: Mean Time Between Failure

MTTF: Mean Time To Failure

MTTR: Mean Time To Repaire

NCOSE: National Council on Systems Engineering

OMG: Object Management Group

Open-PSA: Probabilistic Safety Assessment

PF: Port Feed

PE: Port Engine

PNG: Portable Network Graphics

POT: Port Outer Tank

PIT: Port Inner Tank

PS: Path Set

PSSA: Preliminary System Safety Assessment

RBD: Reliability Block Diagram

RPN: Risk Priority Number

RP: Refuelling Point

SAE: Society of Automotive Engineers

SAM: Safety Argument Manager

SDF: Sûreté De Fonctionnement

SE: Systems Engineering

SF: Starboard Feed

SFT: Static Fault Tree

SIL: Safety Integrity Level

SIT: Starboard Inner Tank

SOT: Starboard Outer Tank

SSA: System Safety Assessment

SysML: System Modeling Language

UML: Unified Modeling Language

XMI : XML Metadata Interchange

Introduction générale

Le succès que peut avoir une entreprise ne se mesure pas uniquement par la qualité du service présenté à sa clientèle ou par le confort qui leur est procuré, mais aussi par le degré de sûreté que l'on peut garantir aux utilisateurs. Ainsi, les efforts se multiplient chaque jour pour perfectionner davantage la sûreté de fonctionnement des systèmes.

En effet, nous sommes entourés par une variété de systèmes qui sont devenus une nécessité pour la vie moderne. Ils sont utilisés pour des fins différentes comme par exemple l'électroménager, le transport, la communication, l'usinage, etc. Afin d'assurer plus de sécurité et de confort pendant leur utilisation, les systèmes sont astreints à réaliser un grand nombre de fonctions, ce qui nécessite l'intégration de plusieurs composants de technologies différentes qui doivent communiquer d'une manière synergique pour réaliser les missions du système. Ceci entraîne une complexité croissante dans les systèmes fabriqués.

En plus de cette complexité croissante, le contexte industriel actuel est sollicité par une forte contrainte de compétitivité. Cela conduit à l'intégration de nouvelles fonctionnalités et à l'amélioration des performances des systèmes conçus, ce qui augmente, de nouveau, la complexité. Le contexte concurrentiel impose également de raccourcir les délais de mise sur le marché et de réduire les coûts tout en fournissant des produits fiables et efficaces.

Ainsi, l'approche Ingénierie Système Basée sur les Modèles (en anglais : Model Based Systems Engineering (MBSE)) est indispensable pour la conception des systèmes complexes critiques. Une recherche sur les approches MBSE révèle que SysML, le langage de modélisation des systèmes d'Object Management Group (OMG), est largement utilisé pour supporter ce genre d'approche. En effet, SysML permet d'exprimer les principaux concepts inhérents aux différents aspects du développement du système. Ce langage est utilisé pour construire un modèle de système utilisé comme référence commune pour tous les domaines métier impliqués.

Pour les systèmes critiques, un fonctionnement incorrect peut avoir des conséquences graves sur la vie humaine, allant de la blessure jusqu'au décès. Pour ces systèmes critiques, des

normes et des réglementations rigoureuses de sûreté de fonctionnement spécifient le niveau de risque acceptable et imposent des contraintes précises sur la conception du système et les méthodes de vérification pour valider le système conçu. Le non-respect de ces normes conduit inévitablement à des systèmes non qualifiés qui ne peuvent pas être commercialisés.

Ces systèmes mécatroniques nécessitent une validation rigoureuse de leur comportement et de leurs performances. Les risques potentiels de tels systèmes doivent être soigneusement identifiés et traités pour les amener à un niveau de sûreté acceptable. Le développement de ces systèmes est alors difficile et nécessite des approches d'ingénierie système adéquates ainsi que des techniques d'analyse de sûreté rigoureuses afin de gérer la complexité et de satisfaire les performances du système et les exigences de sûreté.

D'autre part, nous constatons que l'absence de lien entre le modèle système et les artéfacts de la sûreté de fonctionnement est une source d'incohérence entre ces deux modèles. Cette incohérence inhibe la prise en compte des contraintes de sûreté au plus tôt dans le processus de conception, ce qui peut s'avérer très coûteux lorsque les problèmes sont détectés tardivement. On note également que l'absence de traçabilité et de cohérence entre les modèles système et les modèles de la sûreté de fonctionnement empêche la mise à jour dynamique de l'architecture d'un système critique afin de prendre en compte les modifications nécessaires lorsqu'une évolution du système est envisagée suite à l'analyse de sûreté de fonctionnement. Pour une utilisation réussie, les techniques d'analyse de sûreté doivent être intégrées efficacement dans le processus de conception. Cela ne peut se faire qu'avec l'utilisation d'outils et de méthodes adéquats pour faciliter la cohérence entre les deux domaines.

Ce travail fait suite à la thèse de doctorat de Faïda MHENNI intitulée « Vers une approche intégrée d'analyse de sûreté de fonctionnement des systèmes mécatroniques » [25]. L'objectif de son travail était l'intégration des analyses de sûreté dans le processus de conception du système afin de combler le fossé entre les deux approches en améliorant la communication et la cohérence. L'objectif de ce travail est de renforcer davantage l'intégration des analyses de sûreté dans le processus de conception du système et la prise en compte de la sûreté de fonctionnement lors du choix d'architectures des systèmes complexes afin d'améliorer l'architecture et le comportement des systèmes.

Le langage SysML a été utilisé dans les travaux de thèse de Mhenni qui avaient comme objectif la proposition d'une approche pour intégrer l'analyse de sûreté de fonctionnement dans le processus de conception des systèmes mécatroniques. Dans le cadre de ce travail de thèse,

nous allons bâtir sur les travaux de Mhenni afin de renforcer davantage l'approche intégrée, i.e. le processus de conception - sûreté de fonctionnement pour les systèmes mécatroniques. Nous allons d'abord proposer la génération automatique des arbres de défaillances dynamiques à partir des modèles système. En plus, nous allons proposer une méthode d'intégration des résultats de l'analyse de ces arbres de défaillance dynamiques pour mener des changements d'architecture ayant pour objectif l'amélioration de la sûreté de fonctionnement du système étudié.

Ce manuscrit de thèse est organisé comme suit.

Le chapitre 1 présente l'état de l'art et les principaux concepts utiles pour cette thèse. Premièrement, on définit le système complexe en présentant les principales caractéristiques de la complexité. Ensuite, on présente les concepts clés tel que les processus, les normes, les méthodes et les outils de l'Ingénierie Système (en anglais : Systems Engineering (SE)). La deuxième partie du chapitre est consacrée à l'analyse de la sûreté de fonctionnement. Cette partie commence par la définition des risques en précisant les différentes étapes, les stratégies et les outils de gestion des risques. Par la suite, on présente les principales méthodes et les standards d'analyse de la sûreté de fonctionnement. Enfin, le chapitre présente un état de l'art de l'intégration des analyses de sûreté dans les approches d'ingénierie système ainsi qu'une étude bibliographique sur les travaux qui proposent des méthodes de génération des arbres de défaillances statiques ou dynamiques à partir des modèles systèmes.

Le chapitre 2 commence par la définition de la notion de tolérance aux fautes et l'identification des différents types de redondance. Puis, on présente la méthodologie proposée dans ce travail qui définit le processus d'intégration de l'ingénierie système et de la sûreté de fonctionnement. Ensuite, on décrit les deux premières étapes de cette méthodologie. Dans la première étape, on commence par la description de la génération des FMEA à partir des diagrammes SysML. Par la suite, on passe aux choix d'architecture pour intégrer des modifications structurelles du système suite à l'analyse des FMEAs dans la deuxième étape. Ce chapitre se termine par un cas d'étude qui illustre cette approche.

Le chapitre 3 commence par la description de la méthode de la génération des arbres de défaillances dynamiques à partir des modèles systèmes. Pour cela, un profil SysML qui permet la description du comportement de la redondance ainsi que l'algorithme de la génération des arbres de défaillances dynamiques sont présentés. Ensuite, une étude d'un système de contrôle d'un aileron d'avion est faite afin d'appliquer l'algorithme proposé et de générer l'arbre de

défaillance dynamique en utilisant la structure du système et des informations sur les redondances du système.

Le chapitre 4 présente deux cas d'études. Le premier cas d'étude, traitant l'exemple d'un moteur électromécanique, illustre le processus général de la méthodologie proposée ainsi que le lien entre les différentes étapes. Tout d'abord, on commence par l'étape de la génération des FMEAs à partir des diagrammes SysML, puis on passe à l'étape du choix d'architecture afin que la structure du système atteigne un niveau de sûreté satisfaisant. Ensuite, on applique l'étape de génération des arbres de défaillances dynamiques. Finalement, l'étape de création de la machine à états à partir de l'analyse qualitative des arbres de défaillances dynamiques est placée. Le deuxième cas d'étude est le système de distribution de carburants pour les avions. Il est utilisé pour la validation de l'algorithme de génération des arbres de défaillances dynamiques qui est la troisième étape de notre méthodologie.

Chapitre 1
Etat de l'art : contexte MBSE/MBSA

1.1 Introduction

Les systèmes mécatroniques sont de plus en plus complexes suite à l'intégration d'un grand nombre de composants et d'une variété de technologies. Les composants logiciels sont également de plus en plus intégrés, pour des fins de contrôle-commande, dans des systèmes qui contiennent déjà une diversité d'autres composants de domaines différents tels que des dispositifs électroniques, des capteurs, des actionneurs et des structures mécaniques [1, 2].

Un système mécatronique complexe est dit critique lorsque son dysfonctionnement ou sa défaillance peut entraîner des problèmes de sûreté de fonctionnement. Le développement d'un système critique doit prendre en compte et tenter de minimiser les risques potentiels du système non seulement dans des situations nominales mais aussi dans certaines situations dégradées [3]. Pour cet effet, plusieurs méthodes d'analyses de sûreté de fonctionnement ont été mises en œuvre afin d'améliorer la conception des systèmes (architecture et comportement) et de les rendre plus sûrs.

L'objectif de cette thèse est la contribution à l'intégration de l'analyse de la sûreté de fonctionnement dans le processus de conception de l'Ingénierie Système (IS). Ce premier chapitre intitulé « Le contexte du MBSE/MBSA » introduit le cadre du travail et fournit les connaissances nécessaires sur les deux thèmes principaux de cette thèse que sont l'ingénierie système et l'analyse de sûreté de fonctionnement ainsi que l'état de l'art sur leur intégration. Ce chapitre est organisé en trois parties et une conclusion. Dans la première partie, on traite l'ingénierie système des systèmes mécatroniques complexes. Tout d'abord, on commence par la définition d'un système complexe en présentant les principales caractéristiques de la complexité. Puis, on présente les définitions, les normes et les processus de l'IS. Dans la deuxième partie, on commence par la définition des risques en précisant les différentes étapes, les stratégies et les outils de gestion des risques. Par la suite, on présente les principales méthodes et les standards d'analyse de la sûreté de fonctionnement. Dans la troisième partie, on étudiera des travaux qui intègrent la MBSE et l'analyse de sûreté basée sur des modèles (en anglais : Model Based Safety Assessment (MBSA)). Aussi, une étude bibliographique sur les travaux qui proposent des méthodes de génération d'arbres de défaillances statiques ou dynamiques à partir des modèles systèmes est faite. Ce chapitre sera clôturé par une conclusion qui définit la problématique de notre travail.

1.2 Ingénierie système pour les systèmes mécatroniques complexes

1.2.1 Systèmes mécatroniques complexes

Dans cette partie, on commence par la définition d'un système, de la complexité, et les systèmes complexes.

1.2.1.1 Définition d'un système

Plusieurs ouvrages ont donné des définitions d'un système. Jean-Pierre MEINADIER dans son livre « Ingénierie et intégration des systèmes » [4] a défini un système comme un ensemble de personnel, de matériels et de logiciels organisés pour que leur interfonctionnement permette, dans un environnement donné, de remplir les missions pour lesquelles il a été conçu. Selon cette définition, on constate que les humains peuvent également faire partie d'un système.

L'INCOSE (International Council on Systems Engineering) [5] a proposé une définition qui décrit le système en tant qu'un ensemble d'éléments interdépendants qui interagissent de façon organisée et forment un ensemble unique.

L'Association Française d'Ingénierie Système (AFIS) a proposé sa propre définition qui considère que le système est un ensemble d'éléments en interaction entre eux afin de rendre à son environnement les services correspondant à sa finalité.

On constate que ces différentes définitions se réunissent sur le fait que l'interaction entre les composants est un point indispensable pour la formation d'un système.

1.2.1.2 Complexité

De nos jours, on parle de plus en plus des systèmes complexes ce qui explique l'existence de plusieurs tentatives de définition de la complexité dans plusieurs domaines. D'après Nancy Leveson [6], la complexité des systèmes introduit des inconnues faisant que le comportement de ces systèmes ne peut pas être complètement compris et anticipé. Afin de mieux concevoir les systèmes complexes, des modèles sont nécessaires pour aider à gérer la complexité.

La complexité est définie par SAE-ARP-4754A [7] comme un attribut d'une fonction, d'un système ou d'un élément qui rend les opérations et les modes de défaillances difficiles à comprendre sans recours à des méthodes analytiques.

Romaric Guillerm [8] a défini un système complexe comme étant un "système capable de fournir des fonctions de haut niveau, dont le comportement global est difficile à prévoir et

dont la structure présente un graphe d'interaction non-trivial, souvent pourvu de boucles de rétroactions, et associant la plupart du temps plusieurs technologies de par l'implication de nombre de constituants".

D'après ces définitions de la complexité et d'un système, on remarque que cette complexité varie selon plusieurs facteurs. Par exemple, la complexité d'un système augmente par la croissance du nombre des composants et surtout des relations, par les interactions entre les éléments du système en particulier les boucles de rétroaction et l'utilisation des technologies nouvelles et sophistiquées [6, 7]. De plus, si ces composants en interaction proviennent de domaines différents et échangent des types différents de flux (continus ou discrets, objets ou données, etc.) la complexité croît davantage.

1.2.2 Ingénierie système

1.2.2.1 Historiques

Le terme « ingénierie système » remonte aux « Bell Telephone Laboratories », qui est une société américaine de recherche et de développement scientifique appartenant à la société finlandaise Nokia, dans les années 1940 [9]. La nécessité d'identifier et de manipuler les propriétés d'un système dans son ensemble a motivé diverses industries. L'armée américaine était la première à utiliser l'ingénierie système et ses premières utilisations étaient en 1950 pour des projets de développement des systèmes spatiaux et nucléaires [10].

Lorsque l'évolution de la conception pour l'amélioration d'un système et les outils existants ont montré leurs limites devant les demandes croissantes de la technologie, de nouvelles méthodes ont commencé à être développées pour répondre directement à la complexité [11]. L'évolution continue de l'ingénierie système comprend le développement et l'utilisation de nouvelles méthodes et des techniques de modélisation. Ces méthodes aident à mieux comprendre la conception du système ainsi que le contrôle du développement qui devient de plus en plus complexe, notamment dû à l'augmentation des participants (parties prenantes).

En 1990, une société professionnelle pour l'ingénierie système, le National Council on Systems Engineering (NCOSE), a été fondée par des représentants d'un certain nombre de sociétés et d'organisations américaines. NCOSE a été créé pour répondre à la nécessité d'améliorer les pratiques et l'éducation en ingénierie système. À la suite de la participation croissante des ingénieurs système de l'extérieur des États-Unis, le nom de l'organisation a été changé et devenu l'International Council on Systems Engineering (INCOSE) en 1995 [16].

En 1999, une association française, l'Association Française d'Ingénierie Système (AFIS), a regroupé de grandes entreprises comme Renault, Airbus, RATP, Peugeot pour promouvoir l'ingénierie systèmes et favoriser son usage et son développement. L'AFIS représente le « chapitre français de l'INCOSE ».

1.2.2.2 Définition de l'ingénierie système

On distingue une variété de définitions de l'ingénierie système. Par exemple, la norme militaire américaine MIL-Std-499A a défini l'ingénierie système, en 1974 [17], comme une application des efforts scientifiques et d'ingénierie pour :

1. La transformation d'un besoin opérationnel en une description des paramètres de performance du système et en une configuration du système à l'aide d'un processus itératif d'analyse, de synthèse, de conception, de test et d'évaluation ;
2. l'intégration des paramètres techniques et l'assurance de la compatibilité entre toutes les interfaces physiques de manière à optimiser la définition et la conception totale du système ;
3. l'intégration de la fiabilité, la maintenabilité, la sûreté, la capacité de survie et d'autres facteurs de l'ingénierie technique pour répondre aux contraintes de coût, au planning et aux objectifs de performance technique.

Dans cette première définition, on constate l'apparition de quelques concepts (ou aspects) de l'IS comme le fait qu'il s'agisse d'un processus itératif, la compatibilité et quelques facteurs comme la sûreté, la fiabilité, etc.

Dix-neuf ans après, une nouvelle définition apparait dans une nouvelle version de la norme. Elle définit l'ingénierie système par une approche interdisciplinaire qui englobe l'ensemble des efforts techniques pour faire évoluer et vérifier un ensemble équilibré de produits et de processus système qui répondent aux besoins des clients. D'après cette nouvelle définition, l'ingénierie système englobe :

1. le développement, la fabrication, la vérification, les opérations de développement, le soutien ;
2. le management de la configuration des systèmes ;
3. le développement de l'information pour la gestion de la prise de décision.

L'AFIS a donné une autre définition de l'ingénierie système qui est : "L'ingénierie système est une démarche méthodologique coopérative et interdisciplinaire qui englobe l'ensemble des activités adéquates pour concevoir, développer, faire évoluer et vérifier un ensemble de produits, processus et compétences humaines apportant une solution économique et performante aux besoins des parties prenantes et acceptable par tous''.

Howard Eisner [18] a donné une autre définition de l'ingénierie système où il l'a définie par un processus itératif de synthèse descendante, de développement et du fonctionnement d'un système qui satisfait, d'une manière quasi-optimale, l'ensemble des exigences du système.

A partir de toutes ces définitions, nous déduisons que l'ingénierie système est une approche méthodologique interdisciplinaire qui intègre tous les aspects (la fiabilité, la maintenabilité, la sûreté et les compétences humaines, etc..) afin d'aider le concepteur d'avoir un produit complexe bien développé et qui respecte les exigences données avec un meilleur coût.

1.2.2.3 Cycle de vie et cycle de développement d'un produit

Lorsqu'on parle d'ingénierie système, il est indispensable d'examiner le cycle de développement du produit ainsi que son cycle de vie. Le cycle de développement (ou modèle) en V est le modèle le plus connu. Il est apparu, tout d'abord, dans le domaine informatique [19].

Le cycle de développement en V présente un flux général pour le processus de développement d'un produit. Il commence par l'identification des besoins de l'utilisateur et l'expression des exigences. Le cycle en V se termine par la validation et la maintenance du système par l'utilisateur. Pour arriver au produit final, chaque étape de la définition du produit doit être validée [20].

Au cours des phases de conception générale et de conception détaillée, les sous-systèmes du système sont identifiés et décomposés en composants. Les exigences initiales sont raffinées puis allouées aux composants du système et aux interfaces spécifiées en détail. Dès que la phase de conception détaillée est terminée, on passe à la réalisation du système. Le but principal des tests unitaires, tests d'intégration et de la validation est de valider chaque étape correspondante dans la partie gauche du cycle en V [20].

L'ingénierie système n'agit pas seulement sur l'étape de développement du système mais aussi sur les différentes phases du cycle de vie du produit. Ce cycle de vie contient,

essentiellement, quatre phases ; la conceptualisation, le développement, l'exploitation et le retrait du produit comme l'indique la Figure 1.1.

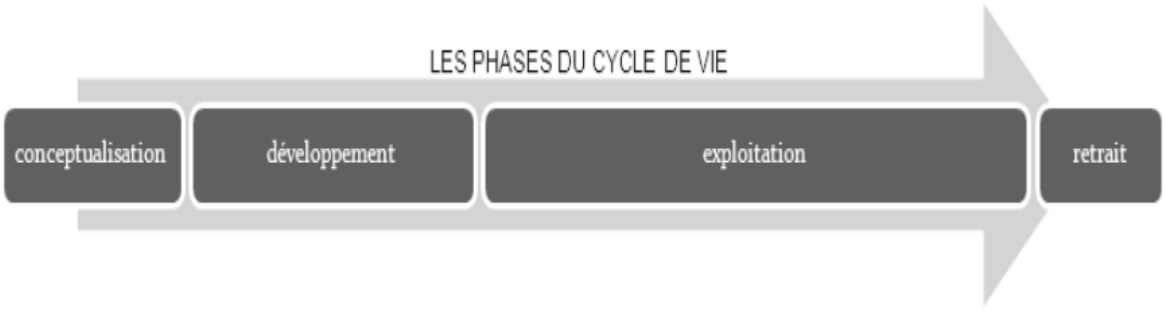


Figure 1. 1: Cycle de vie d'un système [8]

1.2.2.4 Les processus de l'ingénierie système

L'approche de la SE repose sur un ensemble de processus, de méthodes et d'outils. Les processus définissent le **QUOI FAIRE ?** c'est-à-dire les activités à réaliser et les résultats attendus. Les méthodes répondent à la question **COMMENT FAIRE ?** c'est-à-dire de définir la manière de réalisation des activités. Enfin, les outils donnent une réponse à la question **AVEC QUOI FAIRE ?** ils ont pour rôle l'amélioration de l'efficacité de la réalisation des tâches.

On définit un processus par une suite logique de tâches à accomplir afin d'aboutir à un objectif spécifique [21]. En terme d'activités à accomplir, un processus définit le « Quoi ? » sans préciser le « Comment ».

On définit quatre types de processus qui sont décrits par les normes générales de SE. Le premier type correspond aux processus techniques qui ont pour objectif la transformation du besoin en solution. Les processus techniques de développement et les processus techniques après mise en service sont deux groupes de processus techniques. Les processus techniques de développement sont traités par toutes les normes générales de SE et s'appliquent pendant le cycle de vie du système avant mise en service opérationnel. Par contre, les processus techniques après mise en service sont traités uniquement par la norme ISO 15288.

Les seconds processus sont les processus de management qui ont pour tâche la maîtrise des processus techniques. On définit trois processus généraux de management qui sont les processus de planification, de suivi et de maîtrise [22]. On identifie, aussi, des processus spécifiques à l'ingénierie système : 1) Les processus de décision qui ont pour rôle de prendre les meilleurs choix en tenant compte des causes, des conséquences, des risques associés et des stratégies de l'entreprise ; 2) Les processus de gestion et de configuration qui assurent la

cohérence entre les entrées et les sorties après les avoir identifiées, enregistrées et maîtrisées ; 3) Les processus de management de l'information qui collectent les informations puis les transforment pour les fournir aux parties concernées.

Le troisième groupe processus regroupe les processus contractuels qui garantissent les relations entre acquéreur et fournisseur par la définition des stratégies d'acquisition, la préparation d'un contrat, de son suivi et de sa clôture.

La quatrième classe concerne les processus d'entreprise assurant le pilotage stratégique de l'entreprise et des projets au travers de sous-processus que sont les processus de management de l'environnement de l'entreprise, de l'investissement dans les projets et de l'innovation et de l'évolution des lignes de produits. Ils assurent, aussi, le management et le partage des ressources communes par les processus de management des processus, des ressources humaines et des infrastructures.

1.2.2.5 Les normes

L'ingénierie système se base sur trois normes qui décrivent les processus d'IS ; ISO/IEC 15288, IEEE 1220 et EIA 632. La Figure 1.2 présente les parties couvertes par chaque norme. Ces trois normes décrivent la définition du système par les processus techniques mais seule l'ISO 15288 couvre l'ensemble du cycle de vie du système.

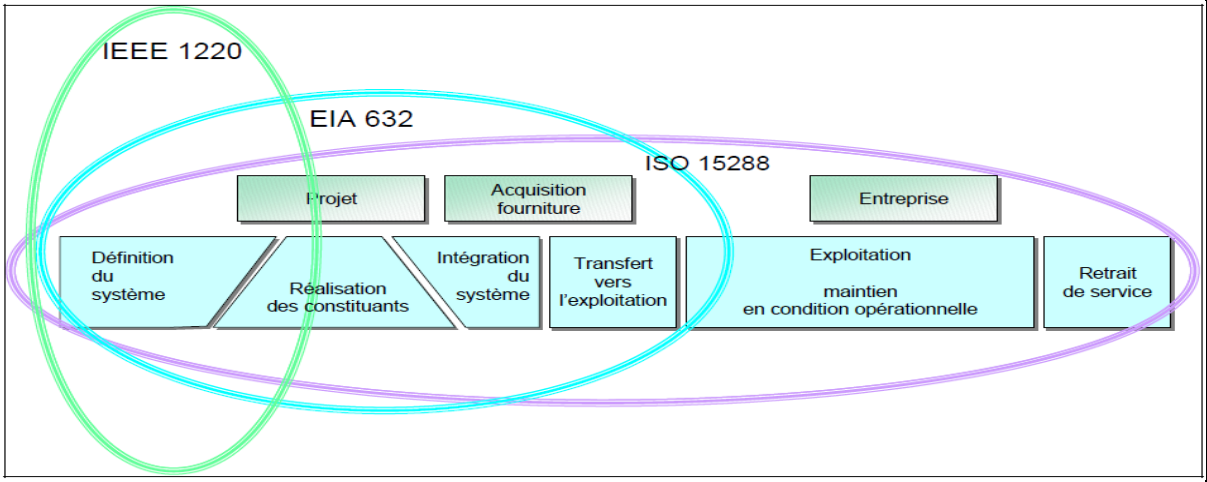


Figure 1. 2: Les domaines de couverture par chaque norme de l'IS [23]

1.2.2.5.1 ISO 15288

La norme ISO/IEC 15288 couvre tous les processus de cycle de vie du système [24]. Elle permet d'étendre les processus techniques à l'ensemble du cycle de vie du système

en couvrant les opérations, la maintenance opérationnelle et l'élimination lors des processus de fin de vie. Elle couvre également les processus de projet, les processus d'entreprise et les processus de contractualisation (client-fournisseur) comme l'indique la Figure 1.3.

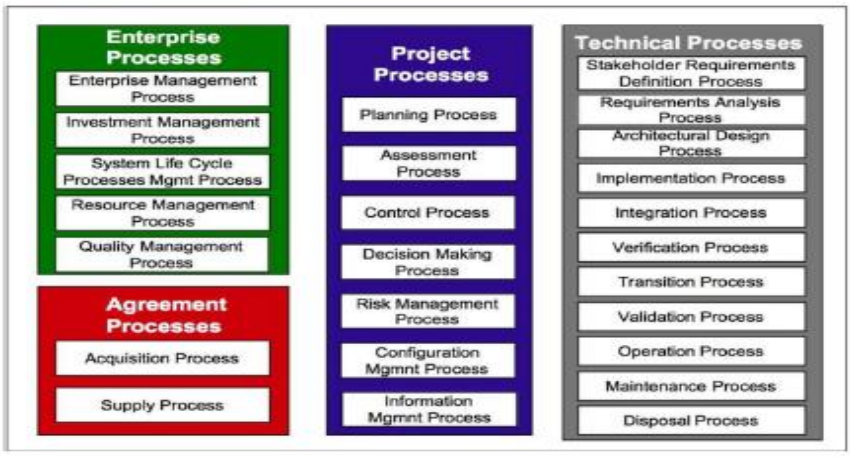


Figure 1. 3: Les processus de la norme ISO 15288 [extrait de 25]

1.2.2.5.2 IEEE 1220

La norme IEEE 1220 [26] permet l'application et la gestion des processus de SE. Cette norme décrit les processus de l'analyse des exigences, de l'analyse fonctionnelle et les processus de l'allocation et de synthèse. Chaque processus est validé avant de passer au processus suivant. La norme spécifie les entrées et les sorties de chaque processus comme l'indique la Figure 1.4.

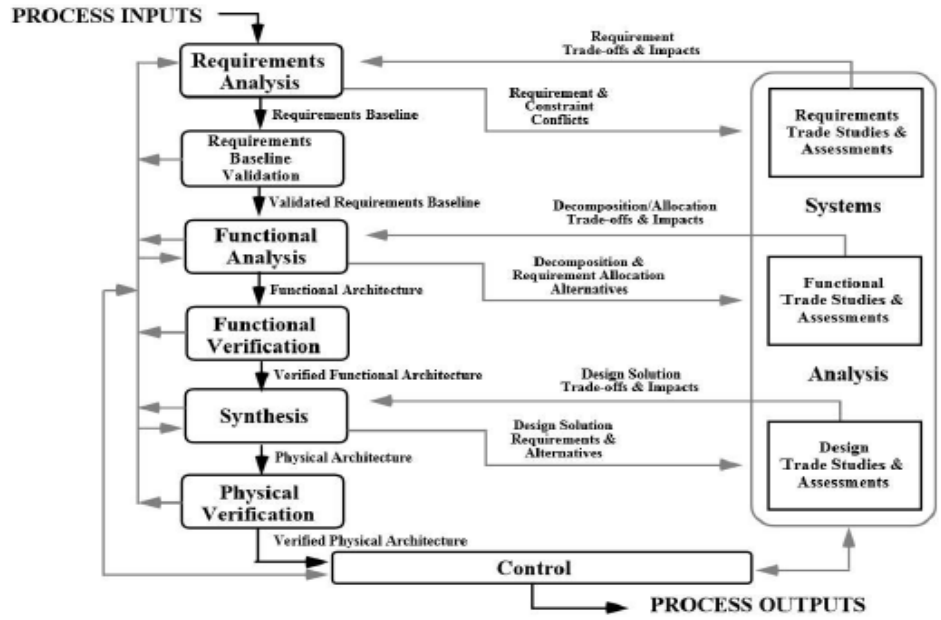


Figure 1. 4: Les processus de la norme IEEE 1220 [25]

1.2.2.5.3 EIA 632

La norme EIA 632 [27] est une norme qui couvre un nombre de processus plus important que la norme IEEE 1220 mais moins important que la norme ISO 15288. Elle couvre les processus techniques de management, les processus d'implémentation des produits et les processus techniques d'évaluation. Une vue d'ensemble de ces processus et de leur relation est donnée à la Figure 1.5.

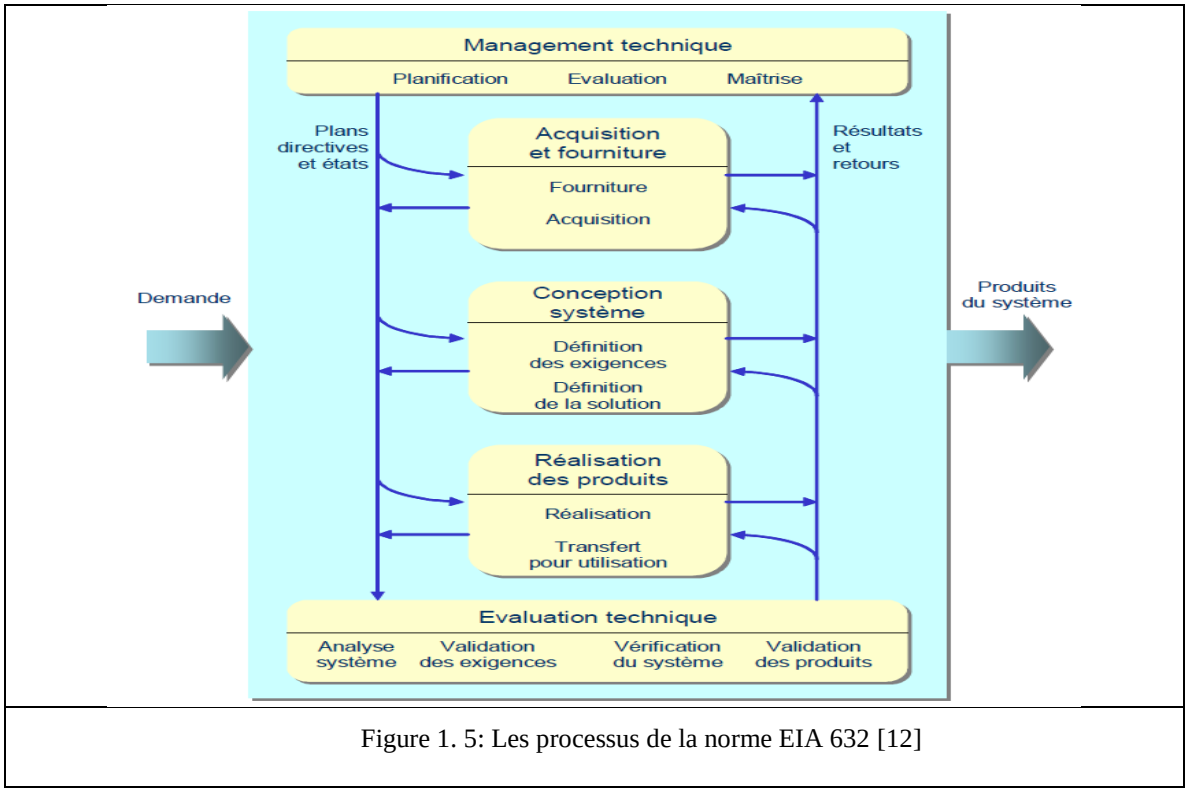


Figure 1. 5: Les processus de la norme EIA 632 [12]

1.2.2.6 Les méthodes de l'ingénierie système

Les méthodes sont celles qui répondent à la question **COMMENT FAIRE ?** pour définir la manière de la réalisation des activités et des tâches définies dans les processus. On distingue deux aspects pour les méthodes de l'ingénierie système : l'aspect démarche qui représente la logique ou l'algorithmique de l'activité des processus et l'aspect de modélisation et de simulation qui permet la compréhension des problèmes et de proposer des solutions à travers des méthodes élémentaires.

1.2.2.7 Les outils de l'ingénierie système

Un outil de l'ingénierie système permet de répondre à la question **AVEC QUOI FAIRE ?** afin de faciliter l'accomplissement des **COMMENT FAIRE.** Il est défini par un instrument qui permet l'amélioration d'une tâche pour qu'elle soit accomplie d'une manière efficace.

1.2.2.8 Ingénierie système basée sur les modèles

Le premier objectif de l'ingénierie système est d'offrir une aide pour la conception et la gestion des systèmes complexes. En effet, le problème est que les processus d'ingénierie traditionnels dépendent fortement des documents ce qui oblige les concepteurs à allouer une grande partie de leurs efforts à la gestion des documents plutôt qu'à l'ingénierie [28]. En effet, ce problème se manifeste lors du classement, de la mise à jour et de la recherche des documents nécessaires. Aussi, on rencontre la problématique de la gestion des différentes versions d'un document et d'être sûr que tout le monde travaille sur la dernière version. En outre, les approches basées sur le texte sont inefficaces pour trouver les erreurs dans la conception d'un système. Elles ne sont pas adéquates non plus pour tester les performances et comparer les solutions possibles [29]. En plus, un système complexe est caractérisé par un comportement qui ne peut pas être soigneusement planifié, compris et anticipé [6], ce qui rend très important l'utilisation de modèles pendant la conception. De plus, ces derniers sont plus expressifs pour décrire les systèmes et plus faciles à être compris que les descriptions textuelles. En plus, la simulation des modèles offre un moyen plus simple et efficace pour effectuer des comparaisons et des compromis entre les conceptions alternatives. Aussi, l'utilisation des modèles facilite l'établissement de la traçabilité entre les différents points de vue et entre les différents niveaux d'abstraction ce qui représente un très grand avantage car il permet de maintenir la cohérence et la continuité entre les différents modèles. Ces avantages ont poussé les ingénieurs systèmes à utiliser les modèles à la place des descriptions textuelles donnant lieu à ce qu'on appelle « l'Ingénierie Système Basée sur les Modèles ».

Un modèle est une abstraction du système réel et ne représente qu'une partie des caractéristiques de ce dernier. Cependant, l'abstraction permet de construire des modèles plus légers, plus faciles à comprendre et moins longs en simulation. Les modèles utilisent quelques caractéristiques intéressantes du système tels que le temps, le comportement du système ou les diverses mesures de performance.

Dans l'approche MBSE, on identifie plusieurs langages et outils de modélisation qui peuvent être utilisés en fonction des différents domaines impliqués dans le système, en fonction du niveau de détail et en fonction des aspects du système à modéliser. Par exemple, nous pouvons utiliser IBM Doors pour la modélisation des exigences, Modelica pour les systèmes physiques, CATIA ou SolidWorks pour la Conception Assistée par Ordinateur (CAO), Matlab Simulink pour les systèmes dynamiques, etc.

Cette thèse se concentre principalement sur les modèles conceptuels. En particulier, on se concentre sur la première étape de conception où les exigences initiales sont définies et progressivement affinées et les modèles fonctionnels abstraits du système sont produits et tracés, selon les exigences correspondantes. Pour ces raisons, SysML, le langage de modélisation des systèmes est choisi comme support pour les modèles système.

1.2.2.9 SysML

SysML, est un langage de modélisation dédié à l'ingénierie système [32]. Il est un profil d'UML (Unified Modeling Language) initialement développé pour l'informatique. Il représente une version étendue d'UML 2.0 et est basé sur les expériences d'utilisation de ce dernier dans certains projets d'ingénierie système. Comme UML, SysML est un langage de modélisation graphique, semi-formel et orienté-objet. L'utilisation du SysML, basé sur un formalisme graphique, rend les modèles plus faciles à expliquer et à comprendre par toutes les personnes impliquées dans le projet. Cela justifie l'utilisation des langages tels que SysML pour couvrir les premières phases de spécification à l'aide d'une modélisation graphique qui favorise le partage des connaissances entre les ingénieurs impliqués. Aussi, SysML permet d'avoir un modèle complet du système qui couvre l'ensemble des étapes partant des exigences jusqu'à la solution physique.

SysML est conçu pour fournir des constructions simples mais puissantes permettant de modéliser les différents problèmes d'ingénierie système [32]. En conséquence, SysML est particulièrement efficace pour spécifier les exigences, la structure, le comportement, les allocations et les contraintes techniques [32]. Ce nouveau profil, SysML, permet de réaliser plusieurs tâches de conception telles que la modélisation de systèmes complexes et embarqués [34]. Il est également de plus en plus adopté par les industriels tels que Valeo, fournisseur de systèmes automobiles [35] et Airbus, concepteur et intégrateur de systèmes aéronautiques [36] pour n'en citer que peu.

SysML contient neuf diagrammes au lieu des treize diagrammes d'UML. Les neuf diagrammes sont divisés en trois catégories : un diagramme d'exigences, quatre diagrammes de comportement et quatre diagrammes de structure. Les diagrammes structurels sont le diagramme de définition de blocs, le diagramme de bloc interne, le diagramme de package et le diagramme paramétrique. Les diagrammes comportementaux sont le diagramme d'activité,

le diagramme de séquence, le diagramme d'états et le diagramme de cas d'utilisation. La Figure 1.6 présente les différents diagrammes dans SysML.

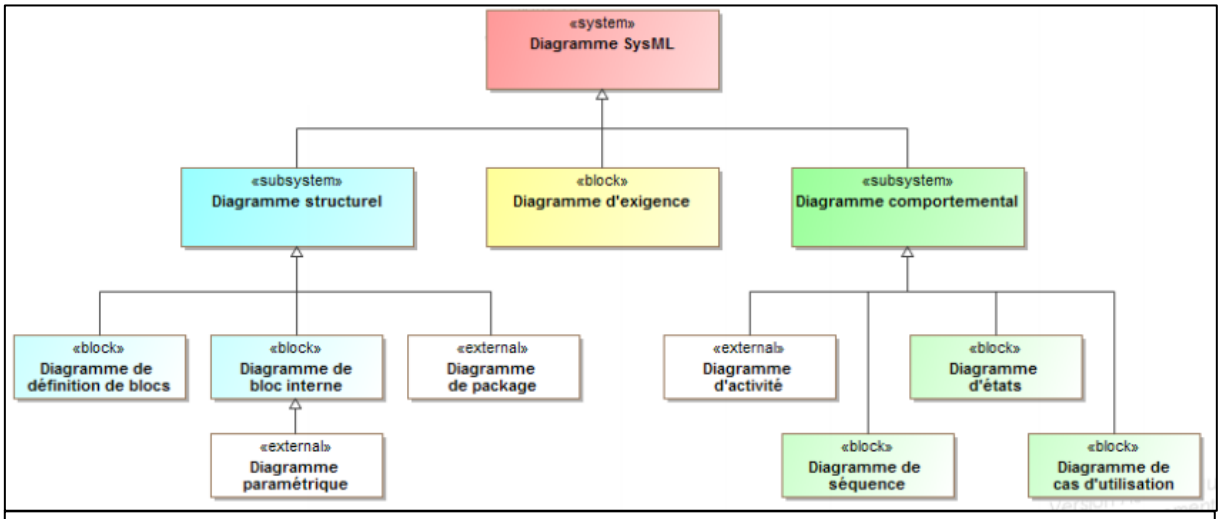


Figure 1. 6: Récapitulatif des diagrammes SysML

1.3 La sûreté de fonctionnement

1.3.1 Les risques

Dans cette partie, nous allons d'abord donner la définition d'un risque, puis présenter les différentes étapes de gestion des risques, les stratégies ainsi que quelques outils de gestion des risques.

1.3.1.1 Définition des risques

La première définition scientifique d'un risque a été donnée en 1738 par Daniel Bernoulli qui a dit que « le risque est l'espérance mathématique d'une fonction de probabilité d'évènements ». Une autre définition donnée par Guillerm [8], « Un risque correspond à une perte ou une dégradation potentielle, identifiée et souvent quantifiable. C'est un danger éventuel plus ou moins prévisible qui peut affecter l'issue du projet. Il est nécessairement lié à une situation ou une activité et est associé à la probabilité de l'occurrence d'un événement ou d'une série d'événements ».

Le risque est caractérisé par sa criticité. La Figure 1.7 montre que la criticité des risques dépend de deux grandeurs qui sont la probabilité d'occurrence du risque et la sévérité qui est une évaluation des effets causés par ce risque. Afin de rendre le risque d'un système acceptable, des stratégies de réduction de la criticité des risques sont mis en place. Ces stratégies sont soit la réduction de la probabilité de l'occurrence du risque par la prévention (par exemple

l'utilisation de la redondance qui est le cas dans ce travail, voir chapitre 2) soit par la diminution de la sévérité par des mécanismes de protection.

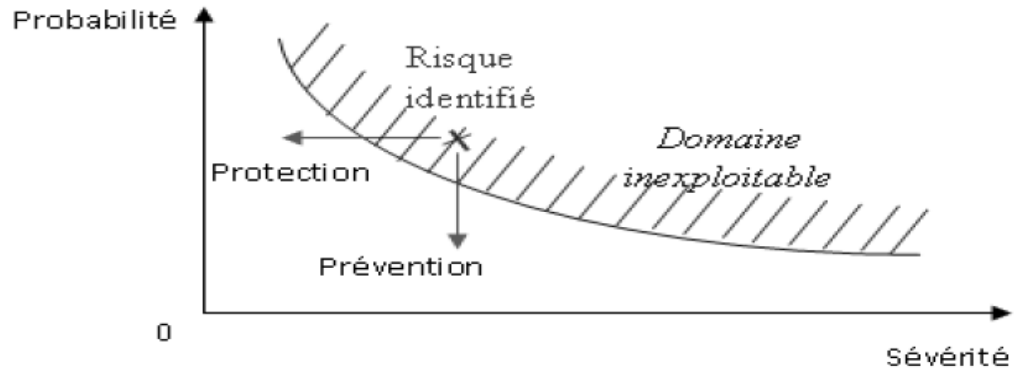


Figure 1. 7 : Criticité des risques [8]

1.3.1.2 Etapes de gestion des risques

Pour bien maitriser les risques d'un système, certaines étapes doivent être suivies La première étape est l'identification des risques qui est représentée par la collection de tout type d'information afin de créer un inventaire des risques. La deuxième étape est l'évaluation et le classement des risques. Dans cette étape, on évalue la probabilité d'occurrence et la sévérité afin de déterminer la criticité du risque. Cette évaluation permet la génération d'un classement entre les différents risques selon leur criticité. Un risque avec une criticité élevée doit être traité. Par contre un risque avec une faible criticité peut être accepté sans aucune intervention. Dans la pratique, le coût des dégâts et du traitement du risque est un facteur très important qui intervient dans la décision d'acceptation ou du traitement. Finalement, la troisième étape est le réexamen des sous-systèmes déjà traités dans la partie précédente afin de vérifier si de nouveaux risques sont apparus. C'est l'étape d'analyse des conséquences et de suivi de l'évolution de la criticité pendant le projet.

1.3.1.3 Stratégies de la gestion des risques

Dans cette partie, on identifie quelques actions à appliquer afin de réduire les risques possibles d'un système [8]. On cite les deux actions suivantes :

- En cas d'un risque avec une faible criticité c'est-à-dire un risque avec une probabilité d'occurrence faible ou une conséquence très limitée, ce risque peut être accepté. Cette stratégie ne permet pas la réduction des risques mais elle reste acceptable surtout si le coût de l'intervention est cher.
- La diminution de la probabilité d'occurrence ou de la sévérité des risques.

1.3.1.4 Outils de la gestion des risques

Afin de réduire le risque d'un système, plusieurs outils de gestion des risques sont utilisés afin d'aboutir à un bon résultat [8]. A titre d'exemple, on cite les outils suivants :

- La prévention qui permet la réduction de la probabilité de l'occurrence d'un risque ;
- Les actions correctives qui permettent l'allégement des conséquences du risque lorsqu'il se produit ;
- La diversification qui consiste à scinder le risque en plusieurs risques indépendants afin d'amortir les effets du risque initial.

1.3.2 Les méthodes de la sûreté de fonctionnement

Les méthodes d'analyse de la Sûreté De Fonctionnement (SDF) ont pour objectif l'évaluation de la sûreté de fonctionnement du système pendant la phase de conception et de s'assurer que les systèmes conçus vérifient un niveau de sûreté satisfaisant. Ces analyses peuvent être classées en analyses qualitatives et analyses quantitatives. De plus, on distingue des méthodes inductives qui permettent l'analyse du particulier vers le général et des méthodes déductives qui assurent l'analyse du général vers le particulier.

Les artefacts d'analyse de sûreté peuvent être construits à partir soit des modèles structurels soit des modèles comportementaux du système. Pour les modèles structurels, on ne considère que l'aspect statique du système par contre pour les modèles comportementaux, on traite l'aspect dynamique du système.

Différentes méthodes et techniques d'analyse de la sûreté existent et elles sont utilisées pour des différents objectifs. Parmi ces méthodes et techniques, on cite l'Analyse des Modes de Défaillances et de leurs Effets (en anglais : Failure Modes Effects and Analysis (FMEA)), la méthode des arbres de défaillances (en anglais : Fault Tree Analysis (FTA)), l'Analyse Préliminaire des Dangers (APD), HAZard and OPerability analysis (HAZOP), la technique de diagramme de fiabilité et la méthode de diagramme de fiabilité dynamique [37,38].

Parmi les méthodes cité, FMEA et FTA, sont les plus utilisées dans l'analyse de sûreté. FMEA est une méthode inductive d'analyse de la sûreté de fonctionnement qui identifie les modes de défaillance des fonctions ou des composants du système, puis détermine leurs effets sur le système [37, 38]. Elle vise à évaluer les effets des modes de défaillance potentiels des composants ou des fonctions, et à en réduire les risques potentiels lors de la phase de conception du système. FTA à l'opposé, est une méthode déductive. Cette méthode peut conduire soit à

une analyse qualitative ou à une analyse quantitative. Les deux analyses partent d'un événement indésirable et combine les composants qui lorsqu'ils tombent en panne ensemble, l'effet indésirable se produit. L'analyse quantitative évalue la probabilité de l'événement indésirable à partir des différentes probabilités des événements élémentaires de l'arbre de défaillance.

Dans cette partie, on présente quelques méthodes d'analyse de sûreté de fonctionnement, parmi celles citées ci-dessus.

1.3.2.1 Analyse des Modes de Défaillances et de leurs Effets

FMEA est la première technique systématique pour l'analyse des défaillances [39, 37]. C'est une méthode de fiabilité préventive utilisée dans la modification des systèmes et qui accompagne le système pendant son cycle de conception pour la modification de composants [40]. Elle a été développée au milieu des années 1960 aux États-Unis par la NASA (National Aeronautics and Space Administration) pour l'étude des problèmes en relation avec les systèmes militaires.

Cette méthode est très utilisée parce qu'elle permet une identification précoce du danger dans le processus de conception. Cette identification avancée élimine les problèmes flagrants issus de la découverte tardive des dangers. Elle réduit, aussi, les dépenses de la maintenance du système. Son objectif est d'évaluer les effets des modes de défaillance des composants ou des fonctions des systèmes afin de les éliminer. Chaque élément du système est traité à part pour identifier ses modes de défaillance ainsi que les causes et les effets de chacun de ces derniers.

FMEA est présentée sous forme d'un tableau où on note pour chaque composant ou fonction les différents modes de défaillances ainsi que les causes et effets de chaque mode de défaillance. Chaque ligne de ce tableau correspond à un élément et les colonnes représentent l'ensemble des propriétés des modes de défaillance comme les causes, les effets, etc.

En résumé, une FMEA permet, en premier lieu, l'identification des modes de défaillance des différentes parties du système. En second lieu, elle assure l'évaluation des effets de chaque mode de défaillance des composants sur les fonctions du système.

1.3.2.2 Méthode de l'arbre de défaillances

La méthode des arbres de défaillances appelée est une approche déductive (Top-down) c'est-à-dire du plus général vers le plus détaillé. Elle est passée par plusieurs étapes dès sa naissance jusqu'à présent. D'abord, cette méthode a été mise en place en 1962, la naissance de cette méthode dans la société Bell Téléphone par Watson. Puis, en 1965, Haasl a réussi à formaliser ces règles de construction. Ensuite, Wasley a commencé à mettre les bases de

l'évaluation quantitative dans les années 70. Enfin, en 1992, en codant les Diagrammes de Décision Binaire (DDB), Madre et Rauzy ont réussi à obtenir une grande efficacité de calcul [41]. Le but de cette méthode est d'identifier les combinaisons possibles qui causent l'évènement indésirable.

L'arbre de défaillance est composé de trois types d'évènements : évènements feuilles, évènements intermédiaires et évènements indésirables. Ces évènements sont reliés entre eux par des portes logiques qui définissent la relation et par la suite la nature de la défaillance possible.

Cette méthode présente plusieurs avantages, on cite par exemple la facilité d'édition car il s'agit d'une méthode graphique qui utilise les symboles ainsi que la facilité de l'insertion d'une modification. En plus, elle offre une grande simplicité pour la compréhension et la lecture de l'arbre de défaillance même pour les gens qui ne connaissent pas la sûreté de fonctionnement. Également, comme on l'a déjà mentionné, cette méthode présente une approche déductive qui permet la construction intuitive de l'arbre.

Comme toute autre méthode, la méthode des arbres de défaillance présente aussi quelques inconvénients. En effet, les arbres de défaillances ne permettent pas l'évaluation de la disponibilité opérationnelle des systèmes réparables. En plus, il faut refaire la construction de l'arbre en cas d'évolution du système.

La FTA permet l'analyse qualitative et quantitative. L'analyse qualitative est utilisée pour identifier la/les combinaison(s) nécessaire(s) et suffisante(s) d'événements feuilles donnant lieu à l'événement indésirable. Les techniques les plus utilisées dans l'analyse qualitative des arbres de défaillances sont les ensembles de coupes minimales (en anglais : Minimal Cut Set (MCS)), les ensembles des chemins minimaux (en anglais : Minimal Path Set (MPS)) et les défaillances à cause commune (en anglais : Common Cause Failure (CCF)). Un ensemble de coupe (en anglais : Cut Set (CS)) est une combinaison de défaillances des composants entraînant la défaillance du système. Un MCS est un ensemble de coupe qui, si un élément est supprimé de cet ensemble, le reste n'est plus un ensemble de coupes. L'ensemble des chemins (en anglais : Path Set (PS)) est l'opposé de CS. Le PS est une combinaison de composants qui, s'ils n'échouent pas, le système reste fonctionnel. Un MPS est un ensemble des chemins qui, si un élément est supprimé de cet ensemble, le reste n'est plus un chemin défini. Les CCF sont des défaillances dépendantes et résiduelles dans lesquelles deux événements de défaillance ou plus existent en même temps en raison d'une même cause

partagée. L'identification de ces derniers aide le concepteur à identifier les points faibles du système [42].

Concernant l'analyse quantitative, elle est effectuée pour calculer la probabilité de l'événement indésirable. L'analyse quantitative comporte deux approches différentes : l'analyse quantitative en temps discret et l'analyse quantitative en temps continu. L'analyse quantitative en temps discret est une approche qui considère la durée de vie complète du système comme un événement singulier. En d'autres termes, chaque composant ne peut être défaillant qu'une seule fois dans un temps fixe. Par contre, l'analyse quantitative en temps continu est une approche qui prend en compte l'évolution des défaillances du système au fil du temps. Il est généralement caractérisé par une fonction de probabilité.

L'arbre de défaillance peut également être transformé en un arbre de succès réussite. Ce dernier présente comment empêcher l'événement indésirable de se produire. Les conditions appliquées sur l'arbre de réussite garantissent que l'événement indésirable ne se produise pas. Donc, l'arbre de succès est un outil précieux qui donne des informations équivalentes à celle de l'arbre de défaillance mais d'un point de vue succès.

1.3.2.3 Comparaison entre FMEA et FTA

En comparant FMEA avec la FTA, on remarque que les combinaisons de défaillance ne font pas partie de FMEA [40]. La FMEA s'occupe davantage de l'évaluation des modes de défaillance d'un système et de leurs effets sur le système, ce qui en fait une bonne source de modes de défaillance possibles pour le FTA. La principale différence entre les deux méthodes est que la FMEA est une méthode inductive (on part du détaillé au général) et la FTA est une méthode déductive (on part du général au détaillé). La méthode FMEA est généralement une bibliothèque de toutes les défaillances potentielles et de leurs conséquences, tandis que la FTA permet une analyse détaillée des relations logiques et temporelles menant à une défaillance de l'évènement indésirable.

1.3.2.4 Analyse Préliminaire des Dangers

En 1960, les Américains ont utilisé l'analyse préliminaire des dangers pour la première fois dans le cadre de l'analyse de sécurité [8]. Ensuite, la société Boeing l'a formalisée pour l'adapter aux applications aéronautiques. Cette méthode a pour but l'identification des dangers d'un système et leurs causes ainsi que l'évaluation de la sévérité des conséquences liée aux situations dangereuses. Pour analyser correctement le système il faut, en premier lieu, identifier le contexte opérationnel du système. En second lieu, il faut identifier les risques potentiels et la

gravité de leurs conséquences. Suite à ces deux dernières étapes, des actions correctives possibles sont à définir. Ensuite, il faut vérifier les conditions de panne et compléter les exigences de sécurité. Enfin, on doit évaluer les objectifs atteints [29].

1.3.2.5 HAZard and OPerability analysis

La méthode HAZOP est l'une des méthodes d'analyse de la sûreté de fonctionnement. Elle a été initialement développée dans les années 1960 pour analyser les principaux procédés chimiques, mais elle a été étendue à d'autres domaines complexes.

Une étude de danger et d'opérabilité est un examen structuré et systématique d'un processus ou d'une opération complexe, planifiée ou existante. Son but est l'identification et l'analyse des risques potentiels d'un système ainsi que les problèmes de fonctionnement qui sont susceptibles de se produire. Elle suit le système durant son cycle de vie. La méthode est basée sur la division de la complexité globale du processus en un certain nombre de sections plus simples appelées « nœuds » qui sont ensuite examinées individuellement. Elle est réalisée par une équipe multidisciplinaire expérimentée lors d'une série de réunions. La méthode HAZOP est qualitative et vise à stimuler l'imagination des participants pour identifier les dangers potentiels et les problèmes d'opérabilité. Le processus d'identification des dangers consiste à identifier les déviations par la comparaison des paramètres à une liste de mots-clés comme : plus, moins, non, inverse, tard, etc., qui sont combinés avec d'autres paramètres comme la vitesse, la distance, la pression, etc… [37].

L'inconvénient principal de cette méthode est que son succès repose, en grande partie, sur la capacité de l'équipe à la prédiction basée sur les expériences passées.

1.3.2.6 La technique de diagramme de fiabilité

La technique de diagramme de fiabilité (en anglais : Reliability Block Diagram (RBD)) est une technique basée sur la schématique pour montrer comment les composants contribuent au succès et à la défaillance d'un système global. Cette méthode est très utilisée vue sa simplicité et son aspect pratique. Ce modèle consiste à avoir un point d'entrée, un point de sortie et un ensemble de blocs connectés entre eux en parallèle ou en série. Chaque bloc représente un composant physique qui fonctionne normalement. Si un des composants est défaillant, il se comporte comme un interrupteur ouvert. Le diagramme reste fonctionnel s'il existe un chemin reliant le point de départ au point final sinon le diagramme sera non opérationnel [43, 45].

L'avantage principal de cette approche est sa simplicité. En effet, les ingénieurs de différents domaines peuvent l'utiliser et la comprendre facilement. Mais cette méthode présente, aussi, un inconvénient majeur qui est le manque d'information sur le comportement du système.

1.3.2.7 Méthode de diagramme de fiabilité dynamique

Vu les limites du RBD à analyser les systèmes dynamique et complexe, la méthode de diagramme de fiabilité dynamique (en anglais : Dynamic Reliability Block Diagram (DRBD)) qui est la dérivée de RBD permet la résolution de ces systèmes. En effet, on obtient un DRBD par l'extension de RBD par de nouveaux outils qui permettent la modélisation dynamique et la dépendance comportementale entre les composants [44].

1.3.2.8 Réseaux de Petri

En présence de pannes, le réseau de Pétri permet la modélisation dynamique des systèmes réparables. Il est développé essentiellement pour l'étude des systèmes dynamiques. Ces derniers passent d'un état à un autre pour chaque transition des composants (réparation, défaillance). Le réseau de Pétri est un graphe bi partie qui contient des cercles et des rectangles. Les cercles représentent les états et les rectangles représentent les transitions. Les états sont reliés entre eux par deux arcs et une transition. Le réseau de Pétri permet la recherche des états non-accessibles, des blocages, des bouclages, des causes d'attentes et des conflits. Il permet l'analyse séquentielle d'un système. Ce réseau présente une facilité de maîtrise par rapport aux autres méthodes. Mais, pour un très grand nombre d'états (des milliers) le réseau sera complexe et la génération d'un graphe de Markov est quasi impossible.

1.3.3 Les normes et standards de la sûreté de fonctionnement

1.3.3.1 ARP4754

L'ARP4754 [7] est une norme nommée "*Guidelines for Development of Civil Aircraft and Systems*". Ce standard a été créé par la « Society of Automotive Engineers (SAE) » en 1996 et a été mis à jour en 2010. Il donne les lignes directrices pour le développement des aéronefs civils et des systèmes civils, en mettant l'accent sur les aspects liés à la sûreté de fonctionnement.

La Figure 1.8 représente les processus de la norme ARP 4754. Cette norme inclue dans son développement des méthodes de sûreté de fonctionnement comme le Functional Hazard Assessment (FHA), Common Cause Analysis (CCA), Preliminary System Safety Assessment (PSSA) et System Safety Assessment (SSA).

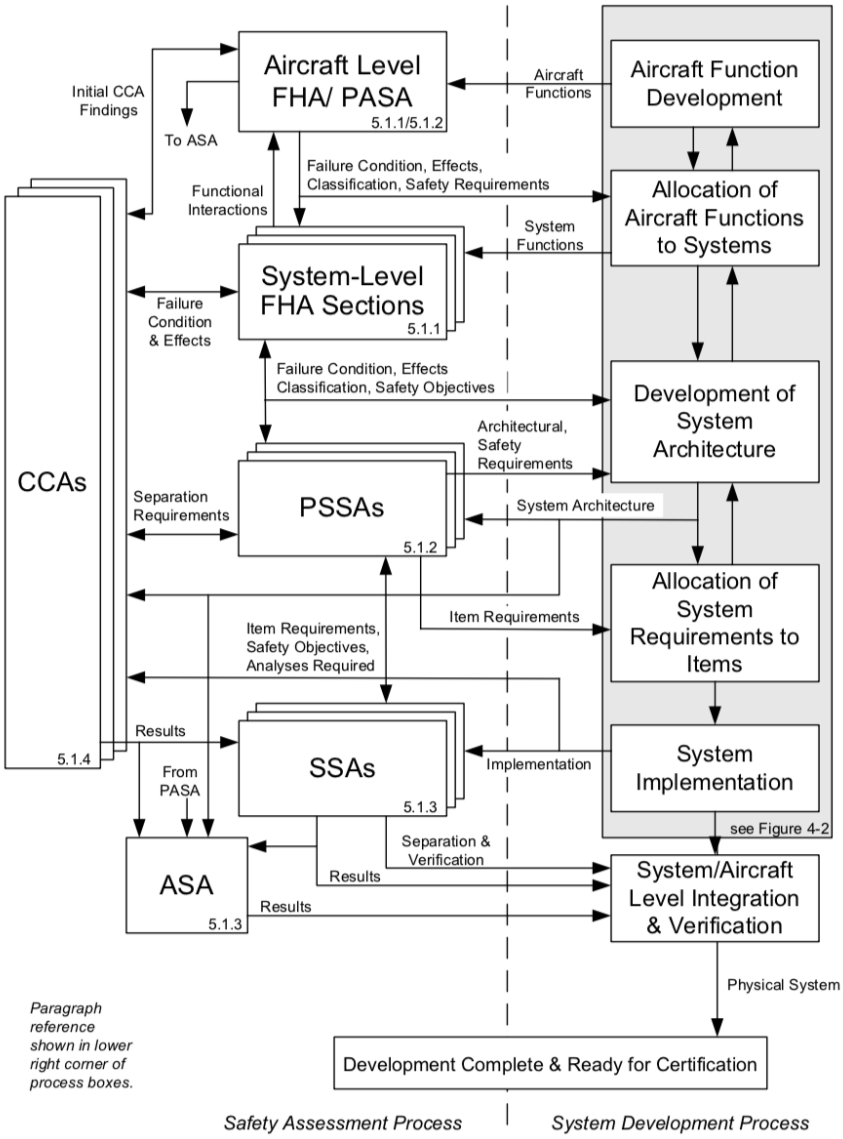


Figure 1. 8 : Les processus d'analyse de sûreté de fonctionnement dans la norme ARP 4754 [7]

Cette norme commence par l'identification des risques potentiels dans le système par une analyse préliminaire des risques. Après, ils sont classés selon leurs criticités. Après une analyse préliminaire, on passe à l'analyse approfondie ainsi que les analyses des causes communes. Enfin, on passe à l'évaluation du système pour valider s'il satisfait les exigences de sûreté de fonctionnement ou pas.

1.3.3.2 ARP4761

La norme ARP 4761 [46] intitulée "*Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*" est une norme de la SAE. Cette norme est destinée à être utilisé simultanément avec la norme ARP 4754 et ne convient que pour les systèmes aéroportés. Ce standard définit un processus utilisant des techniques de

modélisation communes qui permet l'évaluation de la sûreté de fonctionnement des systèmes. Il donne un aperçu de certaines méthodes d'analyse de la sûreté de fonctionnement. Les méthodes couvertes par cette norme sont l'Analyse des Modes de Défaillances et de leurs Effets, la Méthode des Arbres de Défaillances, évaluation du risque fonctionnel, évaluation préliminaire de la sûreté de fonctionnement d'un système et l'analyse des causes communes.

1.3.3.3 CEI 61508 et ses dérivations

La CEI 61508 [47] nommé "*Functional safety of electrical/electronic/programmable electronic safety-related systems*" est une norme générique de sûreté de fonctionnement des systèmes électroniques. Elle représente une référence dans ce domaine pour tous les grands secteurs industriels. Cette norme répond à une problématique qui est apparu dans les années 1980 pour traiter les nouvelles erreurs induites par l'intégration des logiciels de contrôle. Ce standard est composé de 7 parties.

La première partie de la norme représente les exigences générales. Elle définit les activités à effectuer pendant chaque étape du cycle de vie global de la sûreté de fonctionnement d'un système, ainsi que les exigences de documentation et de conformité à la norme.

La deuxième partie définit les exigences relatives à la sûreté de fonctionnement des systèmes électriques, électroniques et électroniques programmables (E / E / PE).

La troisième partie s'intéresse aux exigences logicielles et en interprétant les exigences générales de la première partie.

La quatrième partie donne les définitions et les abréviations des termes utilisés dans la norme.

La cinquième partie donne des exemples de méthodes d'analyse pour la détermination de Safety Integrity Level (SIL) qui est défini comme un niveau relatif de réduction de risque ou une mesure de performance requise pour une fonction de sûreté.

La sixième partie donne des indications sur l'application de la deuxième et la troisième partie.

La septième partie fournit une brève description sur les techniques utilisées en sûreté de fonctionnement et en génie logiciel. Pour plus d'informations, cette partie donne, aussi, des références à des sources d'informations plus détaillées qui concernent la SDF.

Comme on l'a déjà cité, la norme CEI 61508 permet la description, d'une manière courte, des méthodes, des outils et des techniques à implémenter. Afin d'assurer l'application de cette norme dans les différents domaines et secteurs, plusieurs dérivés ont été créés comme l'indique la Figure 1.9.

Les dérivés de la norme CEI 61508 sont :

- En 2001, la création de la norme CEI 61513 qui est utilisée dans le domaine nucléaire.
- En 2003, la création de la norme CEI 61511 qui est utilisée dans les procédés industriels.
- En 2005, la création de la norme CEI 62061 qui est utilisée dans le domaine de la sécurité des machines.
- En 2011, la création de la norme ISO 26262 qui est utilisée dans le domaine automobile.
- Pour le secteur ferroviaire, on identifie la création de trois normes entre 1999 et 2003 et qui sont EN 50126, EN 50128 et EN 50129.

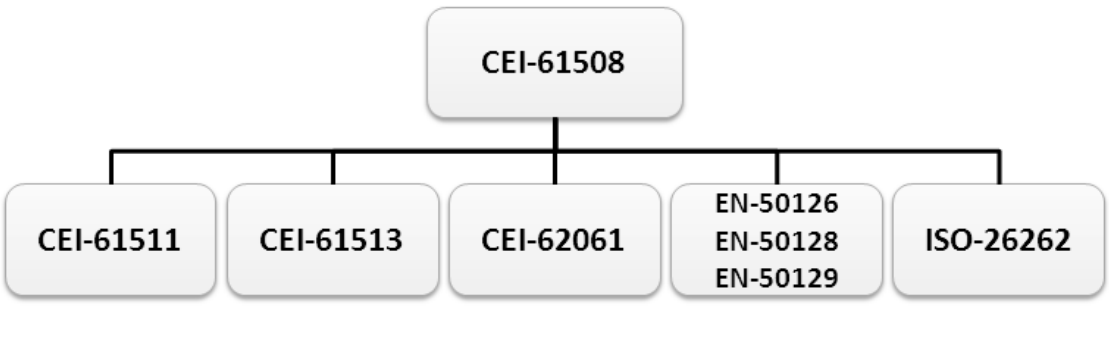


Figure 1. 9 : La norme CEI 61508 et ses dérivées [8]

1.3.4 Analyse de sûreté de fonctionnement basée sur les modèles

Plusieurs méthodes d'analyse de la sûreté de fonctionnement basées sur les modèles sont développées pour formaliser le travail et automatiser certaines étapes de l'analyse de la sûreté de fonctionnement [48]. La plupart de ces travaux sont basés sur des méthodes formelles et

principalement, sur la vérification des modèles. Plusieurs outils sont utilisés afin d'appliquer la vérification des modèles dans l'analyse de la sûreté. Cette section présente certains outils et techniques spécifiques dédiés à l'automatisation de l'analyse de la sûreté.

1.3.4.1 FSAP/NuSMV-SA

FSAP/NuSMV-SA est un outil permettant d'automatiser la génération des arbres de défaillance. FSAP/NuSMV-SA [49], [50], [51], [52] repose sur deux composants : FSAP (Formal Safety Analysis Platform), qui fournit une interface graphique au gestionnaire de tâches d'analyse de la sûreté et le vérificateur de modèle NuSMV2, qui est le moteur d'analyse de la sûreté. FSAP/NuSMV-SA donne certains modes de défaillance prédéfinis ce qui permet à l'utilisateur de spécifier le mode de défaillance d'une entrée / sortie particulière en sélectionnant l'un des modes de défaillance dans la liste prédéfinie. Après, l'utilisateur introduit les défaillances dans le modèle système pour créer un nouveau modèle étendu. Ce nouveau modèle étendu ajoute un comportement dégradé au système d'origine correspondant aux modes de défaillance définis et peut être utilisé pour évaluer la sûreté du système.

1.3.4.2 Galileo – Dynamic Fault Tree Analysis Tool

Galileo [53], [54] est un outil d'analyse et de modélisation des arbres de défaillances dynamiques et statiques qui intègre la méthodologie d'analyse Dynamic Innovative Fault Tree (DIFTree). DIFTree [55] combine des techniques d'analyse des arbres de défaillances statiques et dynamiques en utilisant une approche modulaire. Les arbres de défaillances dynamiques étendent les arbres de défaillances statiques pour permettre la modélisation des systèmes dans lesquels les modes de défaillances peuvent dépendre de l'ordre de défaillances des composants et des défaillances dues à la dépendance fonctionnelle (ex : cause commune). Galileo permet aux utilisateurs d'éditer et d'afficher les arbres de défaillances sous forme textuelle et graphique. Plus d'informations sur Galileo et les autres outils d'analyses des arbres de défaillances sont données dans le chapitre 3.

1.3.4.3 HiP-HOPS

HiP-HOPS (Hierarchically Performed Hazard Origin and Propagation Studies) [56] est une méthode d'analyse de la sûreté découlant d'un certain nombre de techniques classiques telles que l'analyse des défaillances fonctionnelles (en anglais : Functional Failure Analysis (FFA)), l'analyse des modes de défaillances et de ses effets et l'analyse des arbres de défaillances. Cette méthode permet une évaluation du plus bas niveau fonctionnel des modes de défaillances des

composants d'un système complexe. HiP-HOPS est actuellement utilisé par un outil appelé Safety Argument Manager (SAM).

Dans cette méthode, les modèles de défaillance du système, tels que l'analyse des arbres de défaillances et l'analyse des modes de défaillances et des effets, sont construits en établissant la combinaison des effets locaux des défaillances de composants lors de leur propagation dans la structure hiérarchique du système. L'analyse de sûreté compositionnelle fournit des informations préliminaires sur les états-transitions qui représentent la transition entre les états normaux et les états de défaillances du système. Ensuite, dans l'analyse de sûreté comportementale, la vérification du modèle peut être effectuée sur les modèles comportementaux afin de vérifier automatiquement la satisfaction des propriétés de sûreté.

1.3.4.4 AltaRica

AltaRica [57], [58] est un langage de modélisation de haut niveau qui a été conçu pour spécifier formellement le comportement des systèmes en cas de défaillance. Un modèle AltaRica est composé de nœuds caractérisés par leurs états, leurs flux d'entrée et de sortie, leurs événements, leurs transitions et leurs assertions. Les principales phases de l'évaluation de la sûreté avec AltaRica incluent la modélisation du système, les exigences formelles de sûreté, la simulation graphique interactive et l'évaluation de la sûreté [59]. Une fois qu'un modèle système est spécifié dans le langage AltaRica, il peut être compilé dans un formalisme de niveau inférieur, tel que les machines à états finis, les arbres de défaillances et les réseaux de Petri. Les exigences de sûreté sont formalisées à l'aide des opérateurs de logique temporelle et la technique de vérification formelle peut être effectuée par le vérificateur de modèle MEC 5 d'AltaRica. Pour effectuer une analyse de sûreté avec AltaRica, un nouveau modèle système doit être construit en langage AltaRica, ce qui ne garantit pas la cohérence entre les deux modèles. Le maintien de la cohérence entre ces deux modèles nécessite un effort manuel ou des transformations de modèles. Le vérificateur de modèles est limité par la taille des systèmes qu'il peut gérer.

1.4 Intégration entre MBSE et MBSA

1.4.1 Etat de l'art de l'intégration MBSE/ MBSA

Plusieurs méthodologies permettent l'intégration entre l'ingénierie système basée sur les modèles et la sûreté de fonctionnement basée sur les modèles. Dans ce qui suit nous donnons un aperçu des travaux les plus connues.

RAMSAS est une méthode basée sur les modèles, qui permet l'analyse de la fiabilité des systèmes. Cette méthode associe le langage SysML et l'outil Simulink afin de vérifier les performances de fiabilité du système via la simulation. La méthode RAMSAS supporte l'analyse et l'amélioration des propriétés de fiabilité d'un système pendant un processus d'ingénierie système dans les premières phases de conception ainsi que les phases de vérification. RAMSAS permet l'évaluation et la validation des scénarios de configuration et le réglage des paramètres du système afin de guider et de suggérer des choix de conception. Dans les phases de vérification, RAMSAS permet de vérifier, par simulation, les performances de fiabilité du système. RAMSAS est centrée sur quatre phases principales ; l'analyse des exigences de fiabilité, la modélisation du système, la simulation du système et l'évaluation des résultats. Cette méthode a été développée pour la première fois par Garro et al en 2012 [60, 61, 62, 63]. Ensuite, cette méthode a été utilisée avec l'outil Modelica dans [64, 65] et plus récemment dans [66].

MéDISIS est une méthodologie basée sur les modèles, qui permet la génération des FMEA à partir des modèles fonctionnels des systèmes. Cette méthodologie associe le langage SysML qui permet la description du système et le langage AltaRica qui permet la construction des modèles dysfonctionnels afin de calculer les indicateurs de fiabilité. David dans sa thèse de doctorat [67] a décrit la démarche de cette méthodologie qui commence par la génération automatique de FMEA préliminaire. Ensuite, les diagrammes structurels de SysML tels que le diagramme de définition de bloc et le diagramme à blocs interne et les diagrammes comportementaux tels que les diagrammes de séquence et le diagramme d'activité sont analysés en détail pour donner une liste exhaustive des modes de défaillance pour chaque composant et chaque fonction avec leurs causes et leurs effets possibles. Une base de données sur les comportements dysfonctionnels du système est utilisée pour faciliter l'identification des modes de défaillances au cours des différentes phases d'analyse.

Philipp Helle dans [48] présente un processus d'intégration de MBSA dans un MBSE basé sur SysML. Dans ce travail, une extension de SysML permet d'inclure des informations relatives à la sûreté dans le modèle système, ce qui permet à l'ingénieur système de prendre certaines décisions relatives à la sûreté de fonctionnement du système sans l'aide d'un expert de sûreté. Un programme Java, appelé **Safety Analyzer,** est également implémenté. Ce programme utilise le modèle système pour extraire les informations pertinentes. La sortie de ce programme fournit les séquences de coupe minimales pour chaque mode de défaillance et chaque solution système.

Mhenni et al. [68] décrivent une approche intitulée **SafeSysE** intégrant MBSE et MBSA. Cette approche intégrée d'ingénierie systèmes et d'analyse de la sûreté est présenté en 6 étapes. La Figure 1.10 présente le diagramme d'activités du processus intégré ainsi que les échanges entre les différentes étapes. Le diagramme est partagé en deux parties afin de distinguer le processus de l'ingénierie systèmes et de l'analyse de sûreté. Le processus intégré commence par un processus de définition et d'analyse des besoins, avec un ensemble d'exigences initiales décrivant le besoin. Ensuite, les différentes étapes, y compris les activités de conception et les analyses de sûreté, sont effectuées successivement. Les bases de données permettent de stocker les différents artefacts issus de chaque activité.

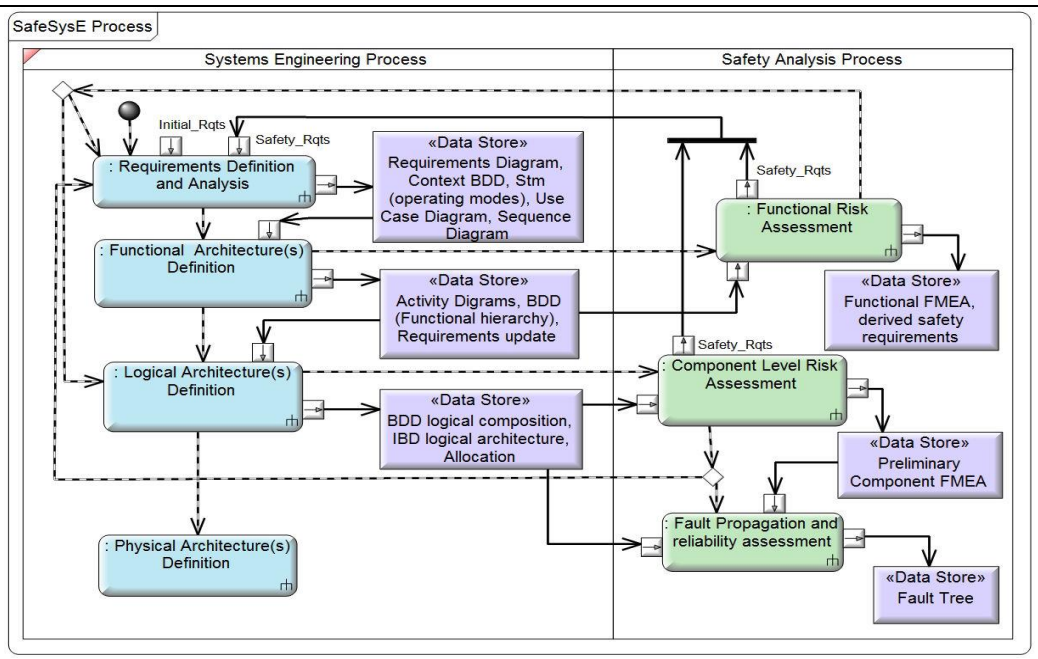


Figure 1. 10: Processus intégré SafeSysE

Etape 1 :

Cette méthodologie commence par la définition et l'analyse des exigences qui permettent la description des fonctionnalités et les interfaces externes du système. Plusieurs diagrammes SysML, tels que les diagrammes de cas d'utilisation et les diagrammes de définition de bloc, peuvent être utilisés pour faciliter l'identification des exigences.

Etape 2 :

Sur la base des exigences fonctionnelles identifiées à l'étape 1, une ou plusieurs architectures fonctionnelles sont identifiées au cours de cette étape. Le résultat final est un modèle hiérarchique de la décomposition de la ou des fonctions principales du système en sous-

fonctions. Dans SysML, les fonctions sont représentées par des activités. Chaque diagramme d'activité représente la décomposition d'une fonction donnée en sous-fonctions. Les diagrammes d'activité montrent également la transformation progressive des flux d'entrée en flux de sortie.

Etape 3 :

Dans la troisième étape, une analyse fonctionnelle du mode de défaillance et de ses effets est utilisée pour identifier les dangers potentiels causés par les défaillances et leurs effets. Dans ce travail, un outil de prototypage a été développé pour générer automatiquement une FMEA partielle basée sur le fichier XML Metadata Interchange XMI généré à partir du modèle SysML. La FMEA générée contient la liste des fonctions et une liste générique des modes de défaillance. L'expert en sûreté procède ensuite à l'analyse et complète FMEA par les données pertinentes. Toutes ces nouvelles informations de sûreté sont ensuite mises à jour dans le modèle SysML via l'extension de profil de sûreté. L'écart entre l'analyse de la sûreté et la modification de la conception est réduit grâce à ce modèle intégré. À la fin de cette étape, les exigences de sûreté sont dérivées et ajoutées à l'ensemble d'exigences.

Etape 4 :

Une fois que l'architecture fonctionnelle est définie en tenant compte des résultats de l'analyse de sûreté de l'étape précédente, une ou plusieurs architectures logiques sont définies en allouant des composants à des fonctions. Un diagramme de définition de bloc (BDD) décrit les composants du système et un schéma synoptique interne (IBD) décrit les interactions entre les composants. L'architecture logique définie à cette étape prend déjà en compte les aspects de sûreté puisqu'elle intègre les résultats de l'évaluation de la sûreté fonctionnelle réalisée à l'étape 3.

Etape 5 :

Lorsque la structure du système est définie, les résultats de l'analyse de la sûreté sont mis à jour et une évaluation des risques au niveau des composants est effectuée. À cette fin, un FMEA composant est généré à partir du fichier XMI, comme à l'étape 3 pour le FMEA fonctionnel. Pour assurer la cohérence avec l'analyse de sûreté précédente, la version FMEA générée, en plus des composants, contient devant chaque composant les fonctions affectées au composant, ainsi que les modes de défaillance identifiés au niveau fonctionnel à titre de rappel. L'expert en sécurité identifie ensuite les modes de défaillance au niveau des composants et

effectue une analyse sur FMEA. S'il existe des risques identifiés avec un niveau inacceptable, ces risques doivent être éliminés ou réduits à un niveau acceptable en apportant des modifications à la conception.

Le problème est que ce travail ne guide pas encore l'architecte système à effectuer des modifications sur l'architecture du système en fonction des recommandations des experts en sûreté. Pour cela, une méthodologie est proposée dans le chapitre suivant afin d'aider l'architecte système à améliorer la structure du système (de point de vue sûreté) en intégrant la redondance pour les composants critiques.

Etape 6 :

La dernière étape est l'analyse des arbres de défaillances. Les arbres de défaillance sont utilisés à la fois pour les analyses qualitatives et quantitatives. Les arbres de défaillances sont générés automatiquement à partir des IBD de SysML décrivant l'architecture du système. Les informations de l'analyse FMEA précédente sont prises en compte pour créer un arbre de défaillance avec des modes de défaillance spécifiques.

1.4.2 Etat de l'art de la génération des arbres de défaillances statiques et dynamiques

Dans cette partie, on traite l'état de l'art de la génération des arbres de défaillances statiques et dynamiques à partir des modèles structurels et comportementaux du système. Plusieurs langages formels et semi-formels peuvent être utilisés pour modéliser la structure d'un système tels que le langage de modélisation unifié (UML), le langage de modélisation des systèmes (SysML), le langage d'analyse et de conception d'architecture AADL (Architecture Analysis and Design Language) [69], qui représente un standard pour les ingénieurs dans le domaine de l'automobile, le langage de modélisation de structure de systèmes S2ML [70] et AltaRica [71]. Pour permettre la génération des artefacts de la sûreté tels que les arbres de défaillances, les modèles système doivent être enrichis par des défaillances (ou un modèle de défaillance), qui représentent le comportement indésirable du système. Divers travaux ont traité la génération des arbres de défaillances à partir des modèles système. Ces travaux peuvent être classés en deux catégories. Tout d'abord, les travaux utilisant la structure du système pour générer des arbres de défaillances statiques et dynamiques : [72], [73], [74], [75], [76], [77], [78] et [81]. Deuxièmement, des travaux qui utilisent le comportement du système [79], [80] pour générer des arbres de défaillances statiques et dynamiques.

Une autre classification peut être faite en fonction du type des arbres de défaillances générés. Certains travaux génèrent des arbres de défaillances statiques [72], [73], [74], [75], [76], et d'autres génèrent des arbres de défaillances dynamiques [81], [82], [83], [79] et [80].

Dans ce qui suit, les travaux qui utilisent la structure du système pour générer des arbres de défaillances sont d'abord présentés. Ces travaux ont des approches assez similaires car ils étendent le modèle système avec des informations de défaillance pour permettre la génération automatique des arbres de défaillances. La principale différence réside dans le langage utilisé pour construire le modèle système.

Mhenni et al. dans [68] décrivent une approche qui cherche à intégrer les analyses de la sûreté de fonctionnement dans le processus de conception du système. Ils proposent une méthode qui permet la génération automatique des arbres de défaillances statiques à partir des diagrammes SysML. Les composants du système et leurs interactions avec l'extérieur sont modélisés par des diagrammes (IBD). Les ports d'entrées/sorties sont utilisés pour modéliser les interactions entre les composants. Ensuite, les diagrammes sont traduits par un graphe dirigé qui contient des sommets et des arcs, les sommets représentent les composants du système et les arcs représentent les relations entres eux. Puis, un algorithme de génération permet, automatiquement, la génération des arbres de défaillances statiques à partir des diagrammes IBD.

Dans [73], Yakymets et al. proposent une méthode pour la génération automatique des arbres de défaillances à partir des modèles SysML en combinant l'approche analytique avec des méthodes de vérification formelles. Dans ce travail, les diagrammes SysML IBD et BDD sont utilisés pour créer la structure du système. Ces modèles sont annotés avec le comportement de défaillance du système et se sont utilisés pour la génération des arbres de défaillances.

Dans [78], Hofig et al. utilisent les couches architecturales pour étendre la génération des arbres des défaillances des composants (Component Fault Trees (CFT)). C'est une version améliorée des arbres de défaillances statiques qui cherche à les rapprocher de la structure du système. Le diagramme IBD de SysML est utilisé pour décrire la structure du système. Contrairement aux autres méthodes qui ne prennent pas en charge la propagation de défaillance verticale, cette description du système considère à la fois les propagations de défaillances verticales d'un système (division du système en couches logicielles et matérielles) et horizontales (encapsulation de fonctionnalités différentes dans des composants distincts). Les auteurs utilisent également un modèle de défaillance pour enrichir le modèle du système.

Ensuite, toutes ces informations seront intégrées dans des modules, ce qui permet la génération des CFT. Cette méthode ne couvre que l'analyse qualitative des arbres de défaillances.

Zhao dans [74] décrit une approche permettant la génération des arbres de défaillances statiques. La génération de ces arbres de défaillances est réalisée à l'aide d'une transformation de modèle à partir d'UML. Pour créer un arbre de défaillance à l'aide de la transformation de modèle, deux étapes sont nécessaires. Tout d'abord, le modèle UML doit être enrichi pour intégrer les informations relatives à l'arbre de défaillance. Ensuite, le langage ATL (Atlas Transformation Langage) est utilisé pour implémenter la transformation du modèle UML enrichi pour la génération des arbres de défaillances.

Joshi et al. dans [75] proposent une méthode pour la génération des arbres de défaillances statiques à partir des modèles AADL. Le modèle des défauts est obtenu en combinant la description du système faite par AADL et les modèles de défaillances. Trois étapes sont nécessaires pour assurer la génération des arbres de défaillances. Tout d'abord, la génération d'un graphe à partir d'un modèle structurel dans AADL dont les nœuds représentent les composants et les arrêtes représentent les connexions entre les composants. Ensuite, la génération d'un arbre en utilisant un algorithme de graphe récursif est effectué. Enfin, un outil commercial appelé CAFTA utilise l'arbre généré pour produire un arbre de défaillance. CAFTA assure l'analyse qualitative des arbres de défaillances alors que l'analyse quantitative n'est pas abordée dans ce travail.

Dans [76], Papadopoulos et Maruhn proposent une méthode pour la génération automatique des arbres de défaillances statiques à partir des modèles Simulink. L'algorithme développé permet la génération des arbres de défaillances en utilisant la structure du système et les dépendances fonctionnelles entre les composants. Les défaillances sont identifiées en examinant les composants du système par la méthode HAZOP (expliquée dans le premier chapitre).

Castet et al. dans [84] décrivent une approche qui vise à générer d'une manière automatique des artefacts de sûreté de fonctionnement tels que FMEA et les arbres de défaillances statiques à partir des modèles SysML. La redondance est prise en compte lors de la génération des arbres de défaillances. Cependant, cette génération ne prend pas en charge la redondance active, standby, mixte, la porte logique Voting et les portes logiques dynamiques. En outre, il ne prend pas en compte l'ordre de défaillance des composants.

Concernant la génération des arbres de défaillances dynamiques à partir de la structure du système, on trouve que dans [81], Pai et Dugan proposent une approche permettant la génération des arbres de défaillances dynamiques à partir des diagrammes UML. La structure du système et les dépendances entre les composants sont modélisées à l'aide d'UML. Chaque composant est caractérisé par des erreurs standards et des propriétés de défaillances. Ensuite, la structure du système est parcourue et les propriétés de défaillance de chaque composant sont intégrées dans le modèle de défaillance, qui regroupe toutes les propriétés du composant. Le modèle de défaillance a la structure d'un arbre de défaillances où les portes logiques représentent les nœuds. Les portes logiques de l'arbre de défaillance sont identifiées à l'aide des dépendances fonctionnelles entre les composants. La génération des arbres de défaillances est terminée lorsque toutes les portes logiques sont identifiées et attribuées aux nœuds respectifs du modèle de défaillance.

Dans [82], Dehlinger et Dugan proposent une méthode pour la génération des arbres de défaillances dynamiques en étendant l'approche d'Anjali [75]. L'approche précédente a été enrichie pour prendre en compte les composants redondants et les séquences temporelles de défaillances. Finalement, l'arbre de défaillance est ensuite généré et analysé par un outil appelé Galileo.

Tajarrod et Latif-Shabgahi dans [83] proposent une méthode qui permet la génération des arbres de défaillances statiques et dynamiques à partir des modèles Simulink. La structure du système est décrite par les modèles Simulink puis elle est enrichie par un modèle de défaillance. La structure du système est parcourue et seuls les sous-systèmes qui affectent l'évènement indésirable sont pris en compte. Ensuite, les combinaisons logiques sont identifiées et incluses dans la génération des arbres de défaillances par les portes logiques correspondantes à l'aide de la structure du système et des dépendances fonctionnelles.

Dans ce qui suit, on donne un résumé des travaux qui génèrent les arbres de défaillances statiques et dynamiques à partir des modèles de comportement du système.

Dans [79], Li et Li proposent une méthode de génération des arbres de défaillances en utilisant des modèles AltaRica. Les modèles AltaRica décrivent le comportement indésirable des composants et identifient l'évènement redouté. La génération du modèle de défaillance se fait avec l'approche Failure Logic Modeling (FLM) ce qui permet la description du comportement indésirable des composants. La première étape consiste à établir une seule logique de défaillance qui est représentée par un graphe simplifié. Après, on construit un modèle

AltaRica et la génération des arbres de défaillances se fait conformément à la logique de défaillance. Enfin, la méthode de génération sera spécifiée en comparant le modèle AltaRica et l'arbre de défaillance correspondant. Ce travail ne couvre que l'analyse qualitative de l'arbre de défaillance.

Dans [80], Rauzy propose une approche qui permet la génération des arbres de défaillances à partir du comportement du système décrit par une machine à états. L'algorithme de génération des arbres de défaillances contient 4 étapes. Tout d'abord, on décrit le comportement du système à partir des machines d'états. Ensuite, on parcourt le graphe qui décrit tous les états accessibles. Après, on détermine les expressions logiques nécessaires pour atteindre un état. Enfin, seules les expressions cohérentes sont conservées et la génération des arbres de défaillances est terminée.

Brameret et al. dans [85] proposent un algorithme de génération des chaînes partielles de Markov à partir des descriptions de haut niveau. Ce travail vise à éviter la construction manuelle de chaînes de Markov. Ce qui nous intéresse dans ce travail est que la compilation des chaînes partielles de Markov donne les mêmes résultats que le calcul des ensembles de coupes minimales des arbres de défaillances. Ainsi, ce travail fournit la même analyse qualitative que les arbres de défaillances.

Ce qu'on peut déduire de la discussion sur l'état de l'art de la génération des arbres de défaillances est que les travaux dans [73], [75], [76], [78], [79] et [80] permettent la génération des arbres de défaillances à partir de différents modèles (SysML , UML, AltaRica et AADL) sans traiter la génération de la redondance qui est l'objectif principal de cette partie du travail. Les travaux dans [81] et [82] permettent la génération des arbres de défaillances dynamiques à partir d'UML et AADL. Ils traitent la génération des arbres de défaillances en tenant compte de la génération de la redondance, des défaillances dépendantes et des activités de reconfiguration. Les limites de ces travaux sont qu'ils ne gèrent pas la redondance active, standby (avec et sans commutateur) et mixte et qu'ils n'utilisent que les portes logiques statiques et la porte logique SPARE. La génération des arbres de défaillances avec redondance est également traitée dans [74], [100], [84] et [83]. Dans ces travaux, seules les portes logiques statiques sont utilisées, ce qui handicape l'arbre de défaillance de prendre en compte l'ordre de défaillances des composants lors de l'analyse qualitative.

1.5 Problématique

Dans cette thèse, on est parti d'un contexte d'intégration entre MBSE et MBSA afin d'éviter l'écart entre les deux modèles. On s'est basé sur les travaux de Mhenni (SafeSysE) [68] qui permettent la génération des FMEAs à partir d'un modèle système. Cependant, le travail de Mhenni ne permet pas la mise à jour de la structure du système en utilisant les résultats de l'analyse des FMEAs. Pour cela, on proposera, dans un premier temps, de modifier la méthodologie SafeSysE pour améliorer la sûreté du système d'un point de vue structurel.

Dans le cadre de ce travail, on traite des systèmes mécatroniques critiques. Pour cela, on utilisera la solution de la redondance pour diminuer la criticité du système étudié. Donc, on obtient une structure du système avec des redondances à gérer et à analyser.

La première solution qui nous semble la plus évidente est l'utilisation des arbres de défaillances statiques comme dans SafeSysE. Mais les arbres de défaillances statiques ne permettent que la détection des dysfonctionnements structuraux et ne peuvent pas détecter les dysfonctionnements comportementaux du système. En plus, les arbres de défaillances statiques ne prennent pas en compte l'ordre de défaillance des composants et par conséquent ils ne traitent pas les redondances. Pour cela, on se propose de développer une méthode de génération des arbres de défaillances dynamiques avec un profil de redondance pour prendre en compte l'ordre de défaillance des composants et la génération de la redondance via des portes logiques temporels et dynamiques.

1.6 Conclusion

Dans ce chapitre, on a défini qu'est-ce qu'un système mécatronique complexe et on a présenté les principaux concepts qui seront abordés dans ce travail comme le cycle de vie de conception d'un système, les processus, les méthodes et les outils ainsi que les différentes normes. De même, on a défini la sûreté de fonctionnement des systèmes complexes et les différentes méthodes d'analyses et standards pratiquées dans ce domaine. Dans la dernière partie de ce chapitre, on a présenté les anciens travaux qui traitent la problématique de l'intégration entre MBSE et MBSA ainsi que l'état de l'art des travaux qui génèrent les arbres de défaillances à partir des modèles systèmes. Enfin et à la lumière de cet état de l'art, on a pu définir la problématique à résoudre dans ce travail de thèse.

Chapitre 2 : Approche intégrée basée sur les modèles d'ingénierie systèmes et de sûreté de fonctionnement

2.1 Introduction

Les nouveaux systèmes conçus par l'être humain sont de plus en plus complexes car ils impliquent un nombre croissant de composants de différentes disciplines comme la mécanique, l'électrique, l'électronique, etc... Ces composants interagissent entre eux de manière synergique et échangent des flux de type différent. La forte dépendance entre les composants rend inadéquate l'approche traditionnelle consistant à diviser la conception en différents domaines. Au lieu de cela, les concepteurs de différents domaines ont besoin de collaborer ensemble autour d'un modèle de système afin de s'assurer que les contraintes spécifiques de tous les domaines soient prises en compte. À cette fin, une approche MBSE est nécessaire pour gérer la complexité, améliorer la cohérence et permettre la modélisation et la simulation de l'ensemble du système. Comme les différents collaborateurs travailleront sur le même modèle, un langage unifié est nécessaire. Ce qui donne l'importance de la création d'un modèle référence auquel chaque contributeur doit se référer lors de la conception du système, à tout moment, pour disposer de données uniques et actualisées partagées par tous. L'objectif de ce modèle est que tout le monde puisse facilement y trouver les informations nécessaires. Cependant, ce modèle doit garantir les différentes représentations du système principal qui doivent être associées à des liens de traçabilité et être cohérentes les unes avec les autres. SysML est un langage semi-formel de modélisation des systèmes complexes qui vise à fournir une norme unifiée pour la spécification, l'analyse, la conception et la vérification des systèmes complexes pouvant inclure du matériel, des logiciels, des informations, du personnel, des procédures et des installations [32]. Ce langage permet, aussi, la création des modèles multi-vues et des liens de traçabilité. Pour toutes ces raisons, SysML est choisi comme support pour la modélisation du système aux premières étapes de la conception de ce travail.

Avec la complexité importante des systèmes, on remarque l'influence du nombre important des systèmes critiques sur la sûreté de fonctionnement. La sûreté de fonctionnement des systèmes critiques est soumise à des exigences très précises concernant la performance, la fiabilité et la sûreté de ces systèmes [1]. Plusieurs techniques d'analyse de la sûreté de fonctionnement sont établies afin d'évaluer et de gérer les risques potentiels résultant des défaillances ou des dysfonctionnements. Ces analyses reposent toutefois sur des outils indépendants qui sont effectuées séparément par des ingénieurs en sûreté. L'extraction des informations du modèle du système se fait généralement manuellement. En conséquence, ces analyses peuvent causer des erreurs et des pertes de temps. Aussi, au cours de l'analyse de la sûreté de fonctionnement du système, la conception continue à évoluer mais les études de la

sûreté de fonctionnement déjà effectuées sont adaptées à une ancienne version du modèle de conception. Afin d'assurer la cohérence entre les analyses de la sûreté de fonctionnement et la conception du système, les analyses de la sûreté de fonctionnement doivent être effectuées dès les premières étapes de la conception. En effet, ces analyses doivent être réalisées au plus tôt pour faire les bons choix d'architecture et du comportement du système dès les premières étapes de la conception sans recourir à des modifications tardives et coûteuses. Pour répondre à ces exigences, nous avons développé une méthode pour intégrer les analyses de la sûreté de fonctionnement au processus de la conception. Dans ce contexte et afin de réduire la probabilité d'erreur et le temps de développement, une génération automatisée des artefacts de sûreté de fonctionnement via des méthodes génériques est réalisée. Suite à des analyses de la sûreté de fonctionnement, on a proposé une procédure de modification systématique du modèle SysML du système touchant son architecture et son comportement Pour cela, un profil UML est développé pour étendre la sémantique SysML.

Dans ce chapitre, nous allons décrire notre méthodologie d'intégration de la sûreté de fonctionnement dans l'approche de l'ingénierie système. Ce travail représente la suite des travaux de Faida Mhenni et sa méthodologie appelée SafeSysE [25]. Basée sur un modèle SysML, cette méthodologie fournit un modèle système de référence commun à tous les contributeurs, prenant en compte les contraintes de tous les domaines. Sur la base de ce modèle, la méthodologie développée dans notre travail automatise certaines étapes de génération d'artefacts de la sûreté de fonctionnement. De plus, elle permet la détection des modes de défaillance ainsi que les fonctions et les composants critiques et propose des modifications sur l'architecture du système ainsi que son comportement.

Ce chapitre est organisé comme suit. On commence par la définition de la notion de la tolérance aux pannes et l'identification des différents types de redondance. Puis, on présente la méthodologie qui définit le processus d'intégration de l'ingénierie système et de la sûreté de fonctionnement. Ensuite, on décrit les deux premières étapes de la méthodologie. On commence par la description de la génération des FMEAs à partir des diagrammes SysML. Puis, on passe aux choix d'architecture par la recommandation des modifications structurelles du système suite à l'analyse des FMEAs. Ce chapitre se termine par un cas d'étude qui illustre cette approche.

2.2 Tolérance aux pannes et redondance

2.2.1 Définition de tolérance aux pannes

La tolérance aux pannes est la capacité d'un système de continuer à exécuter les fonctions prévues en présence de pannes [86]. Au sens large, la tolérance aux pannes est associée à la fiabilité, au bon fonctionnement et à l'absence des pannes. Un système à tolérance aux pannes doit être capable de gérer les pannes des composants matériels ou logiciels individuels, les pannes de courant ou de tout autre type de problèmes imprévus.

La tolérance aux pannes est nécessaire car il est pratiquement impossible de construire un système parfait. Le problème fondamental des nouveaux systèmes est que leur fiabilité diminue dès l'augmentation de leur complexité (par l'augmentation de nombre de composants). Il est aussi probable que certains facteurs environnementaux imprévus ne soient pas pris en compte en dépit des perfections du système. L'objectif de la conception à tolérance de pannes est de minimiser la probabilité des pannes qui gênent les utilisateurs et entraînent ; une perte d'argent, de blessures humaines ou de catastrophes environnementales ou autres.

Il existe différentes approches pour atteindre un bon niveau de tolérance aux pannes. L'une des approches la plus réputée est l'utilisation de la redondance. Il peut s'agir d'un composant matériel répliqué, d'un bit de contrôle supplémentaire associé à une chaîne de données numériques ou de quelques lignes de code de programme vérifiant l'exactitude des résultats du programme. L'idée d'intégrer la redondance afin d'améliorer la fiabilité d'un système a été initiée par John Von Neumann dans les années 1950 dans son travail intitulé « Probabilistic logic and synthesis of reliable organisms from unreliable components » [88].

Deux types de redondance sont possibles [89] : la redondance spatiale (Space redundancy) et la redondance temporelle (Time redundancy). La redondance spaciale est composée de deux types ; la redondance matérielle et la redondance logicielle. Par contre, pour la redondance temporelle, on traite plutôt la répétition des calculs ou la transmission des données et la comparaison des résultats obtenues à une copie stockée des résultats précédents. Dans ce travail, on s'intéresse, principalement, à la redondance matérielle active issue de la redondance spatiale et comportant la redondance active, redondance Standby et redondance Mixte comme l'indique la Figure 2.1.

2.2.2 La redondance logicielle

Les techniques de tolérance aux pannes logicielles peuvent être divisées en deux groupes : les techniques à version unique et les techniques à versions multiples. Les techniques à version unique visent à améliorer la tolérance aux pannes d'un composant logiciel en y ajoutant des mécanismes de détection, de confinement et de reprise. Les techniques à versions multiples utilisent des composants logiciels redondants développés selon les règles de diversité de conception [89].

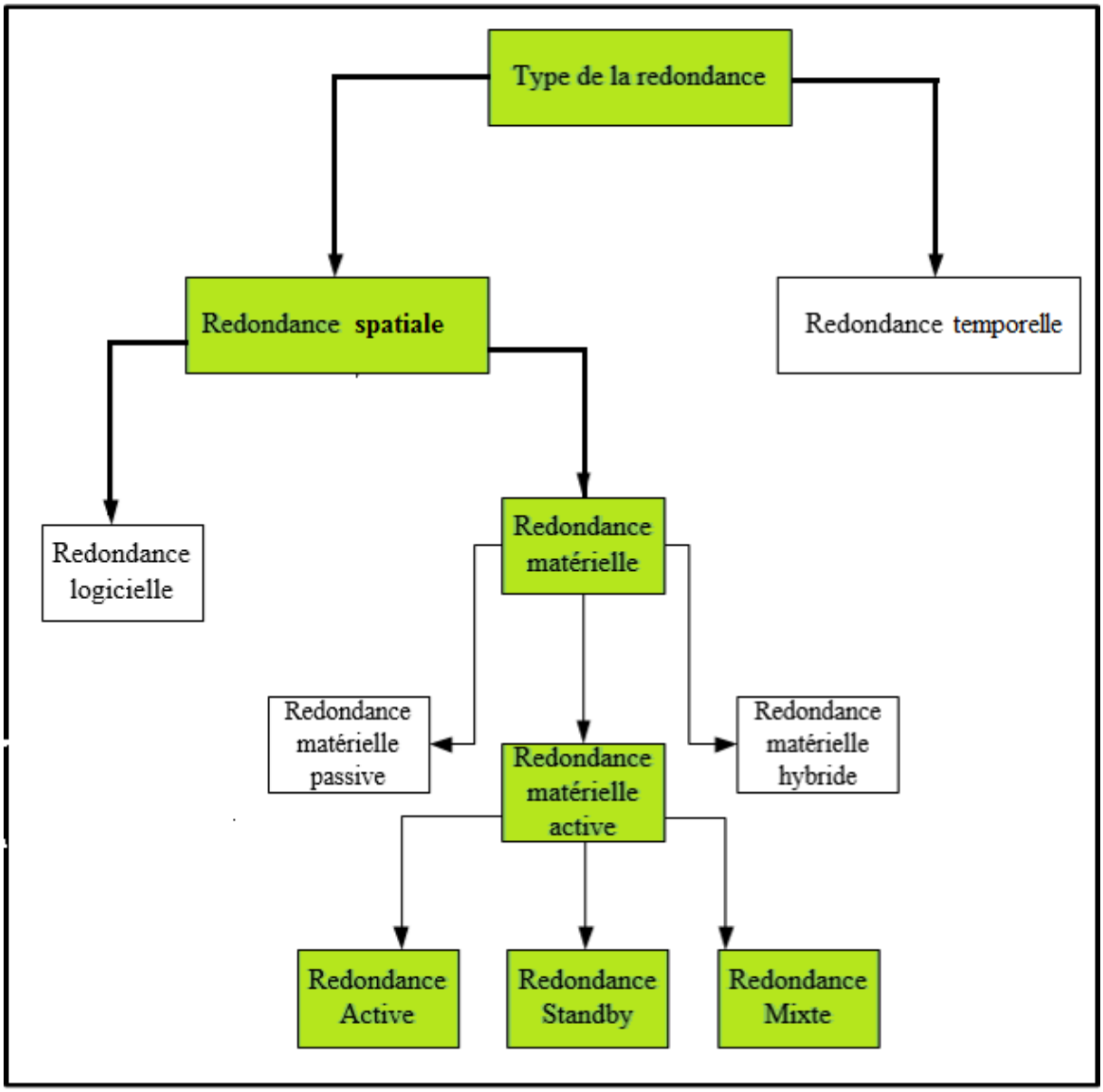


Figure 2. 1: Types de redondances

2.2.3 La redondance matérielle

La redondance matérielle est obtenue en fournissant deux copies physiques ou plus d'un composant matériel. Par exemple, un système peut contenir des processeurs, des mémoires, des bus ou des blocs d'alimentation redondants. La redondance matérielle a un certain nombre de désavantages comme l'augmentation du poids, de la taille, de la consommation électrique, du coût ainsi que du temps nécessaire à la conception, à la fabrication et au test. Pour cela, il existe un certain nombre de choix à faire pour déterminer le meilleur moyen d'intégrer la redondance dans un système et minimiser le poids et le coût du système [90].

À l'origine, les techniques de redondance étaient utilisées pour faire face à la faible fiabilité des composants matériels de base. Les concepteurs des systèmes informatiques anciens ont triplé des composants de bas niveau tels que des portes logiques ou des bascules et ont utilisé le « Majority Voting » pour corriger les erreurs [91]. Au fur et à mesure de l'amélioration de la fiabilité des composants de base, la redondance a été déplacée à des niveaux plus élevés. Les composants plus volumineux, tels que les mémoires ou les unités de traitement, ont été répliqués.

Il existe trois types de redondances matérielles : la redondance passive, la redondance active et la redondance hybride.

2.2.3.1 La redondance matérielle active

La redondance active permet d'atteindre la tolérance aux pannes en détectant d'abord les pannes, puis en effectuant les actions nécessaires pour ramener le système à un état opérationnel. Les techniques de redondance active sont généralement utilisées dans les applications nécessitant une haute disponibilité. Dans ce mode de redondance, les erreurs occasionnelles sont autorisées, à condition que le système regagne son fonctionnement normal dans un délai spécifié. Après la détection de la panne, des actions de localisation, de confinement et de récupération sont effectuées pour supprimer le composant défectueux du système afin d'être remplacé par un autre composant.

2.2.3.1.1 Redondance Active

La redondance Active est la forme basique de la redondance matérielle active. Elle est caractérisée par au moins deux modules identiques qui fonctionnent en parallèles (en même temps et qui accomplissent la même fonction). Ce type de redondance permet de comparer les résultats et d'envoyer un signal d'erreur en cas de problème.

2.2.3.1.2 Redondance Standby

La redondance Standby [92] est une autre technique de la redondance matérielle active. La configuration de la redondance en Standby exige que l'un des n composants soit actif. Les n-1 composants restants sont des composants de réserve. En cas de défaillance du composant principal, un commutateur permet le basculement vers un composant secondaire.

Il existe trois types de redondance en Standby : chaude, tiède et froide. Pour le mode de la redondance en Standby chaude, les composants opérationnels et de rechange sont sous tension. Ainsi, les composants de rechange peuvent être mis en service immédiatement après la défaillance du composant principal. Cela minimise les temps d'arrêt du système dus à la reconfiguration. Pour le mode de la redondance en Standby froide, les composants de rechange sont mis hors tension jusqu'à ce que leur appel soit nécessaire pour remplacer le composant défectueux. Cela augmente le temps de reconfiguration du temps requis pour alimenter et initialiser le composant de rechange mais le type froide est préférable pour les applications où la consommation d'énergie est critique. La redondance en Standby est utilisée dans de nombreuses applications. Par exemple le calculateur de pointage du télescope d'Apollo utilisé dans la première station spatiale américaine Skylab [93]. Dans ce système, deux calculateurs identiques, un principal et un secondaire, sont connectés à un commutateur qui surveille le calculateur principal et qui permet le basculement sur le calculateur secondaire en cas de panne du calculateur principal.

2.2.3.1.3 Redondance Mixte

La redondance Mixte combine les approches de la redondance en Standby et la redondance Active. L'idée est similaire à la redondance en Standby sauf que le système possède deux composants principaux ou plus au lieu d'un seul. Comme dans le cas de la redondance Active, les résultats peuvent être comparés pour détecter une erreur. A partir de ce signal, le calculateur décide lequel des composants est défectueux, le met hors service et le remplace par un composant de rechange.

2.2.3.2 La redondance matérielle hybride

La redondance matérielle hybride combine les avantages des approches passives et actives et permet une reconfiguration sans temps d'arrêt du système. Le masquage des pannes est utilisé pour empêcher le système de produire des résultats momentanés erronés. Puis, en s'inspirant de la redondance active, on procède à la détection, la localisation et la récupération des composants défaillants pour les remplacer par les composants de rechange. Les techniques

de la redondance matérielle hybride sont généralement utilisées dans des applications critiques pour la sûreté, telles que les systèmes de contrôle de processus chimiques, les centrales nucléaires, les armes, les équipements médicaux, les avions, les automobiles, etc.

2.3 Processus d'intégration entre SE et SA

Dans cette section, une vue d'ensemble du projet est présentée. L'objectif de ce travail est l'intégration de la sûreté de fonctionnement dans le processus de l'ingénierie système afin d'assurer la cohérence entre les modèles et d'améliorer la structure et le comportement du système. Le processus développé d'intégration entre l'ingénierie système et la sûreté de fonctionnement est réalisé en 4 étapes. Dans la Figure 2.2, un organigramme décrit ce processus ainsi que les échanges entre les différentes étapes.

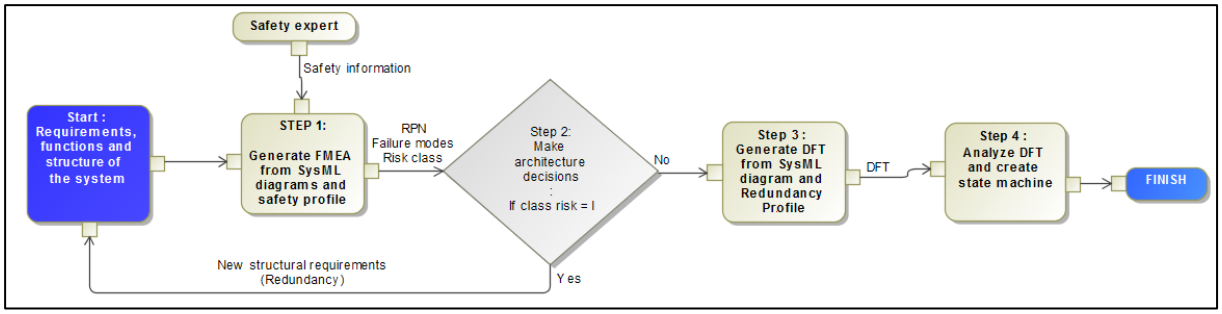


Figure 2. 2: Organigramme de la méthodologie générale

1ère étape :

La première étape de cette méthodologie consiste à générer automatiquement une analyse préliminaire du mode de défaillance et de ses effets (PFMEA) à partir des diagrammes SysML et à l'aide du profil de sûreté. Ensuite, un expert de sûreté de fonctionnement complète la PFMEA avec d'autres informations afin d'obtenir un FMEA contenant des paramètres tels que la Sévérité (S), l'Occurrence (O) et la Détectabilité (D). La section 2.4.1 fournit plus de détails sur la génération des FMEA.

2ème étape :

Dans la deuxième étape, le FMEA généré est analysé et utilisé pour détecter les composants critiques et apporter des modifications sur la structure du système afin d'améliorer sa fiabilité. Dans ce travail, la redondance est utilisée comme une solution pour réduire le risque des composants critiques. La section 2.4.2 comporte plus de détails sur les modifications de l'architecture afin de répondre aux exigences de la sureté de fonctionnement.

3ème étape :

Cette étape consiste à spécifier le comportement de la redondance ajoutée dans l'étape précédente. Par conséquent, un profil de redondance est utilisé ce qui permet la génération des arbres de défaillances dynamiques à partir des diagrammes SysML. La section 3.2 donne plus de détails sur le processus de génération des arbres de défaillances dynamiques,

4ème étape :

Dans la quatrième étape, l'arbre de défaillance généré est analysé et les séquences de coupe minimales sont utilisées pour créer une machine d'états qui décrit le comportement du système. La section 3.2.3 donne plus de détails sur l'étape de création des machines à états à partir des séquences de coupes minimales.

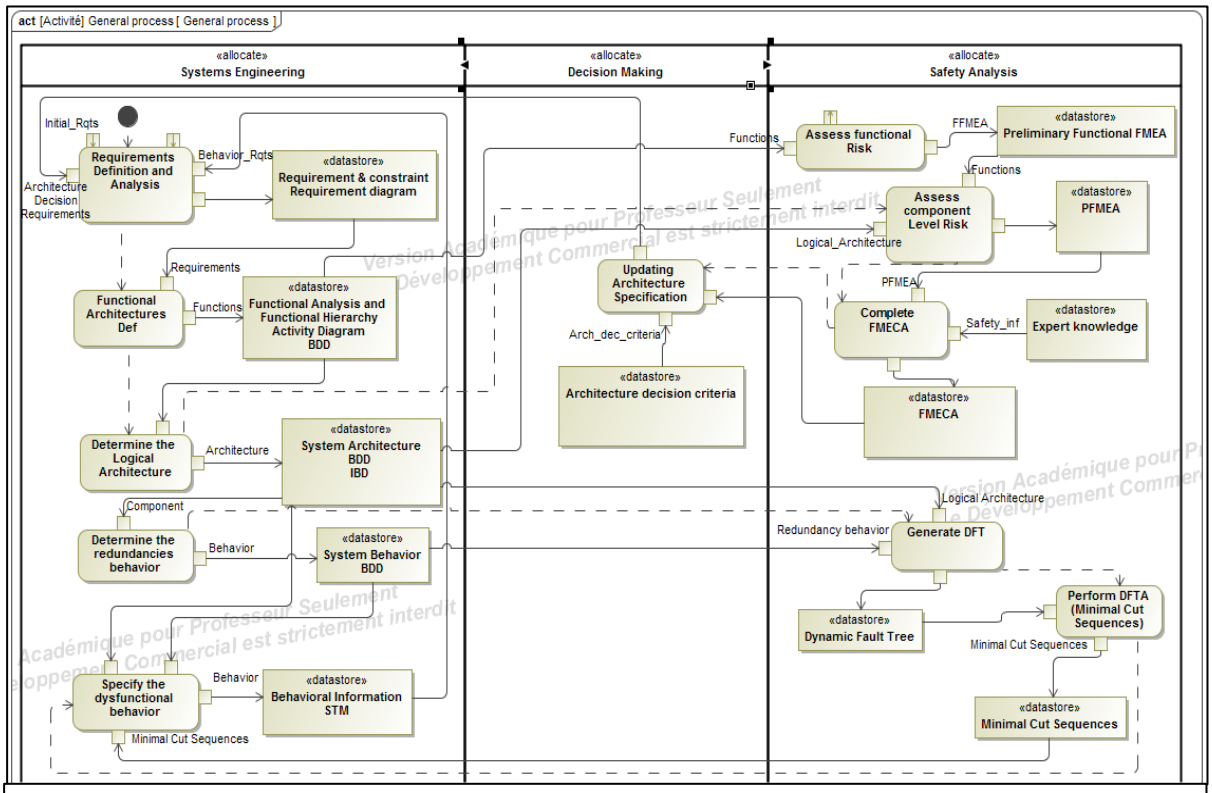


Figure 2. 3: Processus détaillé de la méthodologie générale

Nous pouvons, également, décrire ce processus d'une autre façon, plus intégrée, au travers du diagramme des activités de la Figure 2.3. La partie « prise de décision » représentant l'étape 2 vient s'insérer entre la partie « ingénierie système » comportant la définition de l'architecture et le comportement du système et une partie de l'étape 4 et la partie « analyse de la sûreté de fonctionnement » représentant les étapes 1 et 3. Le point de départ du processus est

un ensemble d'exigences initiales décrivant les besoins du système. Chaque activité a une base de données qui permet le stockage de différentes informations comme les exigences, les fonctions ou les composants du système.

2.4 Génération de FMEA et choix d'architecture

2.4.1 Génération des FMECA

Dans cette section, on décrit les deux premières étapes de la méthodologie proposée dans ce travail et qui tend à intégrer la sûreté de fonctionnement à l'ingénierie système afin d'améliorer la cohérence entre les modèles, assurer la mise à jour dynamique de la structure et garantir un certain niveau de fiabilité des systèmes mécatroniques.

La première étape, comme indiquée dans la Figure 2.2, est la génération de FMEA à partir des diagrammes SysML. Afin d'assurer cette génération, certains processus, représentés dans la Figure 2.3, sont obligatoires. On commence par la description des exigences du système ainsi que ses interfaces externes. Pour cela, plusieurs diagrammes SysML tels que les diagrammes d'exigences, les diagrammes de cas d'utilisation et les diagrammes de définition des blocs peuvent être utilisés pour participer à l'identification des exigences. Afin de valider l'étape de détermination des exigences, il faut passer par les sous-étapes suivantes. En premier lieu, il faut déterminer la mission et les objectifs du système en utilisant les diagrammes d'exigences pour définir les exigences initiales du système. Ensuite, il faut identifier le cycle de vie du système afin de prendre en compte les contraintes de chaque phase en utilisant la machine à état qui permet la définition des différentes étapes du cycle de vie du système ainsi que les conditions de transitions d'une étape à une autre. De plus, il faut définir les limites du système et identifier ce qu'il doit être développé depuis les premiers stades du processus en utilisant les BDD pour représenter l'interaction du système avec les éléments extérieurs. Pour chaque phase du cycle de vie du système, il faut prendre en considération le contexte et les interactions du système avec son environnement. Puis, il faut définir les interfaces externes du système et modéliser le comportement externe du système par les modes de fonctionnement de l'utilisateur ou les services fournis par le système et les scénarios fonctionnels en utilisant le diagramme de cas d'utilisation et les diagrammes de séquences. Toutes ces données seront stockées dans une base de données pour les réutiliser en cas de besoin.

Après, on passe à l'activité de la définition de l'architecture fonctionnelle du système. On remarque que cette activité a deux entrées; la première est la réception des exigences à partir de

la base des données des exigences via un flux d'objet et la seconde est la transition de l'étape de définition des exigences à l'étape de définition de l'architecture fonctionnelle par un flux de contrôle qui indique la fin de l'étape de collecte des exigences et le début de l'étape de spécification des fonctions du système. A partir des exigences fonctionnelles modélisées dans l'analyse de l'étape précédente, une ou plusieurs architectures fonctionnelles du système sont modélisées au cours de cette étape. Le résultat final de cette étape est un modèle hiérarchique de la répartition de la ou les fonction(s) principale(s) du système en sous-fonctions.

Ensuite, on passe à l'activité de détermination de l'architecture logique du système. On remarque que cette activité a deux entrées. La première entrée permet la réception des fonctions du système à partir de la base des données par un flux d'objet. La seconde activité représente la transition de l'étape de définition de l'architecture fonctionnelle à l'étape de détermination de l'architecture logique du système. Dès qu'on définit l'architecture fonctionnelle du système, une ou plusieurs architectures logiques sont déterminées en accordant chaque composant à la ou les fonctions adéquates. Un BDD décrit les composants du système et montre les allocations entre les fonctions et les composants. Dans ce diagramme, une ou plusieurs fonctions sont attribuées à un composant logique. Les composants logiques sont maintenant identifiés, mais la façon dont ils interagissent n'est pas encore spécifiée. Donc, on utilise un diagramme IBD qui représente l'architecture du système avec les différentes interactions entre les composants et les ports d'entrée/sortie. Dans cette étape on a deux flux de sortie; un flux d'objet qui transmet les données de l'architecture logique à la base des données et un flux de contrôle qui indique le passage de l'étape de détermination de l'architecture logique du système à l'étape de génération du FMEA composant préliminaire qui est la même que la PFFMEA avec la liste des composants.

Une FMEA préliminaire fonctionnel, basé sur le format XMI, est généré automatiquement à partir des modèles SysML. Cette FMEA contient la liste des fonctions et une liste générique des modes de défaillance. Puis, on génère un FMEA composant préliminaire qui contient les données de FMEA fonctionnelle préliminaire avec les composants associés à chaque fonction.

La sévérité (S) se caractérise par un entier compris entre 1 et 10 et représente la sévérité du résultat causé par cette défaillance (des morts, des blessures, etc...). L'occurrence (O) est un entier entre 1 et 10. Elle représente l'occurrence de défaillance au cours de la durée de vie du système. La détection (D) est également classée entre 1 et 10. Elle représente la possibilité pour

qu'une défaillance ne soit pas identifiée en raison des problèmes de détection. On note RPN le facteur de criticité qui est calculé par la multiplication des trois paramètres (RPN = S * O * D).

2.4.2 Choix d'architecture

Une fois que la consolidation des FMEA est faite, une analyse des valeurs obtenues doit être faite. L'analyse se base essentiellement sur les valeurs obtenues dans RPN en multipliant les trois facteurs S, O et D. Tout d'abord, une valeur seuil (α) est définie selon le domaine, le besoin et les exigences du système. Cette valeur α présente la valeur maximale de RPN acceptée. Au-delà de la valeur α, le composant et son mode de défaillance seront considérés comme critiques et la structure du système doit être mise à jour pour réduire le risque de défaillance de ce composant et par conséquent l'amélioration de la fiabilité du système.

Une fois la criticité d'un ou de plusieurs composants est identifiée, la structure du système doit évoluer d'une façon à éliminer cette criticité. La solution utilisée dans ce travail est la redondance ce qui veut dire que le composant doit être doublé par un deuxième composant ou sous-système qui assure la même fonction que le premier composant. De nouvelles exigences seront intégrées dans le modèle système ce qui provoque la modification de la structure du système (diagramme IBD). Dès qu'on termine la modification de la structure, une nouvelle FMECA est générée et une nouvelle analyse doit être faite. Dans ce stade, on est devant deux possibilités :

- Si la nouvelle FMEA présente encore des anomalies, alors la méthode décrite précédemment doit être refaite jusqu'à avoir un FMEA avec des RPN inférieurs à la valeur α.
- Si la nouvelle FMEA n'a que des RPN inférieurs à α alors le système est validé et considéré comme un système sûr.

Ainsi, on peut passer à la troisième étape qui est la génération des arbres de défaillances dynamiques du nouveau modèle qui est décrit dans le chapitre 3. L'arbre de défaillances dynamiques est généré et les ensembles de coupes minimales de ce système sont déterminés. Le problème détecté par cette analyse est qu'on trouve des ensembles de coupes minimaux de rang 1 (qui contient la défaillance d'un seul composant) avec une sévérité de défaillance « catastrophique » ce qui n'est pas acceptable. Pour cela, cette méthode a évolué et l'analyse de FMEA intègre d'autres facteurs. La génération des FMEA doit inclure une nouvelle colonne qui donne les classes des risques de chaque mode de défaillance comme le montre le Tableau 1.

Tableau 1: Exemple d'un FMEA avec la colonne classes des risques

Composant	Fonction	Mode de défaillance	S	O	D	RPN	Classes des risques

Le paramètre classes des risques dépend seulement de la sévérité et de l'occurrence et il est partagé en 4 classes comme indiqué dans le Tableau 2.

Tableau 2: Les niveaux de risques selon la norme internationale IEC 61508

Occurrence// Sévérité	Catastrophique	Critique	Marginal	Insignifiant
Fréquent	I	I	I	II
Probable	I	I	II	II
Occasionnel	I	II	III	III
Rare	II	III	III	IV
Improbable	III	III	IV	IV
Invraisemblable	IV	IV	IV	IV

On remarque que la probabilité de l'occurrence est classée sous 6 différents niveaux qui décrivent le degré de la défaillance du système : fréquente, probable, occasionnelle, rare, improbable ou invraisemblable.

- Si la défaillance d'un système est fréquente, cela veut dire qu'elle est présente plusieurs fois dans la vie du système avec un taux d'occurrence supérieur à 10^{-3} ;
- Si la défaillance d'un système est probable, cela veut dire qu'elle est présente plusieurs fois dans la vie du système avec un taux d'occurrence entre 10^{-4} et 10^{-3} ;
- Si la défaillance d'un système est occasionnelle, cela veut dire qu'elle est présente seulement une fois dans la vie du système avec un taux d'occurrence entre 10^{-5} et 10^{-4}.
- Si la défaillance d'un système est rare, cela veut dire qu'elle est peu probable dans la vie du système avec un taux d'occurrence entre 10^{-6} et 10^{-5} ;
- Si la défaillance d'un système est improbable, cela veut dire qu'elle est très peu probable dans la vie du système avec un taux d'occurrence entre 10^{-7} et 10^{-6} ;
- Si la défaillance d'un système est invraisemblable, cela veut dire qu'elle ne devrait jamais arriver dans la vie du système avec un taux d'occurrence inférieur à 10^{-7}.

On remarque, aussi, que les conséquences dues suite à une défaillance sont classées en 4 niveaux de sévérité. Les conséquences peuvent être : catastrophiques, critiques, marginales ou insignifiantes.

- Si les conséquences d'une défaillance sont considérées comme catastrophiques, cela veut dire que cette défaillance cause des pertes humaines multiples ;
- Si les conséquences d'une défaillance sont considérées comme critiques, cela veut dire que cette défaillance cause la perte d'une seule vie humaine ;
- Si les conséquences d'une défaillance sont considérées comme marginales, cela veut dire que cette défaillance cause des blessures majeures à une ou plusieurs personnes ;
- Si les conséquences d'une défaillance sont considérées comme insignifiantes, cela veut dire que cette défaillance cause des blessures mineures.

La combinaison des deux facteurs (Occurrence, Sévérité) donne quatre niveaux de risques selon la norme IEC61508 qui sont les risques de classe I, classe II, classe III et classe IV.

- Si la combinaison de l'occurrence et de la sévérité donne un risque de classe I, cela veut dire que le risque est inacceptable en toute circonstance ;
- Si la combinaison de l'occurrence et de la sévérité donne un risque de classe II, cela veut dire que le risque est non désiré mais tolérable seulement si la réduction du risque n'est pas faisable ou si les coûts sont largement disproportionnés par rapport au gain de réduction du risque ;
- Si la combinaison de l'occurrence et de la sévérité donne un risque de classe III, cela veut dire que le risque est tolérable si le coût de réduction de risque dépasse le gain ;
- Si la combinaison de l'occurrence et de la sévérité donne un risque de classe IV, cela veut dire que le risque est acceptable mais il nécessite une surveillance.

Jusque-là, la FMEA générée inclut les RPN et les classes des risques de chaque mode de défaillance. L'analyse de FMEA se base, maintenant, sur les deux paramètres RPN et classes des risques et il faut que le RPN soit inférieur à la valeur seuil α et la classe des risques des modes de défaillance soit différents de la classe I. L'analyse commence par la vérification des

RPN et des classes de risques de chaque mode de défaillance. Les quatre cas suivants sont possibles pour chaque mode de défaillance:

- Si le mode de défaillance a un RPN supérieur à α et sa classe de risques est de classe I, alors ce mode de défaillance et son composant sont considérés comme éléments critiques et la solution de la redondance doit être appliquée ;
- Si le mode de défaillance a un RPN inférieur à α et sa classe de risques est de classe I, alors ce mode de défaillance et son composant sont considérés comme éléments critiques et la solution de la redondance doit être appliquée ;
- Si le mode de défaillance a un RPN supérieur à α et sa classe de risques est de classe différente de la classe I, alors ce mode de défaillance et son composant sont considérés comme éléments critiques et la solution de la redondance doit être appliquée ;
- Si le mode de défaillance a un RPN inférieur à α et sa classe de risques est de classe différente de la classe I, alors aucune modification n'est demandée sur la structure du système.

Une fois que la structure du système est modifiée par l'ajout de la redondance, un nouveau FMEA est générée comme décrite ci-dessus. Ce nouveau FMEA contient les anciens composants ainsi que les nouveaux composants avec leurs fonctions, modes de défaillances, RPN et classes de risques. Si la nouvelle analyse indique que le système est validé (RPN inférieur à la valeur seuil et classe de risque différent de niveau I), alors on peut passer à l'étape suivante qui est la génération des arbres de défaillances dynamiques. Une application de cette méthode ainsi qu'une explication du problème produit suite à l'utilisation de RPN seule sont données dans le cas d'étude de ce chapitre et dans le cas d'étude général.

2.5 Cas d'étude : Actionneur électromécanique (EMA)

Dans cette partie, on étudie l'architecture interne d'un Actionneur Electromécanique (en anglais : Electro-Mechanical Actuator (EMA)) pour illustrer la méthode de génération des FMEA et du choix d'architecture. Ce système permet le contrôle de l'aileron d'avion. Cette fonction est décomposée en deux sous-fonctions qui sont « Control and command » et « Actuate Aileron ». Cette fonction a deux entrées ; le « PilotInstructions » qui englobe les instructions demandées et « ElecPwr » qui représente l'alimentation du système. Elle contient deux sorties ; «F_Back_to_CtrlU » qui représente le retour d'information pour l'unité de contrôle et « MechPwr » qui représente l'énergie mécanique sortante. Cette fonction est représentée dans le diagramme d'activité de la Figure 2.4.

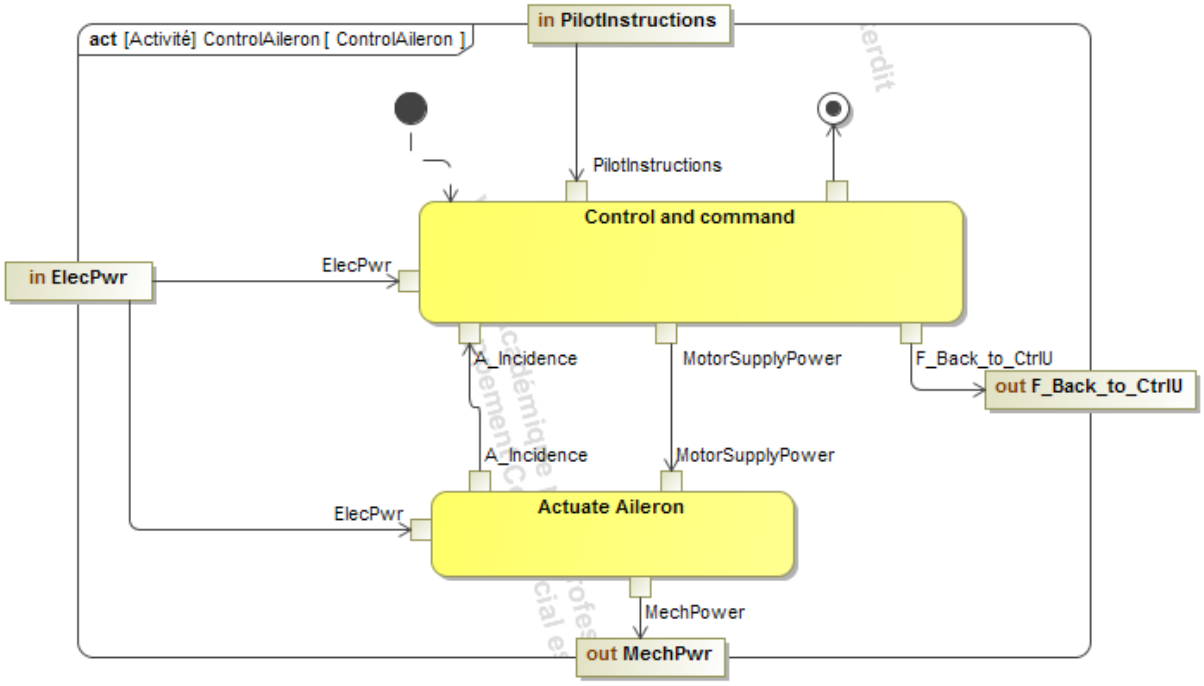


Figure 2. 4: Diagramme d'activité de la fonction "Control Aileron"

Ensuite, la fonction « Control and command » est décomposée en deux sous-fonctions qui sont « Translate Pilot Instructions » et « Regulate Electrical Energy ». Le diagramme d'activité qui décrit cette fonction est présenté dans la Figure 2.5.

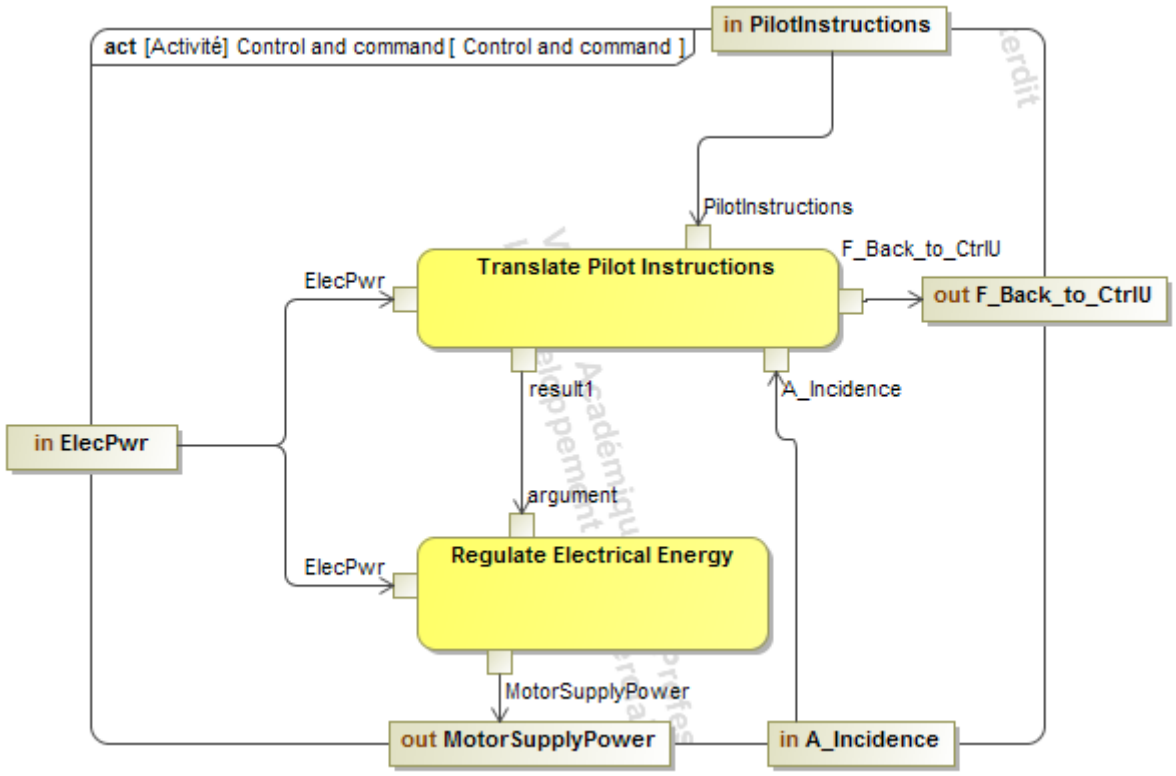


Figure 2. 5: Diagramme d'activité de la fonction "Control and command"

De même, la fonction « Actuate Aileron » est décomposée en quatre sous-fonctions qui sont « Transform Electrical/Mechanical Energy », « Adapt Mechanical Energy », « Mesure Incidence » et « Transmit Mechanical Energy ». Le diagramme d'activité qui décrit cette fonction est présenté dans la Figure 2.6.

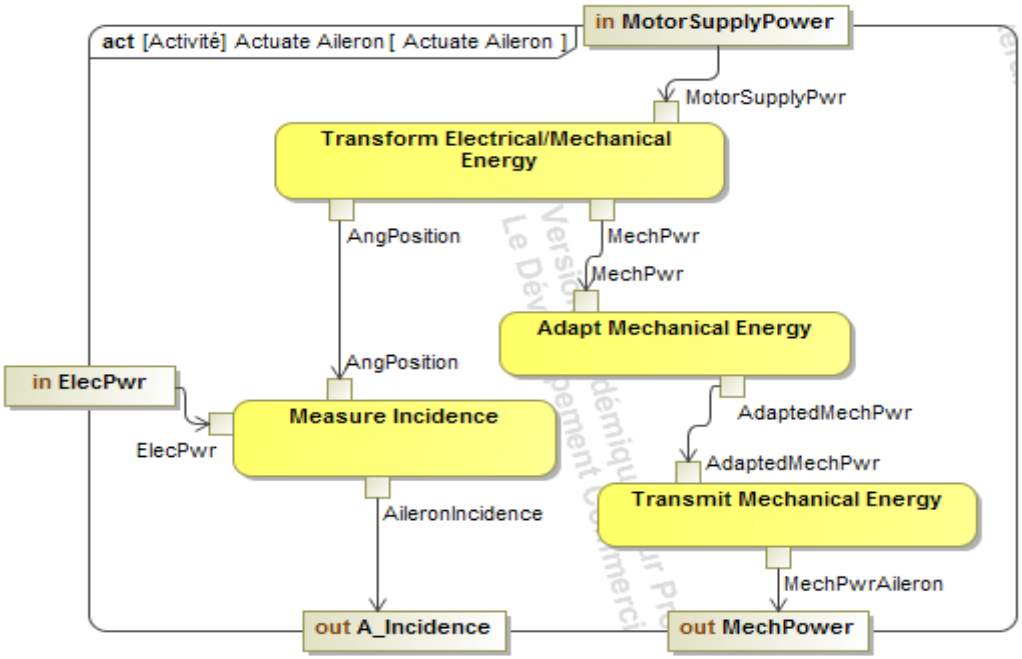


Figure 2. 6: Diagramme d'activité de la fonction "Actuate Aileron"

La fonction « Transform Electrical/Mechanical Energy », elle-même, est décomposée en trois sous-fonctions qui sont « Conduct Electric Current », « Generate Magnetic Field » et « Deliver Mechanical Energy ». Cette fonction est décrite dans le diagramme d'activité de la Figure 2.7.

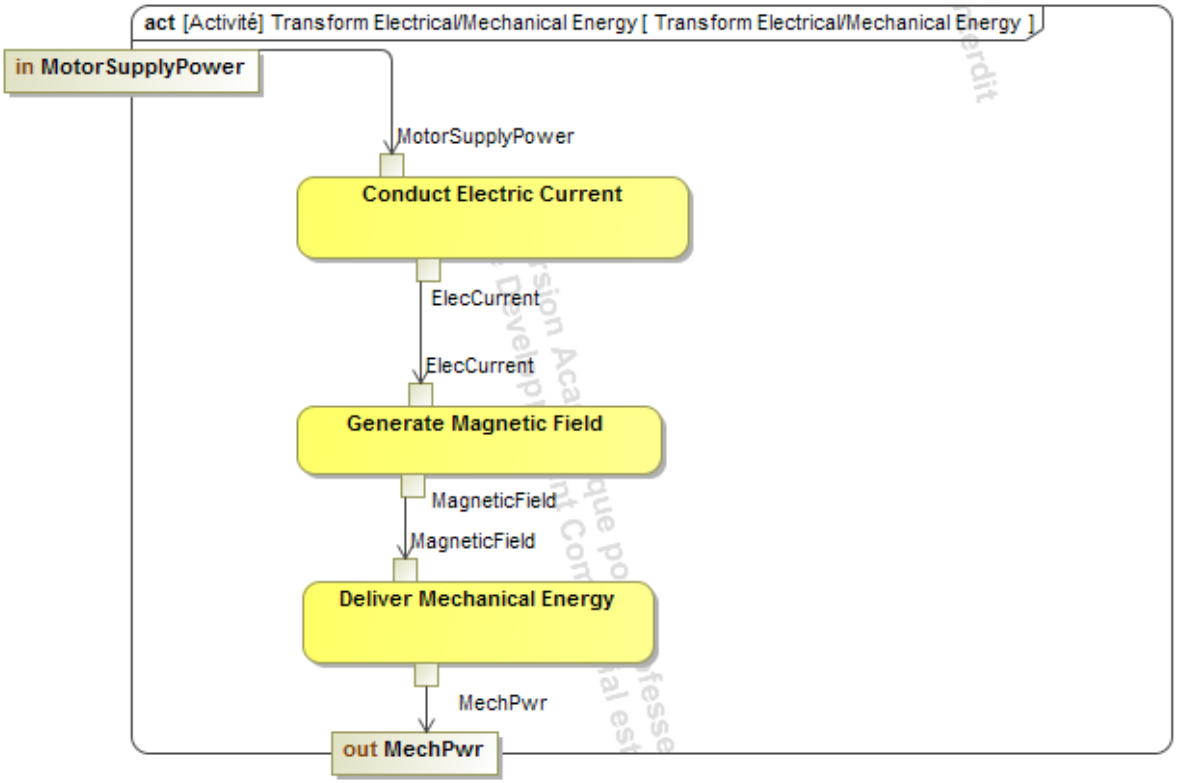


Figure 2. 7: Diagramme d'activité de la fonction "Transform Electrical/Mechanical Energy"

Puis, on définit la structure du système en utilisant le diagramme IBD. Ce diagramme est représenté dans la Figure 2.8. L'EMA est composé par un microcontrôleur embarqué « EMC », un moteur à courant continu « Motor » et une transmission mécanique « Mech-Transmission ». Le moteur à courant continu, lui-même, est composé d'un bobinage « Winding » et d'un rotor « Rotor ». Cette structure a deux entrées externes qui sont l'alimentation « Elec-Energy » et les consignes du pilote « Input ». La sortie est une énergie mécanique qui permet la manipulation de l'aileron d'avion.

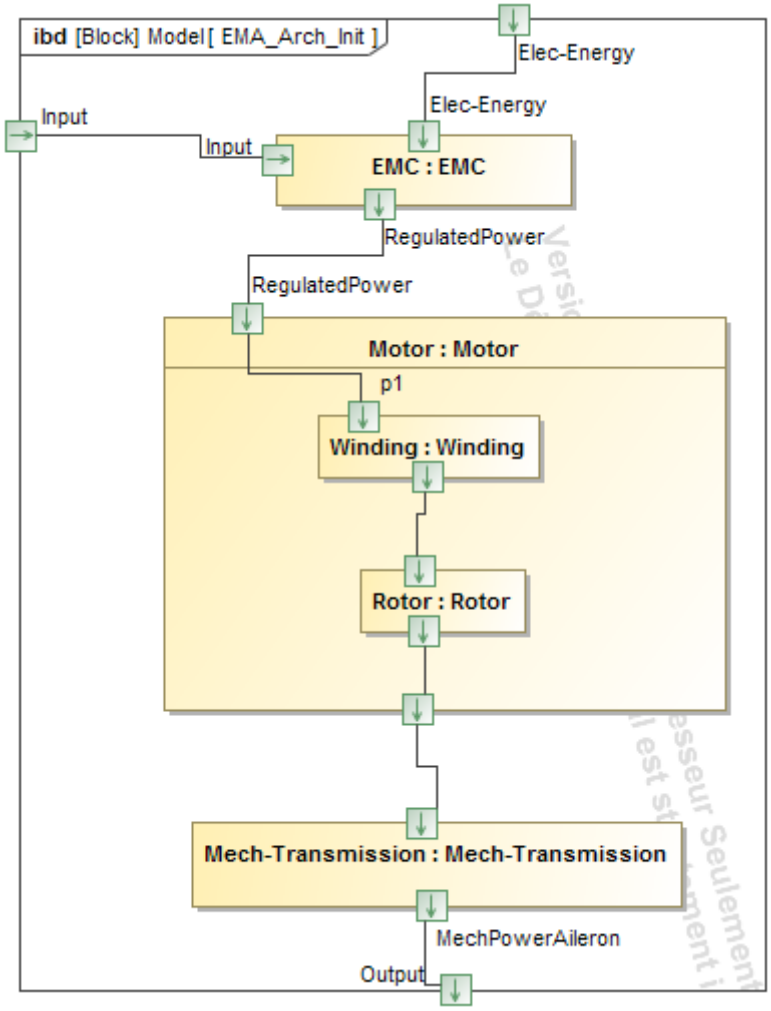


Figure 2. 8: Structure initiale de l'EMA

Après, on alloue la ou les fonctions appropriées à chaque composant. Cette allocation est décrite par le diagramme BDD de la Figure 2.9. La fonction « Control and command » et ses deux sous-fonctions « Regulate Electrical Energy » et « Translate Pilot Instructions » sont allouées au composant « EMC ». De plus, les sous-fonctions « Transmit Mechanical Energy » et « Adapt Mechanical Energy » de la fonction « Actuate Aileron » sont allouées à la transmission mécanique « Mech-Transmission ». Par contre, la troisième sous-fonction « Transform Electrical/Mechanical Energy » est allouée au moteur « Motor ». Les trois sous-fonctions de la dernière fonction sont distribuées comme suit : la fonction « Conduct Electric Current » est allouée à « Winding » et les fonctions « Generate Magnetic Field » et « Deliver Mechanical Energy » sont allouées à « Rotor ».

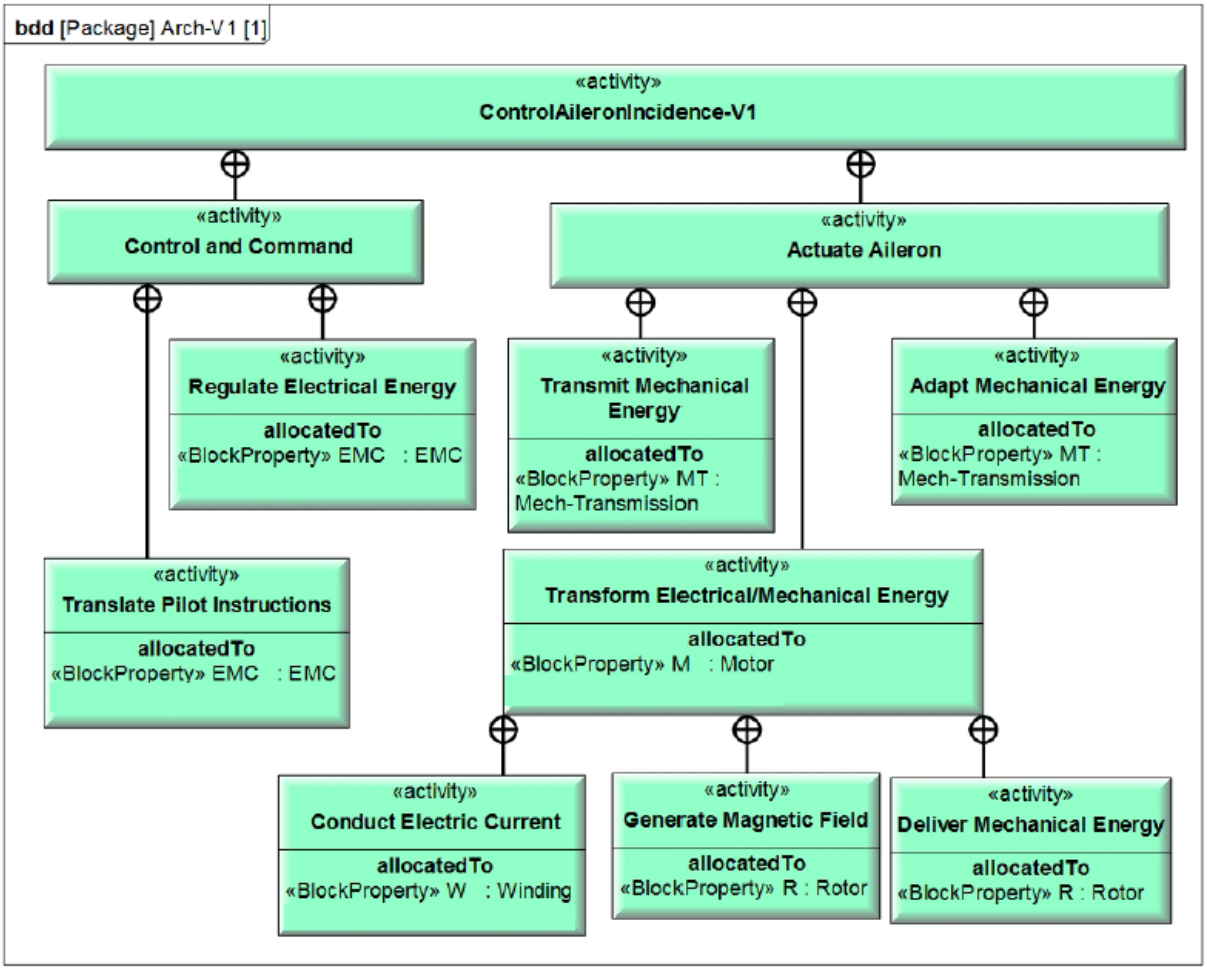


Figure 2. 9: Allocation des composants aux fonctions de l'EMA

Après, à partir des diagrammes qui décrivent les fonctions et la structure de l'EMA, on génère un FMEA préliminaire qui englobe les composants, les fonctions et leurs modes de défaillances génériques comme indiqué dans le Tableau 3.

Tableau 3: FMEA préliminaire de l'EMA

Component	Fonction	Mode de défaillance fonctionnelle	Mode de défaillance	S	O	D	RPN	Classe risque
EMC	Regulate Electrical Energy	Fails to perform	Hardware defect					
		Performs incorrectly	Hardware defect					
	Translate Pilot Instructions	Fails to perform	Software error					
			Hardware defect					

			Synchronization error					
			Memory defect					
		Performs incorrectly	Software fault					
		Performs incorrectly	Synchronization error					
Motor	Measure Incidence	Fails to perform	Mechanical drive defect					
			Power loss					
			Internal Component failure					
		Performs incorrectly	Mechanical drive defect					
			Low or high voltage					
			Measure fault					
	Transform Mechanical Energy	Fails to perform	Jamming					
			Loss of structural integrity					
			Short-circuit between two winding					
		Performs incorrectly	Short-circuit in one winding					
Winding	Conduct Electric Current	Fails to perform	Power loss					
			Internal Component Failure					
Rotor	Generate Magnetic Field	Fails to perform	Hardware defect					
	Deliver Mechanical Energy	Fails to perform	Hardware defect					
		Performs incorrectly	Hardware defect					
Mechanical Transmission	Transmit Mechanical Energy	Fails to perform	Loss of structural integrity					
			Jamming					
		Performs incorrectly	Poor efficiency					

	Adapt Mechanical Energy	Fails to perform	Loss of structural integrity					
		Fails to perform	Jamming					
		Performs incorrectly	Poor efficiency					

Puis, en utilisant l'historique d'utilisation de l'EMA, un expert en sûreté remplit le FMEA par les différentes parties ; S, O, D, RPN et classe des risques comme indiqué dans le Tableau 4.

Tableau 4: FMEA initiale de l'EMA

Component	Fonction	Mode de défaillance fonctionnelle	Mode de défaillance	S	O	D	RPN	Classe risque
EMC	Regulate Electrical Energy	Fails to perform	Hardware defect	8	7	4	224	II
		Performs incorrectly	Hardware defect	6	7	4	168	II
	Translate Pilot Instructions	Fails to perform	Software error	8	4	6	192	III
			Hardware defect	9	7	7	441	I
			Synchronization error	7	5	6	210	II
			Memory defect	7	4	5	140	III
		Performs incorrectly	Software fault	6	7	4	168	II
		Performs incorrectly	Synchronization error	5	5	6	150	III
Motor	Measure Incidence	Fails to perform	Mechanical drive defect	7	4	5	140	III
			Power loss	7	8	3	168	I
			Internal Component failure	6	5	8	240	III
		Performs incorrectly	Mechanical drive defect	5	7	6	210	II
			Low or high voltage	5	3	7	105	IV
			Measure fault	4	7	5	140	III
	Transform Mechanical Energy	Fails to perform	Jamming	6	7	8	336	I

			Loss of structural integrity	10	3	5	150	II
			Short-circuit between two winding	8	5	4	160	III
		Performs incorrectly	Short-circuit in one winding	7	4	6	168	III
Winding	Conduct Electric Current	Fails to perform	Power loss	8	8	2	128	I
			Internal component failure	8	4	3	96	III
Rotor	Generate Magnetic Field	Fails to perform	Hardware defect	5	4	6	120	III
	Deliver Mechanical Energy	Fails to perform	Hardware defect	4	5	5	100	III
		Performs incorrectly	Hardware defect	5	5	6	150	III
Mechanical Transmission	Transmit Mechanical Energy	Fails to perform	Loss of structural integrity	9	3	5	135	III
			Jamming	6	5	3	90	III
		Performs incorrectly	Poor efficiency	5	6	5	150	III
	Adapt Mechanical Energy	Fails to perform	Loss of structural integrity	6	6	4	144	III
		Fails to perform	Jamming	9	4	8	288	II
		Performs incorrectly	Poor efficiency	2	8	3	48	II

Ensuite, on passe à l'étape du choix d'architecture par l'analyse de FMEA générée et par la suite les modes de défaillance les plus critiques sont détectés. L'approche se repose sur l'utilisation de la valeur de RPN et des classes de risques de chaque mode de défaillance. Pour cela, il faut définir une valeur seuil qui exprime les valeurs interdites de RPN.

Pour cet exemple, la valeur seuil est $\boldsymbol{\alpha = 400}$.

Afin de considérer un mode de défaillance comme critique, il faut qu'il satisfasse, au moins, l'un des deux critères qui sont un RPN >= 400 ou classe de risque = I.

L'analyse de ce FMEA montre l'existence de quatre modes de défaillance des différentes fonctions :

1/ fails to perform the translation of pilot instructions because of a hardware defect of the EMC component;

2/ fails to perform the measure incidence because of a power loss of the motor;

3/ fails to perform the transformation of the mechanical energy because of a jamming of the motor;

4/ fails to perform the conduct Electric Current of a power loss of the winding.

Afin d'améliorer l'architecture du système, les trois composants critiques « ECM », « Motor» et « Winding » doivent être doublés et le diagramme IBD sera mis à jour. Le nouveau IBD comporte deux ECM redondants, deux moteurs redondants et deux enroulements redondants pour chaque moteur comme illustré dans la Figure 2.10.

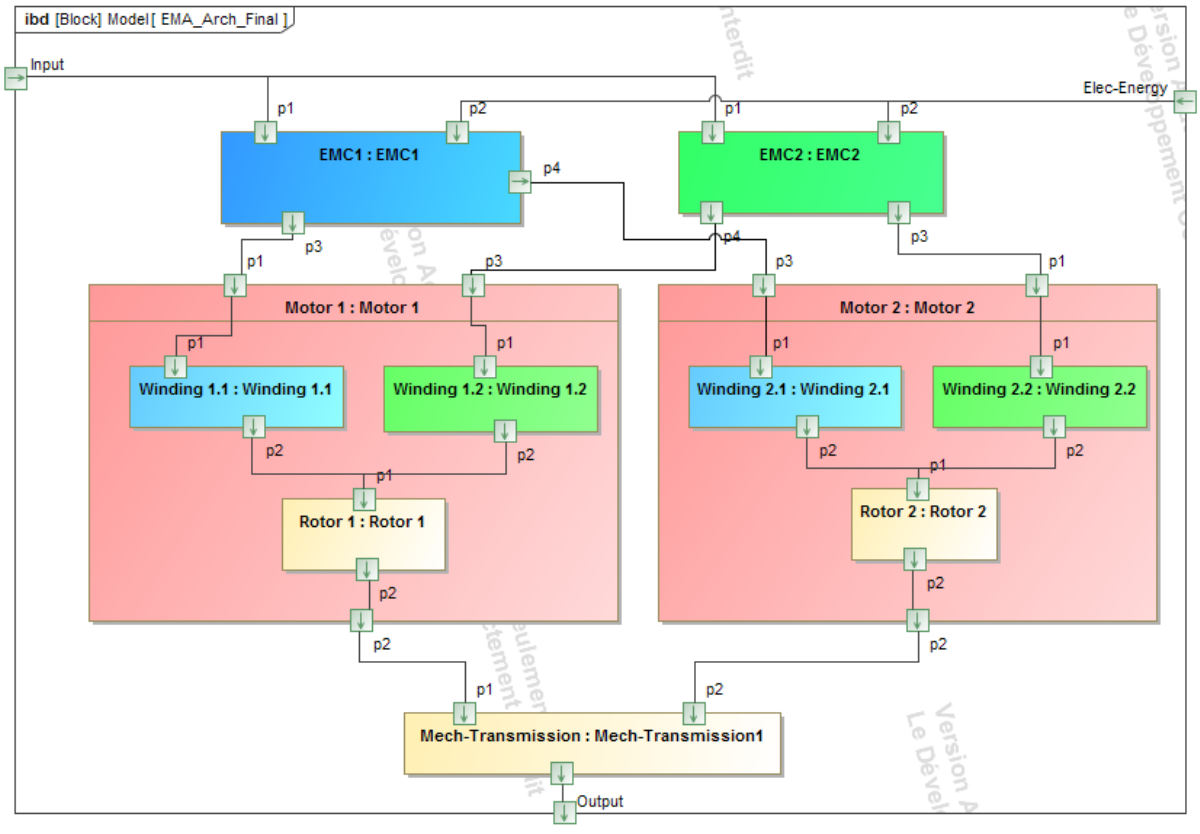


Figure 2. 10: La structure de l'EMA après les modifications

Ensuite, pour chaque nouveau composant dans le système, une allocation des fonctions est obligatoire. La mise à jour du diagramme d'allocation des fonctions aux composants est dans la Figure 2.11.

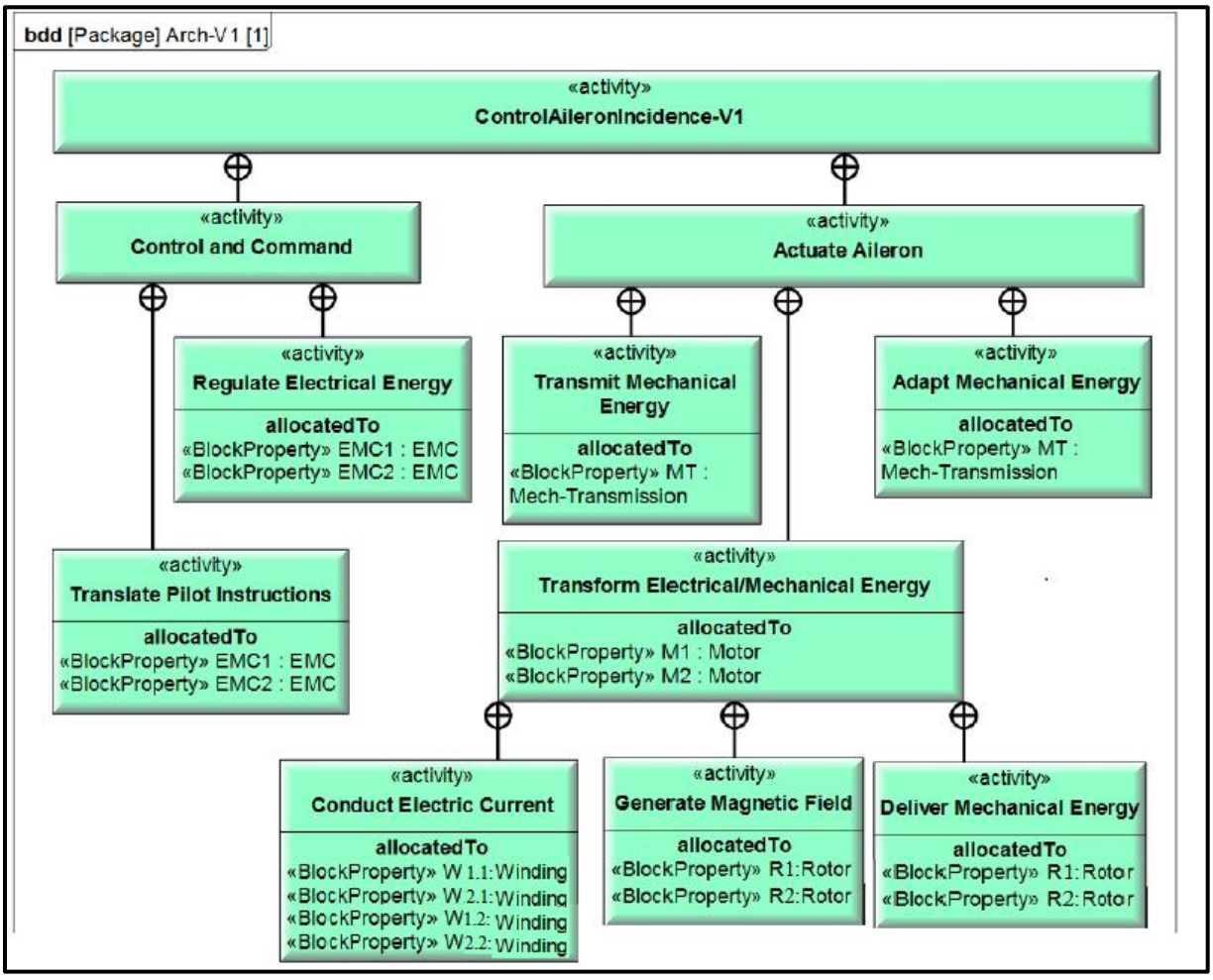


Figure 2. 11: Allocation des composants aux fonctions mis à jour

Après, une nouvelle FMEA doit être générée pour vérifier le niveau de sûreté du système. Elle contient les composants, les fonctions et les modes de défaillance avec les nouvelles valeurs de S, O, D, RPN et de classe de risque. Si les classes risques de tous les modes de défaillance sont différentes de la classe I et leurs RPN < 400, la structure du système est validée. Sinon, une nouvelle itération doit être effectuée et des nouvelles modifications structurelles sont recommandées. Dans notre cas et comme le montre le Tableau 5, tous les modes de défaillance de fonction ont une classe de risque différente de la classe I et un RPN < 400.

Tableau 5: Nouveau FMEA de l'EMA

Component	Fonction	Mode de défaillance fonctionnelle	Mode de défaillance	S	O	D	RPN	Classe risque
EMC	Regulate Electrical Energy	Fails to perform	Hardware defect	8	4	4	128	III
		Performs incorrectly	Hardware defect	6	4	4	96	III
	Translate Pilot Instructions	Fails to perform	Software error	8	1	6	48	IV
			Hardware defect	9	2	7	126	III
			Synchronization error	7	2	6	84	III
			Memory defect	7	1	5	35	IV
		Performs incorrectly	Software fault	6	2	4	48	III
		Performs incorrectly	Synchronization error	5	3	6	90	III
Motor	Measure Incidence	Fails to perform	Mechanical drive defect	7	2	5	70	III
			Power loss	7	4	3	84	III
			Internal Component failure	6	3	8	144	III
		Performs incorrectly	Mechanical drive defect	5	4	6	120	III
			Low or high voltage	5	2	7	70	IV
			Measure fault	4	4	5	80	III
	Transform Mechanical Energy	Fails to perform	Jamming	6	4	8	192	III
			Loss of structural integrity	10	2	5	100	III
			Short-circuit between two winding	8	3	4	96	III
		Performs incorrectly	Short-circuit in one winding	7	2	6	84	III
Winding	Conduct Electric Current	Fails to perform	Power loss	8	4	2	64	II
			Internal component failure	8	2	3	48	III

Rotor	Generate Magnetic Field	Fails to perform	Hardware defect	5	4	6	120	III
	Deliver Mechanical Energy	Fails to perform	Hardware defect	4	5	5	100	III
		Performs incorrectly	Hardware defect	5	5	6	150	III
Mechanical Transmission	Transmit Mechanical Energy	Fails to perform	Loss of structural integrity	9	3	5	135	III
			Jamming	6	5	3	90	III
		Performs incorrectly	Poor efficiency	5	6	5	150	III
	Adapt Mechanical Energy	Fails to perform	Loss of structural integrity	6	6	4	144	III
		Fails to perform	Jamming	9	4	8	288	II
		Performs incorrectly	Poor efficiency	2	8	3	48	II

2.6 Conclusion

Dans ce chapitre nous avons présenté notre méthodologie pour intégrer la sûreté de fonctionnement dans le processus de conception. Cette méthodologie contient quatre étapes, la génération et l'analyse des FMEA, la proposition des modifications de la structure du système afin d'améliorer sa fiabilité.

L'analyse du FMEA se repose sur deux critères qui sont obligatoires pour déterminer les modes de défaillance critiques. Le premier est le RPN qui ne doit pas dépasser une valeur seuil à définir par le concepteur. Le second critère est d'avoir une classe de risque différente de I. En cas de détection d'un composant critique, ce dernier doit être doublé par un second composant ayant la même fonction. A la suite de cette redondance une nouvelle structure est déterminée et une nouvelle analyse est faite pour vérifier de nouveau la sûreté de fonctionnement du système.

Après avoir créé une structure qui contient de la redondance, il reste à définir et analyser son comportement afin d'améliorer la fiabilité et la disponibilité du système. Pour cela, on génère des arbres de défaillances dynamiques à partir d'un modèle système. Ce travail est décrit dans le chapitre prochain.

Chapitre 3 : La génération automatique des arbres de défaillances dynamiques

3.1 Introduction

Dans ce chapitre, nous proposons une méthode pour la génération automatisée des arbres de défaillances dynamiques à partir des diagrammes SysML et du profil de la redondance. Cette génération consiste en une exploration automatisée du modèle du système pour identifier des modèles spécifiques, puis pour chaque modèle, on génère le sous-arbre de défaillance correspondant en utilisant un algorithme de génération. L'arbre de défaillance général est obtenu en assemblant tous les sous-arbres de défaillances.

L'arbre de défaillances est généré sous deux formats différents: un format image pour une meilleure compréhension de la propagation des fautes et un format Open-PSA (Probabilistic Safety Assessment) pour permettre l'analyse de l'arbre de défaillance par les différents outils d'analyse. La méthode de génération est décrite plus en détail dans ce chapitre après un rappel des concepts de base des FTA, de ses outils d'analyses et d'une discussion sur les travaux qui concernent la génération automatisée des arbres de défaillances.

Ce chapitre est organisé comme suit. D'abord, une étude sur les outils d'analyses des arbres de défaillances gratuits, open source et commerciaux, afin de les comparer est présentée dans la section 3.2. Ensuite, l'algorithme de génération des arbres de défaillances dynamiques à partir du diagramme IBD de SysML et des informations sur la redondance est détaillé dans la sous-section 3.3. Pour cela, un profil de la redondance qui permet la description de son comportement y est présenté. Enfin, une étude d'un système de contrôle d'un aileron d'avion est faite afin d'appliquer l'algorithme proposé et de générer son arbre de défaillances dynamiques en utilisant la structure du système et des informations sur les redondances du système.

3.2 Génération automatique des arbres de défaillances dynamiques

3.2.1 Profil SysML pour la gestion de la redondance

Dans cette section, nous présentons un profil SysML qui inclut les propriétés de la redondance dans le modèle système afin d'enrichir l'architecture du système et de faciliter la génération automatique des arbres de défaillances dynamiques. La deuxième sous-section est consacrée à l'explication de cette extension UML pour permettre le traitement de la redondance.

3.2.1.1 Terminologie de la redondance

En ingénierie, la redondance est définie par la duplication des composants ou des fonctions critiques d'un système afin d'améliorer sa fiabilité et / ou sa disponibilité [47]. Selon [90] et [107], il existe trois types de la redondance matérielle qui sont la redondance active, la redondance passive et la redondance hybride. En outre, il existe trois types de stratégies de la redondance Hardware active: i) Duplication and Comparison (Active), ii) Standby et iii) la redondance Pair-And-A-Spare (Mixte). On parle de la redondance Active si tous les composants redondants fonctionnent ensemble à partir du temps zéro et sont alimentés en permanence et partagent la même charge. Pour la redondance en Standby, les composants redondants ne commencent à fonctionner que lorsque l'unité principale tombe en panne. En fait, lors de la défaillance du composant en fonctionnement, la redondance de secours bascule sur l'un des composants redondants pour poursuivre le fonctionnement du système. Il existe trois types de redondance en Standby qui sont : la redondance froide, la redondance chaude et la redondance tiède. La redondance en Standby froide est utilisée lorsque le temps de réponse à une défaillance n'a pas une priorité élevée. Par contre, la redondance en Standby tiède est utilisée pour les processus non critiques qui nécessitent encore un temps de réponse plus rapide en cas de défaillance que dans la redondance froide. La redondance en Standby chaude est utilisée lorsque le temps de réponse à une défaillance doit être instantané.

La redondance Mixte [108] est la combinaison de la redondance Active et de la redondance Standby qui vise à améliorer la fiabilité des systèmes. Le Tableau 6 récapitule les quatre scénarios de redondance possibles et spécifie le nombre initial de composants Actifs et en Standby pour chaque cas.

Tableau 6: Stratégies de la redondance

Stratégie de la redondance	Nombre de composants actifs	Nombre de composants en Standby
Pas de redondance	= 1	= 0
Redondance Active	>1	= 0
Redondance Standby	= 1	>= 1
Redondance Mixte	>1	>= 1

3.2.1.2 *Profil de la redondance*

Afin de faciliter la génération des arbres de défaillances dynamiques avec la prise en compte de la redondance, on a modélisé un profil SysML qui permet l'enrichissement du modèle système avec des informations sur la redondance.

Pour cela, un méta-modèle du profil de redondance est donné dans le diagramme de classes de la Figure 3.1. Ce diagramme représente une version étendue du diagramme de classes dans [100] et décrit la décomposition d'un arbre de défaillance en portes logiques et événements. Ce diagramme de classes spécifie qu'un arbre de défaillance est caractérisé par un type arbre de défaillance qui peut être statique ou dynamique. Rappelons que les arbres de défaillances statiques sont utilisés lorsque l'ordre dans lequel les événements de défaillance se produisent n'a pas d'importance. Par contre, les arbres de défaillances dynamiques sont utilisés lorsque l'ordre de défaillance des composants du système a un impact sur l'événement indésirable.

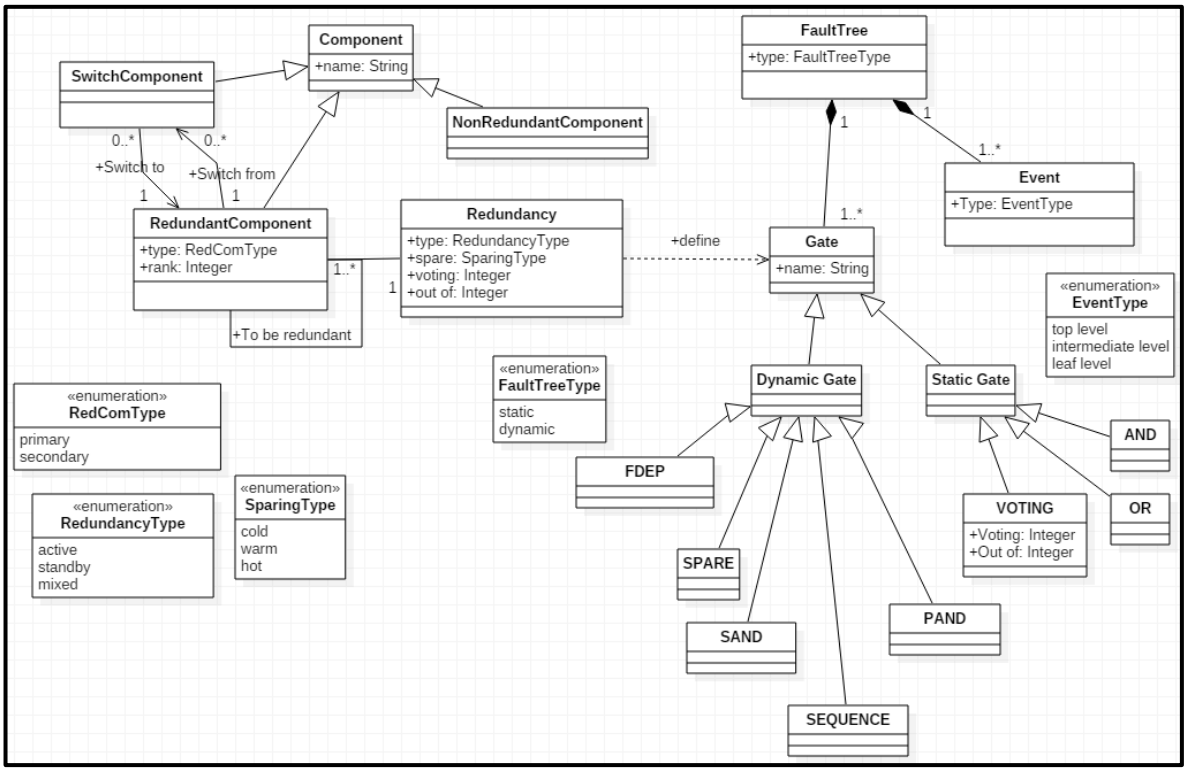


Figure 3. 1: Diagramme de classes pour la gestion de la redondance

Chaque arbre de défaillance doit avoir au moins un événement. Un événement peut être de trois types. L'évènement principal qui représente l'événement indésirable. Les événements feuilles sont des événements élémentaires qui représentent les feuilles de l'arbre de

défaillances. Les événements intermédiaires qui se mettent entre l'événement redouté et les évènements feuilles.

Deux types de portes logiques visent à décrire la propagation des défaillances menant à l'événement indésirable. Les portes logiques statiques telles que «OR», «AND» et «VOTING» sont utilisées dans la construction des arbres de défaillances statiques et dynamiques. Les portes logiques dynamiques [104] telles que "PAND", "SEQUENCE", "SPARE" et "FDEP" sont utilisées dans la construction des arbres de défaillances dynamiques.

Comme indiqué dans la partie gauche de la Figure 3.1 dans le diagramme de classes, une association de généralisation entre la classe « Component » et les trois sous-classes « NonRedundantComponent », « RedundantComponent » et « SwitchComponent » est utilisée. La première catégorie représente les composants non-redondants qui sont des composants non-dupliqués. La deuxième catégorie représente les composants redondants liés entre eux. En cas de redondance en Standby, les composants redondants sont caractérisés par un autre attribut permettant de spécifier l'ordre dans lequel ils fonctionnent (primaire, secondaire, etc.). Enfin, les composants de commutation représentent les composants qui permettent le basculement du composant principal au composant Standby de la redondance. De plus, deux associations, «SwitchFrom» et «SwitchTo», permettent de définir l'ordre d'activation des différents composants redondants.

Pour pouvoir intégrer les informations de la redondance, décrites dans le diagramme de classes, une extension UML appelée «Profil de redondance» a été créée. La Figure 3.2 représente le diagramme du profil de la redondance où les composants « NonRedundantComponent », « RedundantComponent » et « SwitchComponent » sont des extensions de blocs SysML. Ainsi, le stéréotype «Component» étendra la méta-classe «Classe» d'UML. La relation de redondance entre les composants redondants peut être considérée

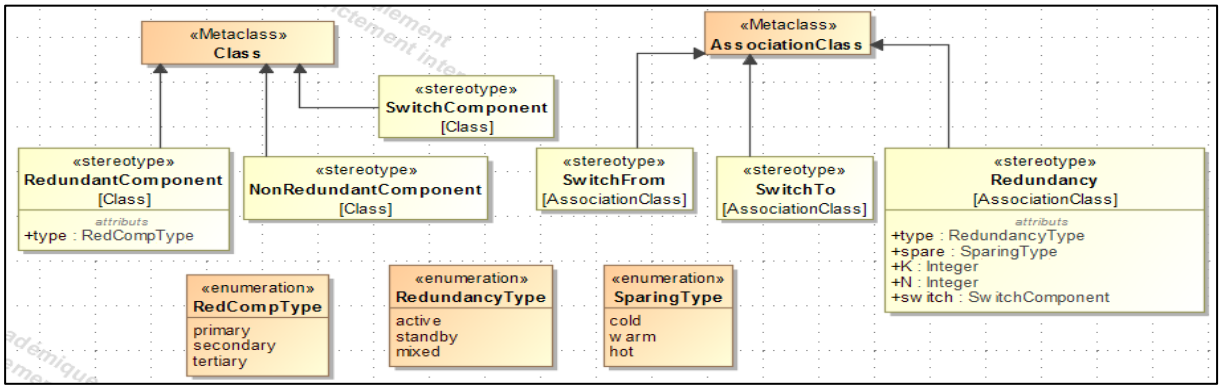


Figure 3. 2: Diagramme de profil pour la gestion de la redondance

comme une extension de la méta-classe «Association Class». Elle sera représentée par un « AssociationBloc » dans le BDD. Les caractéristiques de la redondance, telles que le « type » et « spare », peuvent être considérées comme la définition du stéréotype «Redundancy». En outre, «K» et «N» représentent respectivement le nombre minimal de composants exigé pour maintenir le bon fonctionnement du système et le nombre total de composants redondants.

Par conséquent, pendant la phase de la conception et afin de spécifier la redondance de deux composants ou plus, les concepteurs du système doivent intégrer les informations de la redondance dans le modèle SysML en utilisant le profil de la redondance. Plus tard, la génération de l'arbre de défaillance dynamique sera effectuée en se basant sur ces informations.

3.2.2 Algorithme de génération

Dans cette partie, on explique l'algorithme de génération des arbres de défaillances dynamiques à partir du modèle système enrichi par des informations de redondance. L'algorithme proposé dans ce travail étend l'algorithme de génération des arbres de défaillances statiques élaboré par Mhenni [100]. Les différences entre ces deux algorithmes sont:

- Premièrement, l'ancien algorithme permet la génération d'un arbre de défaillances statique, alors que le nouvel algorithme permet la génération d'un arbre de défaillances dynamique.
- Deuxièmement, l'ancien algorithme génère un arbre de défaillance qui ne prend en compte que la structure du système. Cependant, le nouvel algorithme permet la génération des arbres de défaillances qui regroupent les différents sous-arbres de défaillances.

À cette étape, on possède la structure interne du système, y compris les informations concernant la redondance exprimée sous la forme d'un diagramme SysML (IBD).

Un événement indésirable ou défaillance est défini comme une erreur de sortie externe du système. Pour chaque événement indésirable, un arbre de défaillances est généré. Par conséquent, un port externe intitulé «o» correspondant à l'évènement indésirable à analyser sera considéré comme une entrée dans notre algorithme.

Si l'IBD étudié contient plusieurs ports de sorties externes, un arbre de défaillance dynamique sera généré pour chaque sortie externe. Le résultat de cet algorithme est un arbre de défaillances dynamique.

L'algorithme commence par une première recherche en profondeur dans le diagramme IBD, considéré comme un graphe orienté, afin de détecter toutes les entrées externes et tous les composants du système qui ont un impact sur la sortie de l'événement indésirable «o». Ensuite, une classification, des informations collectées pendant le parcours de l'IBD et les informations obtenues par le profil de redondance, sera faite. L'algorithme classe tous ces éléments en trois ensembles :

- Le premier ensemble (EI) inclut toutes les entrées externes du système qui ont un impact sur l'évènement redouté « o » ;
- Le second ensemble (NRC) contient tous les composants non-redondants du système qui conduit à l'événement indésirable « o » ;
- Le troisième ensemble (RC) comprend des groupes de composants redondants entre eux Chaque groupe contient des composants et des commutateurs liés les uns aux autres afin d'assurer la relation de la redondance.

Une erreur sur un port d'entrée externe (EI) ou sur un composant non-redondant (NRC) entraîne une défaillance, c-à-dire un événement redouté et par la suite entraîne la défaillance du système. Pour cela, l'algorithme génère un sous-arbre sous forme d'un évènement feuille pour chaque élément de l'ensemble EI et de l'ensemble NRC.

De même, pour chaque groupe de redondance dans l'ensemble RC, l'algorithme génère un sous-arbre de défaillance approprié au type de redondance, Active, Standby ou mixte. Dans ce cas, des informations supplémentaires telles que le « RedComponentType » (primaire, secondaire, tertiaire, etc.) seront utilisées. Pour des défaillances des composants de type « SwitchComponent » (les commutateurs), L'algorithme distingue deux modes:

- Le premier mode de défaillance affecte le passage de l'alimentation aux composants et, dans ce cas, le commutateur provoque la défaillance du système en cas de défaillance.
- Le deuxième mode de défaillance affecte le basculement d'un composant à un autre et dans ce cas, le commutateur ne provoque pas directement la défaillance du système mais dépend des autres composants.

1er cas : Si le type de la redondance est Actif et que le système doit avoir au moins «K» sur «N» composants assurant le fonctionnement du système, nous utilisons la porte logique «VOTING (K / N)» comme l'indique la Figure 3.3.

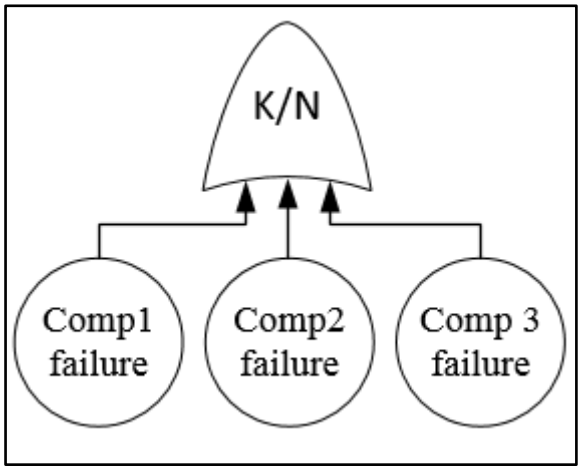


Figure 3. 3: Redondance Active

2e cas : Si le type de redondance est Standby et qu'on n'a pas besoin d'un commutateur pour basculer d'un composant à un autre alors la porte logique «SPARE» est utilisée. De plus, le type des composants redondants doit être défini : composants primaire, secondaire, etc. Dans ce cas, l'algorithme génère un sous arbre de défaillance comme l'indique la Figure 3.4.

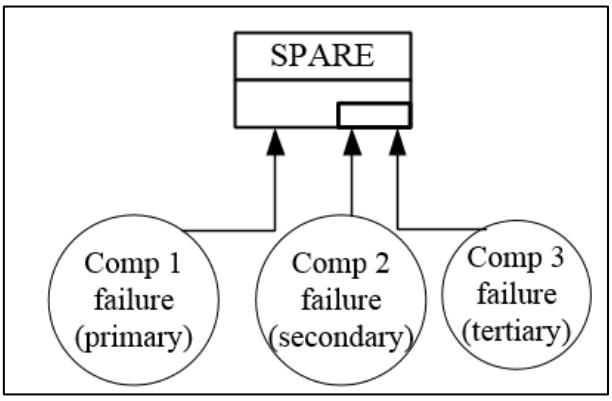


Figure 3. 4: Redondance en Standby sans commutateur

3e cas: Si le type de la redondance est en Standby et un commutateur est nécessaire pour basculer d'un composant à un autre, les portes logiques «SPARE» et «PAND» sont utilisées. Si un seul composant en Standby est utilisé alors l'algorithme génère un sous-arbre de défaillance comme l'indique la Figure 3.5. Dans ce cas, la défaillance du système n'est obtenue que si les composants principaux et secondaires tombent en panne ou si le commutateur tombe en panne avant le composant principal, car dans ce cas, il ne basculera pas vers le composant secondaire.

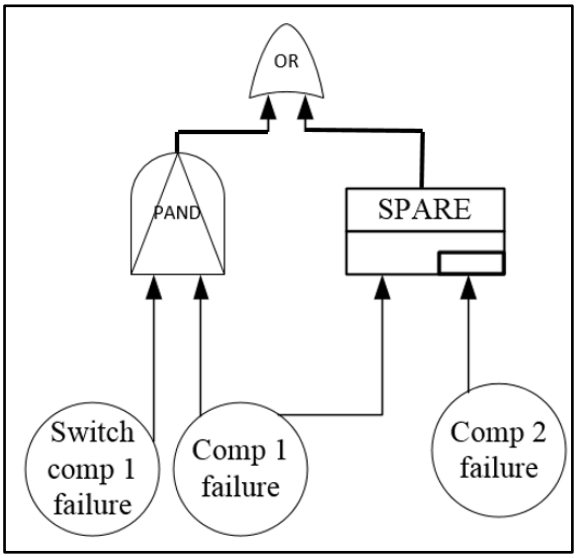


Figure 3. 5: Redondance en Standby avec un commutateur

4e cas : Si en plus des conditions du 3e cas, plusieurs composants de secours sont utilisés avec plusieurs commutateurs, les portes logiques sont présentées de la manière illustrée à la Figure 3.6.

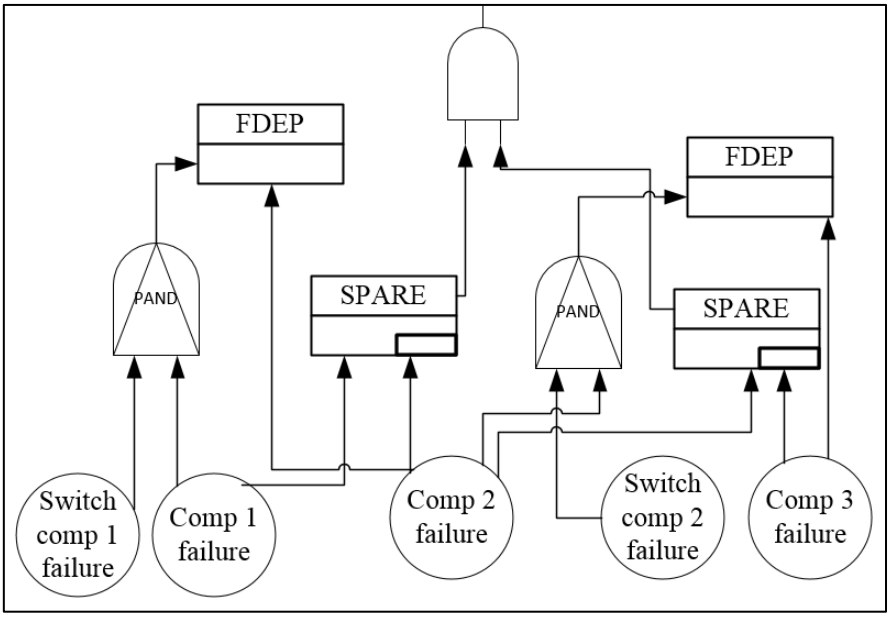


Figure 3. 6: Redondance en Standby avec plusieurs composants de secours et plusieurs commutateurs

5e cas : Si le type de la redondance est Mixte et on n'a pas besoin de commutateur pour basculer entre les composants, alors le résultat de cette situation est la combinaison du sous-arbre de défaillances de la redondance Actif (Figure 3.3) et du sous arbre de défaillance de la redondance Standby sans commutateur (Figure 3.4).

6e cas : Si le type de redondance est Mixte et on a besoin d'un commutateur pour basculer d'un composant à un autre, alors le résultat de cette situation est la combinaison du sous arbre de défaillances de la redondance Actif (Figure 3.3) et du sous-arbre de défaillance de la redondance Standby avec commutateur (Figure 3.5).

7e cas : Si le type de la redondance est Mixte et plusieurs composants de secours sont utilisés avec plusieurs commutateurs, alors le résultat sera la combinaison du sous-arbre de défaillance de la redondance Actif (Figure 3.3) et du sous-arbre de défaillance de la redondance Standby avec plusieurs composants de secours et plusieurs commutateurs.

Enfin, tous les sous-arbres de défaillance générés sont rassemblés afin de créer l'arbre de défaillances dynamiques globales à l'aide de la fonction «Concatenate». L'arbre de défaillance dynamique contient une porte logique « OR » principale et tous les sous-arbres de défaillances générés sont connectés directement à cette porte logique. L'algorithme qui permet la génération de cet arbre de défaillance dynamique à partir du diagramme IBD et des informations sur la redondance est donné ci-dessous.

```
Algorithm DynamicFaultTreeGeneration
Input:
        1) External output port o
        2) IBD as an oriented graph
        3) Information from redundancy profile
Output: Dynamic Fault Tree

Do a depth first search of the IBD from o
        Detect all external inputs that are connected with o
        EI := Set of External Inputs
        Detect all components that are connected with o
        NRC := Set of Non-Redundant Components
        RC := Set of groups of Redundant Components
        i := 1
For each element in EI do
        tree[i] := GenerateSubTree(leaf_event)
        i++;
For each element in NRC do
        tree[i] := GenerateSubTree(leaf_event)
        i++;

For each group of redundancy in RC do
        switch RedundancyType :
                case Active :
                        Read(K)
```

```
                Read(N)
                tree[i] := GenerateSubTree(active_redundancy)
        case Standby :
                if (SwitchComp = True) then
                        tree [i] := GenerateSubTree(standby_with_switch)
                else
                        tree[i] := GenerateSubTree(standby_without_switch)
        case Multi-standby :
                if (SwitchComp = True) then
                        tree[i] := GenerateSubTree(multi_standby_with_switch)
                else
                        tree[i]:= GenerateSubTree(standby_without_switch)
i++;

For j from 1 to i do
        result := Concatenate(tree[j])
return result
```

Dans le cas où l'architecture du système est composée de plusieurs niveaux d'abstraction, alors on procède par le même algorithme d'une manière récursive. Au début, on génère l'arbre de défaillances qui ne prend en compte que le haut niveau d'architecture. Ensuite, on génère les arbres de défaillances des bas niveaux d'architecture avec la même méthode de génération. Les arbres de défaillances des niveaux inférieurs seront concaténés avec l'arbre de défaillance principal.

3.2.3 Création des machines à états

Dans cette sous-section, on décrit la quatrième étape de la méthodologie. L'objectif de cette étape est la création d'une machine à états qui décrit le comportement du système. Une fois que l'arbre de défaillance dynamique est généré à partir du modèle du système, celui-ci sera analysé afin d'obtenir les séquences de coupes minimales du système. Ces séquences de coupes minimales sont utilisées pour créer manuellement des machines à un ou plusieurs états. La machine à états démarre avec l'état de fonctionnement normal puis pour chaque défaillance d'un composant, on passe à l'étape suivante qui affiche le nouveau comportement du système. La défaillance d'un composant représente la transition qui permet le passage d'un état à autre. Cette méthode est répétée jusqu'à parcourir l'intégralité de la séquence de coupes minimales qui représente la panne du système. La quatrième étape de cette méthodologie vise à inclure toutes les informations structurelles et comportementales dans le même modèle SysML. En outre, les machines à états et les IBD générés dans cette étape offrent la possibilité d'utiliser les séquences de coupes pour effectuer des simulations des systèmes à l'aide d'un outil de simulation

(exemple : Cameo Systems Modeler [109]). Cela aide les concepteurs de système à mieux comprendre les défaillances et les dysfonctionnements du système.

3.3 Cas d'étude : Aileron

Dans cette partie, on considère l'exemple d'un système de contrôle des ailerons d'avion avec une architecture redondante pour illustrer l'algorithme de génération des arbres de défaillances dynamiques.

La structure du système est décrite par un diagramme IBD comme l'indique la Figure 3.7. Le système est composé d'une carte de commande « Command Card » qui reçoit les ordres du pilotage et fournit les ordres aux moteurs pour commander l'aileron, trois moteurs redondants «Motor1», «Motor2» et « Motor3 », deux commutateurs « Switch1 » qui permet de basculer de« Motor1 » à« Motor2 » et« Switch2 » qui permet le basculement de« Motor3 » à« Motor2 » et une transmission mécanique « Mechanical Transmission ».

La description du comportement de la redondance du système est décrite dans un diagramme BDD, à l'aide du profil de la redondance présenté dans la Figure 3.8. Dans ce diagramme, on commence par la déclaration des composants redondants et des commutateurs. Tous les composants non déclarés sont considérés comme des composants non-redondants. Dans cette étape, il est simplement demandé de mentionner les composants redondants, leurs types (primaire, secondaire, ..) et les composants de commutation. Ensuite, des blocs d'association entre deux composants redondants sont utilisés pour déclarer et définir leur redondance mutuelle. Des blocs d'association sont utilisés pour déclarer le sens de basculement entre les composants redondants, dans le cas de défaillance, par les composants de commutation en utilisant les classes d'association « SwitchFrom » et « SwitchTo ».

Le scénario proposé dans cette application montre que les composants non redondant sont NRC= {Command card, Switch1, Switch2, Mechanical transmission} et les composants redondants sont des RC= {Motor1, Motor2, Motor3}. «Motor1» et «Motor3» sont primaires et «Motor2» est secondaire, ce qui signifie que si «Motor1» ou «Motor3» tombe en panne, le «Motor2» le remplace. Les composants «Switch1» et «Switch2» sont déclarés comme des composants de commutation. «Motor1» et «Motor3» sont liés par une association de redondance, ce qui signifie qu'ils sont redondants entre eux. La redondance est active et il faut au moins 2 moteurs disponibles pour que le système continue à fonctionner.

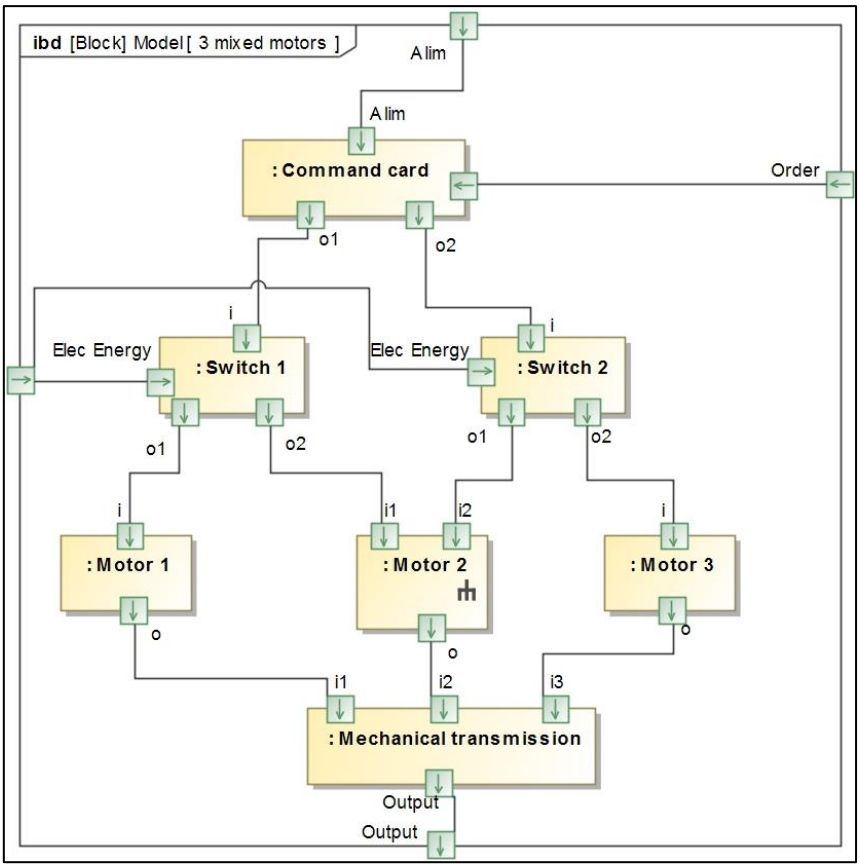


Figure 3. 7: IBD d'un système de contrôle des ailerons d'avion

De plus, on a deux associations de redondance « Motor1 » / « Motor2 » et « Motor3 » / « Motor2 ». Ces deux associations de redondance sont chacune de type Standby. Ainsi, les trois moteurs sont redondants l'un avec l'autre et la redondance est appelée redondance Mixte. D'un autre coté, le composant « Switch1 » permet de basculer de « Motor1 » à « Motor2 » et le composant « Switch2 » permet de basculer de « Motor3 » à « Motor2 » en cas de défaillance du « Motor1 » ou « Motor3 ».

Nous commençons par le choix de l'erreur du port de sortie qui représente l'événement indésirable. Après, on applique un parcours de graphe en profondeur à travers la structure du système pour déterminer ses entrées externes.

Dans cet exemple, trois entrées externes sont détectées : l'alimentation de la carte commande «Alim», les ordres de pilotage de l'aileron «Order» et l'alimentation des moteurs «Elec Energy» (EI= {Alim, Order, Elec Energy}). Pour chaque entrée externe, un sous arbre de défaillance de type événement feuille est généré. Les défaillances des composants non-redondants sont représentées par-dessous des arbres de défaillances de type événement feuille.

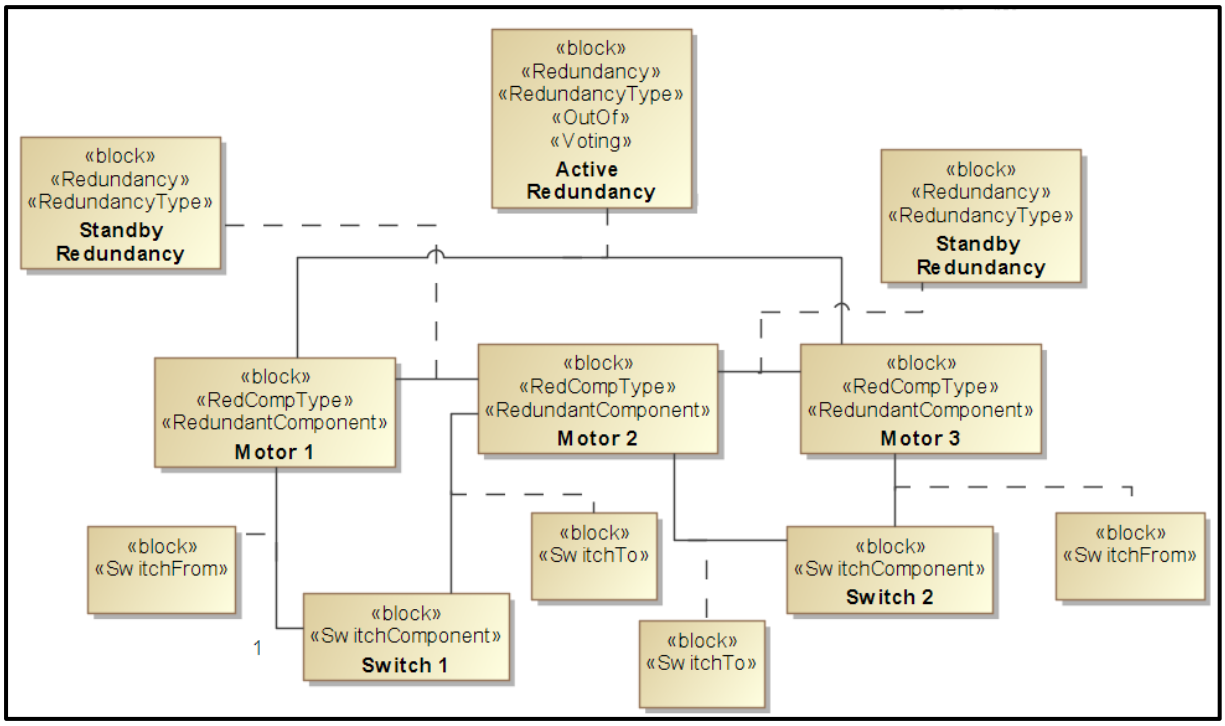


Figure 3. 8: BDD d'un système de contrôle des ailerons d'avion avec des informations de redondance

Les moteurs « Motor1 » (primaire) et « Motor2 » (secondaire) ont une redondance en Standby entre eux et le commutateur « Switch1 » permet le basculement de « Motor1 » à « Motor2 » en cas de défaillance du « Motor1 ». Alors, un sous-arbre de défaillance de la redondance en Standby avec un commutateur est généré, comme illustré dans la Figure 3.5, et qui représente les combinaisons de défaillances possibles de « Motor1 », « Motor2 » et « Switch1 ».

On utilise le même raisonnement pour les composants « Motor3 », « Motor2 » et « Switch2 ». Un sous-arbre de défaillance de la redondance en Standby avec un commutateur est généré.

Nous savons aussi que les moteurs « Motor1 » et « Motor3 » ont une redondance Active entre eux. Pour cela, un sous arbre de défaillance de la redondance Active est généré, comme illustré dans la Figure 3.3.

Étant donné que la distribution générale des composants est en série, tous les événements de défaillance sont directement liés à la porte logique « OR » principale. Le résultat de ce travail est un arbre de défaillance dynamique qui combine les différents sous- arbres de défaillances comme le montre la Figure 3.9. L'analyse de cet arbre de défaillance dynamique garantit, à la fois, l'analyse qualitative et l'analyse quantitative.

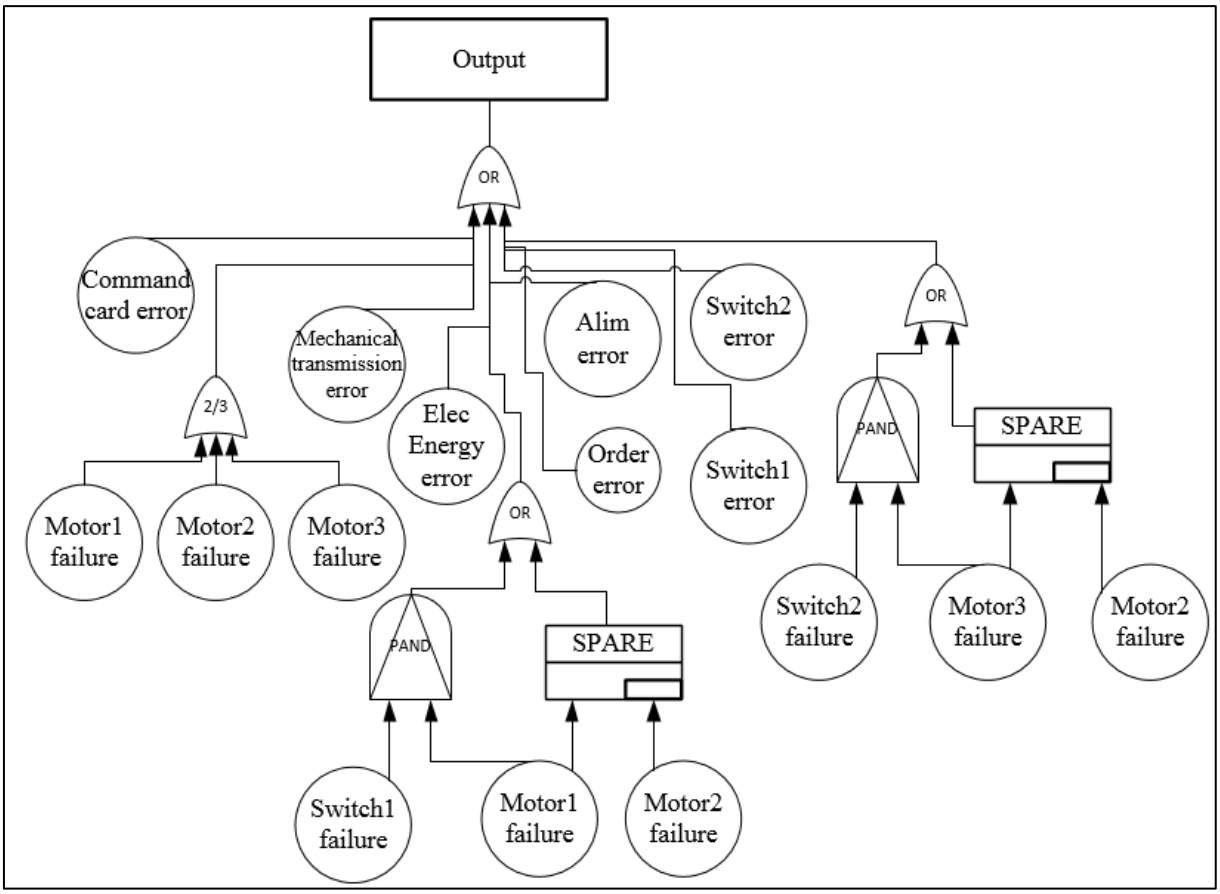


Figure 3. 9: Arbre de défaillance dynamique du système de contrôle des ailerons d'avion

L'analyse qualitative de l'arbre de défaillance de cet exemple donne les séquences de coupes minimales suivantes :

* Motor1 failure & Motor2 failure
* Motor2 failure & Motor1 failure
* Switch1 failure & Motor1 failure
* Motor2 failure & Motor3 failure
* Motor3 failure & Motor2 failure
* Switch2 failure & Motor3 failure
* Motor1 failure & Motor3 failure
* Motor3 failure & Motor1 failure
* Command card error
* Mechanical transmission error
* Elec energy error
* Alim error

* Order error
* Switch1 error
* Switch2 error.

L'objectif de l'utilisation des séquences de coupes minimales à la place des ensembles de coupes minimaux est d'avoir l'information de l'impact de la défaillance des composants dans un ordre donné sur la défaillance du système. Cela veut dire que la défaillance du composant 1 puis composant 2 peut ne pas avoir le même impact sur le système si le composant 2 tombe en panne puis le composant 1.

Concernant l'analyse quantitative, différents résultats peuvent être obtenus tels que la probabilité de défaillance de l'événement redouté, MTTF, MTBF, etc…

Un outil de preuve de concept a été mis en place pour la génération des arbres de défaillances. Tout d'abord, le modèle SysML, qui est représenté par les diagrammes IBD, BDD et le profil de la redondance, est exporté dans le format XMI (XML Metadata Interchange). Ensuite, notre algorithme analyse le fichier XMI à l'aide des bibliothèques du langage de programmation Python. Cet algorithme collecte et analyse toutes les informations utiles, telles que les composants et les informations de redondance, afin de créer l'arbre de défaillance dynamique appropriée. La génération de l'arbre de défaillance peut être exportée sous deux formats :

- Soit le format image PNG (Portable Network Graphics) utilisé pour la visualisation des résultats.
- Soit le format d'échange de modèle Open-PSA [110]. L'arbre de défaillances au format Open-PSA peut être analysé avec des outils utilisant XFTA [111]. Il garantit à la fois l'analyse qualitative telle que les ensembles de coupes minimales et l'analyse quantitative comme les évaluations probabilistes.

3.4 Conclusion

Dans ce chapitre, nous avons présenté la génération automatisée des arbres de défaillances dynamiques. C'est une des étapes de la méthodologie qui vise à intégrer l'analyse de la sûreté de fonctionnement dans l'approche d'ingénierie système. Un profil de redondance a été développé pour permettre la description du comportement des redondances dans le système. Il fournit des informations telles que le type de la redondance, le type des composants, etc. L'algorithme de génération prend en compte la structure du système et toutes les

informations sur la redondance déjà intégrées dans les modèles SysML. Une étude sur un système de contrôle des ailerons d'avion a été utilisée pour illustrer l'algorithme de génération.

La génération automatique des arbres de défaillances dynamiques permet l'amélioration de la cohérence entre les modèles en créant un lien entre le modèle système et les artefacts de la sûreté de fonctionnement. Les travaux futurs consistent à améliorer encore la cohérence entre le modèle système et les artefacts d'analyse de la sûreté en permettant la mise à jour dynamique du modèle système à l'aide de l'analyse des arbres de défaillances dynamiques générés. La structure et le comportement du système seront mis à jour afin d'améliorer la fiabilité et la disponibilité du système. De plus, dans les travaux futurs, l'outil de preuve de concept développé avec Python devrait être amélioré et testé pour différentes tailles d'application afin de valider l'évolutivité de l'algorithme de génération en termes de consommation de mémoire et de temps d'exécution.

Chapitre 4 : Cas d'études

4.1 Introduction

Dans ce chapitre, on illustre la méthodologie proposée par deux cas d'études. Le premier cas d'étude traite l'exemple d'un actionneur électromécanique afin de valider l'ensemble de la méthodologie. Le deuxième cas d'étude traite l'exemple d'un système de distribution de carburant pour l'avion. C'est un exemple plus complexe qui permet la validation de l'algorithme de génération des arbres de défaillances dynamiques.

4.2 Cas d'étude 1 : Actionneur électromécanique (EMA)

Pour des raisons de confidentialité, aucune information industrielle sur l'EMA n'a pu être obtenue. On a essayé de l'étudier avec notre connaissance limitée du système réel et sans retour d'expérience. Les modèles et les analyses fournis ici expriment notre point de vue.

L'EMA vise à actionner les ailerons d'avion et il est de plus en plus utilisé dans les avions car il présente plusieurs avantages comme [112]:

- Un meilleur respect de l'environnement avec la suppression des risques de puissance hydraulique et des fuites d'huile ;
- Un gain de poids sur les ailes ;
- Une réduction des coûts de maintenance ;
- Une augmentation des performances avec une précision de la vitesse grâce aux actionneurs électriques.

L'aileron d'avion est une partie montée sur l'aile de l'avion. Dans les grands avions, on peut avoir quatre ailerons (2 ailerons externes et 2 ailerons internes) comme indiqué dans la Figure 4.1.

Les ailerons sont placés à l'extrémité de l'aile et commandés par le manche. Leurs mouvements asymétriques permettent de modifier l'inclinaison de l'avion sur l'axe de roulis. Les ailerons externes sont utilisés aux basses vitesses alors que les ailerons internes sont utilisés aux grandes vitesses.

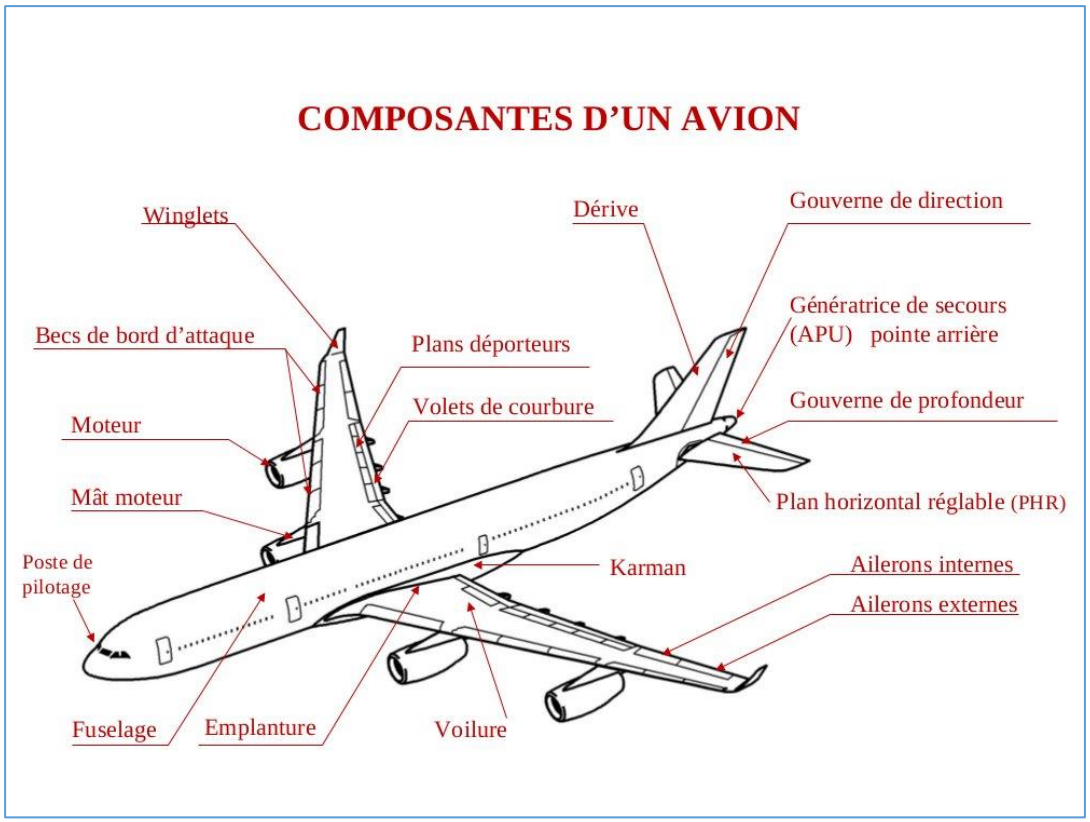


Figure 4. 1: Composantes d'un avion [113]

4.2.1 Etape 1 : Génération automatique de FMEA à partir des modèles SysML

La première étape de cette approche est la génération automatique de FMEA à partir des modèles systèmes. Pour le faire, il faut, tout d'abord, définir les exigences du système, le périmètre de l'étude de ce système ainsi que son architecture fonctionnelle et logique. Toutes ces informations sont intégrées dans des diagrammes SysML.

Dans la Figure 4.2, on trouve un extrait des exigences initiales de l'EMA. Ce diagramme décrit la décomposition de la fonction principale assurée par le système.

Cette même décomposition de la fonction principale est représentée par le diagramme d'activité de la Figure 2.4 du chapitre 2. Ensuite, une décomposition supplémentaire est réalisée; les sous-fonctions « Control and command » et « Actuate Aileron » sont détaillées, aussi, dans

des diagrammes d'activité, respectivement dans les Figures 2.5 et Figure 2.6. Ces diagrammes permettent la définition de l'architecture fonctionnelle du système.

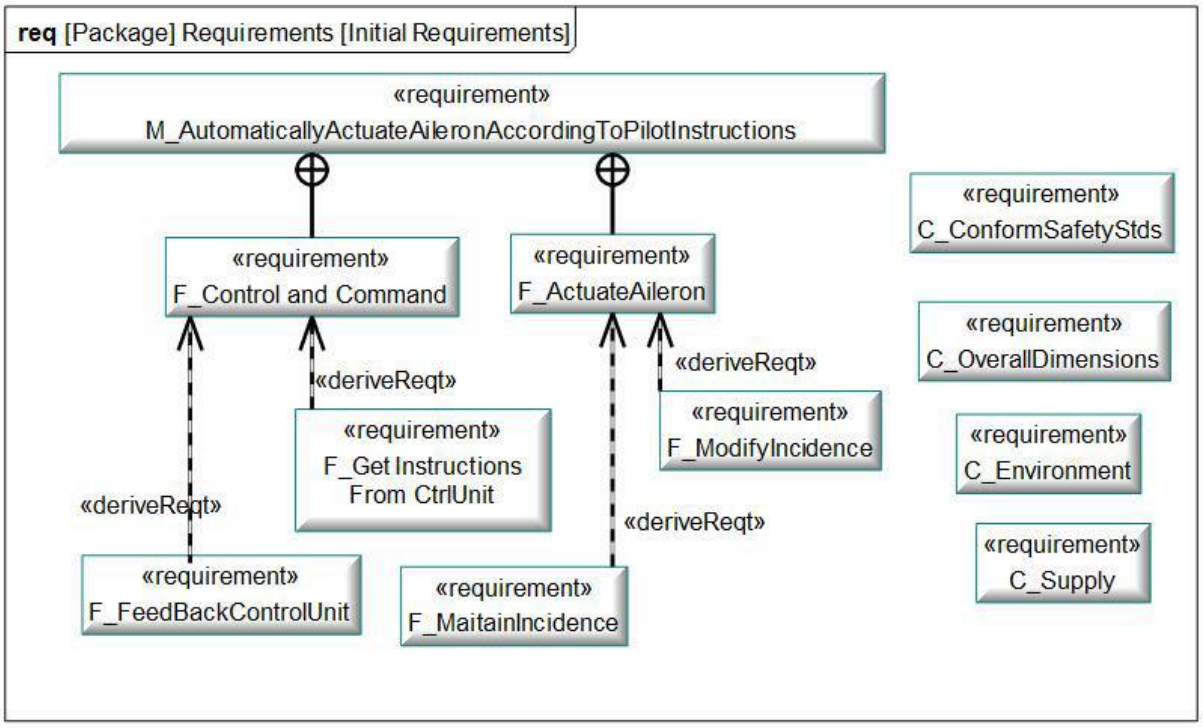


Figure 4. 2: Extrait des exigences initiales de l'EMA

Le diagramme d'activité principal décompose la fonction principale assurée par ce système qui est « Automatically actuate aileron according to pilot instructions » en deux fonctions qui sont « Control and Command » et « Actuate Aileron ». La fonction « Control and Command » est décomposée en deux sous-fonctions qui sont « Translate Pilot Instructions » et « Regulate Electrical Energy ». De même, la fonction « Actuate Aileron » est décomposée en 4 sous-fonctions qui sont « Transform Electrical/Mechanical Energy », « Adapt Mechanical Energy », « Transmit Mechanical Energy » et « Measure Incidence ».

Après, en se basant sur l'architecture fonctionnelle, des composants seront choisis pour réaliser les différentes fonctions identifiées dans l'architecture fonctionnelle. Les fonctions à réaliser sont affectées aux composants correspondants. La règle pour l'attribution est qu'une fonction ne peut être attribuée qu'à un seul composant à la fois, alors qu'un composant peut avoir plus d'une fonction à exécuter. La raison pour qu'une fonction ne soit affectée qu'à un seul composant est la difficulté de définition de la responsabilité de chaque composant vis-à-vis de la fonction (c.-à-d. le composant remplit quelles parties de la fonction ?). Cette règle ne s'applique pas dans le cas des composants redondants qui remplissent la même fonction. Lorsqu'une fonction est exécutée par plusieurs composants, elle doit ensuite être décomposée

jusqu'à ce que ses sous-fonctions soient attribuées sans ambiguïté à des composants respectant la règle ci-dessus.

La Figure 4.3 représente le diagramme IBD de l'EMA. Ce diagramme est composé de trois composants: une partie électronique « Embedded MCU with Power Bridge », un motoréducteur électrique avec codeur « Geared Motor with encoder » et une transmission mécanique « Mech Transmission ». En outre, les entrées du système sont « Electric Power » et le "CtrlData". La sortie du système est la puissance mécanique « Mech Power Aileron » qui fournit le mouvement de sortie de l'aileron.

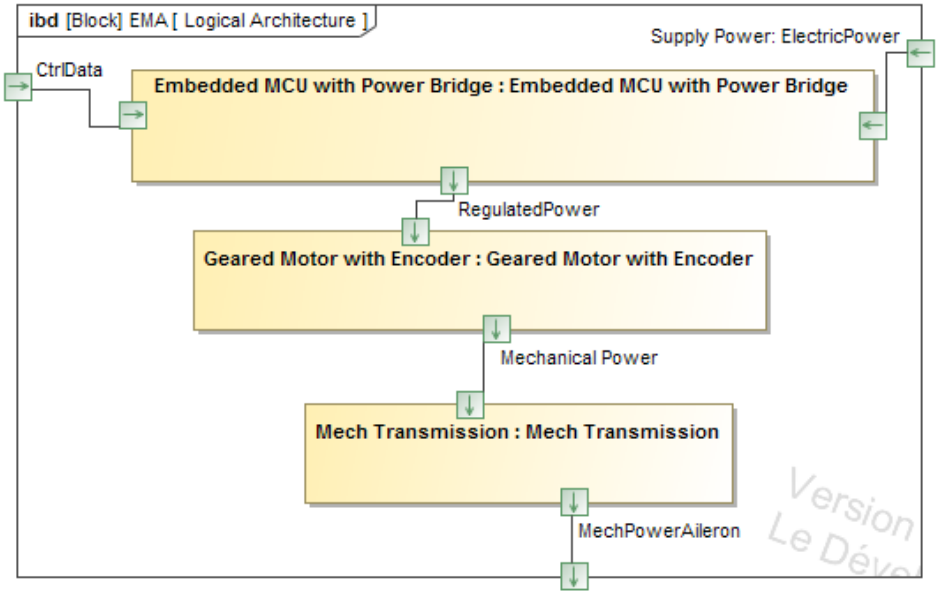


Figure 4. 3: La structure initiale de l'EMA

En utilisant les diagrammes d'activités et le diagramme IBD, on génère automatiquement un FMEA préliminaire qui contient les composants, les fonctions et leurs modes de défaillances génériques.

En se basant sur ce FMEA préliminaire, un expert en sûreté de fonctionnement remplit le tableau par les données de sévérité, occurrence, détectabilité, RPN et les classes des risques. On obtient le FMEA initial de ce système dans le Tableau 7.

Tableau 7: FMEA initial de l'EMA

Component	Fonction	Mode de défaillance fonctionnelle	Mode de défaillance	S	O	D	RPN	Classe risque
EMCU	Regulate Electrical Energy	Fails to perform	Hardware defect	8	7	4	224	II
		Performs incorrectly	Hardware defect	6	7	4	168	II
	Translate Pilot Instructions	Fails to perform	Software error	8	4	6	192	III
			Hardware defect	9	7	7	441	I
			Synchronization error	10	5	6	300	I
			Memory defect	7	4	5	140	III
		Performs incorrectly	Software fault	6	7	4	168	II
		Performs incorrectly	Synchronization error	5	5	6	150	III
Geared Motor with encoder	Adapt Mechanical Energy	Fails to perform	Loss of structural integrity	10	6	4	240	I
		Fails to perform	Jamming	9	4	8	288	II
		Performs incorrectly	Poor efficiency	2	8	3	48	II
	Measure Incidence	Fails to perform	Mechanical drive defect	7	4	5	140	III
			Power loss	7	8	3	168	I

			Internal component failure	6	5	8	240	III
		Performs incorrectly	Mechanical drive defect	5	7	6	210	II
			Low or high voltage	5	3	7	105	IV
			Measure fault	4	7	5	140	III
	Transform Mechanical Energy	Fails to perform	Jamming	6	7	8	336	I
			Loss of structural integrity	10	3	5	150	II
			Short-circuit between two winding	8	5	4	160	III
		Performs incorrectly	Short-circuit in one winding	7	4	6	168	III
Mechanical Transmission	Transmit Mechanical Energy	Fails to perform	Loss of structural integrity	9	3	5	135	III
			Jamming	6	5	3	90	III
		Performs incorrectly	Poor efficiency	5	6	5	150	III

4.2.2 Etape 2 : Choix d'architecture

Dans cette étape, le FMEA généré est analysé et les modes de défaillance les plus critiques sont détectés. La première approche se repose sur l'utilisation de la valeur de RPN. Pour cela, il faut définir une valeur seuil qui exprime les valeurs interdites de RPN.

Pour cet exemple, la valeur seuil est **α = 400**. Cette valeur est prise à titre d'exemple qui permet, juste, la modélisation du scénario proposé. Dans le cas réel, chaque entreprise choisira une valeur seuil selon son domaine d'activité et le degré de sûreté à chercher.

Par conséquent, les modes de défaillances qui ont un RPN >= 400 sont classés comme des modes de défaillances critiques. En analysant le FMEA, on trouve qu'on a un seul mode de défaillance critique qui est « Fails to perform hardware defect » de la fonction « Translate Pilot Instructions » du composant « EMCU ». Pour cela, le composant « EMCU » doit être doublé par un autre « EMCU » redondant pour minimiser la criticité de ce composant.

Le problème dans cette approche est que, parfois, on peut avoir des modes de défaillances qui ont une sévérité et une occurrence élevées mais, en même temps, ils ont un RPN < 400 ce qui pose un problème pour la sûreté de fonctionnement de ce système. Ce problème existe dans cet exemple, regardons, par exemple, le mode de défaillance « Fails to perform power loss » de la fonction « Measure Incidence » du composant « Geared Motor with encoder » qui a une sévérité et occurrence élevées égale à 7 et 8 respectivement. Malgré ces valeurs élevées de la sévérité et de l'occurrence, on trouve un RPN < 400. Donc, ce mode de défaillance ne sera pas classé comme un mode de défaillance critique.

On propose une nouvelle approche qui se repose non seulement sur RPN mais aussi sur les classes de risques de la norme IEC61508. Le but de l'analyse de FMEA est de détecter les modes de défaillances critiques qui ont un RPN >= 400 ou classe de risque = I.

L'analyse de ce FMEA montre l'existence de cinq modes de défaillance des différentes fonctions avec une classe de risque égale à I.

1/ fails to perform the translation of pilot instructions because of a hardware defect of the EMCU component;

2/ fails to perform the translation of pilot instructions because of a synchronization error of the EMCU component;

3/ fails to perform the adaptation of the mechanical energy because of loss of structural integrity of the geared motor with encoder;

4/ fails to perform the measure incidence because of a power loss of the geared motor with encoder;

5/ fails to perform the transformation of the mechanical energy because of a jamming of the geared motor with encoder.

Afin d'améliorer l'architecture du système, les deux composants critiques « EMCU » et « Geared Motor with encoder » doivent être doublés et le diagramme IBD sera mis à jour. Le nouveau IBD comporte deux EMCU redondants et deux motoréducteurs redondants avec un composant de commutation permettant la commutation du composant principal vers le composant secondaire comme illustré dans la Figure 4.4.

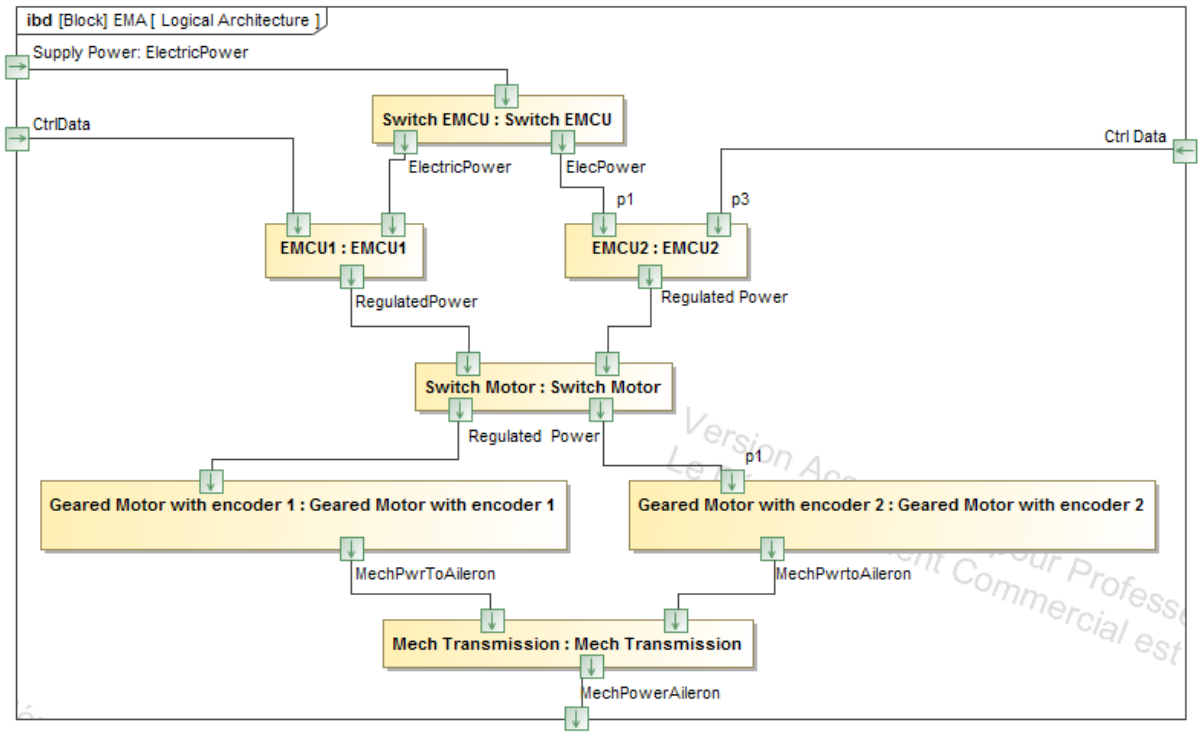


Figure 4. 4: La nouvelle structure du système

Ensuite, une nouvelle FMEA doit être générée pour vérifier le niveau de sûreté du système. Ce nouveau FMEA inclut les nouveaux composants tels que «Switch EMCU» et «Switch Motor». Si les classes risques de tous les modes de défaillance sont différentes de celui de la classe I et leurs RPN < 400, la structure du système est validée. Sinon, une nouvelle itération doit être effectuée et de nouvelles modifications structurelles sont recommandées. Dans notre cas et comme le montre le Tableau 8, tous les modes de défaillance de fonction ont une classe de risque différente de la classe I et un RPN < 400. La nouvelle structure du système est validée et on passe à la troisième étape qui est la génération des arbres de défaillances dynamiques à partir du modèle système.

Tableau 8: Nouveau FMEA de l'EMA

Component	Fonction	Mode de défaillance fonctionnelle	Mode de défaillance	S	O	D	RPN	Classe risque
EMCU	Regulate Electrical Energy	Fails to perform	Hardware defect	8	4	4	128	III
		Performs incorrectly	Hardware defect	6	4	4	96	III
	Translate Pilot Instructions	Fails to perform	Software error	8	2	6	96	III
			Hardware defect	9	2	7	126	III
			Synchronization error	10	2	6	120	III
			Memory defect	7	1	5	35	IV
		Performs incorrectly	Software fault	6	3	4	72	III
		Performs incorrectly	Synchronization error	5	3	6	90	III
Switch EMCU	Transmit Electric Energy	Fails to perform	Hardware defect	9	2	5	90	III
	Switch-over the Electric Energy	Fails to perform	Hardware defect	7	3	5	105	III
Geared Motor with encoder	Adapt Mechanical Energy	Fails to perform	Loss of structural integrity	10	3	4	120	II
		Fails to perform	Jamming	9	2	8	144	III

		Perform incorrectly	Poor efficiency	2	4	3	24	IV
	Measure Incidence	Fails to perform	Mechanical drive defect	7	2	5	70	III
			Power loss	7	4	3	84	III
			Internal Component failure	6	3	8	144	III
		Performs incorrectly	Mechanical drive defect	5	4	6	120	III
			Low or high voltage	5	2	7	70	IV
			Measure fault	4	4	5	80	III
	Transform Mechanical Energy	Fails to perform	Jamming	6	4	8	192	III
			Loss of structural integrity	10	2	5	100	III
			Short-circuit between two winding	8	3	4	96	III
		Performs incorrectly	Short-circuit in one winding	7	2	6	84	III
Switch Motor	Transmit Electric Energy	Fails to perform	Hardware defect	9	2	5	90	III
	Switch-over the Electric Energy	Fails to perform	Hardware defect	7	3	5	105	III

Mechanical Transmission	Transmit Mechanical Energy	Fails to perform	Loss of structural integrity	9	3	5	135	III
			Jamming	6	5	3	90	III
		Performs incorrectly	Poor efficiency	5	6	5	150	III

4.2.3 Etape 3 : Génération des arbres de défaillances dynamiques à partir des modèles systèmes

Dans la troisième étape, on étudie la nouvelle architecture du système. Puisque ce nouveau système contient de la redondance, un diagramme BDD est créé en utilisant le profil de la redondance afin de décrire les relations entre les composants redondants et par la suite de faciliter la génération automatique de l'arbre de défaillance dynamique.

Le nouveau système contient deux groupes de composants redondants avec commutateur. Le BDD de la Figure 4.5 décrit à la fois la redondance des motoréducteurs et la redondance des EMCU «EMCU1», «EMCU2», «Geared Motor with encoder 1» et «Geared Motor with encoder 2» sont des composants redondants. « EMCU1 » et « EMCU2 » sont redondants l'un avec l'autre et le type de redondance est Standby. «EMCU1» est primaire et «EMCU2» est secondaire. Le «Switch EMCU» est un composant de commutation qui permet de passer de «EMCU1» à «EMCU2» en cas de défaillance de «EMCU1». De la même manière que EMCU, «Geared Motor with encoder 1» et le «Geared Motor with encoder 2» sont redondants entre eux et le type de redondance est Standby. « Geared Motor with encoder 1 » est primaire et « Geared Motor with encoder 2 » est secondaire. Le «Switch Motor» est un composant de commutation permettant de passer du «Geared Motor with encoder 1» au «Geared Motor with encoder 2» en cas de défaillance du «Geared Motor with encoder 1».

Ensuite, l'algorithme de génération des arbres de défaillances dynamiques utilise la structure du système (IBD) représentée dans la Figure 4.4 et les informations de la redondance (BDD) représentées dans la Figure 4.5 afin de générer les arbres de défaillances dynamiques. Premièrement, l'erreur dans le port de sortie est considérée comme un événement indésirable de l'arbre. Deuxièmement, la première recherche en profondeur dans la structure du système est appliquée afin de déterminer les entrées externes du système.

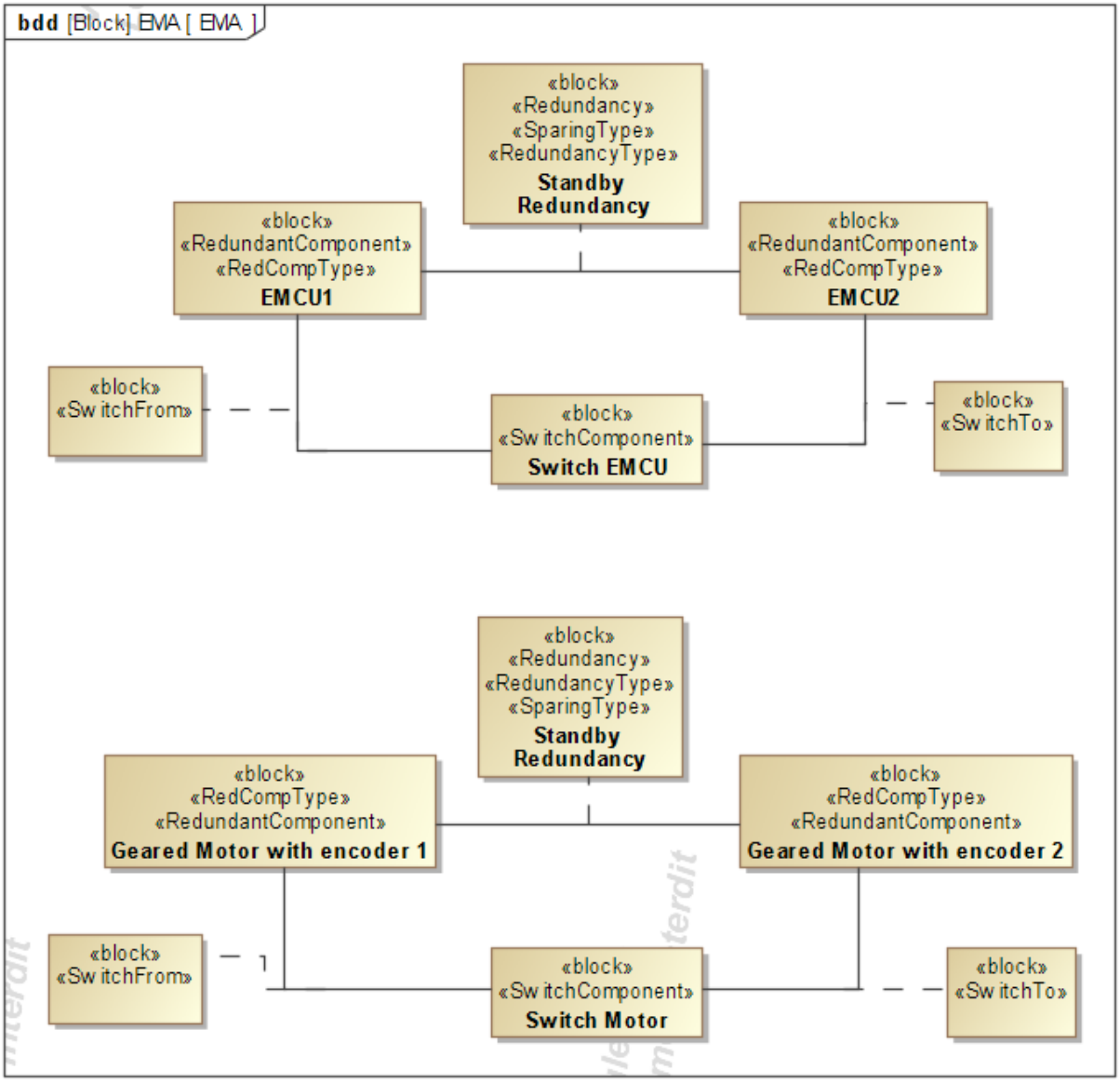


Figure 4. 5: BDD de l'EMA avec les informations sur la redondance

Dans l'exemple de l'EMA, deux entrées externes sont atteintes qui sont «CtrlData» et «Electric Power». Pour chaque entrée externe, un événement de défaillance feuille est généré. De plus, les erreurs des composant non-redondantes sont représentées par un événement de défaillance feuille dans l'arbre de défaillance dynamique. Dans cet exemple, seul le composant «Mechanical Transmission» est non-redondant. Les deux groupes de redondance sont des redondances en Standby avec un composant commutateur, donc un sous-arbre de défaillance d'une redondance en Standby avec commutateur est généré comme indiqué dans la Figure 3.15. Sachant que la distribution générale des composants est en série, tous les sous arbres de défaillances générés sont directement liés à la porte logique principale « OR ». La sortie de

l'algorithme est l'arbre de défaillance dynamique de la Figure 4.6 qui fournit toutes les combinaisons de défaillances possibles.

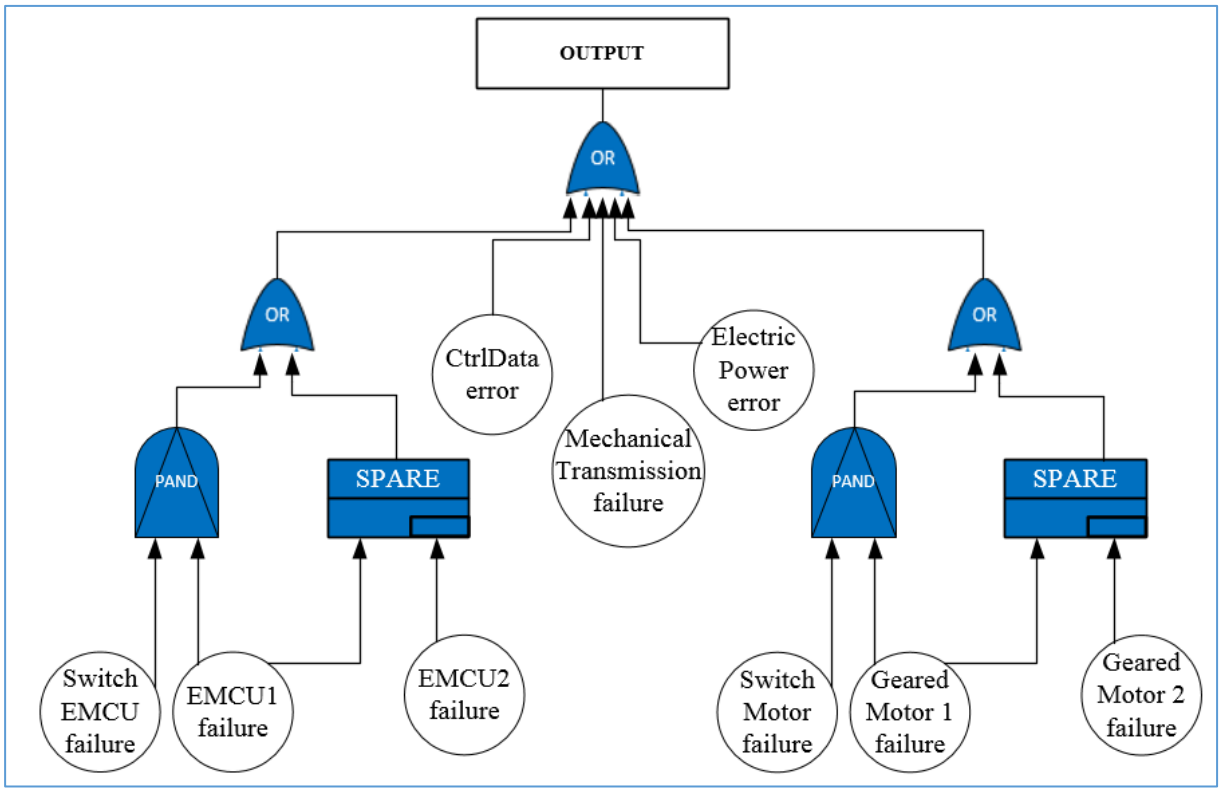


Figure 4. 6: Arbre de défaillance dynamique de l'EMA

4.2.4 Etape 4 : Création de la machine d'état

Après la génération de l'arbre de défaillance dynamique, on passe à l'analyse de cet arbre. Dans cette partie, on se contente, seulement, de l'analyse qualitative. Le résultat de l'analyse est un ensemble de séquences de coupe minimales. Dans cet exemple, neuf séquences de coupes minimales sont détectées, à savoir:

* EMCU1 failure &EMCU2 failure
* EMCU2 failure &EMCU1 failure
* Switch EMCU failure &EMCU1 failure
* Geared motor 1 failure &Geared motor 2 failure
* Geared motor 2 failure &Geared motor 1 failure
* Switch motor failure &Geared motor 1 failure
* CtrlData error
* Electric power error

* Mechanical transmission failure

La quatrième étape de cette méthodologie consiste à créer une machine à états décrivant le comportement du système en fonction des séquences de coupes minimales générées. Dans chaque état, le comportement de tous les composants est décrit. Lorsque la défaillance se produit, le nouveau comportement du système est décrit dans l'état suivant. Cette itération est répétée jusqu'à la défaillance du système qui est représenté par la fin de la séquence de coupes minimales. Le résultat de ce travail est présenté dans la Figure 4.7. Les séquences de coupe minimales avec un seul élément ne sont pas prises en compte dans ce travail.

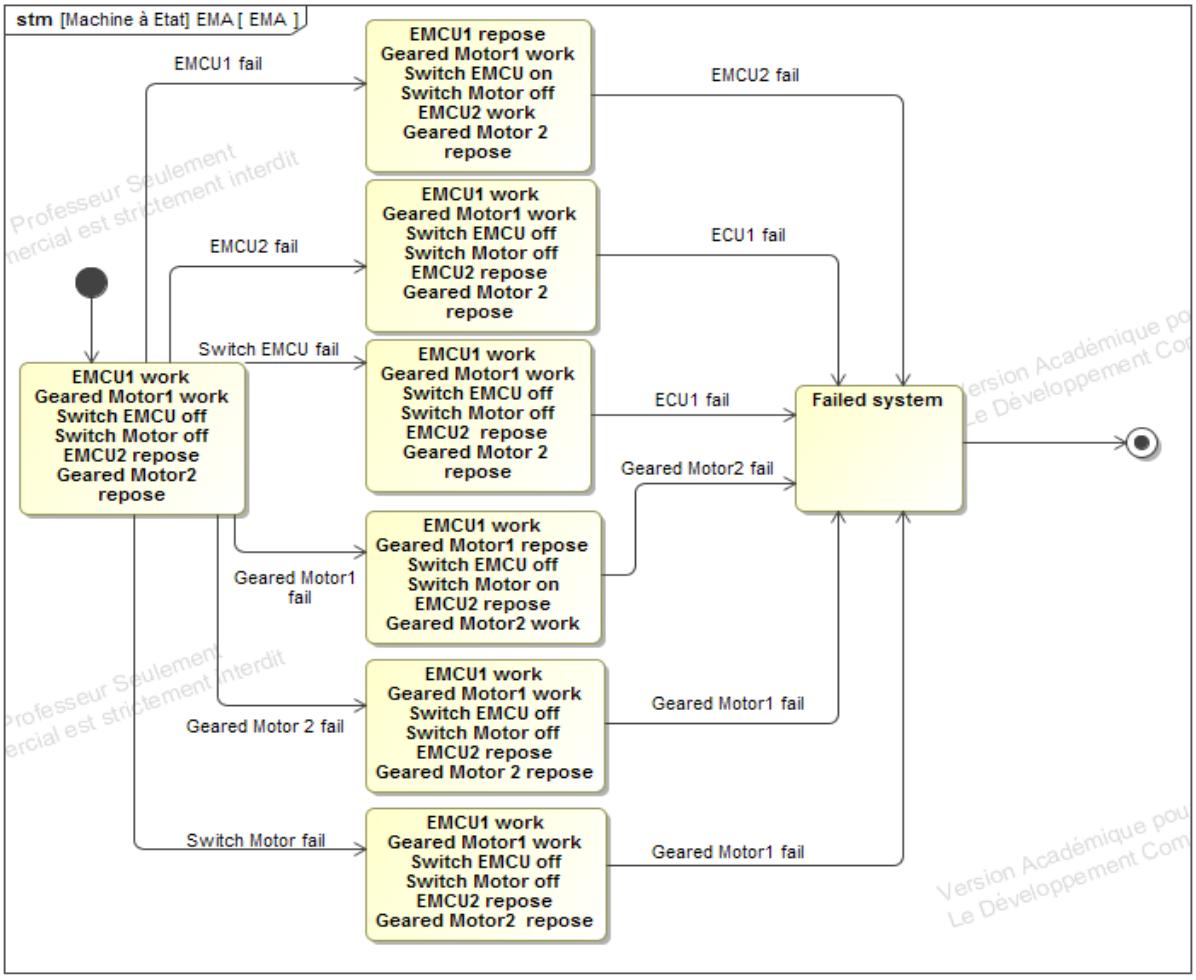


Figure 4. 7: Machine à états de l'EMA

4.3 Cas d'étude 2 : Aircraft Fuel Distribution System (AFDS)

Dans cette partie, on étudie un système complexe qui est le système de distribution de carburant d'avion (en anglais : Aircraft Fuel Distribution System (AFDS)) avec une structure redondante pour illustrer l'algorithme de génération des arbres de défaillances dynamiques (étape 3 de la méthodologie).

Ce système est utilisé dans l'étude de cas du travail dans [114]. L'AFDS assure deux fonctions principales qui sont le stockage du carburant dans les réservoirs pendant la phase de ravitaillement (en anglais : refuelling phase) et la distribution du carburant dans tout le système pendant la phase de consommation. La structure du système décrite dans la Figure 4.8 est composée d'un moteur gauche (en anglais : Port Engine (PE)) et un moteur droit (en anglais : Starboard Engine (SE)), 5 réservoirs de carburant qui stockent le carburant qui sont le réservoir extérieur gauche (en anglais : Port Outer Tank (POT)), le réservoir intérieur gauche (en anglais : Port Inner Tank (PIT)), le réservoir extérieur droit (en anglais : Starboard Outer Tank (SOT)), le réservoir intérieur droit (en anglais : Starboard Inner Tank (SIT)) et le réservoir de réservation central (en anglais : Central Reservation Tank (CRT)), 7 pompes à carburant qui envoient le carburant dans tout le système (CRP, PCP, SCP, CPP, CSP, PJV et SJP) et 11 vannes permettant l'ouverture ou la fermeture de certains chemins (CRV, CPV, CJV, CPV, CSV, PIV, SIV, POV-PJV, SIV-SOV, PCV-CPV et SCV-CSV). Le système est divisé en trois sous-systèmes principaux qui sont l'alimentation gauche (en anglais : Port Feed (PF)), l'alimentation droite (en anglais : Starboard Feed (SF)) et la réservation centrale (en anglais : Central Reservation (CR)). Chaque sous-système est composé de plusieurs composants comme l'indique la Figure 4.8. Le système AFDS a comme point de départ le point de ravitaillement (en anglais : Refuelling Point (RP)) qui permet de stocker le carburant dans CRT. De plus, le système dispose de 3 sorties différentes qui sont la sortie principale des moteurs, le point de largage (en anglais : Jettison point) (côté PF) et le point de vidange (côté SF).

La description du comportement de la redondance du système est donnée par le diagramme BDD, qui est un diagramme structurel du SysML, à l'aide du profil de la redondance. On commence par la déclaration des composants redondants et des commutateurs. Tous les composants non déclarés sont supposés des composants non redondants. Dans cette étape, il est simplement demandé de mentionner les composants redondants, leurs types (primaire, secondaire, ..) et les composants de commutation. Ensuite, des blocs d'association entre deux composants redondants sont utilisés pour déclarer que ces composants sont redondants entre eux, pour définir son type de redondance, etc. De plus, les blocs d'association sont utilisés pour déclarer le sens du basculement entre les composants redondants dans le cas de défaillance en utilisant les classes d'association SwitchFrom et SwitchTo.

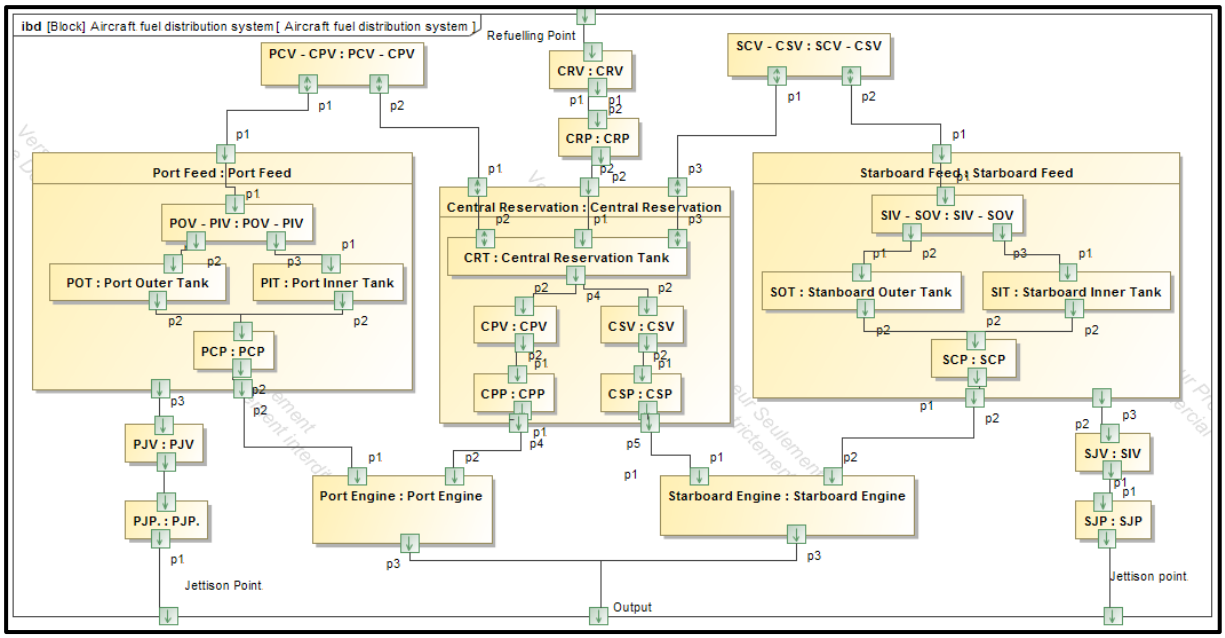


Figure 4. 8: La structure du système de distribution de carburant de l'avion

Pendant la phase de consommation et dans les conditions de fonctionnement normales, le sous-système SF alimente le moteur SE et le sous-système PF alimente le moteur PE. Lorsque le SF ou le PF tombe en panne, le sous-système CR prend le relais pour fournir du carburant aux moteurs. Pour assurer le passage de SF (ou PF) à CR, le SCV (ou PCV pour le côté PF) doit être fermé et le CSV (ou CPV pour le côté PF) doit être activé. Par conséquent, les sous-systèmes SF, PF et CR sont déclarés en tant que composants redondants. Les sous-systèmes PF et CR ont une redondance en Standby entre eux, avec PF primaire et CR secondaire (de même pour SF et CR). De plus, les vannes PCV-CPV et SCV-CSV sont déclarées comme composant de commutation et permettent la commutation du composant primaire au composant secondaire. Le diagramme BDD illustré à la Figure 4.9 décrit la structure de la redondance du système.

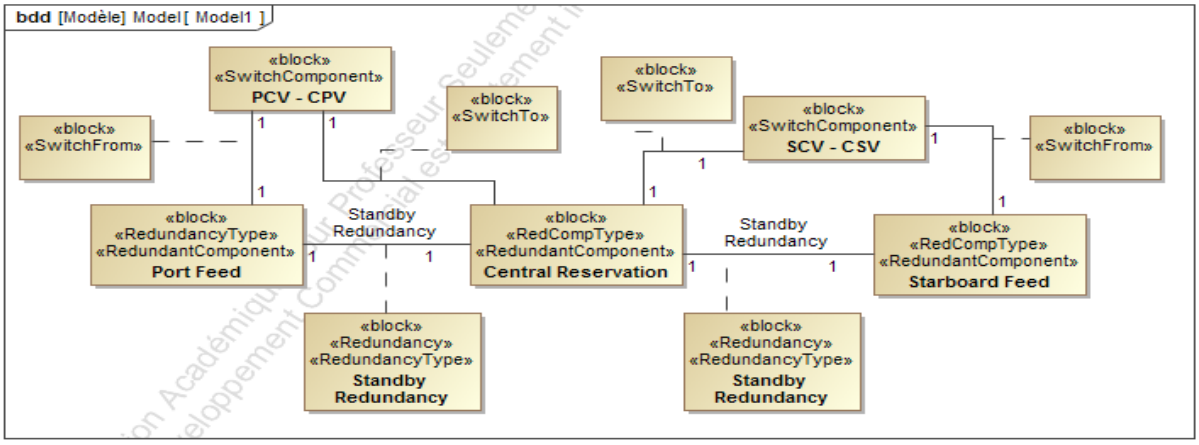


Figure 4. 9: BDD du système de distribution de carburant d'avion avec informations de redondance

En outre, le sous-système PF est composé de deux réservoirs à port redondant (POT et PIT). La redondance est en Standby avec POT primaire et PIT secondaire. La vanne POV-PIV est déclarée en tant que composant de commutation et permet la commutation entre les composants en fermant POV et en activant PIV. La sortie du sous-système PF fournit du carburant au PE. La Figure 4.10 décrit la structure de la redondance dans le sous-système PF.

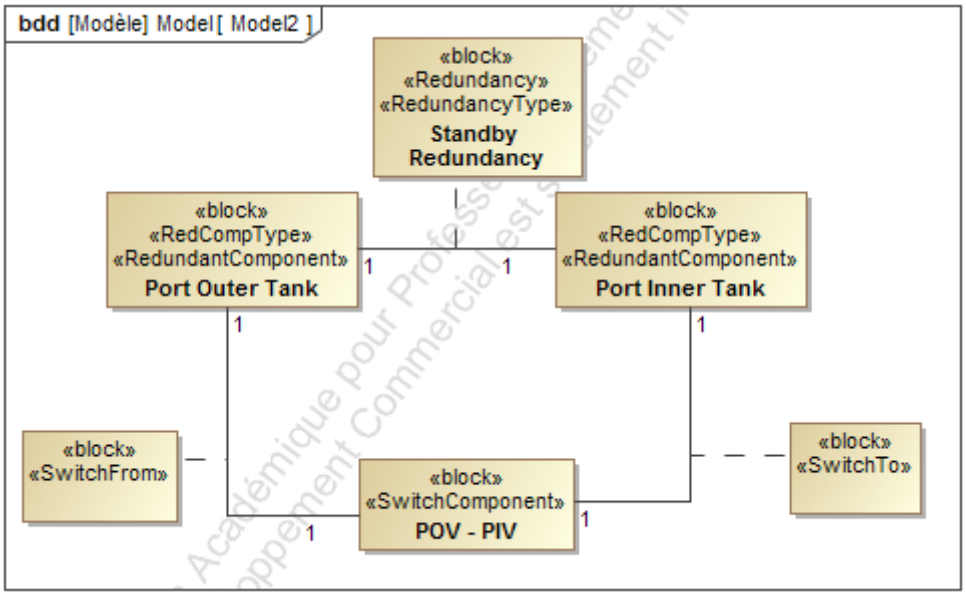


Figure 4. 10: BDD du sous-système de l'alimentation de port avec informations de redondance

Symétriquement, le sous-système SF est composé de deux réservoirs redondants (SOT et SIT). La redondance est en Standby avec SOT primaire et SIT secondaire. La vanne SCV-CSV est déclarée comme composant de commutation et permet la commutation entre les composants en fermant SOV et en activant SIV. La sortie du sous-système SF fournit du carburant au SE. La Figure 4.11 décrit le comportement de la redondance dans le sous-système SF.

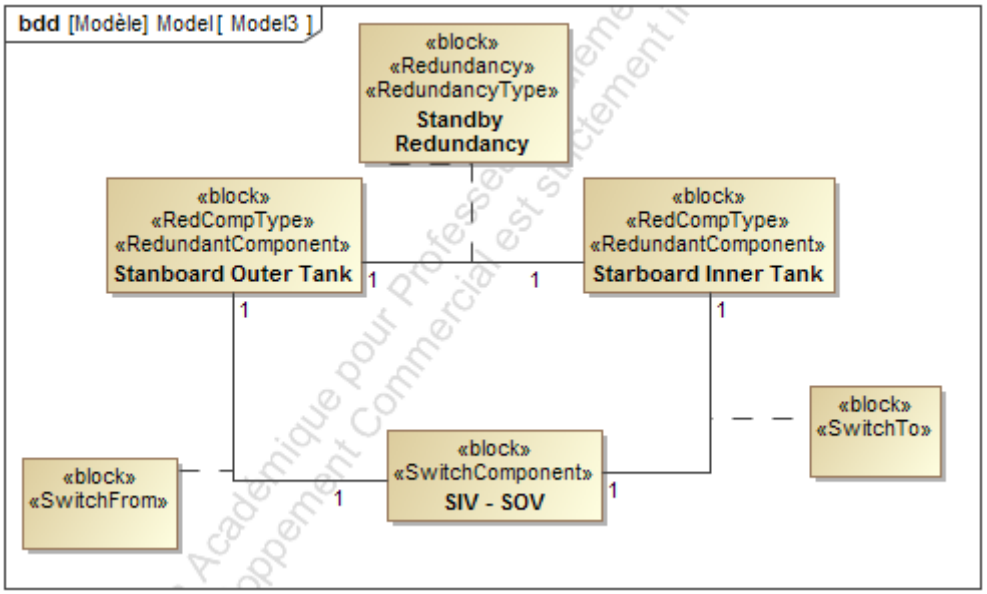


Figure 4. 11: BDD du sous-système d'alimentation tribord avec informations de redondance

Pour créer l'arbre de défaillance du système, on considère l'erreur du port de sortie des moteurs comme un événement indésirable. Ensuite, nous appliquons la première recherche de profondeur à travers la structure du système pour déterminer les entrées externes du système. Dans cet exemple, une seule entrée externe est atteinte: le « Refuelling point ». Pour cette entrée externe, un événement de défaillance de type feuille est généré. De plus, les erreurs des composants non redondants sont représentées par des événements de défaillances de type feuille. L'ensemble des NRC est composé de [PCV-CPV, CRV, SCV-CSV, CRP, PE et SE]. Le sous-système PF (principal) et le sous-système CR (secondaire) ont une redondance en Standby entre eux. Le composant PCV-CPV permet le basculement de PF à CR. Ainsi, un sous arbre de défaillance d'une redondance en Standby avec commutateur comme l'indique la Figure 3.15 est utilisée pour représenter les défaillances possibles du sous-système PF, du sous-

système CR et du composant PCV-CPV. Le même raisonnement est suivi pour le sous-système SF, le sous-système CR et le composant SCV-CSV.

Étant donné que la distribution générale des composants est en série, tous les événements de défaillance sont directement liés à la porte logique « OR » principale. Le premier résultat de ce travail est un arbre de défaillance dynamique partiel comme le montre la Figure 4.12.

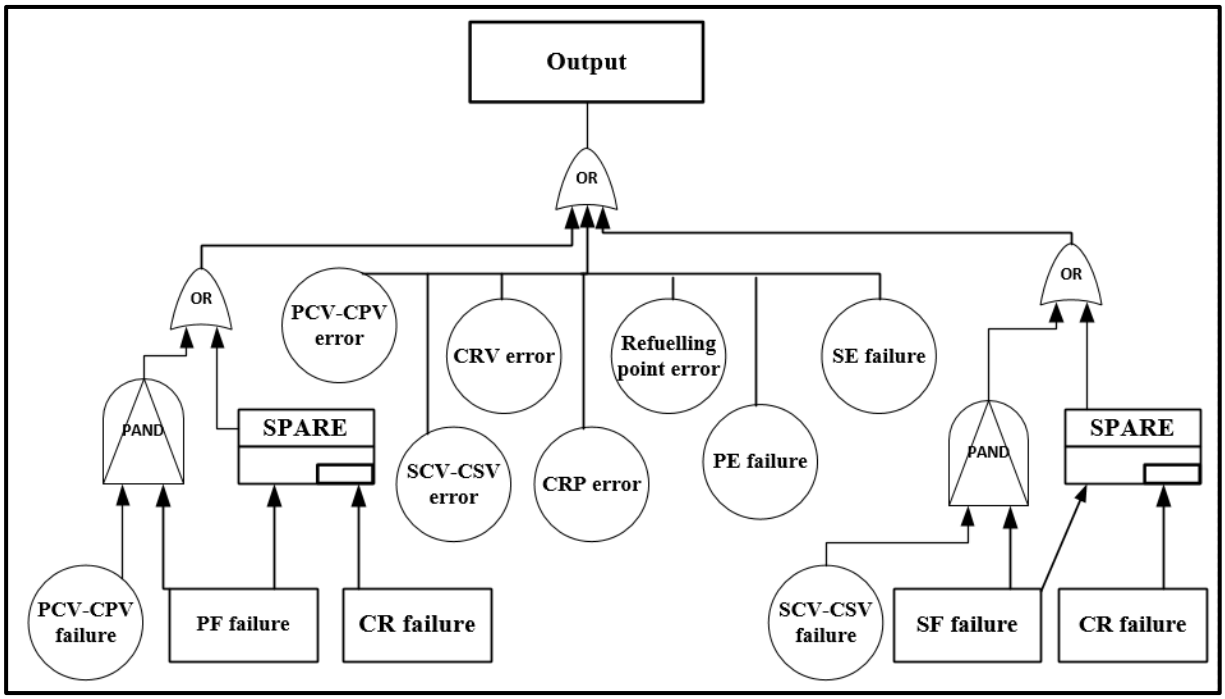


Figure 4. 12: Arbre de défaillance dynamique partiel de l'AFDS

Ensuite, le même algorithme est appliqué pour chaque sous-système afin de générer un arbre de défaillance dynamique de l'erreur de la sortie correspondante. Commençant par le sous-système PF, il contient un composant non redondant (PCP) et deux réservoirs à ports redondants. La redondance est en Standby avec POT (primaire), PIT (secondaire) et le composant POV-PIV est le composant commutateur permettant le basculement de POT à PIT. Donc, un sous arbre de défaillance de la redondance en Standby avec un commutateur, comme illustré dans la Figure 3.15, est utilisée pour représenter les défaillances possibles des composants POT, PIT et POV-PIV. Le même raisonnement est conduit pour le sous-système SF.

Dans le sous-système CR, les composants CRT, CPV, CSV, CPP et CSP sont des composants non redondants. Donc, un événement de défaillance de type feuille est généré pour chaque erreur de composant. Le résultat final de la génération de l'arbre de défaillance est un arbre de défaillance dynamique général comme illustré dans la Figure 4.13.

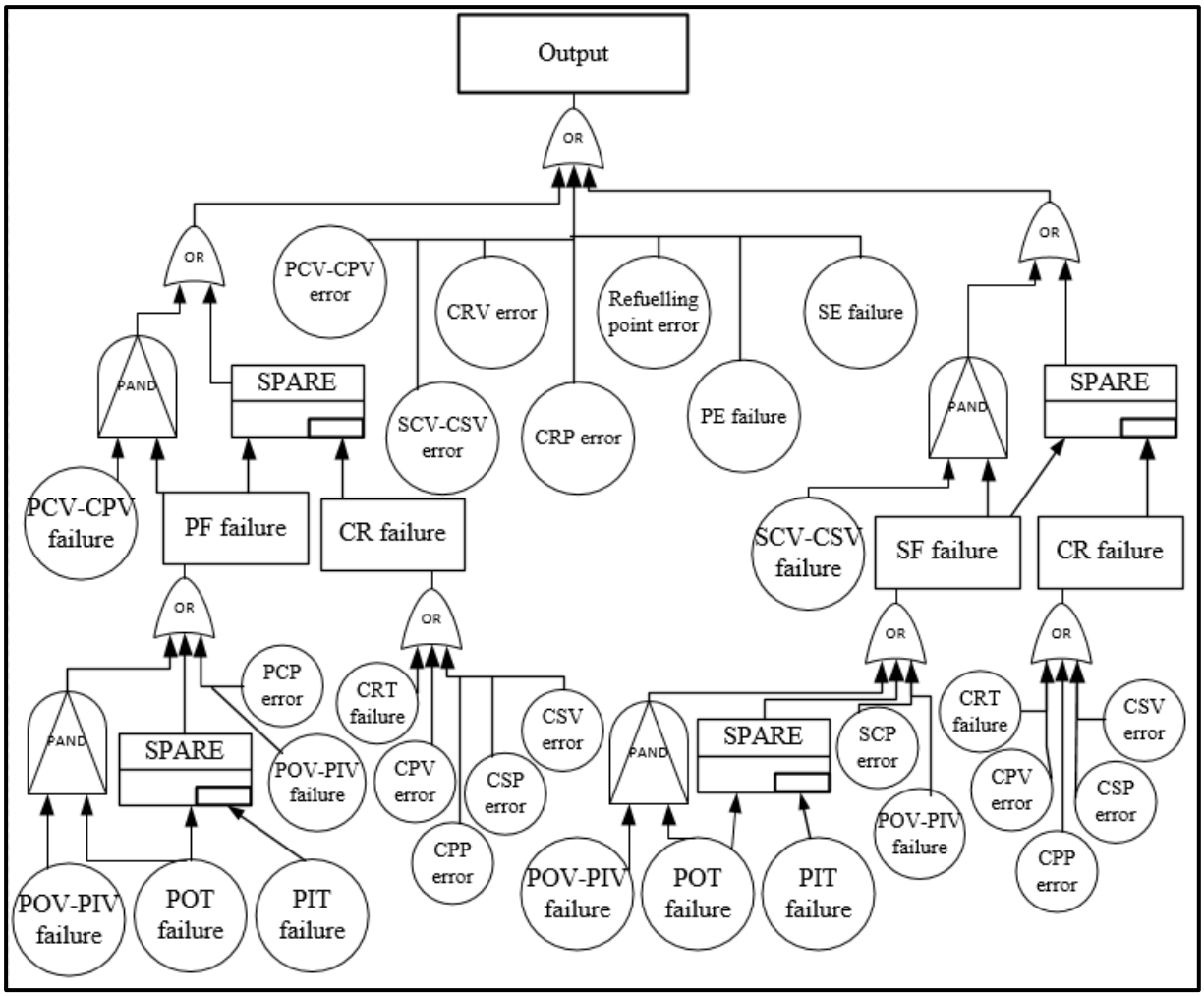


Figure 4. 13: Arbre de défaillance dynamique général de l'AFDS

4.4 Conclusion

Ce chapitre a présenté deux différents cas d'étude pour pouvoir bien tester et valider la méthodologie à quatre étapes pour l'amélioration de la sûreté de fonctionnement et assurer la cohérence entre les différentes vues des modèles. Le premier cas d'étude est un EMA qui possède une architecture relativement simple, alors que le second cas d'étude, d'une architecture plus complexe qui est un système de distribution de carburant dans les avions. A partir de la simple structure du système, un FMEA est généré. Puis, en l'analysant, on propose de nouvelle architecture du système qui améliore sa fiabilité. Ensuite, on génère un arbre de défaillance dynamique à partir de la nouvelle architecture obtenue. Cet arbre permet la description du comportement du système est surtout le comportement de ses redondances. Après l'analyse de l'arbre de défaillance dynamique généré, on crée une machine à états qui enrichit le modèle système par le comportement du système en cas de défaillance de l'un de ces

composants. Cette machine à états permet avec le diagramme IBD de simuler le comportement du système en utilisant un simulateur.

Conclusion générale

La sûreté de fonctionnement des systèmes mécatroniques complexes représente un réel défi pour les experts de sûreté, les ingénieurs système et les concepteurs, en raison de la complexité croissante des systèmes. Tout d'abord, les ingénieurs systèmes sont devant l'obligation de gérer la fusion de plusieurs domaines d'ingénierie (mécanique, électronique, informatique, etc.) tout en étant limités par le comportement physique et les interférences de plusieurs domaines de la physique qui impliquent certaines contraintes supplémentaires comme les couplages thermiques, la Compatibilité Electro-Magnétique (CEM), les vibrations, etc.

La conception des systèmes critiques demande une attention particulière aux exigences de sûreté pour se conformer aux contraintes imposées par les réglementations et les normes de sûreté de fonctionnement. Mais jusqu'à présent, il semble évident que les ingénieurs système ne sont pas entièrement équipés et formés pour faire face à ces exigences. Les experts de sûreté de fonctionnement, de leur point de vue, doivent traiter les aspects liés à la sûreté afin de vérifier que toutes les exigences sont satisfaites par le système. Les méthodes d'évaluation de sûreté traditionnelles, tels que FMEA et FTA, offrent une aide précieuse aux architectes systèmes car ils fournissent des rapports et des recommandations qui permettent l'ajustement ou la correction du système. Cependant, leurs exécution est généralement très longue. S'ils se produisent tard dans le processus de conception, cela peut provoquer une source d'incohérence entre les modèles, ce qui ne permet pas la prise en compte des contraintes de sûreté au plus tôt dans le processus de conception, ce qui peut s'avérer très coûteux lorsque les problèmes sont détectés tardivement dans le processus.

L'objectif de ce travail consiste à réduire l'écart entre les modèles de l'ingénierie système et les artéfacts de la sûreté de fonctionnement en intégrant les informations de la sûreté de fonctionnement dans le processus d'ingénierie système. Pour atteindre cet objectif, une méthodologie composée de quatre étapes est développée.

Le premier objectif de ce travail est développé dans les deux premières étapes de cette méthodologie. La première étape commence par la modélisation du système en partant d'un ensemble d'exigences cohérent et traçable. A partir du modèle système crée, on génère des

FMEAs préliminaire enrichi par des informations de sûreté. Après, en analysant les FMEAs obtenus, on propose des modifications structurelles afin d'améliorer la fiabilité du système. La solution choisie dans la méthode développée est la redondance et particulièrement la redondance matérielle. La structure du système est mise à jour et le même processus est répété jusqu'à l'élimination de la criticité de tous les composants.

Une fois que la structure du système est mise à jour, on passe à la satisfaction de notre deuxième objectif de génération des arbres de défaillances dynamiques à travers l'analyse du comportement du système modifié intégrant la redondance ajoutée. Les arbres de défaillances dynamiques générés prennent en compte la génération de la redondance, pour cela, un profil de redondance décrivant la structure de la redondance dans le système est créé. Ce profil permet la déclaration de la redondance, le type de la redondance (Active, Standby ou Mixte), les composants redondants, les commutateurs, la direction de commutation, etc.. L'algorithme de génération des arbres de défaillance commence par une recherche dans la structure du système qui est considéré comme un graphe orienté afin de détecter toutes les entrées externes et tous les composants du système qui ont un impact sur la sortie de l'événement indésirable. Ensuite, une classification, des informations collectées pendant le parcours de la structure et les informations obtenues par le profil de redondance, sera faite. Puis, l'algorithme classe tous ces éléments en sous-ensembles. La sortie de cet algorithme est un arbre de défaillance dynamique qui donne toutes les combinaisons de défaillances possibles. Aussi, un outil de preuve de concept a été mis en place pour la génération des arbres de défaillances. Tout d'abord, le modèle SysML, qui est représenté par les diagrammes IBD, BDD et le profil de la redondance, est exporté sous le format XMI. Après, notre algorithme analyse le fichier XMI à l'aide des bibliothèques du langage de programmation Python. Cet algorithme collecte et analyse toutes les informations utiles, tels que les composants et les informations de redondance, afin de créer l'arbre de défaillance dynamique appropriée. La génération de l'arbre de défaillance peut être exportée sous deux formats ; Format image pour la visualisation des résultats ou format d'échange de modèle Open-PSA pour une analyse qualitative ou quantitative de l'arbre.

Dans la dernière étape, l'arbre de défaillance généré est analysé et un ensemble de séquences de coupe minimales est obtenu. Ensuite, en utilisant ces séquences, on crée un diagramme de machine à états qui permet la description du comportement du système en cas de défaillance d'un composant. Dans le cadre de ce travail, cette machine à états est le premier et l'unique diagramme qui décrit le comportement du système dans le modèle SysML.

La méthodologie est appliquée sur deux cas d'étude afin de la valider et de montrer l'amélioration obtenue sur le comportement du système.

A la fin de ce travail, on apporte une méthodologie qui permet l'amélioration de la sûreté de fonctionnement de la structure et le comportement des systèmes mécatroniques critiques. On a réussi à modifier la structure des systèmes mécatroniques complexes suite à l'analyse de sûreté. Cette structure contient de la redondance pour réduire la criticité des composants critiques et par la suite réduire la criticité du système. Egalement, on a apporté une méthode de génération des arbres de défaillances dynamiques à partir d'un modèle système qui permet l'analyse des systèmes mécatroniques critiques afin d'identifier les anomalies structurelles et comportementales du système.

Cependant, la méthodologie proposée présente certaines limites qui vont faire l'objet des futurs travaux. Par exemple, l'ajout de la redondance n'est pas toujours la solution optimale en termes de poids et de coût supplémentaires. De plus, lors de la génération des arbres de défaillances à partir des modèles système, l'algorithme proposé ne peut pas prendre en compte la redondance des portes d'entrées externes du système. Pour ces raisons, dans les travaux futurs, il conviendrait d'améliorer la cohérence entre les modèles système et les artefacts d'analyse de la sûreté en proposant des solutions structurelles autres que la redondance. En outre, le profil de la redondance doit être amélioré pour prendre également en compte la redondance des entrées externes du système. Enfin, le résultat de l'analyse des arbres de défaillances dynamiques sera utilisé, non seulement pour la description du comportement du système, mais également pour permettre la mise à jour de son modèle.

Références

[1] P. David, V. Idasiak, and F. Kratz. Reliability Study of Complex Physical Systems Using SysML. Reliability Engineering and System Safety, 95(4):431 – 450, 2010. ISSN 0951-8320.

[2] Verein Deutscher Ingenieure. Design Methodology for Mechatronic Systems, June 2004. VDI 2206.

[3] M. Bozzano and A. Villafiorita. Integrating Fault Tree Analysis with Event Ordering Information. In Safety and Reliability: Proceedings of the ESREL 2003 Conference, Maastricht, the Netherlands, 15-18 June 2003.

[4] J.-P. Meinadier, Ingénierie et Intégration des Systèmes, Editions Hermès, 1998.

[5] Systems Engineering Handbook. International Council on Systems Engineering (INCOSE) Working Group, 2004.

[6] N. Leveson., O. Hammami and J-L. Voirin. Complex Systems Design and Management, Proceeding of the Second International Conference on Complex Systems Design and Management CSDM 2011, pages 27–39, 2011.

[7] Society of Automotive Engineers International. Guidelines for Development of Civil Aircraft and Systems, 2010. SAE-ARP-4754A.

[8] R. Guillerm. Intégration de la Sûreté de Fonctionnement dans les Processus de l'Ingénierie Système. PhD thesis, Université de Toulouse, 2011.

[9] J. Schlager, Systems Engineering: Key to Modern Development. IRE Transaction on Engineering Management, July 1956.

[10] A. Hall. A Methodology for Systems Engineering. Van Nostrand Reinhold, 1962.

[11] A-P. Sage. Systems Engineering. Wiley IEEE, 1992.

[12] Electronic Industries Alliance. EIA-632 – Processes for Engineering a System, January 1999.

[16] INCOSE Resp Group, Genesis of INCOSE, June 2004. Retrieved 11-07-2006.

[17] US Department of Defense (DoD). Engineering Management, May 1974. Mil Std-499A.

[18] H. Eisner. Essentials of Project and Systems Engineering Management. Wiley Publisher, 2002.

[19] K. Forsberg, H. Mooz, Visualizing Project Management: Models and Frameworks for Mastering Complex Systems. Proceeding of the 1991 INCOSE Symposium.

[20] K. Forsberg, H. Mooz, System Engineering for Faster, Cheaper, Better, in: Proceedings of the 9th Annual International Symposium of the INCOSE 1998, Vancouver, Canada, 1998: pp. 1–8.

[21] J. Estefan. Survey of Model-Based Systems Engineering Methodologies, Rev. B. INCOSE MBSE Initiative, 23 Mai 2008.

[22] Les 3 Grandes Familles de Processus: Management, Réalisation et support https://www.pyx4.com/blog/3-familles-processus-management-realisation-support/

[23] AFIS, Association Française d'Ingénierie Système, *www.afis.fr*.

[24] F. Schneider and B. Berenbach. A Literature Survey on International Standards for Systems Requirements Engineering. Procedia Computer Science, 16(0):796 – 805, 2013. 2013 Conference on Systems Engineering Research.

[25] F. Mhenni. Safety Analysis Integration in a Systems Engineering Approach for Mechatronic Systems Design, PhD thesis, École Centrale Paris, 2014.

[26] IEEE Computer Society. Standard for Application and Management of the Systems Engineering Process, 1999. IEEE-1220.

[27] Electronic Industries Alliance. Processes for Engineering a System, 1998. EIA-632.

[28] INCOSE. INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities. International Council of Systems Engineering, Cecilia Haskins edition, June 2006.

[29] D-M. Buede. The Engineering Design of Systems: Models and Methods. John Wiley & sons, 2009.

[32] Object Management Group, Systems Modeling Language (SysML). *www.omgsysml.org*

[33] M. Héon, J. Basque and G. Paquette. Validation de la Sémantique d'un Modèle Semi-Formel de Connaissances avec Onto Case. In 21èmes Journées Francophones d'Ingénierie des Connaissances, Nimes, France, 2010.

[34] N. Belloir, J-M. Bruel, N. Hoang, and C-D. Pham. Utilisation de SysML pour la Modélisation des Réseaux de Capteurs. In Actes de la Conférence Langages et Modèles à Objets (LMO 08) Montréal, Canada, 2008.

[35] E. Adrianarison and J-D. Piques. SysML for Embedded Automotive Systems: A Practical Approach. In Embedded Real Time Software and Systems ERTS 2010, 19th - 21st may 2010, Toulouse, France. 2010.

[36] D. Chapon and G. Bouchez. On the Link between Architectural Description Models and Modelica Analyses Models. In Proceedings 7th Modelica Conference, Como, Italy, Sep. 20-22, 2009.

[37] C.A. Ericson, II. Hazard Analysis Techniques for System Safety; Wiley: New York, NY, USA, 2005; pp. 1–528.

[38] E. Balz and J. Goll. Use Case-Based Fault Tree Analysis of Safety Related Embedded Systems. In Proceedings Software Engineering and Applications, 2005.

[39] N. Xiao, H-Z. Huang, Y-L-L. He, and T. Jin. Multiple Failure Modes Analysis and Weighted Risk Priority Number Evaluation in FMEA. Engineering Failure Analysis, 18:1162–1170, 2011.

[40] B. Bertsche, Reliability in Automotive and Mechanical Engineering, 2008 Springer.

[41] C. Pagetti, Sûreté de Fonctionnement, ENSEEIHT, 2012.

[42] E. Ruijtersy and M. Stoelinga, Fault Tree Analysis: A survey of the State-of-the-Art in Modeling, Analysis and Tools, Computer Science Review, Volumes 15–16, February–May 2015, Pages 29–62.

[43] A. Talon, D. Boissier and L. Peyras, Analyse de Risques : Identification et Estimation: Démarches d'Analyse de Risques - Méthodes Qualitatives d'Analyse de Risques, 2009.

[44] H. Xu, L. Xing, and R. Robidoux: Dynamic Reliability Block Diagrams for System Reliability Modeling. International Journal of Computers and Applications, 31, 202–, 2009.

[45] M. Rausand and A. Hoyland. System Reliability Theory- Models, Statistical Methods, and Applications. John Wiley & sons, 2008.

[46] Society of Automotive Engineers International. Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996. SAE-ARP-4761.

[47] The International Electrotechnical Commission. Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related systems. parts 1 to 7, 1998. IEC 61508.

[48] P. Helle. Automatic SysML-based Safety Analysis. In Proceedings of the 5th International Workshop on Model Based Architecting and Construction of Embedded Systems, 2012.

[49] M. Bozzano, A. Cavallo, M. Cifaldi, L. Valacca, and A. Villafiorita. Improving Safety Assessment of Complex Systems: An Industrial Case Study. In Proceedings of Formal Methods 2003, Volume 2805 of LNCS, Springer-Verlag, pages 208-222, 2003.

[50] M. Bozzano and A. Villafiorita. Integrating Fault Tree Analysis with Event Ordering Information. In Proceedings of ESREL 2003, pages 247-254, Maastricht, The Netherlands, 15-18 June, 2003.

[51] M. Bozzano and A. Villafiorita. Improving System Reliability via Model Checking: the FSAP / NuSMV-SA Safety Analysis Platform. In Proceedings of SAFECOMP 2003, pages 49-62, Edinburgh, Scotland, United Kingdom, 23-26 September, 2003.

[52] FSAP/NuSMV-SA. *http://sra.itc.it/tools/FSAP/*

[53] J. Bechta, B. Dugan, K-J. Sullivan, and D. Coppit. Developing a High-Quality Software Tool for Fault Tree Analysis. In Proceedings of the International Symposium on Software Reliability Engineering, pages 222-31, Boca Raton, Florida, 1-4 November 1999.

[54] K-J. Sullivan, J-B. Dugan, and D. Coppit. The Galileo Fault Tree Analysis Tool. In Proceedings of the 29th Annual International Symposium on Fault-Tolerant Computing, pages 232-5, Madison, Wisconsin, 15-18 June 1999.

[55] J. Bechta, B. Dugan, B. Venkataraman, and R. Gulati. DIFTree: A Software Package for the Analysis of Dynamic Fault Tree Models. In Annual Reliability and Maintainability Symposium 1997Proceedings, Philadelphia, PA, 13-16 January 1997.

[56] Y. Papadopoulos and J-A. McDermid, HiP -HOPS: Hierarchically Performed Hazard Origin and Propagation Studies. In SAFECOMP ’99, Toulouse, LNCS 1698, pages 139-152, Sept. 1999.

[57] G. Point. Alta-Rica: Contribution à l’Unification des Méthodes Formelles et de la Sûreté de Fonctionnement. PhD thesis, Université de Bordeaux I, 2000.

[58] A. Arnold, A. Griffault, G. Point and A. Rauzy. The AltaRica Language and Its Semantics. Fundamenta Informaticae, 34:109–124, 2000.

[59] P. Bieber, C. Castel, and C. Seguin. Combination of Fault Tree Analysis and Model Checking for Safety Assessment of Complex System. In *Proceedings Fourth European Dependable Computing Conference (EDCC4). Toulouse (France),* pages pp 19–31, October 2002.

[60] A. Garro and A. Tundis, Enhancing the RAMSAS Method for System Reliability Analysis: An Exploitation in the Automotive Domain, Proceedings of the 2nd International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2012), Rome, Italy, 28-31 July 2012.

[61] A. Garro and A. Tundis, Modeling and Simulation for System Reliability Analysis: The RAMSAS Method, Proceedings of the 7th IEEE International Conference on System of Systems Engineering (IEEE SoSE 2012), Genoa, Italy, 16-19 July 2012.

[62] A. Garro, J. Groß, M-R. Gen. Richter and A. Tundis, Experimenting the RAMSAS Method in the Reliability Analysis of an Attitude Determination and Control System (ADCS), Proceedings of the Int. Workshop on Applied Modeling and Simulation (WAMS 2012), jointly held with the NATO CAX FORUM, Rome, Italy, 24-27 September 2012.

[63] A. Garro, J. Groß, M. Riestenpatt Gen. Richter and A. Tundis, Reliability Analysis of an Attitude Determination and Control System (ADCS) through the RAMSAS method, Journal of Computational Science, 5(3):439-449, 2014, Elsevier B.V., Amsterdam, The Netherlands.

[64] A. Garro and A. Tundis, Enhancing the RAMSAS Method for Systems Reliability Analysis through Modelica, Proceedings of the 7th Workshop on Model-Based Product Development (MODPROD 2013), Linköping University, Sweden, 5-6 February 2013.

[65] A. Garro and A. Tundis, RAMSAS4Modelica: a Simulation-driven Method for System Dependability Analysis Centered on the Modelica Language and Related Tools, Proceedings of the Symposium On Theory of Modeling and Simulation (TMS) at SpringSim 2014, Tampa, FL, USA, 13-16 April 2014.

[66] A. Garro and A. Tundis, On the Reliability Analysis of Systems and SoS: the RAMSAS Method and Related Extensions, IEEE Systems Journal (IJS), 9(1):232-241, 2015, IEEE Systems Council.

[67] P. David. Contribution à l'Analyse de Sûreté de Fonctionnement des Systèmes Complexes en Phase de Conception: Application à l'Evaluation des Missions d'un Réseau de Capteurs de Présence Humaine. PhD thesis, Université d'Orléans, Novembre 2009.

[68] F. Mhenni, N. Nguyen and J.-Y. Choley, SafeSysE: A Safety Analysis Integration in Systems Engineering Approach, IEEE Systems Journal, vol. 12, pp. 161–172, 3 2018.

[69] M. Hecht, C. Vogl, and A. Lam, Application of the Architectural Analysis and Design Language (AADL) for Qualitative System Reliability and Availability Modeling, Aerotech, Seattle, WA, November 2009.

[70] M. Batteux, T. Prosvirnova and A. Rauzy, System Structure Modeling Language (s2ml), HAL, 2015.

[71] A. Rauzy, Model-Based Safety Assessment with Altarica 3.0, European Safety and RELiability conference, Portoroz, Slovenia, June 2017.

[72] F. Mhenni, J-Y. Choley and N. Nguyen, An Integrated Design Methodology for Safety Critical Systems, 2016 Annual IEEE Systems Conference (SysCon), pp. 1–6, April 2016.

[73] N. Yakymets, H. Jaber and A. Lanusse, Model-Based System Engineering for Fault Tree Generation and Analysis, In Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2013), Barcelona, Spain, pp. 210–214, Feb 2013.

[74] Z. Zhao and P. Dorina, UML Model to Fault Tree Model Transformation for Dependability Analysis, Proceedings of the International Conference on Computer and Information Science and Technology Ottawa, Ontario, Canada, no. 127, May 2015.

[75] A. Joshi, S. Vestal and P. Binns, Automatic Generation of Static Fault Trees from AADL Models, DSN Workshop on Architecting Dependable Systems, Edinburgh International Conference Centre, Edinburgh, UK, June 2007.

[76] Y. Papadopoulos and M. Maruhn, Model-Based Synthesis of Fault Trees from Matlab-Simulink Models, Proceedings of the International Conference on Dependable Systems and Networks, Goteborg, Sweden, pp. 77–82, July 2001.

[77] B. Kaiser, P. Liggesmeyer and O. Mackel, A New Component Concept for Fault Trees, Proceedings of the 8th Australian workshop on Safety critical systems and software, Darlinghurst, Australia, vol. 33, no. January, pp. 37–46, 2003.

[78] K. Hofig, M. Zeller and R. Heilmann, ALFRED: A Methodology to Enable Component Fault Trees for Layered Architectures, Proceedings- 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2015, pp. 167–176, 2015.

[79] S. Li and X. Li, Study on Generation of Fault Trees from Altarica Models, 3rd International Symposium on Aircraft Airworthiness, ISAA 2013.

[80]A. Rauzy, Mode Automata and Their Compilation into Fault Trees, Reliability Engineering and System Safety, vol. 78, no. 1, pp. 1–12, 2002.

[81] G. Pai and J. Dugan, Automatic Synthesis of Dynamic Fault Trees from UML System Models, 13th International Symposium on Software Reliability Engineering, Annapolis, MD, USA, pp. 243–254, 2002.

[82] J. Dehlinger and J-B. Dugan, Analyzing Dynamic Fault Trees Derived from Model-Based System Architectures, Nuclear Engineering and Technology, vol. 40, no. 5, pp. 365–374, 2008.

[83] F. Tajarrod and G. Latif-Shabgahi. A Novel Methodology for Synthesis of Fault Trees from MATLAB-Simulink Model. World Academy of Science, Engineering and Technology, 17(5):1256–1262, 2008.

[84] J. Castet, M. Bareh, J. Nunes, S. Okon, L. Garner, M. Chacko and M. Izygon, Failure Analysis and Products in a Model-Based Environment, 2018 IEEE Aerospace Conference, Big Sky, MT, 2018.

[85] A. Brameret, P-A. Rauzy, and M. Roussel, Automated Generation of Partial Markov Chain from High Level Descriptions, Reliability Engineering and Systems Safety, December 2015.

[86] F-P. Brooks, No silver bullet: Essence and Accidents of Software Engineering. IEEE Comput. 20(4), 10–19 (1987)

[87] J-C. Laprie, Dependable Computing and Fault Tolerance: Concepts and Terminology. In: Proceedings of 15th International Symposium on Fault-Tolerant Computing (FTSC-15), pp. 2–11 (1985)

[88] V. Neumann, Probabilistic Logics and Synthesis of Reliable Organisms from Unreliable Components. In: Shannon, C., McCarthy, J. (eds.) Automata Studies, pp. 43–98. Princeton University Press, Princeton (1956)

[89] A. Avižienis, Fault-Tolerant Systems. IEEE Trans. Comput. 25(12), 1304–1312 (1976)

[90] E. Dubrova, Fault-Tolerant Design, Springer, New York 2013.

[91] E. Moore and C. Shannon, Reliable Circuits Using Less Reliable Relays. Inst. 262(3), 191–208 (1956)

[92] J. Losq, Influence of Fault-Detection and Switching Mechanisms on the Reliability of Standby Systems. In: Digest 5th International Symposium on Fault-Tolerant Computing, pp. 81–86 (1975)

[93] L-A. Ferrara, Summary Description of the AAP Apollo Telescope Mount. Technical Report TM-68-1022-3, National Aeronautics and Space Administration (NASA) (1968)

[94] US Department of Defense. Procedure for Performing a Failure Mode, Effects and Criticality Analysis, November 1980. MIL-STD-1629A.

[95] R. Banach and M. Bozzano. Retrenchment, and the Generation of Fault Trees for Static, Dynamic and Cyclic Systems. In Janusz Gorski, editor, Computer Safety, Reliability, and Security. Proceeding of the 25th International Conference, SAFECOMP, volume 4166 of Lecture Notes in Computer Science, pages 127–141. Springer Berlin Heidelberg, Gdansk, Poland, September 27-29, 2006.

[96] M. Walker, L. Bottaci, and Y. Papadopoulos, Compositional Temporal Fault Tree Analysis, Proceedings of the 26th International Conference on Computer Safety, Reliability and Security (SAFECOMP'07), pp. 106– 119, 2007.

[97] J-B. Dugan, S. Bavuso and M. Boyd, Dynamic Fault Tree Models for Fault Tolerant Computer Systems, IEEE Transactions on Reliability, vol. 41, no. 3, pp. 363–377, Sep 1992.

[98] Pragmadev, *www.pragmadev.com/2016_SurveyFullResult.html*

[99] OpenFTA, *http://www.openfta.com/*

[100] N. Nguyen, F. Mhenni and J-Y.Choley, Redundancy Handling with Model-Based Systems Engineering, 26th European Safety and Reliability Conference (ESREL 2016), Glasgow, Scotland, September 2016.

[101] OpenAltaRiaca, Water Supply System.

[102] ALD, Fault Tree Analyzer, web-based version, *http://www.fault-tree-analysis-software.com/*

[103] F. Arnold, A. Belinfante, F-V-D. Berg, D. Guck and M. Stoelinga, DFTCalc: A Tool For Efficient Fault Tree Analysis.

[104] M. Delic, S. Ilic, J. Glisovic and D. Catic, Dynamic Fault Tree Analysis of Lawnmower, 9th International Quality Conference, June 2015.

[105] ITEM TOOLKIT, http://www.itemsoft.com/fault_tree.html

[106] Isograph FaultTree+, *www.isograph.com/software/reliability-workbench/fault-tree-analysis/*

[107] K. Ding, A. Morozov and K. Janschek, Classification of Hierarchical Fault-Tolerant Design Patterns, 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, 2017.

[108] M. Abouei Ardakan and A. Zeinal Hamadani, Reliability Optimization of Series-Parallel Systems with Mixed Redundancy Strategy in Subsystems, Reliability Engineering and System Safety, vol. 130, pp. 132–139, Oct 2014.

[109] No Magic, Cameo Systems Modeler, USER GUIDE 18.1; 2015.

[110] S. Epstein and A. Rauzy, Open-PSA Model Exchange Format, May 2008.

[111] A. Rauzy, XFTA: An Open-PSA Fault Tree Engine, October 2015.

[112] D. Mami. Définition, Conception et Expérimentation de Structures d'Actionneurs Electromécaniques Innovants Incluant par Conception des Fonctionnalités de Sûreté et de Sécurité de Fonctionnement. PhD thesis, Université de Toulouse, Institut National de Polytechnique de Toulouse, 2010.

[113] Wikipédia, www.wikipedia.org

[114] S. kabir, M. Walker and Y. Papadopoulos, Dynamic System Safety Analysis in HiP-HOPS with Petri Nets and Bayesian Networks, Safety science, vol 105, pp 55-70, 2018.

Annexe

7.1 Introduction

Dans cette partie, on expose les méthodes utilisées dans le rapport comme FMEA et FTA puis on présente une étude comparative sur les outils d'analyses des arbres de défaillances. Tout d'abord, la méthode FMEA est décrite en indiquant la signification des différents paramètres. Egalement, la méthode FTA est décrite en précisant les différents événements et portes logiques qui constituent un arbre de défaillance statique et dynamique. Finalement, une étude comparative entre les outils d'analyse des arbres de défaillances gratuits, open source et commerciaux est présentée. Cette étude vise à préciser l'efficacité des outils gratuits et open source contre les outils commerciaux.

7.2 Analyse des modes de défaillances et de leurs effets (FMEA)

L'analyse des modes de défaillance et de leurs effets (FMEA) est un outil d'analyse de la fiabilité et des modes de défaillances d'un système. C'était l'une des premières techniques systématiques d'analyse des défaillances. Elle a été créée dans les années 1950 par des ingénieurs de fiabilité afin de mieux étudier et prévenir les conséquences des dysfonctionnements ou/et des défaillances des systèmes militaires et d'évaluer l'impact de ces défaillances sur la fiabilité des systèmes. Ensuite, elle a été utilisée dans plusieurs autres domaines et elle est devenue une pratique courante pour la plupart des projets.

FMEA est une méthode d'évaluation ascendante qui peut être appliquée à n'importe quel niveau de conception. Elle est applicable aux fonctions, aux composants, aux systèmes ou sous-systèmes. FMEA est utilisée, généralement, dans l'analyse de la partie matérielle et des processus mais elle est utilisée aussi dans l'analyse de la partie logicielle du système afin d'évaluer les effets des modes de défaillances de ses fonctions.

Le but de l'analyse des FMEA est l'identification des conditions de fiabilité qui sont potentiellement inacceptables. Cela nécessite une compréhension approfondie des fonctions et de la conception du système. Par conséquent, pour une bonne analyse des modes de défaillances, la FMEA doit être réalisée par une équipe multidisciplinaire qui comprend au minimum des experts de sûreté et des concepteurs des systèmes. Si des risques inacceptables sont identifiés, une liste des actions correctives pour éliminer ou réduire ces risques est fournie.

Ces actions correctives impliquent de nouvelles itérations dans le processus de conception du système pour la modifier et éliminer ou réduire les risques. L'exécution de la FMEA nécessite du temps et des ressources humaines. Pour être efficaces, ces efforts doivent être

effectués le plus tôt possible au cours du processus de conception afin d'influencer les choix de conception en temps réel et éviter les modifications tardives et coûteuses. Si les problèmes ne sont pas découverts et résolus suffisamment tôt, ils pourraient être découverts à des stades très avancés, comme la phase de production. Plus les actions correctives sont tardives, plus elles sont compliquées et coûteuses.

La procédure de création de FMEA d'un système suit les 5 étapes suivantes [40] :

1. Les éléments et la structure du système ;
2. Les fonctions et les structures fonctionnelles ;
3. Les analyses de défaillances ;
4. L'évaluation des risques ;
5. L'optimisation.

La première étape consiste à définir le système à analyser puis le diviser en sous-systèmes, des groupes de fonctions et des composants. La deuxième étape permet la définition des fonctions du système et l'allocation de chaque fonction à son composant approprié. La troisième étape commence par la détermination d'une liste des modes de défaillances potentiels, des causes de défaillances et les effets de ces défaillances. La quatrième étape permet l'évaluation de la criticité des modes de défaillances d'un composant. Cette évaluation passe par l'analyse des trois facteurs suivants : la Sévérité (S), l'Occurrence (O) et la Détectabilité (D). Aussi, le RPN donne une visibilité sur la criticité pour en servir, peut-être, pour prendre des décisions d'optimisation sur la structure du système. La cinquième partie est celle où on utilise l'analyse qui a été faite pour effectuer des actions d'optimisations comme des actions d'empêchement des causes de défaillances pour réduire l'occurrence des défaillances comme l'implémentation de la redondance. Aussi, on peut avoir des actions qui augmentent la probabilité de détection des défaillances par l'implémentation d'un module de diagnostic par exemple.

On remarque l'existence des FMEA fonctionnelles et des FMEA des composants. Dans une FMEA fonctionnelle, l'analyse se concentre sur les différentes fonctions du système utilisé pour achever sa mission. Le point de départ d'un FMEA fonctionnel est l'architecture fonctionnelle du système définie par les concepteurs. L'architecture fonctionnelle décrit la liste des différentes fonctions que le système doit exécuter pour remplir sa mission, ainsi que les flux d'entrée et de sortie de chaque fonction. Par conséquent, il décrit comment les entrées externes

du système sont progressivement transformées en sorties. Ensuite FMEA analyse les modes de défaillances potentiels de chaque fonction afin d'identifier les causes et les effets de chaque mode de défaillance. La liste des fonctions est insérée dans la FMEA. Pour chaque fonction, les modes de défaillance potentiels sont identifiés. Ensuite, pour chaque mode de défaillance, les causes et les effets sont déterminés, puis la criticité du mode de défaillance est évaluée. Si le mode de défaillance a des effets négatifs au niveau du système, l'expert de sûreté tente d'identifier une liste d'actions correctives. Cette liste est priorisée en fonction de la criticité des modes de défaillance.

La FMEA composant se concentre sur les composants du système plutôt que sur ses fonctions. Elle est effectuée lorsque le processus de conception est plus avancé et lorsque les composants du système sont déjà définis. Elle est liée à FMEA fonctionnelle et doit être compatible avec la dernière. En effet, les composants sont affectés aux fonctions du système, leurs modes de défaillance correspondent aux modes de défaillance fonctionnels mais ils sont expliqués plus en détail en tenant compte des aspects physiques. La sémantique de FMEA composant est assez similaire à celle de la FMEA fonctionnelle. La différence entre les deux est que FMEA composant est basée sur les composants plutôt que sur les fonctions. Dans la FMEA composant, les colonnes sont légèrement différentes de la FMEA fonctionnelle et, par exemple, la colonne «Composant» est ajoutée.

On remarque, aussi, l'existence d'une version plus détaillée de FMEA qui permet l'analyse des modes de défaillance, de leurs effets et de leurs criticités (FMECA). La FMECA est similaire à FMEA, avec une différence qui est l'ajout d'une évaluation de la criticité pour chaque mode de défaillance. La FMECA est basée sur trois facteurs:

- La Sévérité (S) qui indique la gravité des conséquences des défaillances ;
- L'Occurrence (O) qui indique la fréquence de défaillances du système ou des sous-systèmes ;
- La Détection (D) qui donne une idée sur la facilité avec laquelle les défaillances peuvent être détectées.

Ensuite, on définit le nombre de priorités de risque qui est calculé en multipliant ces trois facteurs. Il est considéré comme un paramètre de fiabilité qui permet l'identification des composants critiques et ensuite il aide à la hiérarchisation des actions correctives.

$$RPN = S \times O \times D$$

Dans FMECA, plusieurs colonnes sont utilisées dont de nombreux vocabulaires nécessitent des définitions comme les modes de défaillances, les causes des défaillances, les effets des défaillances, les méthodes de détection des défaillances et les actions correctives.

- ## Les modes de défaillances

Un mode de défaillances est défini par la manière de défaillance d'un élément. C'est le mode ou l'état dans lequel se trouve l'élément après sa défaillance [37].

La première étape de la réalisation de FMEA est l'identification des différents modes de défaillance de chaque élément. Les modes de défaillance peuvent être déterminés de différentes manières. Ils peuvent être déterminés en utilisant une base des données existante fournie par l'expérience acquise sur des produits similaires. Certaines normes peuvent collecter de telles données à partir de l'expérience de plusieurs industriels et fournir une liste des fonctions classifiées avec leurs modes de défaillance respectifs. Lorsque ce type de données est disponible, elles peuvent être très utiles car elles permettent d'éviter certaines erreurs. Mais, ce n'est pas toujours le cas, surtout pour les produits totalement nouveaux. Dans ce cas, la connaissance de l'expert de sûreté est la seule source d'informations sur laquelle la réalisation de FMEA peut s'appuyer. Aussi, les normes fournissent des règles qui permettent la génération systématique des modes de défaillance. En effet, au niveau fonctionnel, les modes de défaillance sont assez génériques et peuvent être résumés dans la liste suivante [67], [94]:

- La fonction n'arrive pas à s'exécuter ;
- La fonction s'exécute mais sans avoir les performances exigées ;
- la fonction est intermittente ;
- la fonction est arrêtée pendant son exécution ;
- la fonction n'est pas exécutée dans le temps exigé ;
- La fonction devrait s'arrêter mais elle fonctionne toujours ;
- Autres modes de défaillance propres à la fonction.

- ## Les causes de défaillances

Une cause de défaillance est un processus responsable du lancement du mode de défaillance. Les processus possibles pouvant entraîner une défaillance de fonctionnement sont les défaillances physiques, les défauts de fabrication, les défauts de conception, etc. [37].

- ## Les effets de défaillances

Les effets de défaillances sont les conséquences d'un mode de défaillance sur le fonctionnement du système, la fonction à accomplir ou l'état d'un élément du système [37].

- **Les méthodes de détection**

Cette partie identifie comment un mode de défaillance spécifique peut être détecté après son apparition et avant toute conséquence grave.

- **Les actions correctives**

Cette partie identifie les méthodes qui permettent l'élimination ou la réduction des effets du mode de défaillance potentielle. Plusieurs actions peuvent être prises comme l'ajout d'un composant redondant pour augmenter la fiabilité et assurer le bon fonctionnement du système.

7.3 Analyse des arbres de défaillances (FTA)

Dans cette section, les concepts et les utilisations de l'analyse d'arbre de défaillances (FTA) sont présentés. Les arbres de défaillances sont largement utilisés pour l'évaluation de la sûreté de fonctionnement et de la fiabilité des systèmes [83]. FTA est une technique déductive utilisée dans les analyses de sûreté. Partant d'un état non souhaité, FTA vise à identifier toutes les causes possibles de cet état de défaillance par une analyse descendante de l'évènement sommet de l'arbre de défaillance jusqu'à atteindre les évènements feuilles. Le résultat de l'analyse est un graphique qui a la forme d'un arbre et qui s'appelle ''arbre de défaillances''. Un arbre de défaillance décrit les différents chemins dans un système qui mène à un événement de défaillance potentielle non désirée. Ces chemins lient les événements et les conditions qui contribuent à l'événement indésirable du système à l'aide des portes logiques standards. On peut utiliser des portes logiques statiques ainsi que des portes logiques dynamiques. Les arbres de défaillances avec seulement des portes logiques statiques sont des arbres de défaillances statiques et décrivent la défaillance de la structure du système. Par contre, les arbres de défaillance qui utilisent des portes logiques statiques et dynamiques sont des arbres de défaillances dynamiques et elles décrivent le comportement du système et prennent en compte l'ordre de défaillance des évènements (cas de la redondance par exemple).

Un arbre de défaillances peut être traduit en un modèle mathématique pour calculer les probabilités de défaillance dans le cadre d'une analyse quantitative. Ce modèle exprime les relations entre la probabilité des événements intermédiaires et haut niveau et les probabilités des événements feuilles. Pour chaque porte logique, une formule mathématique donne la

relation entre la probabilité de l'événement de sortie et les probabilités des événements d'entrée de la porte logique. A la fin de l'analyse, la probabilité de l'événement haut niveau du modèle est déterminée.

De plus, l'arbre de défaillance peut être analysé d'une manière qualitative pour identifier les points faibles de la structure et du comportement du système. Ceci est obtenu en identifiant les combinaisons nécessaires et suffisantes d'événements feuilles qui provoquent la défaillance de l'événement haut niveau. Ces combinaisons nécessaires et suffisantes sont appelées ensemble de coupes minimales.

7.3.1 Type d'évènements des arbres de défaillances

L'arbre de défaillance peut contenir trois différents types d'événements qui sont l'événement de haut niveau, les événements intermédiaires et les événements feuilles comme l'indique la Figure A.1.

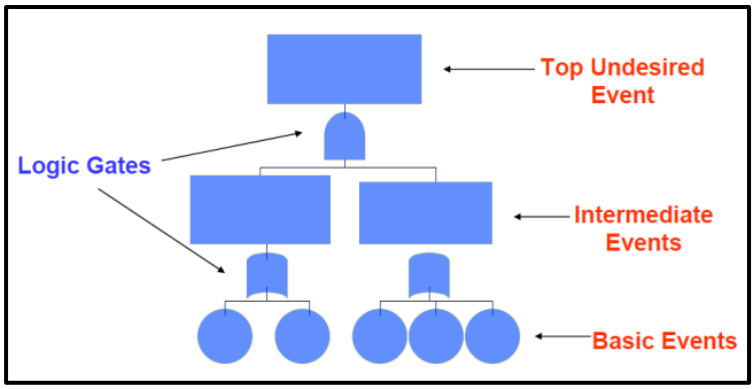


Figure A. 1: Structure de l'arbre de défaillance

Différents types d'événements feuilles peuvent être utilisés dans un arbre de défaillances comme les évènements de base, les événements non-développés et les évènements maison et les évènements de conditionnement. Pour cela, chaque type d'évènement a une forme géométrique spécifique. Les symboles des différents évènements sont donnés ci-dessous.

- Un événement de base est un évènement représenté par un défaut de base qui ne nécessite aucun développement supplémentaire. Le symbole de l'événement de base est donné dans la Figure A.2.

Figure A. 2: Symbole de l'évènement de base

- Un événement non-développé est un événement qui n'est pas développé car il a une conséquence insuffisante ou il a des informations indisponibles. Le symbole d'événement non développé est donné dans la Figure A.3.

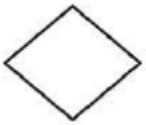


Figure A. 3: Symbole d'un évènement non-développé

- Un événement maison est défini par un évènement non probabilisé car il doit prendre 0 ou 1 avant le traitement de l'arbre. Cet évènement permet d'avoir plusieurs variantes d'arbre de défaillances dans un seul schéma selon la valeur affectée à l'évènement par l'utilisateur. Le symbole de l'évènement maison est donné dans la Figure A.4.

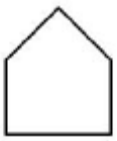


Figure A. 4: Symbole de l'évènement maison

- Un événement de conditionnement est un évènement qui applique une condition ou une restriction spécifique sur les portes logiques. Le symbole d'événement de conditionnement est donné dans la Figure A.5.

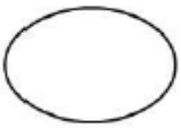


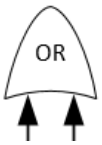
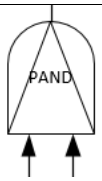
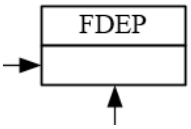
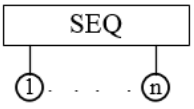
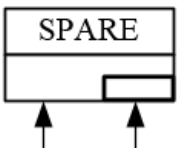
Figure A. 5: Symbole de l'évènement de conditionnement

7.3.2 Portes logiques des arbres de défaillances

Différentes portes logiques peuvent être utilisées dans un arbre de défaillances pour lier les événements qui mènent à l'événement sommet. Les portes logiques décrivent la combinaison logique des événements d'entrée qui mènent à l'événement de sortie. On distingue deux types

de portes logiques : statiques et dynamiques. Les principales portes logiques sont décrites dans le Tableau 9.

Tableau 9: Portes logiques statiques et dynamiques

Porte logique	Type	Description	Symbole
AND	Statique	Elle a deux entrées ou plus et une sortie. L'événement de sortie se produit si toutes les entrées sont vraies.	
OR	Statique	Elle a deux entrées ou plus et une sortie. L'événement de sortie se produit si au moins une des entrées est vraie.	
VOTING (K/N)	Statique	Elle a deux entrées ou plus et une sortie. L'événement de sortie se produit si au moins K entrées sur N sont vraies.	
PAND (Priority AND)	Dynamique	Elle a deux entrées ou plus et une sortie. L'événement de sortie se produit si toutes les entrées sont vraies de gauche à droite [96].	
FDEP (Functional DEPendency)	Dynamique	Elle possède deux entrées ou plus et une entrée de déclencheur (Trigger). Si le déclencheur est activé, tous les événements dépendants seront vrais [97].	
SEQ (SEQuence)	Dynamique	Elle a deux entrées ou plus et une sortie. Cette porte logique force les entrées à suivre un ordre particulier de défaillance. La sortie sera vraie quand toutes les entrées sont vraies dans l'ordre particulier.	
SPARE	Dynamique	Elle possède une entrée principale et une ou plusieurs entrées secondaires et une seule sortie. Si l'entrée principale est vraie, alors la deuxième entrée sera activée et ainsi de suite. L'événement de sortie se produit si toutes les entrées sont vraies.	

7.4 Outils d'analyse des arbres de défaillances

Afin d'analyser les arbres de défaillances, des outils d'analyses sont utilisés. On distingue deux types d'outils que sont les outils d'analyses gratuits et open source et les outils commerciaux. Ces outils permettent l'analyse qualitative et/ou quantitative des arbres de défaillances statiques et/ou dynamiques. Cette partie est consacrée à l'étude et à la comparaison entre les outils d'analyse gratuits et open source et les outils d'analyse commerciaux.

Cette partie présente une étude approfondie de certains outils gratuits et open source pour l'analyse d'arbre de défaillances. Une étude comparative a été faite pour quatre outils différents. Premièrement, Open FTA qui est un outil d'analyse des arbres de défaillances open source basée sur le moteur de calcul XFTA. Deuxièmement, Open AltaRica qui est un outil gratuit qui permet de générer des arbres de défaillances des systèmes complexes. Troisièmement, ALD Fault Tree Analyzer qui est un outil Web gratuit qui permet l'analyse des arbres de défaillances statiques. Enfin, DFTCalc est un outil open source qui permet l'analyse des arbres de défaillances dynamiques en utilisant des techniques de vérification de modèle stochastique. Pour faire des comparaisons, on a étudié aussi des outils d'analyses commerciaux qui proposent des périodes d'essai comme Isograph Fault Tree+ et Item Toolkit.

Le choix de l'étude des outils gratuits et open source et la comparaison avec des outils commerciaux est conforme à la tendance actuelle. En effet, selon l'enquête PragmaDev 2016, 58% des entreprises ont pour politique d'utiliser des outils open source [98]. Cela rendra les outils FTA plus accessibles aux utilisateurs des outils d'analyses.

Afin de comparer ces outils, trois exemples représentatifs sont utilisés. Le premier exemple modélisé par OpenFTA, ALD Fault Tree Analyzer et Isograph Fault Tree ++, est un actionneur électromécanique (EMA) pour les ailerons d'un avion. Le second exemple modélisé par DFTCalc, est une tondeuse à gazon contrôlée à distance. Le troisième exemple modélisé par OpenAltaRica est un exemple de génération d'arbre de défaillances à partir d'un code AltaRica.

7.4.1 OpenFTA

OpenFTA [99] est un produit open source graphique développé par ''Formal Software Construction'' qui permet l'analyse des arbres de défaillances. Cet outil offre plusieurs avantages, tels qu'une interface utilisateur avancée offrant une vue globale, une arborescence de gestion simple et une présentation arborescente optimisée. Il possède une base de données d'événements qui peuvent être utilisée. En utilisant la simulation de Monte Carlo, OpenFTA

propose des analyses qualitatives et quantitatives. Cet outil supporte différentes plates-formes telles que Microsoft Windows (2000, XP, 7 ...) et UNIX (Linux, Sun, etc.).

Open FTA propose une variété de portes logiques statiques tel que les portes «AND», «OR», «NOR», «VOTING (K / N)» et «NAND». Il utilise des lois de probabilité telles que loi «constante», «exponentielle», «Weibull», etc.

On a utilisé OpenFTA pour créer un arbre de défaillances pour un moteur électromécanique (EMA) [100]. La structure interne de l'EMA est illustrée dans la Figure 2.10 à l'aide du diagramme SysML (IBD). Il contient deux microcontrôleurs intégrés EMC1 et EMC2 et deux moteurs M1 et M2. Chaque moteur est constitué de deux enroulements séparés W1 et W2 recevant des signaux d'entrés respectivement de EMC1 et EMC2. Chaque moteur a également un rotor qui est lié aux deux enroulements. Une transmission mécanique fournit alors le mouvement de sortie à l'aileron.

L'arbre de défaillances de l'EMA, construit en utilisant l'outil OpenFTA, est présenté dans la Figure A.6. Cet arbre de défaillances statique peut également être généré automatiquement à partir du modèle SysML, comme expliqué dans [100]. Cet arbre est composé d'événements, de portes logiques« OR » et « AND ». Les portes « AND » représentent la redondance dans le système. La simulation d'OpenFTA permet d'avoir un rapport qui contient l'analyse qualitative et quantitative du système. Il existe 13 ensembles de coupes minimales avec des ordres allant de 1 à 4. Certaines mesures de probabilité sont calculées, telles que la probabilité de défaillances de l'évènement redouté, le temps moyen avant défaillance (en anglais : Mean Time To Failure (MTTF)), le temps moyen entre les défaillances (en anglais : Mean Time Between Failure (MTBF)) et le temps moyen de réparation (en anglais : Mean Time To Repaire (MTTR)).

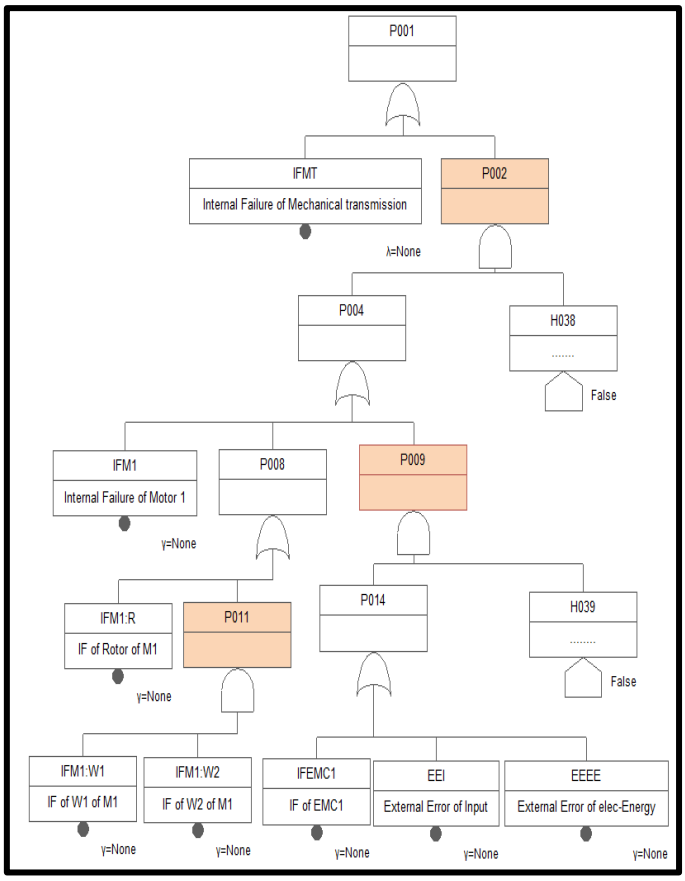


Figure A. 6: Arbre de défaillance de l'EMA à l'aide d'OpenFTA

7.4.2 OpenAltaRica

La plate-forme OpenAltaRica [101] est basée sur le langage de modélisation AltaRica 3.0. Cette plate-forme offre des outils open-source tels qu'ARC et AltaRica Studio. ARC est un outil open source qui vérifie la conformité des modèles AltaRica à l'aide des algorithmes de vérification des modèles. AltaRica Studio est un petit outil graphique utilisé avec ARC qui vise à valider les modèles AltaRica et à offrir un accès rapide aux principaux calculs proposés par le vérificateur de modèles d'ARC.

Pour générer un arbre de défaillance à partir du modèle AltaRica [79], il faut passer par plusieurs étapes. Le modèle AltaRica doit être traduit en logique de défaillance (Failure Logic) puis l'arbre de défaillance doit être généré par une analyse de la logique de défaillance. Finalement, une comparaison entre le code AltaRica et l'arbre de défaillance généré à partir de la même logique de défaillance doit être faite comme illustré à la Figure A.7.

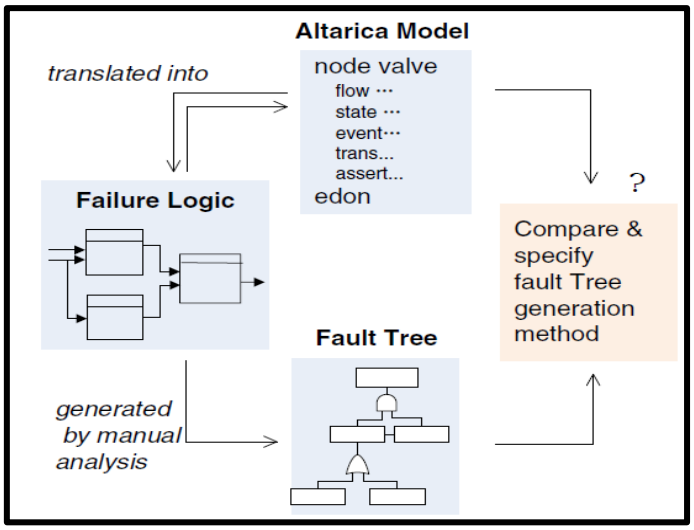


Figure A. 7: Génération des arbres de défaillances à partir d'AltaRica

Selon Li et al. [79], trois étapes sont nécessaires pour générer des arbres de défaillances à partir d'un code AltaRica. Tout d'abord, il faut identifier l'événement redouté du système. Après, il faut définir les événements bas niveau avec les variables d'entrées de chaque composant. Finalement, il faut définir les portes logiques qui décrivent les relations entre les événements hauts et bas niveaux.

Dans le cadre de l'étude de cet outil, un simple exemple d'un composant, cité dans [79], est illustré dans la Figure A.8. A et B sont deux variables d'entrée, X est la variable de sortie et l'événement sommet de l'arbre de défaillance du composant.

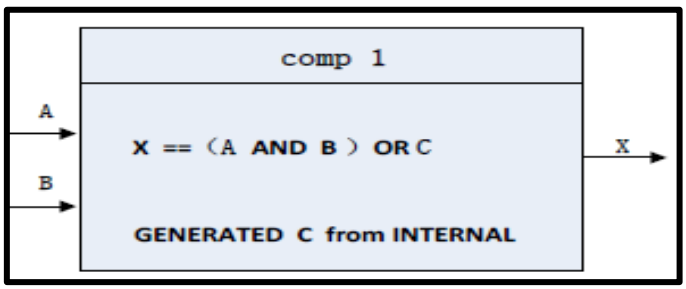


Figure A. 8: Exemple de la logique de défaillance d'un composant

Le code qui décrit le comportement dysfonctionnel du composant 1 est présenté dans la Figure A.9. Il suit les trois étapes déjà mentionnées. La variable X est définie comme un flux de sortie qui représente l'événement redouté de l'arbre de défaillance. Ensuite, les événements de bas niveau sont définis par les variables d'entrée qui sont A et B. Enfin, la définition des portes logiques après la description de la relation entre les évènements de haut et bas niveau.

```
node comp1
 flow
  A : default_nominal_failed : in;
  B : default_nominal_failed : in;
  X : default_nominal_failed : out;
 state
  C : {nominal,failed} ;
 event
  fail_C_nominal_failed;
 init
 C := nominal;
 trans
  ((C = nominal)) |- fail_C_nominal_failed -> C := failed;
 assert
  X = if ((((A=failed) and (B=failed)) or (C=failed))) then failed else nominal;
edon
```

Figure A. 9: Exemple d'un code AltaRica d'un composant

Le résultat de cette méthode est la génération d'un arbre de défaillance comme le montre la Figure A.10. Après l'obtention de l'arbre de défaillances, différents types d'analyse peuvent être établis.

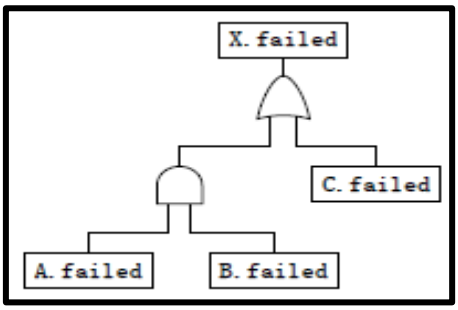


Figure A. 10: Exemple d'un arbre de défaillance d'un composant

7.4.3 ALD Fault Tree Analyzer

ALD Fault Tree Analyzer [102] est un outil en ligne et gratuit qui permet l'analyse d'arbre de défaillance statique. C'est un outil graphique avec une interface utilisateur simple et intuitive. Il offre une taille illimitée de hauteur d'arbre de défaillance avec la possibilité d'exporter ou d'importer des arbres de défaillances vers et depuis un PC. Il donne la possibilité d'importer un arbre de défaillance à partir des outils d'analyse commerciaux tels que « Ram Commander », « Cafta », « Isograph Fault Tree ++ » et « Safety Commander ». De plus, il permet l'exportation des arbres de défaillances sous format image. Il utilise, seulement, trois types de portes logiques: «OR», «AND» et «VOTING (K/N)». Il inclut différents types d'événements tels que les événements à « probabilité constante », « réparables », « non réparables », « évidents » et « externes ».

De même qu'OpenFTA, on a utilisé ALD Fault Tree Analyzer pour créer un arbre de défaillances pour l'exemple du moteur électromécanique présenté dans la Figure A.11. Cet arbre de défaillance statique est composé de plusieurs événements, de portes logiques « OR » et « AND ». De plus, cet outil offre une analyse qualitative (ensemble de coupes minimales) et

une analyse quantitative. Ces résultats sont résumés dans un rapport qui peut être enregistré localement sur le PC.

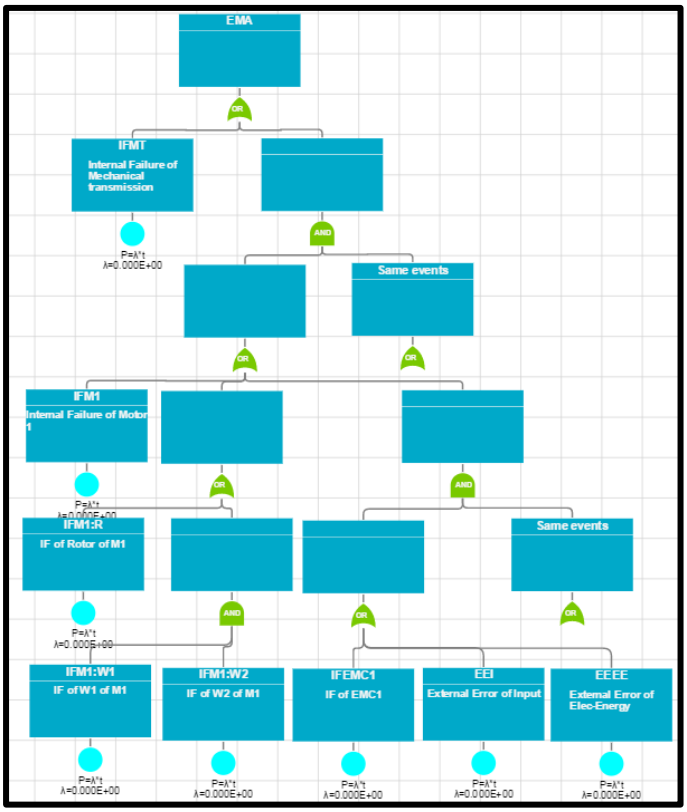


Figure A. 11: Arbre de défaillance de l'EMA à l'aide d'ALD FAULT TREE ANALYZER

7.4.4 DFTCalc

DFTCalc [103] est un outil open source qui assure l'analyse des arbres de défaillances dynamiques. Il prend comme entrée un arbre de défaillances dynamique le langage Galileo avec un choix de quelques paramètres de fiabilité, telles que : a) Le MTTF, qui décrit la durée de travail prévue du système pour avoir la première panne, b) La fiabilité temporelle qui donne la probabilité que le système tombe en panne dans un temps ou un intervalle de temps donné, c) La fiabilité, qui est la probabilité que le système tombe en panne. La sortie est donnée sous forme de graphique ou de chiffres.

On a utilisé DFTCalc pour construire un arbre de défaillance dynamique pour une tondeuse à gazon contrôlée à distance étudiée dans [104]. Il contient un moteur électrique, un coupe herbe, les sous-systèmes de mouvement et du manœuvre. Le sous-système de gazon à couper contient trois lames. Deux roues attachées à l'arbre assurent le mouvement de l'ensemble du système. Les batteries alimentent la tondeuse à gazon en alimentant le moteur électrique. Le

système est contrôlé par une télécommande et le système de manœuvre contient des capteurs, un récepteur de rechange et une manette de jeu.

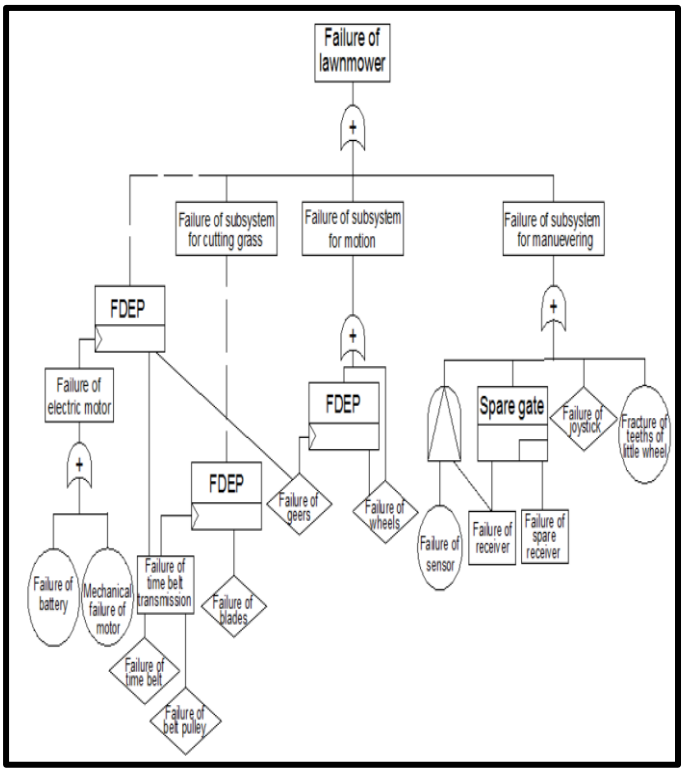


Figure A. 12: Arbre de défaillance dynamique de la tendeuse à gazon contrôlée à distance

L'arbre de défaillance dynamique du système est donné dans la Figure A.12. Il contient l'événement indésirable de défaillance de la tondeuse à gazon. On y trouve les événements intermédiaires, les événements feuilles, les portes logiques statiques comme les portes logiques « OR » et « AND » et les portes logiques dynamiques comme les portes « FDEP », « SPARE » et « PAND ».

```
toplevel "SYSTEM";
"SYSTEM" or "FDEP" "CUTTING" "MOTION" "MANUEVERING";
"FDEP" fdep "ELECTRIC" "TRANSMISSION" "GEERS";
"ELECTRIC" or "BATTERY" "MOTOR";
"CUTTING" fdep "TRANSMISSION" "BLADES";
"MOTION" or "FDEP" "WHEELS";
"FDEP" fdep "GEERS" "WHEELS";
"MANUEVERING" or "PAND" "SPARE" "JOYSTICK" "LITTLE";
"PAND" pand "SENSOR" "RECEIVER";
"SPARE" spare "RECEIVER" "SPARE";
"SPARE" lambda=5.0e-5 dorm=0;
"RECEIVER" lambda=5.0e-5 dorm=0.5;
"JOYSTICK" lambda=2.0e-5 dorm=0;
"LITTLE" lambda=2.0e-5 dorm=0;
"SENSOR" lambda=1.0e-6 dorm=0;
"WHEELS" lambda=1.0e-4 dorm=0;
"GEERS" lambda=1.0e-4 dorm=0;
"BLADES" lambda=1.0e-4 dorm=0;
"MOTOR" lambda=1.0e-4 dorm=0;
"BATTERY" lambda=1.0e-4 dorm=0;
```

Figure A. 13: Extrait du code de DFTCalc pour l'exemple de la tondeuse à gazon avec le langage Galileo

Afin de créer l'arbre de défaillance dynamique avec DFTCalc, on utilise le langage Galileo qui décrit le DFT. Un extrait du code de DFTCalc est présenté dans la Figure A.13. Le résultat de la simulation de DFTCalc peut être un graphique ou des chiffres. La Figure A.14 donne un graphique qui décrit la probabilité d'occurrence de l'évènement indésirable par rapport au temps.

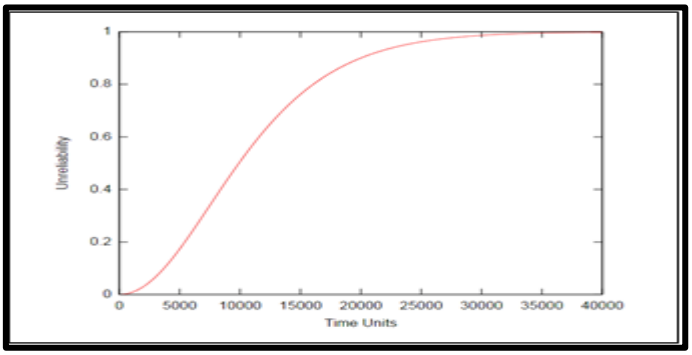


Figure A. 14: Simulation de DFTCalc

7.4.5 Outils commerciaux

Généralement, les outils d'analyse des arbres de défaillances commerciaux fournissent une version d'essais gratuite pour tester le produit. Par exemple, ITEM ToolKit et PTC Windchill FTA proposent une version d'essais gratuite pendant 30 jours et Isograph FaultTree + propose une licence temporaire pour une période de 7 jours seulement. En outre, ALD RAM Commander propose une version de démonstration.

ITEM ToolKit [105] est un outil d'analyse des arbres de défaillances. Il offre une grille avancée et une vue tabulaire. ITEM ToolKit offre une variété d'assistant graphique puissant. Il donne une facilité d'importation et d'exportation des fichiers. Il utilise deux moteurs de calcul basés sur le diagramme de décision binaire et la méthode d'approximation. Il assure l'analyse qualitative et quantitative des arbres de défaillances.

Isograph FaultTree + [106] est un outil d'analyse des arbres de défaillances qui offre un package avec trois modules. Le premier module permet la construction et l'analyse des arbres de défaillances. Ensuite, le deuxième module permet l'analyse des arbres d'événements qui cherche à analyser les résultats suite à l'occurrence d'un événement. Puis, le troisième module offre une analyse de Markov qui est une méthode d'analyse de fiabilité et de disponibilité des systèmes complexes. Il autorise, aussi, la construction des modèles de Markov avec une forte interdépendance. Afin d'illustrer cet outil, le même exemple (EMA) que OpenFTA et ALD Fault Tree Analyzer est étudié avec Isograph Fault Tree ++ et la Figure 3.15 donne l'arbre de défaillance obtenue.

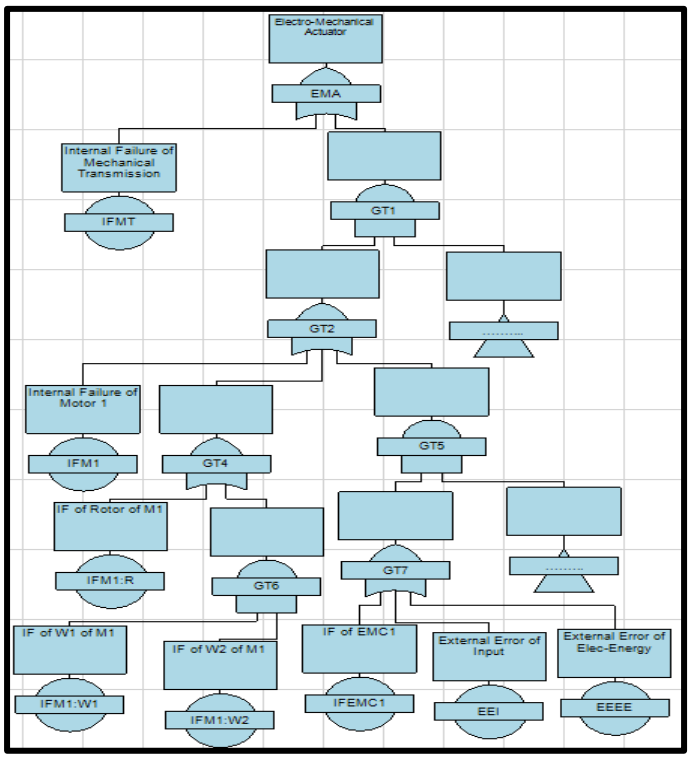


Figure A. 15: Arbre de défaillance de l'EMA avec Isograph FaultTree +

7.4.6 Comparaison

Le Tableau 10 donne une comparaison entre les différents outils d'analyses des arbres de défaillances statiques et dynamiques (gratuits, open source et commerciales). Différents

critères sont utilisés tels que le type des arbres de défaillances, type des portes logiques (statiques et dynamiques), la méthode de l'analyse, la facilité d'importation et d'exportation, etc...

OpenFTA et OpenAltaRica, deux outils open source et gratuits, garantissent tous les deux une analyse qualitative et quantitative. Des outils commerciaux tels que Isograph Fault Tree ++ et Item Toolkit offrent les mêmes types d'analyse avec des options de plus. Par exemple, OpenFTA propose un certain nombre de lois de probabilité, telles que des lois exponentielles, constantes et de Weibull tandis que Isograph Fault Tree ++ offre un grand nombre de lois de probabilité comme les lois binomiales, de Weibull, de Poisson, etc., avec la possibilité de créer de nouvelles lois.

OpenFTA est un outil facile à utiliser tandis qu'OpenAltarica utilise un langage de programmation spécifique qui nécessite un expert de développement dans le domaine de la sûreté de fonctionnement. DFTCalc utilise également un langage de programmation spécifique, au format Galileo, mais très facile à mettre en œuvre. DFTCalc, qui est un outil open source, offre une analyse quantitative des arbres de défaillances dynamiques. Par contre, les outils d'analyses commerciaux ne fournissent aucune version d'essai gratuite pour l'analyse des arbres de défaillances dynamiques.

L'avantage des outils commerciaux est qu'ils offrent une grande variété de formats de fichiers d'importation et d'exportation et plus de documentations que les outils gratuits et open source.

Les outils d'analyses gratuits, open source et commerciaux ciblent les utilisateurs académiques, les chercheurs et les industriels. Il utilise pratiquement les mêmes portes logiques statiques que sont les portes « OR », « AND », « NOR » et « NAND ». Il offre généralement une interface simple, une importation et exportation faciles des fichiers et une installation facile si besoin car ALD Fault Tree Analyzer et DFTCalc sont deux outils en ligne et qui ne nécessitent, donc, pas une installation.

Tableau 10: Résumé des critères des différents outils

	OpenFTA	OpenAltaRica	ALD Fault Tree Analyzer	Isograph Fault Tree ++	Item Toolkit	DFTCalc
Type de FT	Statique	Statique	Statique	Statique	Statique	Dynamique
Portes logiques statiques	AND, OR, Voting, NOR et NAND	AND, OR, Voting, NOR et NAND	AND, OR et Voting	AND, OR, Voting, NOR, NULL et NAND	AND, OR, Voting, NOR, NULL et NAND	AND, OR, Voting, NOR, NULL et NAND
Portes logiques dynamiques	Pas de portes logiques dynamiques	Pas de portes logiques dynamiques	Pas de portes logiques dynamiques	Pas de portes logiques dynamiques	Pas de portes logiques dynamiques	FDEP, SPARE et PAND
Type d'analyse	Analyse qualitative et quantitative	Analyse qualitative et quantitative	Analyse qualitative et quantitative	Analyse quantitative	Analyse qualitative et quantitative	Analyse quantitative
Méthodes d'analyse	Simulation de Monte Carlo et la méthode algébrique pour la génération des MCS	Simulation de Monte Carlo et la méthode algébrique pour la génération des MCS	Pas trouvée	Analyse des chaines de Markov	Diagramme de décision binaire et méthode d'approximation	Analyse des chaines de Markov
Utilisateurs	Académique, recherche et industriel	Académique, recherche et industriel	Académique, recherche et industriel	Académique, recherche et industriel	Académique, recherche et industriel	Académique, recherche et industriel
Facilité de l'utilisation	Utilisation facile (graphique)	Besoin d'un langage de programmation	Utilisation facile (graphique)	Utilisation facile (graphique)	Utilisation facile (graphique)	Utilisation facile
Format d'importation	Open-PSA	Open-PSA généré automatiquement	De RAM Commander, Cafta, Isograph Fault Tree ++ et Safety Commander	Excel, XML, text, SQL,Server databases, Oracle databases, etc…	Excel, Acess, text, etc…	Format Galileo
Format d'exportation	.HTML	.HTML	.HTML ou .png	Excel, XML, text, SQL, etc...	Excel, Acess, text, etc…	.gzip
OS supporté	Windows (32/64 bit), Linux (32/64 bit), Mac, etc.	Windows (32/64 bit), Linux (32/64 bit), Mac, etc.	Windows (32/64 bit), Linux (32/64 bit), Mac, etc.	Windows (32/64 bit), Linux (32/64 bit), Mac, etc.	Windows (32/64 bit), Linux (32/64 bit), Mac, etc.	Windows (32/64 bit), Linux (32/64 bit), Mac, etc.
Licence	Open source	Gratuit	Gratuit	Période d'essai (7 jours)	Période d'essai (30 jours)	Open source
Installation	Facile	Pas facile	N.A (outil en ligne)	facile	facile	Pas besoin (outil en ligne)

Dans cette section, une comparaison entre les outils gratuits, open source et commerciaux d'analyses des arbres de défaillances statiques et dynamiques a été faite. En conclusion, les outils open source OpenFTA et la plateforme gratuite OpenAltaRica, tous les deux basés sur XFTA, sont des outils complets pour les projets académiques et les petits projets industriels. Pour les grands projets industriels nécessitant un grand nombre de lois de probabilités et un format spécifique d'importation et d'exportation, les outils gratuits et open source restent assez limités et l'utilisation d'outils commerciaux, qui proposent un nombre illimité de lois de probabilités et un grand nombre de format d'importation et d'exportation, est inévitable.

Vu la facilité d'utilisation des outils graphiques d'analyse des arbres de défaillance, on recommande OpenFTA si l'arbre de défaillance doit être créé directement. Par contre, si l'arbre de défaillance doit être généré automatiquement à partir d'un modèle, on recommande OpenAltaRica qui offre un cadre plus générique dans lequel il est possible de concevoir le modèle avec le langage AltaRica, de le simuler et de générer automatiquement des arbres de défaillances. De plus, si on cherche à analyser des arbres de défaillances dynamiques, un outil qui permet ce type d'analyse doit être utilisé. Cependant, aucun outil commercial n'offre une version d'essai pour l'analyse des arbres de défaillance dynamiques, mais l'outil open source DFTCalc qui se base sur les techniques de contrôle de modèle stochastique peut résoudre ce problème et propose une analyse quantitative des arbres de défaillances dynamiques. Étant donné que de grands efforts ont été faits par la communauté open source pour les outils d'analyse d'arbre de défaillances, il convient de promouvoir leur utilisation, ce qui rendra l'analyse d'arbre de défaillances plus accessible.

Publication

- L'acceptation et la publication d'un article intitulé "**Dynamic Fault Tree Generation for Safety Critical Systems within a Systems Engineering Approach"** dans une revue Internationale avec comité de lecture intitule **IEEE Systems Journal.**

- L'acceptation et la publication d'un article intitulé **«Improved Safety Analysis Integration in a Systems Engineering Approach** » dans une revue internationale avec comité de lecture intitulé Applied Science. L'article a été publié le 25 Mars 2019 avec le MDPI publisher.

- L'acceptation et la présentation d'un papier intitulé **« Free and Open Source Fault Tree Analysis Tools Survey »** dans une conférence internationale « **SYSCON » en CANADA (Montréal) du 23 Avril jusqu'à 27 Avril 2017**.

- L'acceptation et la présentation d'un papier intitulé **«Enhanced System Architecture and Behavior Using FMEA Recommendations»** à **CMSM en Tunisie (Hammamet) du 27 Mars jusqu'à 29 Mars 2017**. Le papier propose un aperçu sur la méthodologie d'intégration des modèles basés sur l'ingénierie systèmes et les modèles basés sur la sûreté de fonctionnement.

- L'acceptation et la présentation d'un papier intitulé **«Improved System Architecture and Behavior Based on FMEA Recommendations »** dans une conférence internationale « **ESREL » en Slovénie (ljubljana) du 19 Juin jusqu'à 22 Juin 2017**.

- La participation à un papier intitulé « **Topology-Based Safety Analysis for Safety Critical CPS** » publié à la conférence Complex Adaptative Systems.

université PARIS-SACLAY
ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : Prise en compte de la sûreté de fonctionnement lors du choix d'architectures des systèmes complexes

Mots clés : Ingénierie systèmes, système complexe, redondance, sûreté de fonctionnement, FMEA/FTA

Résumé : Le but de cette thèse est l'intégration de l'analyse de la sûreté de fonctionnement dans une approche d'ingénierie système basée sur des modèles afin d'assurer la cohérence entre la conception du système et les artefacts de sûreté de fonctionnement. Cette intégration permet l'amélioration continue de la structure et du comportement du système. Cela réduit également le temps de développement du système et empêche la détection tardive des erreurs. Pour atteindre cet objectif, la méthodologie SafeSysE est étendue. Dans SafeSysE, une analyse préliminaire du mode de défaillance et des effets (FMEA) est automatiquement générée à partir d'un modèle SysML. Cette analyse FMEA est ensuite complétée par l'expert de sûreté de fonctionnement, mais aucun développement supplémentaire n'est proposé. La contribution de cette thèse est de suggérer des recommandations basées sur l'analyse FMEA afin d'améliorer la structure du système et de la rendre conforme aux exigences de sûreté de fonctionnement. Après, une structure de système mise à jour peut contenir de la redondance est proposée. Ensuite, un profil de redondance est utilisé pour enrichir le modèle système avec des informations de redondance, ce qui permettra de générer un arbre de défaillance dynamique qui prend en compte le comportement du système. Enfin, l'arbre de défaillance dynamique généré doit être analysé afin de créer un diagramme de machine à états décrivant le comportement du système. La machine à états créée aidera les concepteurs de systèmes à mieux comprendre les dysfonctionnements du système en le simulant. La méthodologie proposée est appliquée à un système d'actionneur électromécanique et un système de distribution de carburant pour l'avion qui sont utilisés dans le domaine aéronautique.

Title : Consideration of safety when choosing architectures of complex systems.

Keywords : Systems Engineering, complex systems, redundancy, safety, FMEA/FTA

Abstract : The goal of this thesis is the integration of safety analysis in a model-based systems engineering approach to ensure consistency between system design and safety artifacts. This integration permits the continuous improvement of the structure and behavior of the system. It also reduces system development time and prevents late detection of errors. To reach this purpose, the SafeSysE methodology is extended. In SafeSysE, a preliminary Failure Mode and Effects Analysis (FMEA) is automatically generated from a SysML model, and this FMEA is then completed by the safety expert but no further development was proposed. The contribution of this thesis is to suggest recommendations based on the FMEA analysis in order to enhance the system design and make it comply with safety requirements. After, an updated system structure that may contain redundancy is proposed. Then, a redundancy profile is used to enrich the system model with redundancy information, which will allow the generation of a dynamic fault tree considering the system behavior. Finally, the generated dynamic fault tree should be analyzed in order to create a state machine diagram that describes the behavior of the system. The created state machine will help the system designers to better understand the system dysfunctions by simulating the system. The proposed methodology is applied to an Electro-Mechanical Actuator system and aircraft fuel distribution system which are used in the aeronautics domain.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France