Vers une ingénierie de systèmes sûrs de fonctionnement basée sur les modèles en conception innovante

Pierre Mauborgne

THÈSE DE DOCTORAT

Université de Lorraine

École doctorale RP2E (Ressources, Procédés, Produit, Environnement)

Pour obtenir le grade de

DOCTEUR
DE
L'UNIVERSITÉ DE LORRAINE

Spécialité : GÉNIE DES SYSTÈMES INDUSTRIELS

Vers une ingénierie de systèmes sûrs de fonctionnement basée sur les modèles en conception innovante

---

par

Pierre MAUBORGNE

Laboratoire Équipe de Recherche sur les Processus Innovatifs

Diplômé de l'École Normale Supérieure de Cachan

Soutenance publique le **03/05/2016**,
devant le jury d'examen :

Rapporteur	Vincent CHAPURLAT	Professeur IMT, École des Mines d'Alès
Rapporteur	Frédéric KRATZ	Professeur des Universités, INSA CVL
Examinateur	Jean-Marc ROUSSEL	Maître de Conférences HDR, ENS Cachan
Co-Directeur	Éric BONJOUR	Professeur des Universités, Université de Lorraine
Co-Directeur	Éric LEVRAT	Professeur des Universités, Université de Lorraine
Co-Encadrant	Samuel DENIAUD	Maître de Conférences, UTBM – UBFC
Examinateur Invité	Jean-Pierre MICAËLLI	Maître de Conférences, Université Lyon 3
Examinateur Invité	Dominique LOISE	Encadrant Industriel Expert IS, Systèmes Conseil
Examinateur Invité	Nicolas BECKER	Expert Sûreté Fonctionnelle, PSA Groupe

SOMMAIRE

REMERCIEMENTS

Ces deux pages sont les dernières à être écrites mais souvent les premières à être lues, donnant un peu d'originalité à ce « pavé » ! Il est vrai que cela peut paraître légèrement trop angélique quand on voit les remerciements mais au final, on retient plus les bons moments que les quelques passages à vide qui ont pu arriver.

Tout d'abord, Frédéric Kratz, je vous remercie pour avoir accepté de rapporter sur ce mémoire et de devenir président du jury. Les échanges lors de cette soutenance ont été très enrichissants. Vincent Chapurlat, après nos échanges forts intéressants lors du « *social event* » de CIGI'13 et du CT SV2S, je vous remercie pour avoir également accepté d'être rapporteur de ce mémoire. Jean-Marc, comme tu l'as dit lors de la soutenance, depuis huit ans, tu suis ma transition d'étudiant de prépa vers prof de prépa à travers stages, présentations et cette soutenance de thèse. Merci pour tous tes conseils. Nicolas Becker, je vous remercie pour avoir accepté de venir assister à ma soutenance, de prendre du temps pour qu'on échange auparavant sur mon sujet de thèse malgré votre emploi du temps chargé.

Bien entendu, je souhaite te remercier Éric B. pour ces 3 ans d'échange comme directeur de thèse. Avant de m'engager dans ce projet de thèse, j'avais demandé à un sage professeur cachanais (certains y verront un jeu de mots foireux) ce qu'il pensait de toi : « *De ce que j'ai entendu, une personne très bien sur le plan scientifique comme sur le plan humain* ». J'ai vérifié ces propos et je peux le valider. Cela peut également s'appliquer à mes autres encadrants. Merci également à toi Éric L. Mes passages à l'AIP ont été bien productifs. Tu m'as prodigué de nombreux conseils sur le fond de ces travaux mais également sur la gestion de ce projet, ce qui a été d'une très grande aide. Je souhaite aussi te remercier Samuel pour ces nombreux échanges que nous avons eu durant cette thèse. Que tu sois à Velizy, Nancy, ou à Belfort, tu avais du temps pour me conseiller sur les travaux à effectuer ! Jean-Pierre, j'aimerais aussi te remercier pour ces 3 ans. Tu m'as permis d'avoir un mémoire dont je peux être fier, ce qui n'était pas gagné au départ. Merci également pour les nombreuses réflexions lors des réunions sur le crobard, la nuance entre humain, corps et fantôme ! Enfin, concernant cette thèse qui a été « bien encadrée », je souhaiterais te remercier Dominique pour la confiance que tu m'as accordée en me permettant de réaliser cette thèse et pour nos nombreux échanges à Vélizy au sein de PSA Groupe et ensuite, quand nos chemins se sont séparés de ce groupe. Ton expertise en IS et SdF a apporté une plus-value importante à ces travaux de thèse.

En parlant de PSA Groupe, je remercie Sébastien Loeb que j'admire pour son parcours sportif (bien que vous ne lisiez sûrement jamais ce mémoire !). Plus sérieusement, je souhaiterais remercier mes collègues de travail que ce soit au sein de l'ancienne équipe de Dominique (les Pascals, Yannick, Véronique, Marc, Fabrice), mes collègues d'open space (l'équipe PLM) et de repas (des gens de (la) Qualité), la communauté des doctorants de PSA et tout particulièrement Mihai pour ce que tu fais pour développer celle-ci et les autres personnes avec qui j'ai pu collaborer (Alexandra, Minh, Nicolas, Stéphane Virginie, …). Enfin, j'aimerais remercier PSA Groupe ainsi que l'ANRT pour leur soutien financier. PSA Groupe a également assuré indirectement mon soutien logistique avec ces 2 formidables voitures que sont mon ancienne C3 et ma 207 !

Une thèse CIFRE permet de faire un lien entre une entreprise (PSA Groupe si vous n'aviez pas compris) mais également un laboratoire qui m'a accueilli, l'ERPI. Je souhaiterais donc remercier Laure Morel, directrice de ce laboratoire pour la qualité de l'accueil lors de mes séjours au laboratoire ainsi que Sandrine puis Noémie et Stéphane pour les différentes aides administratives. Je souhaiterais remercier mes collègues doctorants (Nathalie, Hind, Alexis, Giovanny, Julien(s), Elisa, Daniel, Lamia, Hélène, Alex, Manon, Fabio, Aline et Andrea) avec qui j'ai passé de très bonnes journées mais également de très bonnes soirées. Concernant l'ERPI, je souhaiterais remercier au final les autres professeurs ayant eu des remarques constructives pour la thèse lors des séminaires et autres échanges. Sur un plan professionnel, j'aimerais remercier le service sûreté de fonctionnement de Nexter Systems (Nicolas, Marc, Anne-Sophie, Jonathan) où j'ai pu effectuer un stage qui m'a permis de m'orienter vers cette thèse. Je souhaite aussi remercier les membres du CT SV2S de l'AFIS et ceux du séminaire francilien de sûreté de fonctionnement, deux communautés mêlant universitaires et industriels où les échanges étaient fort intéressants. Pour finir sur ce plan, j'aimerais remercier ceux qui m'entourent depuis septembre 2016 que ce soit ma direction pour m'avoir permis de finir ces travaux, mes collègues (l'équipe de TSI, l'équipe de SI, mon tuteur et ceux m'ayant permis d'avoir de nombreuses ressources pour construire mes cours). L'Ingénierie Système est un sujet arrivant dans les enseignements et j'ai apprécié en discuter avec des collègues comme par exemple aux journées de l'UPSTI où nous avons pu montrer nos convictions avec un autre Pierre ! Une pensée à mes élèves qui liront peut-être ces quelques lignes quand je leur parle de ma thèse !

Sur un plan plus personnel, j'aimerais remercier des gens de mon ancienne école avec qui j'ai pu rester en contact (pas forcément autant que nous le souhaiterions). Je pense à mes amis de promo (Aurélie, Bastien, Champa, Emeric, Julien, Marianne, Marion, Plume, Renaud, Romain, Touski) et ceux qui ont entouré mes années à Cachan, voire avant (Chris, Clotilde, Danielle, Étienne, Gwenaël, H, Jus, Kara, Lorène, ma binôme de prépa, mes anciens partenaires de basket, Mirouf, Pauline, Stall, Toto, Vanessa). J'ai du oublier des personnes auxquelles je n'ai pas pensé en écrivant cela mais auxquelles je tiens, je m'en excuse et les remercie.

Pour finir, je tiens à remercier ceux qui m'ont vu grandir (et grossir) ! Je pense fortement à ma mère, ma sœur, mon beau-frère, ma nièce, mon grand oncle et mon parrain. Ils n'ont pas forcément compris ce que j'allais faire à Nancy mais m'ont fait comprendre qu'ils étaient fiers d'avoir un « *docteur* » dans la famille !

Enfin, je dédie cette thèse à mon père qui aurait aimé lire cette thèse et à ma nièce Chloé qui n'a pas du comprendre grand-chose lors de la soutenance !

INTRODUCTION

We build too many walls and not enough bridge
(propos attribués à Joseph Fort Newton)

De nos jours, tout acheteur de véhicule, tout conducteur, tout passager, etc., entend disposer d'une voiture agréable, personnalisée, dotée d'un nombre croissant de fonctionnalités, mais aussi plus sûre, tout en consommant, en polluant et en coûtant moins. Répondre à ces **exigences** [1] n'est guère aisé. Ceci implique, de la part du constructeur, de lancer sur le marché de nouveaux véhicules à la silhouette et à l'habitacle plus travaillés, aux fonctions embarquées plus nombreuses, etc. Par incidence, cela suppose, pour le concepteur automobile, de ne pas se contenter d'ajouter, de perfectionner, des organes existants, mais aussi de maîtriser une **complexité** fonctionnelle d'ensemble plus grande ; les fonctions du véhicule étant de plus en plus imbriquées, interdépendantes. Dans un tel contexte, un métier devient alors critique : celui d'**architecte système** (AS). C'est à l'AS qu'incombe la charge de bien organiser le **système** et d'en proposer une **architecture fonctionnelle** à même de satisfaire au mieux des exigences de plus en plus nombreuses et sévères. Pour mener à bien ses activités, l'AS peut s'appuyer sur différents théories, langages, modèles, méthodes et outils, proposés ou recommandés par les référentiels ou normes de l'**Ingénierie Système** (IS) comme l'IEEE 1220 (IEEE 1220, 2005), l'ISO 15288 (ISO 15288, 2008), le SEBoK (Pyster & Olwell, 2013), *Découvrir et Comprendre l'IS* (Fiorèse & Meinadier, 2012), etc. Selon l'Association Française d'Ingénierie Système (AFIS) « *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. Cet ensemble est intégré en un système, dans un contexte de recherche d'équilibre et d'optimisation sur tout son cycle de vie* » (Fiorèse & Meinadier, 2012).

1 Les concepts clefs sont définis dans le lexique page 163. Les mots associés sont mis en gras lorsqu'ils sont évoqués pour la première fois dans ce mémoire.

L'AS a pour responsabilité de fournir une solution répondant aux besoins des parties prenantes, y compris sur la dimension **Sûreté de Fonctionnement** (SdF). La complexité croissante des systèmes mécatroniques ou multiphysiques pourrait augmenter les risques sécuritaires d'insatisfaction des clients liés au dysfonctionnement des nouvelles fonctions intégrées dans le véhicule. Pour traiter ce problème, une nouvelle norme est apparue en 2011 pour l'industrie automobile : l'ISO 26262. Elle impose de démontrer que l'ingénierie du système véhicule a été réalisée selon les règles de l'art pour ne pas générer de risques inacceptables pour le conducteur, les passagers ou l'environnement. Or, les études de sûreté de fonctionnement sont souvent réalisées *ex post* par des spécialistes du domaine, après que les **architectures fonctionnelles** et organiques ont été définies. Il en résulte une détection tardive de problèmes de sûreté de fonctionnement et une perte de temps due aux itérations nécessaires entre l'AS et l'ingénieur spécialiste de SdF.

Placée dans le cadre spécifique de l'IS automobile, cette thèse abordera une problématique en conception, à savoir l'intégration et l'adaptation des processus d'IS avec des concepts, modèles et activités de la SdF. L'idée sous-jacente est que l'AS peut mener à bien certaines activités liées à la SdF et que ces activités devraient être intégrées dans les référentiels de l'IS. Pour ce faire, de nouveaux processus, de nouveaux modèles du système doivent être proposés, de sorte à assister l'AS dans ses tâches de spécification, de conception d'architectures fonctionnelle et organique, de vérification et de validation, etc.

Pour éclairer ce dernier point, un exemple trivial peut être donné. On ne peut envisager de la même manière une étude de **fiabilité** relative à une courroie de transmission pour laquelle peuvent être réalisées des expériences en grandeur réelle et celle visant un système multiphysique dont les coûts de prototypage sont très élevés. Les connaissances, le savoir-faire, les modèles physiques ou les outils du mécanicien expérimentateur ne sont pas immédiatement pertinents. Il en est de même pour l'amélioration locale des différents organes, par exemple en les redimensionnant, en changeant les matériaux, les formes, etc. La SdF du système doit être appréhendée de façon globale, abstraite, logique. Comme l'illustre l'incipit, des ponts doivent être bâtis pour permettre à différents métiers de contribuer, de façon collaborative, à la SdF globale du système, l'AS ou le spécialiste SdF ne pouvant faire ce travail isolément.

Le cadre général de la thèse venant d'être posé, nous allons maintenant en préciser le contexte industriel.

CONTEXTE INDUSTRIEL

L'automobile est un produit désormais plus que centenaire. Sa forme, ses performances, les fonctions qu'elle embarque ont changé en un siècle. Depuis deux décennies, l'une des tendances lourdes qui apparaît est l'exigence de disposer de véhicules de plus en plus sûrs. Par exemple, bien que non obligatoire pour la mise en circulation d'un véhicule, l'EuroNCAP (*European New Car Assessment Program*), créée en 1997, est devenu un incontournable, puis une véritable exigence relative à la sécurité à bord des véhicules. L'EuroNCAP est un argument de vente important, qui participe à la réputation du constructeur en matière de sécurité. Parmi les nouvelles fonctions critiques requises pour passer haut la main les tests d'EuroNCAP, on peut citer le régulateur de vitesse, l'alerte contre le risque de collision, la détection de piéton, l'alerte de franchissement involontaire de ligne, etc. Ces nouvelles fonctions produisent des effets systémiques. Pour ce qui concerne la conception organique (*embodiment design*), les satisfaire suppose des composants communs, des ressources partagées, créant ainsi des dépendances entre les fonctions.

L'explosion de la complexité des véhicules automobiles impose donc des méthodes de développement performantes et robustes vis-à-vis de la Sûreté des véhicules. Des normes encadrent ce domaine. On peut de nouveau citer l'ISO 26262:2011, relative à la **sûreté fonctionnelle** des véhicules routiers. Même s'il n'existe pas pour l'instant de certification dans le domaine automobile, contrairement à l'aéronautique ou au ferroviaire, cette dernière norme impose toutefois un certain nombre de démonstration de respect des règles de l'art dans le développement des fonctions complexes et critiques en SdF. Jusqu'à présent, les activités d'Ingénierie et de Sûreté de Fonctionnement étaient séparés et séquentielles. Pour le dire autrement, le **processus de conception** du système et le **processus de SdF** ne sont pas alignés. Ce qui implique deux rôles différents, voire antagoniques de par leurs modes de raisonnement, les modèles et outils utilisés, entre l'AS du véhicule ou de l'un de ses sous-systèmes, et d'autre part **le spécialiste en SdF**. Par exemple, l'AS définit les performances du système lorsque celui-ci fonctionne normalement. Le spécialiste en SdF s'intéresse au système lorsqu'il dysfonctionne. Il doit donc enrichir l'analyse fonctionnelle de l'AS avec les aspects dysfonctionnels, et rendre ses résultats à l'AS après un certain délai. Comme celui-ci progresse en parallèle, les conclusions d'analyses de SdF sont donc partiellement valides. Le désalignement entre l'IS, **fonctionnelle**, et la SdF, **dysfonctionnelle**, se traduit aussi dans l'organisation des bureaux d'études, avec des métiers différents, reposant sur des compétences et utilisant des outils spécifiques. Par exemple, la SdF peut recourir aux réseaux de Petri stochastiques, ce qui suppose des connaissances pointues que n'a pas forcément l'AS.

Dans le cadre de cette thèse, l'analyse du désalignement évoqué et la recherche de processus, de modèles du produit à même de rapprocher le domaine de l'IS avec celui de la SdF n'ont pas

été réalisés avec un point de vue extérieur, abstrait. Dans le cadre de la CIFRE dont nous avons bénéficié, nous avons été affectés à une entité s'occupant des méthodes d'IS et de SdF[2].

PROBLÉMATIQUE

Les pages précédentes ont montré qu'existe un besoin industriel en matière d'alignement entre l'IS et la SdF, plus précisément entre l'activité de l'AS en charge de l'architecture fonctionnelle du véhicule et l'ingénieur en SdF. Pour répondre à ce besoin, et sans épuiser la question, plusieurs pistes peuvent être envisagées. Elles peuvent être organisationnelles (rapprocher physiquement l'AS et l'ingénieur en SdF), cognitives (les former au domaine de l'autre), etc. La piste retenue dans cette thèse concerne les processus et les modèles ; l'IS contemporaine entendant de plus en plus en faire les objets principaux de la conception.

Dès lors, plusieurs voies peuvent être imaginées :

1. enrichir les modèles d'IS pour faire ensuite de la SdF ;
2. enrichir les modèles de SdF pour faire ensuite de l'IS ;
3. adapter des modèles de SdF de sorte à réaliser en parallèle de l'IS ;
4. adapter les modèles d'IS de sorte à réaliser en parallèle de la SdF ;
5. unifier les modèles de l'IS et ceux de la SdF ;
6. créer des passerelles entre les modèles, pour faciliter l'échange de données, d'objets, de résultats de modélisation, etc., entre l'IS et la SdF ;
7. convertir les modèles de sorte à permettre la traduction réciproque des modèles de l'IS et de ceux de la SdF, etc.

Les voies évoquées sont plus ou moins ambitieuses. Elles supposent des travaux de recherche plus ou moins conséquents : la création de passerelles (6) étant moins exigeante que celle de la conversion de modèles (7), sans parler de leur unification (5). Aussi la voie 6 est-elle la plus souvent explorée jusqu'à présent.

Le champ des solutions possibles, en matière d'alignement de l'IS et de la SdF, s'étend si on prend en compte l'aspect procédural de la conception. En effet, l'IS et la SdF reposent non seulement sur des connaissances déclaratives, mais aussi procédurales, dont des processus faisant par exemple l'objet de normes.

[2] *Le tuteur industriel de cette thèse, Dominique Loise, a été directeur d'une entité Méthodes d'IS et de SdF au sein de la conception fonctionnelle des GMP pendant 6 ans avant de devenir le responsable PSA des « méthodes d'ingénierie système et d'intégration » pendant 5 ans.*

Là encore, plusieurs pistes, procédurales, peuvent être imaginées :

1. créer un nouveau processus de conception hybridant l'IS et la SdF ;
2. guider les processus d'IS par la SdF ;
3. intégrer les activités d'IS et celles de SdF, etc.

Parmi toutes les pistes mentionnées, et vu le contexte industriel de notre thèse, nous ferons le choix suivant :

1. pour ce qui concerne les modèles, nous enrichirons les modèles d'IS de sorte à intégrer la SdF (pistes 1 et 4);
2. pour ce qui concerne les processus, nous intégrerons les activités d'IS et de SdF dans un même processus (piste procédurale 3).

C'est donc en adaptant les modèles de SdF à ceux de l'IS (dont les modèles seront enrichis) et en intégrant les activités d'IS et de SdF que nous entendons contribuer à l'élaboration d'une **Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF)**, comme le montre la figure 1.

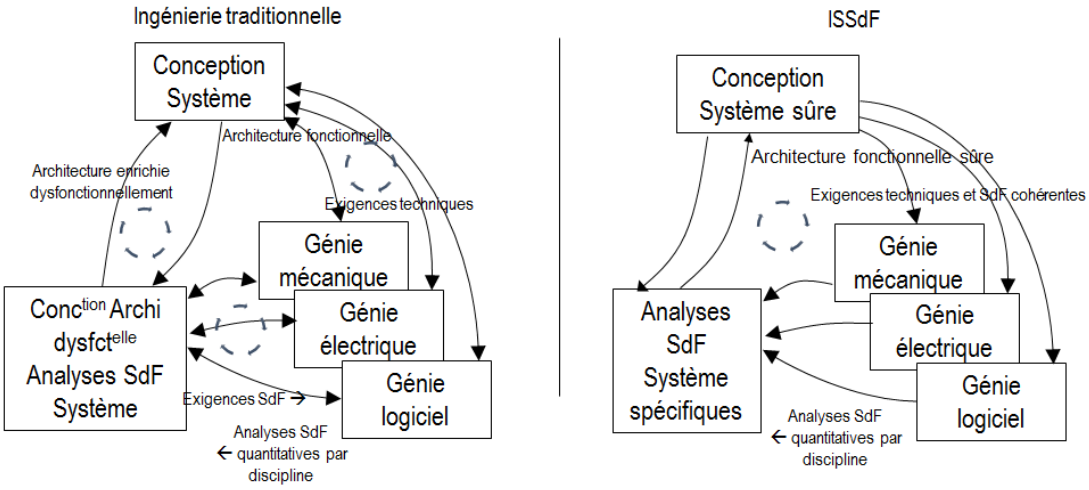


Figure 1 : Approches liant l'IS, la SdF et des spécialités

Cette figure résume deux approches liant l'IS, la SdF et des spécialités[3]. Nous représentons l'approche traditionnelle, qui repose plutôt sur un mode guichet et silo, par la figure de gauche. Les analyses de SdF sont réalisées à partir de données provenant de la conception système (architecture fonctionnelle) et de chaque spécialité ou sciences du génie (génie mécanique, électrique, logiciel...). La SdF développe une vision système, modifie l'architecture fonctionnelle pour prendre en compte les fonctions de sécurité, fait le lien avec les différentes spécialités, leur alloue les exigences de sûreté et intègre le retour de celles-ci. Dans le rôle spécifique, la SdF réalise des études ayant recours à des modèles spécifiques (par exemple, résistance à la fatigue des matériaux d'un composant, analyse de pannes précoces en

[3] *Pour des raisons de clarté, seulement trois spécialités ont été indiquées. Nous sommes conscients qu'il peut y en avoir d'autres.*

électronique, etc). Cette organisation implique des itérations souvent longues avec des impacts forts sur les architectures et les spécialités.

Dans la figure de droite, une seule entité assurant la conception système sûr échange avec les différentes spécialités, préférablement en équipe intégrée, pour définir l'architecture système sûre. Les exigences et les architectures sont considérées simultanément, de façon qualitative mais également quantitative, d'un point de vue fonctionnel et dysfonctionnel. Le rôle de la spécialité SdF évolue au niveau système pour s'intéresser surtout à la démonstration de la tenue des exigences qualitatives et quantitatives de sûreté. De la même façon peuvent intervenir d'autres spécialités transversales, par exemple, l'Acoustique Vibratoire pour vérifier la satisfaction d'exigences de confort ou de filtration de bruit. De plus, pour chaque spécialité, des analyses quantitatives de SdF pourront être effectuées par des experts des domaines concernés. Nous défendons la thèse que l'approche présentée par la figure de droite serait plus performante.

Cette thèse suppose donc une nouvelle répartition des activités de SdF visant à confier à l'AS (ou à son équipe) des activités qualitatives de SdF, éventuellement en collaboration avec les spécialistes de SdF, et à confier aux spécialistes de la SdF des études, souvent quantitatives, nécessitant une expertise spécifique. Il s'agit donc d'un couplage adéquat entre ces deux domaines permettant d'aboutir à un processus unifié d'activités d'IS et de SdF réalisables par un AS. Cette proposition doit tenir compte de la capacité de l'AS et de son équipe à réaliser de nouvelles activités, tout en permettant aux spécialistes de SdF de travailler à partir de modèles plus adaptés à leurs besoins. Il ne s'agit pas de minorer ou de supprimer le rôle de spécialiste en SdF, mais de montrer qu'en ajoutant, en partageant ou en répartissant des concepts, modèles et activités entre AS et lui, le processus de conception se trouverait amélioré.

MÉTHODE SUIVIE

Le contexte industriel ayant été décrit et les pistes de recherche choisies, il convient maintenant de préciser la méthode suivie. Celle-ci repose sur cinq piliers :

1. l'alignement de la démarche de recherche sur les principes de l'IS ;
2. la modélisation systémique ;
3. une revue de la littérature cohérente avec la structure modulaire de la thèse ;
4. des entretiens avec des experts de l'IS ou de la SdF ;
5. le choix d'un exemple emblématique.

Alignement sur l'IS — Le choix a été fait d'utiliser le cadre de référence de l'IS pour chacun des points présentés dans les prochains chapitres. Nous commencerons ainsi chaque chapitre par une analyse du besoin concernant, rappelons-le, le désalignement entre l'IS et la SdF lorsqu'il s'agit de spécifier un système ou d'architecturer les fonctions d'un système. Après une analyse de l'existant, nous émettrons des exigences d'une solution attendue relevant des pistes choisies. Nous réaliserons ensuite une première phase dans laquelle nous étudierons plusieurs solutions candidates existantes. Lors d'une seconde phase, nous choisirons une solution de principe dont nous explorerons les possibilités et qui sera ensuite consolidée. Enfin, nous vérifierons cette solution par rapport aux exigences et nous en réaliserons une validation partielle en l'appliquant à un exemple automobile.

Modélisation systémique — Comme son nom l'indique, l'IS renvoie au concept de système. Appréhender celui-ci, le représenter, raisonner à son sujet suppose de le modéliser. Nous essaierons le plus possible de prendre en compte cette dimension systémique dans les processus ou modèles proposés. Compte tenu de son importance, les référentiels de l'IS seront présentés en premier chapitre.

Revue de la littérature — Afin d'en faciliter la lecture, nous avons fait le choix d'une structure de thèse modulaire. Si un point particulier intéresse le lecteur, la lecture d'un chapitre sera alors suffisante. La revue de la littérature a été découpée pour respecter une telle structure.

Entretiens avec des experts — Nous avons pu alimenter notre travail par des entretiens réalisés avec des experts de la Direction de la Recherche et du Développement de PSA Peugeot Citroën. De plus, grâce à une participation régulière au groupe *Model-Based Safety Assessment* (MBSA) du Comité Technique *Sûreté, Validation et Soutien des Systèmes* (SV2S) de l'AFIS, des échanges avec des experts d'autres entreprises et d'autres secteurs ont été engagés afin de vérifier la validité de nos travaux.

Choix d'un exemple automobile — Idéalement, la validation de nos travaux devrait se faire lors d'un projet de développement d'un véhicule ou de l'un de ses sous-systèmes (groupe motopropulseur, liaison au sol, etc.). Cependant, vu la durée et le caractère confidentiel des données d'un tel projet, cela nous a été impossible. Aussi, afin de pouvoir réaliser une première boucle de validation, nous avons choisi un exemple, relativement simple à comprendre, mais assez complexe pour être significatif. Notre choix s'est ainsi porté sur l'étude d'un phénomène dangereux spécifique, à savoir, l'accélération intempestive du véhicule.

QUESTIONS DE RECHERCHE

L'hypothèse qui sera vérifiée dans ce mémoire est qu'il est possible d'intégrer l'Ingénierie Système et la Sûreté de Fonctionnement afin de réaliser de l'Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF).

La première question qu'il faut se poser est : quelle est la sémantique de l'ISSdF ? En effet, lors du rapprochement de deux domaines, il faut vérifier qu'ils peuvent se comprendre, partagent une terminologie commune. Pour cela, la première contribution de ce mémoire sera de proposer un **modèle conceptuel** de l'Ingénierie de Systèmes Sûrs de Fonctionnement.

La deuxième question porte sur : comment définir les différentes activités permettant d'aboutir à une architecture de système sûr ? La deuxième contribution de cette thèse consistera en un processus de l'Ingénierie de Systèmes Sûrs de Fonctionnement reposant sur les processus d'IS standard.

Il est ensuite nécessaire de se demander si le processus d'ISSdF peut être amélioré par des méthodologies ? Notre troisième contribution sera une méthodologie permettant de déterminer les exigences de système sûr. Elle permettra donc également de définir les ***Safety Goals***, correspondant à un livrable demandé par l'ISO 26262:2011.

La dernière question, analogue à la troisième, s'interroge sur la possibilité de définir une architecture fonctionnelle sûre. Pour cela, la quatrième contribution de ce mémoire est une méthodologie permettant de concevoir une architecture fonctionnelle de système sûr. Le ***Functional Safety Concept***, autre livrable demandé par l'ISO 26262:2011, sera une vue extraite de cette architecture fonctionnelle.

PLAN DU MÉMOIRE DE THÈSE

Les précédentes pages ont permis d'expliciter le contexte et le besoin industriels auxquels se réfère cette thèse, les pistes choisies pour traiter de l'alignement entre la SdF et l'IS, la méthode suivie, et les questions de recherche soulevées par notre recherche. Il est désormais possible de présenter la structure, modulaire, de ce mémoire.

Le premier chapitre présentera les deux sources de l'ISSdF. Nous montrerons ce qu'est l'IS, puis la SdF. Nous complèterons ce chapitre en dressant un état de l'art des travaux cherchant à aligner l'IS et la SdF.

Le deuxième chapitre proposera un modèle conceptuel de l'ISSdF susceptible de réduire les distances cognitives existant entre les domaines de l'IS et de la SdF. Après un état de l'art relatif aux modèles conceptuels utilisés en IS ou en SdF, nous proposerons un modèle conceptuel susceptible de décrire ce que pourrait être ce nouveau domaine qu'est l'ISSdF.

Le troisième chapitre définira le **processus d'ISSdF**. Après un rappel des méthodologies existantes et un raffinement de l'état de l'art en matière de processus, nous proposerons un processus d'ISSdF applicable par l'AS. Conformément à différentes normes encadrant l'IS, nous prendrons en compte, dans une optique de SdF, quatre processus techniques clés en IS, à savoir, la définition des exigences des parties prenantes, l'analyse des exigences système, la définition de l'architecture fonctionnelle et la définition de l'architecture organique. Nous focalisant sur la conception fonctionnelle, nous ne ferons qu'évoquer la définition de l'**architecture organique**. Le processus d'ISSdF proposé comprendra des activités classiques de l'IS enrichies par la SdF, mais aussi de nouvelles activités orientées SdF que devra réaliser l'AS.

Le quatrième chapitre se focalisera sur le processus de définition des exigences de système sûr. En la matière, et classiquement, une **Analyse Préliminaire des Risques** (APR) est réalisée par l'ingénieur en SdF en se basant sur les vues externes du système. Dans une optique d'IS, une telle analyse importe car elle permet de définir les exigences de haut niveau en SdF. Ce chapitre apportera une méthode, liée à la vue opérationnelle, permettant de définir les exigences opérationnelles d'un système sûr.

Le cinquième et dernier chapitre proposera une méthodologie dédiée à la détermination d'une architecture fonctionnelle de système sûr. Elle explicitera la vue structurelle du véhicule comportant les chemins de **propagation** et les concepts de SdF, puis la vue comportementale comprenant les modes nominaux et les modes dégradés des fonctions.

Ce mémoire se conclura par une synthèse de nos résultats, puis par une présentation de perspectives possibles.

Enfin, le présent document comprendra :

1. des annexes ;
2. un lexique des concepts clefs, indiqués en gras lorsqu'ils sont mentionnés pour la première fois ;
3. une table de traduction des termes utilisés, l'IS et la SdF étant des domaines où prévaut l'Anglais ;
4. une liste des sigles et acronymes ;
5. une table des matières.

Afin de rendre aussi explicite que possible la structure de cette thèse, la figure suivante la représente à l'aide d'un diagramme d'activité SysML (OMG, 2012), langage d'IS qui sera utilisé pour les modélisations des systèmes, au cours de cette thèse. L'outil utilisé pour représenter les différents diagrammes SysML est IBM Rhapsody ®.

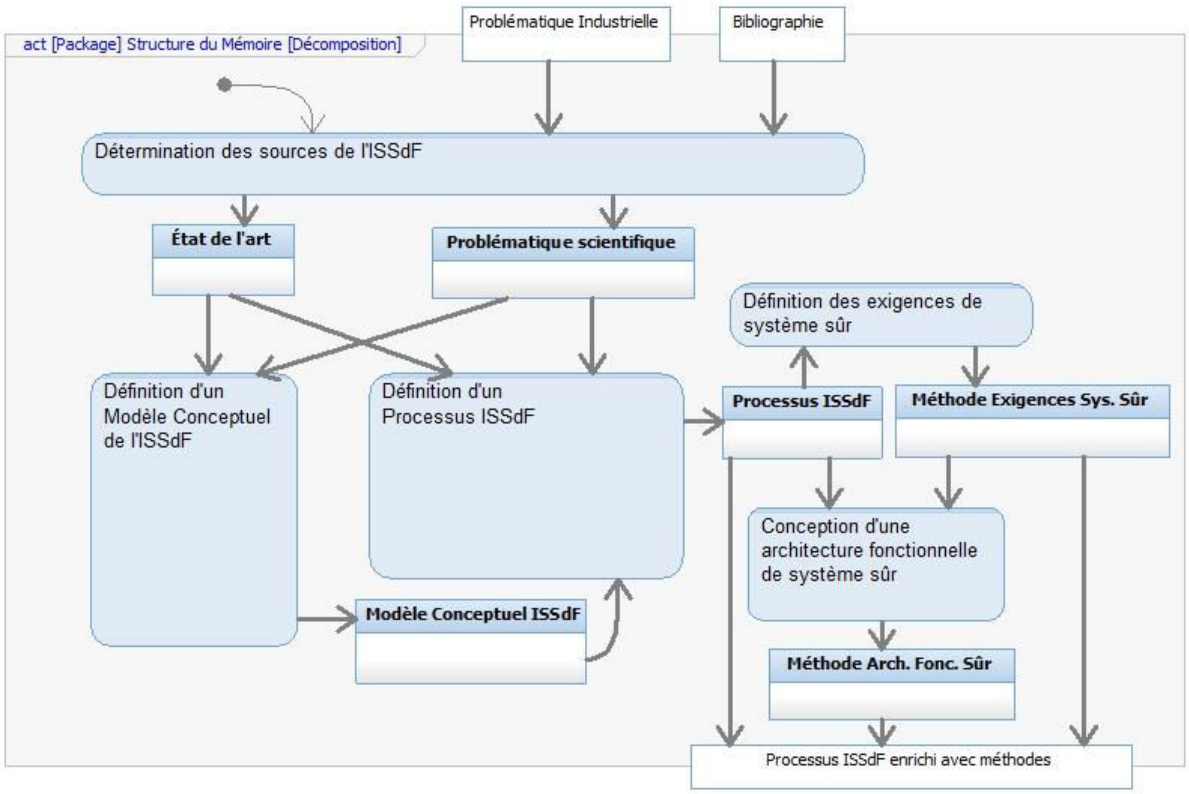


Figure 2 : Organisation du mémoire

CHAPITRE 1
AUX SOURCES DE L'ISSDF

L'introduction du présent mémoire a indiqué la nécessité qu'il y a, pour des industriels en charge de développer des produits complexes comme les automobiles, de coupler le domaine de l'Ingénierie Système (IS) avec celui de la Sûreté de Fonctionnement (SdF). Ce premier chapitre entend montrer qu'il n'existe pas encore sur étagère de solution susceptible de satisfaire un tel besoin. Une telle absence nous obligera à préciser la problématique à laquelle ce mémoire entend apporter une première réponse. Pour traiter ces points, nous dresserons d'abord un panorama de l'IS, puis de la SdF, en nous appuyant dans les deux cas sur les référentiels normatifs ou les guides de bonnes pratiques existants. Nous présenterons ensuite une première série de travaux liant l'IS et la SdF, ce qui nous permettra de préciser la problématique à laquelle nous proposons une contribution par ce travail de thèse.

1.1 UN PANORAMA DE L'IS

L'IS naît dans les années 1960 aux États-Unis (Cook & Ferris, 2007). « *Ce ne sont pas les concepteurs qui ont initié son développement, mais les maîtres d'ouvrages d'artefacts complexes tels que les aéronefs, les systèmes d'armes, les réseaux de télécommunications, les gros progiciels, etc. Ces maîtres d'ouvrage supposaient que la systémique offrait les concepts utiles pour mieux structurer et conduire des projets impliquant de très nombreux maîtres d'œuvre aux compétences variées. En quarante ans, l'ingénierie système s'est diffusée dans de nombreux secteurs industriels* » (Bonjour, Deniaud, & Micaëlli, Conception complexe et ingénierie système, 2009). Le détail de l'histoire de l'IS est donné en ANNEXE 2.

De nos jours, il existe une communauté internationale consacrée à l'IS, l'INCOSE ; communauté réunissant des concepteurs, des maîtres d'ouvrage, des consultants, des offreurs d'outils, des universitaires, etc. Pour l'AFIS se veut le chapitre Français de l'INCOSE. Tant les acteurs de l'INCOSE que de l'AFIS s'intéressent autant au « *système à faire* », c'est-à-dire au véhicule pour ce qui nous concerne, qu'au « *système pour faire* », c'est-à-dire l'organisation de

sa conception automobile[4]. Ce qui donne deux points d'entrée de l'IS : par le produit ou par l'organisation. Compte tenu du besoin industriel auquel cette thèse entend répondre, c'est par celui-ci que nous aborderons l'IS. L'organisation de la conception sera appréhendée non à travers les projets, les équipes, les départements du bureau d'études, mais par les processus.

1.2 LES PROCESSUS DE L'IS

Trois principales normes précisent ce que sont les processus d'IS (Figure 3) : l'EIA 632 (EIA 632, 2003), l'IEEE 1220 (IEEE 1220, 2005) et de l'ISO 15288 dont la première version date de 2002 et la seconde, sur laquelle nous nous appuierons, de 2008 (ISO 15288, 2008). Sa version 2015 n'étant disponible qu'à la fin de la thèse, nous ne l'avons donc pas utilisée. S'ajoute à ces normes une description des processus proposée par les rédacteurs du *Systems Engineering Body of Knowledge* (*SEBoK)* (Pyster & Olwell, 2013). Ce document n'est pas une norme, mais un recueil de connaissances ou de bonnes pratiques.

La Figure 3 atteste que les différentes normes gravitant autour de l'IS ont des périmètres différents, couvrant tout ou partie du cycle de vie du système. C'est l'ISO 15288 qui a le spectre le plus large. Cependant, l'EIA 632:2003 et IEEE 1220:2005 détaillent certains processus ou complètent utilement celle-là. Aussi, dans la suite de ce mémoire sera-t-il fait appel à ces référentiels. Il en sera de même pour le *SEBoK*. À noter enfin, que l'AFIS reconnaît deux référentiels essentiels en IS. Le premier est un livre didactique intitulé *Découvrir et comprendre l'Ingénierie Système* (AFIS, 2007). Le second est le *Systems Engineering Handbook* (INCOSE, 2011), qui sert de support aux certifications en IS. Depuis trois ans, pour clarifier les liens entre tous ces référentiels ou normes, l'ISO 15288, le *SE Handbook* et le *SEBoK* sont réécrits de sorte à être plus cohérents entre eux.

[4] Il serait réducteur de penser que l'IS ne s'intéresse qu'au système-produit et au système-projet puisque l'IS vise à définir également les autres systèmes contributeurs nécessaires dans les différentes phases de vie du système.

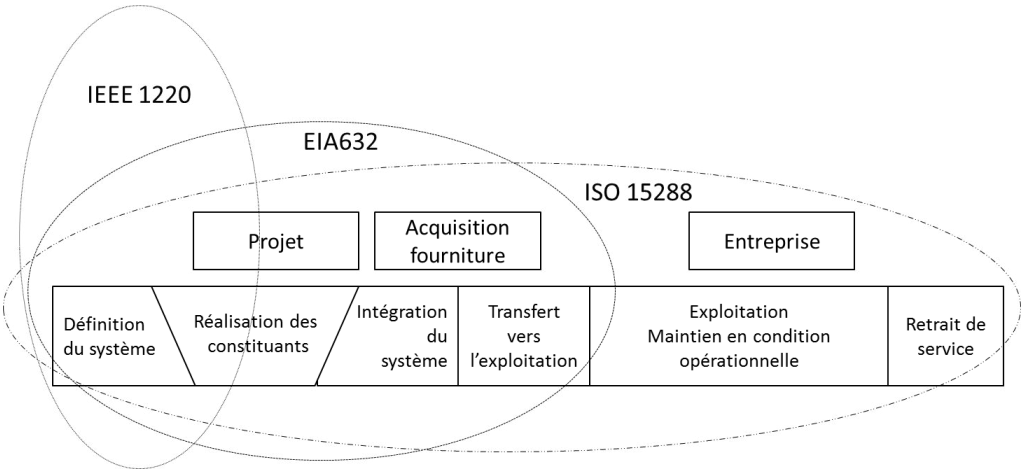


Figure 3 : Normes encadrant l'Ingénierie Système (AFIS, 2007).

Les normes les plus anciennes, à savoir l'EIA 632:2003 et l'IEEE 1220:2005 sont connues de la communauté de l'IS et ont déjà été présentées dans de nombreux mémoires comme la thèse de Samuel Rochet (Rochet, 2007) pour la norme EIA632:2003 ou celle de Clémentine Cornu (Cornu, 2012) pour la norme IEEE1220:2005. L'EIA 632:2003 se focalise sur les processus techniques et les processus support mais détaille peu les activités de conception, alors que l'IEEE 1220:2005 insiste sur les aspects de vérification et de validation des activités de conception, ainsi que sur l'analyse des solutions. L'ISO 15288:2008 est plus générale. Elle propose un nombre élevé de processus de nature différente, organisationnels, managériaux, et pour ce qui nous concerne : techniques. Cette norme reconnaît huit processus techniques couvrant la conception (définition des exigences et de l'architecture du système), la réalisation et la construction du système. La Figure 4, qui reprend le cycle en V, montre l'enchaînement logique de ces huit processus, à savoir : *Stakeholder Requirements Definition Process*, *Requirements Analysis Process* et *Architectural Design Process*. Au cours d'un projet, ces processus sont réalisés en parallèle. Au cours de chaque phase de définition et construction du système, ils le sont aussi, mais avec des intensités différentes.

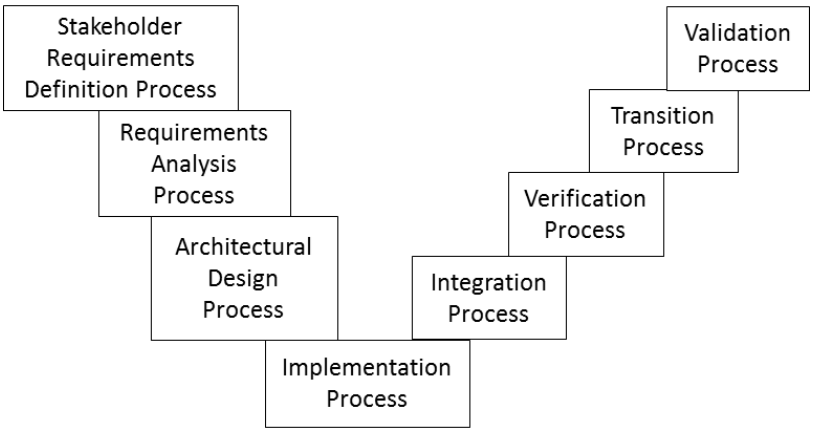


Figure 4 : Les différents processus techniques d'IS de l'ISO 15288:2008

Le *Stakeholder Requirements Definition Process* consiste, pour le concepteur, à répertorier les différentes parties prenantes, à recueillir leurs demandes, souhaits, attentes, besoins, etc. Bien mettre en œuvre ce processus suppose de sa part de savoir identifier les parties prenantes, de recueillir leurs retours d'expériences, de collecter le matériau utile pour la suite, d'observer l'usage de tel ou tel produit, etc. S'ensuit un deuxième processus, le *Requirements Analysis Process*, consistant à traduire les « exigences des parties prenantes » (*stakeholders' requirements*) en « exigences techniques (ou système) » (*technical (or system) requirements*). Ce processus est assuré par le concepteur. Il l'oblige à formaliser les exigences, à spécifier le périmètre fonctionnel du système et les fonctions qu'il devra intégrer. Une fois cette identification fonctionnelle faite, il lui faudra, au cours de l'*Architectural Design Process*, les détailler et les architecturer, de sorte à imaginer comment le système fonctionnera. C'est au cours de ce processus que sera définie l'architecture fonctionnelle voire logique du système (même s'il existe une nuance entre ces deux architectures). Suite aux allocations des fonctions aux composants, la définition de l'architecture organique est réalisée ; celle-ci comprenant des modules organiques qui devront être développés à leur tour.

Le *SEBoK* propose un point de vue de la conception sensiblement différent de celui des normalisateurs de l'ISO. La conception du système est découpée en quatre « domaines de connaissances » (*knowledge areas*) : le « *System Requirements* », la « *Logical Architecture Design* », la « *Physical Architecture Design* », et le « *System Analysis* ». On retrouve là une configuration proche de celle de l'ISO 15288:2008, à deux détails près :

1. l'architecture fonctionnelle, voire logique, du système est clairement distinguée de son architecture organique (nommée *Physical Architecture* dans les publications anglo-saxonnes) ;
2. après analyse du système, le processus d'ensemble est récursif, si bien qu'il peut être rappelé lorsqu'il s'agira de développer les strates inférieures, c'est-à-dire les sous-systèmes ou constituants, et ce jusqu'au composant jugé élémentaire.

Le *SEBoK* comporte également une activité préliminaire de *Business or Mission Analysis*.

Pour faciliter leur comparaison, le tableau 1 met en lice les référentiels évoqués. Il montre qu'il existe des recouvrements significatifs entre ces référentiels.

Tableau 1 : Récapitulatif des différents processus d'IS.

<table>
<tr><th>ISO 15288:2008</th><th>IEEE 1220:2005</th><th colspan="3">EIA 632:2003</th><th>SEBoK v1.4</th></tr>
<tr><td></td><td></td><td colspan="3"></td><td>Business or Mission Analysis</td></tr>
<tr><td>Stakeholder Requirements Definition</td><td></td><td rowspan="4">System Design</td><td rowspan="2">Requirements Definition Process</td><td>Acquirer Requirements & Other Stakeholder Requirements</td><td>Stakeholder Needs and Requirements</td></tr>
<tr><td>Requirements Analysis</td><td>Requirements Analysis & Requirements Validation</td><td>System Technical Requirements</td><td>System Requirements</td></tr>
<tr><td rowspan="2">Architectural Design</td><td>Functional Analysis & Functional Verification</td><td rowspan="2">Solution Definition Process</td><td>Logical Solution Representations</td><td>Logical Architecture Model Development</td></tr>
<tr><td>Synthesis & Design Verification</td><td>Physical Solution Representations</td><td>Physical Architecture Model Development</td></tr>
<tr><td></td><td>System Analysis</td><td colspan="3">System Analysis</td><td>System Analysis</td></tr>
</table>

L'IS proposant plus que des processus, il est possible d'aborder d'autres points de ce domaine ; points cohérents avec l'enjeu de la thèse. C'est ainsi que nous allons nous focaliser sur un concepteur particulier, l'architecte système (AS), et une modalité particulière de son travail, à savoir l'IS fondée sur les modèles, ou *Model-Based Systems Engineering* (MBSE).

1.2.1 Rôle de l'architecte système

"Architecte" est un mot d'origine grecque *arkhitektôn*, qui signifie maître constructeur selon le dictionnaire Larousse (Larousse, 2016). En matière de conception et de construction de bâtiment, le maître d'œuvre interagit avec le maître d'ouvrage, pour comprendre ses besoins, ses attentes, ses contraintes, etc., coordonne les différents corps de métiers opérant sur le chantier, et réceptionne les travaux. L'architecte crée une forme fonctionnelle et esthétique conforme aux exigences du maître d'œuvre. Dans le domaine de l'IS, le rôle de l'architecte diffère :

1. dans le cadre de grands projets, il n'est pas un maître d'œuvre, un chef de projet, mais une personne ayant à déterminer l'architecture du système, sur la base d'exigences venant de parties prenantes avec lesquelles il n'interagit pas forcément, et d'échanger avec les différents métiers, experts, etc., présents dans les bureaux d'études ;
2. il n'est pas responsable du développement de la forme du produit, cette tâche incombant, dans l'automobile, aux *designers ;*

3. il se focalise sur la formalisation des exigences liées aux architectures externes/internes, puis sur la définition d'architectures fonctionnelles ou organiques, et non sur la conception détaillée, comme le font certains architectes en bâtiment qui, à l'instar des plus grands (Gaudi, Le Corbusier, Aalto, etc.) allaient jusqu'à dessiner tous les meubles, toutes les poignées de porte,…

De la sorte, l'AS est le ''pivot'', l'intégrateur entre les exigences des parties prenantes rapportées par d'autres métiers, et les spécialistes de tel ou tel type d'organes ou de comportements du système (Morel, et al., 2013).

1.2.2 Architecturer le système à l'aide de modèles

Il est aisé d'identifier les outils de modélisation traditionnellement utilisés par l'architecte d'un bâtiment. Il se sert du dessin (croquis, esquisse à la main), de maquettes réalistes, physiques ou de plus en plus virtuelles, de dessins côtés, de modèles géométriques permettant de dimensionner la structure, etc. Le cas est semblable pour l'architecte système. Celui-ci s'inscrit de plus en plus dans le cadre du MBSE. En 2007, l'INCOSE a défini la MBSE [5] comme une application de la modélisation pour supporter toutes les activités de l'Ingénierie Système. Nous nous concentrerons dans ce mémoire sur les modèles utiles pour définir la vue externe du système, puis la vue interne correspondant à ses architectures fonctionnelle et organique.

L'un des problèmes liés à l'usage de modèles par l'architecte système concerne la complication et la complexité des connaissances et des données qu'il lui faut traiter. L'une des heuristiques auxquelles recourir est de modulariser les systèmes (Bonjour, Deniaud, & Micaëlli, 2013) (Bonjour, Deniaud, Dulmet, & Harmel, 2009) et de proposer une approche multivues du système à architecturer, voire des processus requis pour l'architecturer. L'idée est simple : il n'est pas possible d'avoir toutes les connaissances et les informations sur un même diagramme, si bien qu'un système compliqué ou complexe peut être représenté à l'aide de vues, donc de diagrammes, spécialisés et complémentaires.

Cette question est traitée par plusieurs acteurs de l'IS. Certains théoriciens du domaine proposent des matrices intégrant les vues selon les facettes du système à faire et pour faire mises en avant. Jean-Michel Penalva (Penalva, 1990) a élaboré une matrice croisant trois vues du système à faire (*Operational, Functional, Constructional*) et trois vues correspondant au niveau de décision dans l'organisation de la conception (*Actions, Tactics, Strategy*). Le processus de conception global consiste alors à enchaîner les neuf vues de la matrice. Daniel Krob (Doufene,

[5] *Model-Based Systems Engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases.*

Hugo, Dauron, & Krob, 2014), propose aussi une matrice de 9 cases, mais le sens des lignes et des colonnes diffèrent. Les trois lignes correspondent aux vues opérationnelle, fonctionnelle et structurelle ; les trois colonnes, à la description des états, à la description statique, et à la description du comportement dynamique. Différents diagrammes SysML (OMG, 2012), composants du MBSE, sont alignés sur la matrice. Il est à noter que les états sont utilisés pour décrire le comportement du système et par conséquent, la colonne état et la colonne comportement dynamique pourraient être fusionnés. Par ailleurs, cette matrice ne montre pas que la nature ou le détail des modèles peuvent changer en cours de conception, lorsque différentes phases d'enrichissements (ou raffinements) successifs et d'itérations sont réalisées.

Les offreurs de méthodes ou d'outils dédiés à l'IS ne sont pas en reste. Avec *Arcadia* ®, Thalès propose cinq vues (Voirin, 2008) : *Operational Analysis, System Analysis, Logical Architecture, Physical Architecture, Contracts for development,* et IVVQ (*Integration, Verification, Validation, Qualification*). On retrouve une logique similaire dans les travaux de Shames et Skipper (2006), pour ce qui concerne l'architecture d'engins spatiaux. Enfin, un groupe comme Airbus utilise les points de vue *Fonctionnel, Logique/L1* (regroupement de fonctions pour un *work-package*[6]) et *Logique/L2-Physique* (allocation des fonctions à un work-package) comme l'a montré Claire Campan (Campan, 2014).

Comme l'a explicité Serge Tichkiewitch (Tichkiewitch, 1996) (Tichkiewitch & Véron, 1997), l'approche multivue paraît pertinente d'un point de vue cognitif (elle facilite la compréhension du système) et pratique (elle facilite l'activité de l'architecte système), si bien qu'elle sera reprise dans ce mémoire. Cependant, énoncer une telle approche n'est pas suffisant pour l'Ingénierie Système. Dans le cadre de l'IS, en effet, il est nécessaire de définir un processus. Celui-ci a pour conséquence d'induire une trajectoire de parcours et d'enrichissement des différentes vues retenues par l'analyse.

1.2.3 Autres aspects de l'IS

Comme cela vient d'être évoqué, l'architecte système travaille sur un certain nombre de vues qu'il doit enrichir, ce qui peut être réalisé par des échanges avec des experts. Une des solutions prometteuses a été traitée par Fabien Bouffaron (Bouffaron, Dupont, Mayer, & Morel, 2014) (Bouffaron, 2016) en réalisant une co-spécification basée sur les modèles.

Bien que l'IS ait de nombreux avantages, son déploiement est assez difficile pour des raisons organisationnelles, culturelles (d'autres routines sont bien ancrées dans les métiers), logicielles, financières, d'interopérabilité avec l'existant, etc. C'est pour cette raison que des travaux visent à élaborer un cadre méthodologique pour la mettre en œuvre en s'adaptant à des contextes

[6] *Notion employée par Airbus équivalente à un ensemble logique.*

industriels variés. Nous pouvons citer à cet égard les travaux de Clémentine Cornu avec Airbus Helicopters (Cornu, 2012) ou ceux de Hervé Panetto (Panetto, 2006).

Une fois l'IS mise en œuvre, une autre problématique concerne l'évaluation, puis l'aide à la décision, en s'appuyant sur différents points de vue évoqués ci-dessus. Nous pouvons citer les travaux de Yun Ye (Ye, 2014), de Fabien Retho (Retho, 2014), ou également de Mambaye Lô (Lô, 2013), qui propose une méthode d'évaluation des architectures en Ingénierie Système.

Dans ce panorama, nous nous sommes surtout concentrés sur les étapes de définition des exigences et de conception de l'architecture. Il existe des travaux montrant également l'importance des activités d'Intégration, de Vérification, de Transition et de Validation comme dans (Chapurlat & Braesch, 2008) ou dans (Chapurlat & Bonjour, 2014).

Enfin, pour réaliser au mieux la conception du produit, il faut aussi que le projet soit bien architecturé. Cela a fait l'objet de recherches comme dans (Harmel, 2007), (Bonjour & Micaëlli, 2010) ou dans l'habilitation à diriger des recherches d'Éric Bonjour (Bonjour, 2008).

1.2.4 Synthèse

L'IS est un cadre de référence utile pour concevoir des systèmes complexes. En s'appuyant sur des vues et des modèles, l'AS peut échanger avec des experts de différents domaines afin d'aboutir à une solution globalement performante satisfaisant les exigences. Cependant, l'AS est focalisé sur l'aspect fonctionnel du système alors qu'il est avéré que tout produit, à un moment ou un autre de son cycle de vie, peut défaillir, tomber en panne, devenir dysfonctionnel, voire provoquer des dommages aux personnes, aux biens ou à l'environnement. Dès lors, comment coupler les domaines de l'IS et de la SdF ? Avant de répondre à cette question, il importe de définir quelques éléments clefs de ce dernier domaine. Ce sera l'objet de la section suivante.

1.3 UN PANORAMA DE LA SDF

Jean-Claude Laprie (Laprie, 1995) définit la SdF, comme « *la propriété qui permet aux utilisateurs du système de placer une confiance justifiée dans le service qu'il leur délivre* ». Compte tenu de son importance, la SdF a fait et fait encore l'objet de très nombreux travaux, tant théoriques que pratiques. Comme l'IS, des normes l'encadrent, elles définissent aussi bien les concepts d'un système sûr que les processus à mettre en œuvre pour développer des produits présentant cette qualité. Par ailleurs, depuis plus de sept décennies, des méthodes et des outils classiques existent en matière de SdF.

1.3.1 SdF : le cadre normatif

Il existe de nombreuses normes sectorielles encadrant la SdF. Pour ce qui concerne la conception de système proprement dite, l'aéronautique a été un secteur pionnier avec les normes DO 178:2012 (DO-178C, 2012), ARP 4754:2010 (ARP 4754, 2010), et ARP 4761:1996 (ARP 4761, 1996). En matière de composants ou de dispositifs électriques, électroniques ou logiciels embarqués, nous pouvons citer l'IEC 61508:2010 (IEC 61508, 2010). En matière de SdF des systèmes électriques embarqués dans l'automobile, une norme fait référence : l'ISO 26262:2011 (ISO 26262, 2011). S'ajoute à ces référentiels normatifs, l'*ISO/IEC Guide 51* de 1999, qui inclut des principes directeurs liés à la SdF et à intégrer dans les normes. Bien sûr, cette liste est loin d'être exhaustive. Elle comprend toutefois les référentiels que nous mentionnerons dans les pages suivantes ; référentiels qui peuvent être agencés fonctionnellement et sectoriellement les uns par rapport aux autres, comme le montre la Figure 5.

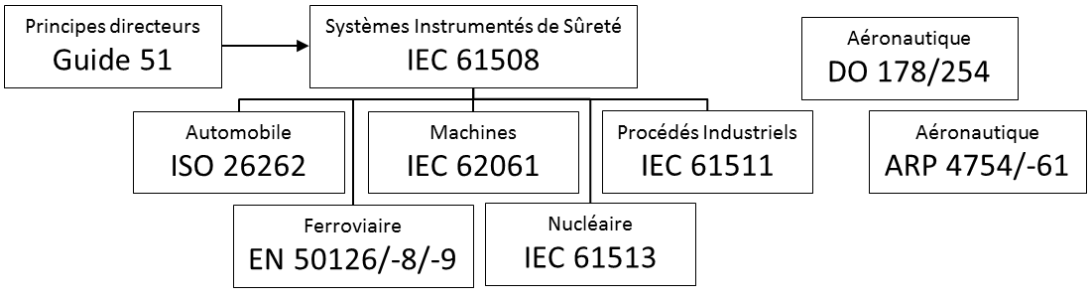


Figure 5 : Hiérarchie des normes de sûreté de fonctionnement adapté de (Saddem, 2012)

La première version de l'*ISO/IEC Guide 51* remonte à 1999. Ce document a été réactualisé en 2014, mais cette dernière version n'était pas disponible au début de ce travail de thèse. Les rédacteurs de ce guide précisent son objet : « *ce guide s'applique à tous les aspects de la sécurité relatifs aux personnes, aux biens ou à l'environnement, ou à l'une de leurs combinaisons (par exemple : personnes seulement; personnes et biens ; personnes, biens et environnement). Le présent guide adopte une démarche visant à réduire le risque engendré par l'utilisation de produits, procédés ou services. Il tient compte de la vie complète du produit, procédé ou service incluant aussi bien l'utilisation prévue que les mauvais usages raisonnablement prévisibles* ». Ce document fournit des recommandations et des exigences pour inclure des aspects de SdF dans des normes, le plus souvent sectorielles comme la montre la précédente figure. Enfin, le guide a été traduit en français, contrairement aux autres normes utilisées dans ce mémoire. Cette traduction est utile car le passage de concepts de l'anglais au français n'est pas toujours évident, comme nous le montrerons dans les chapitres suivants.

L'aéronautique est précurseur en matière de normalisation de la SdF. De ce fait, ce secteur peut servir d'exemple, de référence, de parangon (*benchmark*). Les normes DO 178:2012, ARP 4754:2010 et ARP 4761:1996 pourraient être des sources pertinentes pour qui veut

coupler IS et SdF. Toutefois, il faut faire attention aux spécificités sectorielles. En aéronautique, les phases de vie du système (l'aéronef) sont très bien définies. Les solutions sont fabriquées en petite série, avec un coût et des contrôles de qualité élevés. Il est possible et simple, vu la taille de la plupart des aéronefs, d'ajouter des capteurs, de pratiquer la redondance, de zoner le produit, etc. Enfin, un pilote est conducteur professionnel, bien formé, à même de contrôler l'aéronef dont il est responsable dans des circonstances dangereuses. Toutes ces spécificités s'opposent à l'automobile, qui est un produit de petite taille, fabriqué en masse à moindre coût, utilisable par tous dans des contextes où les **menaces** abondent parfois (autres usagers de la route, chaussée déformée, intempéries, signalétique peu lisible, etc.), voire avec des usages parfois atypiques (surcharge du véhicule, voiture bélier, etc.).

L'IEC 61508:2010 présente aussi un réel intérêt car elle est intersectorielle. Elle traite de sûreté fonctionnelle pour les systèmes instrumentés de sécurité. Elle prescrit un certain nombre de livrables à produire pour que ce système soit le plus sûr possible. Norme générique, elle a ensuite été déclinée dans de nombreux secteurs, et ce, pour répondre à leurs spécificités. La déclinaison qui nous intéressera directement, l'ISO 26262:2011, concerne l'automobile. Elle sera amplement détaillée dans les chapitres suivants. L'ISO 26262:2011 propose des concepts, des livrables intéressant notre problématique de thèse. En revanche, ce qui concerne les menaces, les agressions ou les mauvais usages du véhicule n'entre pas dans le périmètre de cette norme.

En plus d'un référentiel normatif conséquent, la SdF peut aujourd'hui s'appuyer sur des méthodes et des outils classiques, enseignés à de nombreux techniciens ou ingénieurs, et mis en œuvre dans de très nombreux bureaux d'études.

1.3.2 Les classiques de la SdF

Le but de cette section n'est pas de détailler ou de raffiner les méthodes ou outils classiques de la SdF, mais d'apprécier quels services ils peuvent apporter à l'IS et à l'AS.

Une première mention peut être faite à l'Analyse Préliminaire des Risques (APR). Comme l'énonce Yves Mortureux (Mortureux, 2002), l'APR associe une démarche et un outil. La démarche consiste à identifier les événements redoutés d'un système, les analyser, puis trouver des solutions de principe susceptibles de minimiser les conséquences des risques, voire de les empêcher. La démarche d'APR est menée en conception amont, le plus souvent après l'analyse fonctionnelle externe ou, si on adopte le cadre de l'IS, l'analyse d'exigences. La démarche d'APR est menée : (1) à tous les niveaux de stratification du système, du système global, aux sous-systèmes, etc., (2) en tenant compte de toutes les phases de son cycle de vie. L'outil d'APR, quant à lui, aide à bien réaliser la démarche mentionnée. En suivant la démarche d'APR, on imagine un scénario conduisant à l'accident et on remplit le tableau ci-dessous recensant des informations clefs pour exprimer ce scénario.

Tableau 2 : Exemple d'outil d'APR (Mortureux, 2002)

Tableau type de la méthode APR											
Sous-système ou équipement	Phase	Entité dangereuse	Évènement causant une situation dangereuse	Situation dangereuse	Évènement causant un accident	Accident	Effet ou conséquences	Gravité	Occurrence (Fréquence)	Mesures de prévention ou de protection	Application des mesures

La deuxième méthode classique de la SdF est l'Analyse des Modes de Défaillance, de leurs Effets et de leurs Criticités (AMDEC). Cette méthode part d'un item[7] donné (composant, organe, module, sous-système, système) et définit ses différents modes de défaillances, leurs effets à un niveau local ou global, et les criticités, et ce, afin d'avoir une qualification de ces modes de défaillances. L'AMDEC est considérée comme une méthode remontante (*bottom-up*) ; les effets de la défaillance de l'item analysé étant d'abord appréhendés localement, avant de remonter, si besoin est, à la strate supérieure. De ce fait, l'AMDEC est utilisée lors de phases de conception détaillée, donc en aval et non en amont, lors de phases de définition du concept du système. Les résultats d'une AMDEC se présentent sous une forme tabulaire. Enfin, l'élaboration d'une AMDEC fait l'objet de normes. Par exemple, la NF EN 60812:2006 (NF EN 60812, 2006) décrit quel processus mettre en œuvre pour la réaliser d'une façon satisfaisante. Pour information, ce processus est représenté Figure 6.

[7] *Au sens de l'ISO 26262:2011*

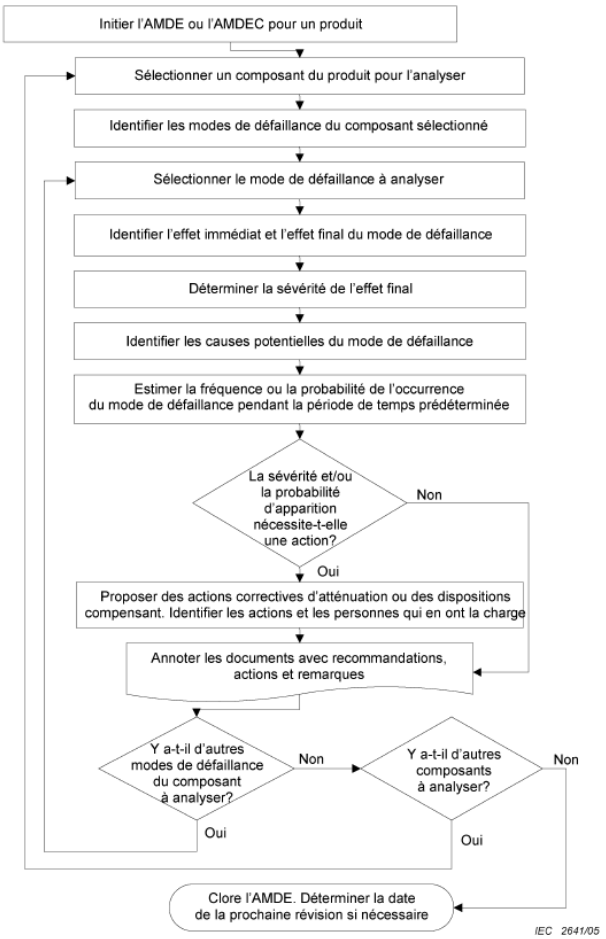


Figure 6 : Processus normalisé d'AMDEC (NF EN 60812, 2006)

La troisième méthode classique de la SdF que nous entendons mentionner est HAZOP, pour *HAZard and OPerability study*. C'est une méthode qui se focalise sur les flux échangés entre composants du système étudié. Elle s'appuie sur un lexique, des mots guides, permettant de définir et de qualifier les états dysfonctionnels des dits composants. L'intérêt des mots-guides proposés est de faire un intermédiaire entre une approche binaire (le composant fonctionne ou ne fonctionne pas) et une approche physique (loi de défaillance du composant). Un tel intermédiaire sera particulièrement intéressant lorsqu'il s'agira d'architecturer fonctionnellement un véhicule.

Tableau 3 : Mots guides de la méthode HAZOP.

Mot Guide	Signification
No or not	Négation
More	Augmentation quantitative
Less	Diminution quantitative
As well as	Augmentation qualitative
Part of	Diminution qualitative
Reverse	Opposé logique
Other than	Substitution
Early	Modification temporelle
Late	Modification temporelle
Before	Modification séquentielle
After	Modification séquentielle

La quatrième et dernière méthode classique de la SdF qui a retenu notre attention est l'arbre de défaillance. Cet arbre est une structure formelle permettant de déterminer toutes les causes possibles d'un évènement redouté. Afin d'expliciter les différentes combinaisons logiques entre causes, des portes sont utilisées. Les plus courantes sont : ET, OU, K/N (K composants fonctionnent sur N possibles). Les défaillances étant le plus souvent modélisées à l'aide de variables booléennes, il est donc possible de traduire cet arbre en équation booléenne. Étant un outil formel, l'arbre de défaillance aide à vérifier les objectifs de sécurité et à déterminer des coupes minimales, ou plus petites combinaisons possibles de causes provoquant l'événement redouté. Il est donc essentiel, pour certains événements redoutés de systèmes critiques, de ne pas avoir de coupes minimales d'ordre 1 (une défaillance amène à l'événement redouté).

1.4 DES RESTRICTIONS NÉCESSAIRES

Pour bien comprendre la suite de cette thèse, il importe d'apporter des restrictions au vaste domaine de la SdF.

La première restriction concernera le périmètre retenu. En effet, la SdF (*Dependability*) au sens large associe plusieurs performances, que sont la disponibilité (*Availability*), la fiabilité (*Reliability*), la sécurité de fonctionnement (*Safety*), ou encore la maintenabilité (*Maintainability*). Seule la sécurité de fonctionnement (*Safety*) d'un point de vue architecture nous intéressera par la suite. Et ce, compte tenu du besoin industriel ayant motivé cette thèse. Les aspects spécifiques de la sûreté comme les analyses stochastiques ne seront donc pas abordés dans ce mémoire.

Concernant l'aspect *Safety*, la terminologie utilisée en France est ambigüe. Ce terme est souvent traduit par sûreté et se définit ainsi : « *the non-occurrence of catastrophic consequences on the environmen*t » (Laprie, 1995). Dans le domaine de la *Safety*, nous pouvons distinguer la sûreté fonctionnelle (ou sécurité fonctionnelle), les menaces, les agressions, ainsi que les mauvais usages. Là encore, un choix sera fait, qui consistera à ne s'intéresser qu'à la sûreté fonctionnelle, c'est-à-dire aux phénomènes dangereux provenant d'un mauvais fonctionnement du système causé par une défaillance aléatoire.

Dans la suite de ce mémoire, nous utiliserons le terme système sûr pour désigner un système dont le fonctionnement ne peut pas produire de conséquence catastrophique sur son environnement. L'impact de l'environnement sur le système sera traité par la conception (définition des interfaces du système) ou par une étude des agressions. Le traitement de la sécurité (*Security*), comme ce qui concerne le *hacking* des véhicules, n'est donc pas dans le périmètre d'étude de ce mémoire.

La deuxième restriction concernera l'objet sur lequel portera la sûreté fonctionnelle. Pour ce qui nous concerne, cet objet sera l'architecture fonctionnelle. Un exemple relatif à l'ingénierie du groupe motopropulseur, à savoir l'accélération intempestive, sera utilisé pour vérifier l'intérêt de nos propositions.

La troisième restriction concernera le cadre de référence retenu pour décrire la conception de l'architecture fonctionnelle du système. Comme indiqué en première section, ce cadre sera celui de l'IS vu que l'on s'intéresse à la conception d'architecture de systèmes multiphysiques et non de systèmes purement mécaniques ou purement logiciels, par exemple.

La quatrième restriction concernera l'acteur de la conception visé par nos résultats. Il s'agira ici de l'architecte système.

Enfin, la cinquième restriction concernera les modèles, méthodes et outils de sûreté retenus. Nous mentionnerons les outils classiques de la SdF, tout en tenant compte du fait qu'ils font l'objet de perfectionnements divers, avec les tendances suivantes :

1. le développement de modèles ou d'outils permettant la vérification formelle. À cet égard, on peut citer (Aubry, et al., 2012) ;

2. le dépassement des limites des outils classiques. Par exemple, les arbres de défaillance donnent une vision statique de l'aspect dysfonctionnel d'un système. Marc Bouissou, avec BDMP (Bouissou & Bon, 2003), Martin Walker avec PANDORA (Walker, Bottaci, & Papadopoulos, 2007), et Merle et al. (Merle, Roussel, & Lesage, 2011) greffent désormais des portes dynamiques sur ces arbres ;

3. la limitation du temps de calcul d'outils combinatoires. Nous pouvons citer les travaux de Brameret et al. (Brameret, Roussel, & Rauzy, 2013), qui donnent une estimation d'objectif de sûreté par exploration d'une chaîne de Markov limitée.

Ces tendances s'expliquent par l'augmentation de la complexité des systèmes, donc la complexification induite de l'analyse de leurs dysfonctionnements. De plus, elles se focalisent aujourd'hui sur des organes, des pièces physiques, des composants électroniques ou logiciels, si bien qu'elles sont proches de problèmes de fiabilité de composants et loin de notre objet, qui est le système dans son ensemble.

Il faut garder présent à l'esprit les perfectionnements évoqués. Toutefois, dans le premier temps qui est le nôtre, nous restreindrons notre périmètre de recherche à ces outils classiques de la SdF. Si l'Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF) n'est pas capable de les intégrer, voire les remplacer, c'est que ses bases seraient considérées comme insuffisantes.

1.5 L'INGÉNIERIE DE SYSTÈMES SÛRS DE FONCTIONNEMENT, UN DOMAINE ÉMERGENT

Dans les deux premières sections, nous avons présenté deux panoramas séparés de l'IS et de la SdF. Nous allons maintenant apprécier si la littérature propose une solution, complète ou partielle, à la question qui est la nôtre, à savoir proposer une Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF). Dans un premier temps, nous vérifierons ce qui est proposé dans les normes mentionnées précédemment, puis nous ferons une synthèse de la littérature sur la conceptualisation des liens entre IS et SdF. Ceux-ci étant faibles, nous pourrons alors poser la problématique précise de ce mémoire.

Les normes de l'IS mentionnent la sûreté. Toutefois, cette mention est succincte et, au mieux, seules sont suggérées des méthodes comme l'AMDEC, et ce, uniquement en fin de conception fonctionnelle et organique. Robin Cressent a proposé un processus liant différentes activités relevant de l'ISSdF (Cressent, David, Idasiak, & Kratz, 2012). Cependant, les deux types d'activités, respectivement liés à l'IS et à la SdF, demeurent séparés et sont dans des « *swimlanes* » différents. La première piste qui consisterait, pour élaborer une ISSdF, à se référer au seul cadre de l'IS, quitte à l'étendre un peu, n'est donc pas pertinente.

Une deuxième piste possible est l'utilisation, voire la reconstruction, par les ingénieurs en SdF, des modèles d'IS. Par exemple, Pierre David (David, 2009) (David, Idasiak, & Kratz, 2010) a développé une méthode, nommée Médisis, permettant de réaliser de nombreuses activités de sûreté comme des AMDEC. Dans (Mauborgne, Labe, Smouts, Stojanovic, & Delgado, 2013), Pierre Mauborgne a élaboré une méthode facilitant l'élaboration des arbres de défaillance à partir de modèles d'IS.

Une troisième piste consisterait à perfectionner les modèles ou outils existants de SdF, pour les intégrer à l'IS. Les outils de sûreté comme Simfia ® ou Safety Architect ® permettent de visualiser les équations dysfonctionnelles associées aux fonctions d'une architecture d'un produit, pour ensuite générer des arbres de défaillance (Dumont, Sadmi, & Vallée, 2011). Belmonte et Soubiran (Belmonte & Soubiran, 2012) ont, quant à eux, proposé un langage métier

(*Specific Domain Language*) permettant de lier dans une arborescence les composants de l'architecture fonctionnelle et les AMDEC. Le langage AltaRica ®, dont la dernière version est détaillée dans (Prosvirnova & Rauzy, 2012) et dans (Batteux, Prosvirnova, Rauzy, & Kloul, 2013), propose de spécifier un système, puis d'en modéliser les dysfonctionnements de haut niveau. Les projets IMOFIS (Projet IMOFIS, 2011) et VERDE (Systematic, 2012) ont également exploré cette voie avec la société Alstom, en définissant un langage métier permettant de passer d'une décomposition fonctionnelle à une décomposition de modèles dysfonctionnels.

En fait, les deuxième et troisième pistes souffrent d'une même insuffisance. Les méthodes ou outils proposés aident au mieux l'expert en SdF à comprendre certains modèles de l'IS, et non l'architecte système à définir une architecture sûre. Et ce, à cause de formalismes dédiés nécessitant des compétences spécifiques, pointues, dans le domaine de la SdF.

Une quatrième et dernière piste est proprement logicielle. Elle vise à convertir les modèles d'un domaine en des modèles d'un autre domaine. Le CEA a proposé de nombreuses passerelles entre langages (Yakymets, Jaber, & Lanusse, 2012), par exemple entre SysML et AltaRica, NuSMV, etc. Ces passerelles ne concernent en fait qu'une partie des diagrammes proposés par SysML, à savoir les *Block Definition Diagram* (BDD), *Internal Block Diagram* (IDB) et *State Machine Diagram* (STM). Il est impossible d'utiliser les cas d'utilisation, les diagrammes d'activité ou de séquence, alors que leur emploi est pertinent pour un architecte système désirant définir la situation opérationnelle du système ou en comprendre le fonctionnement à un niveau global. Faida Mhenni (Mhenni, 2014) utilise le format XMI pour rendre interopérables les différents modeleurs utilisant SysML. Il s'agit là d'une voie prometteuse, même si les essais que nous avons réalisés sur différents modeleurs proposés sur le marché – Papyrus ®, Artisan Studio ®, et Rhapsody ® – se sont avérés peu probants.

1.6 UNE ISSDF EN CHANTIER

Du fait de l'absence actuelle de processus, de méthode, d'outil, permettant de bien articuler les domaines de l'IS et de la SdF pour former une ISSdF, il convient donc de considérer cette question comme problématique. Le présent mémoire de thèse entend donc proposer une contribution, une première étape, à l'ISSdF. Pour ce faire, il est possible de se référer au schéma conceptuel suivant, proposé par l'INCOSE (INCOSE, 2011).

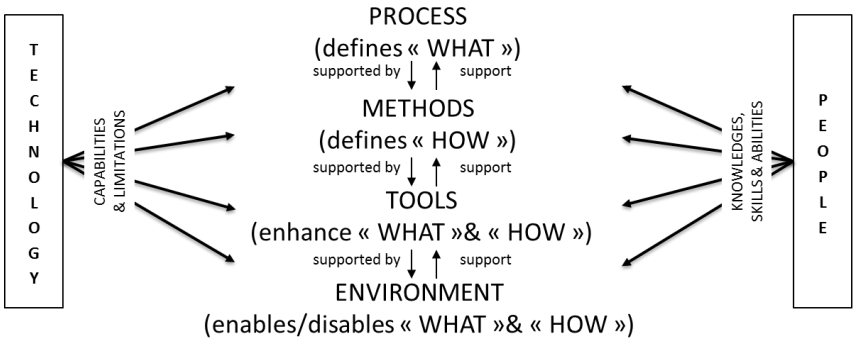


Figure 7 : Liens entre processus, méthode et outil selon l'INCOSE (INCOSE, 2008)

Celui-ci a pour mérite de mettre en évidence les aspects à prendre en compte pour développer telle ou telle proposition nouvelle, ici en matière de couplage entre IS et SdF. Si nous le reprenons, nous pouvons alors expliciter les façons dont nous traiterons l'élaboration de l'ISSdF. Partant d'une analyse du cadre normatif existant en IS et en SdF, nous définirons d'abord les connaissances minimum nécessaires à l'ISSdF, et ce, du point de l'architecte système et de l'ingénieur de SdF (*people*). Pour ce faire, un modèle conceptuel de ce domaine émergent sera proposé. Ce modèle sera ensuite mis en œuvre dans un processus d'ISSdF. Celui-ci cherchera à intégrer le mieux possible les activités liées à la conception et à la sûreté (*process*). Certaines de ces activités sont instrumentées par les méthodes et outils classiques de la SdF. Dès lors, nous proposerons des méthodes (*methods*) jugées pertinentes pour ce qui concerne les études de la sûreté. Des outils de modélisation seront utilisés pour vérifier si des outils peuvent supporter les méthodes mentionnées (*tools*). L'application de tels outils ne formera pas le cœur de la thèse. Il en sera de même pour la mise en œuvre, dans les projets de développement du véhicule, des méthodes ou processus proposés. Enfin, nous puiserons dans un fonds technologique existant pour traiter le problème. Celui-ci comprend les normes, les recueils de bonnes pratiques, un langage comme SysML, des modeleurs, etc. (*technologies*).

1.7 CONCLUSION

Ce premier chapitre a permis de présenter les deux sources de l'ISSdF, à savoir l'IS et la SdF. L'IS dispose de référentiels qui proposent des concepts décrivant le produit et des processus utiles pour maîtriser le développement de systèmes complexes. L'IS repose sur une vue fonctionnelle du produit : la solution développée est supposée sûre, fiable, contrôlable, facilement maintenable, etc. Malheureusement, même si la SdF est évoquée dans certaines normes de l'IS, cette mention est trop succincte. Réciproquement, la SdF s'appuie sur des normes, des méthodes, des outils qui ont fait leur preuve, mais leur intégration avec ceux de l'IS n'est pas évident. Et ce, que le couplage entre la SdF et l'IS passe par une extension des processus d'IS, par le transfert de modèles d'un domaine à l'autre, ou par la transformation de modèles. Des pistes de couplage ont été abordées, mais les résultats ne sont pas forcément probants par rapport à notre problématique. Du fait de l'absence actuelle de processus, de méthodes, d'outils, permettant d'aligner l'IS et la SdF pour former une ISSdF, il convient donc de considérer cette question comme non-encore résolue. Le présent mémoire de thèse entend ainsi proposer une contribution à l'élaboration de l'ISSdF. Pour ce faire, le premier travail à réaliser est d'élaborer un modèle conceptuel de l'ISSdF ; modèle satisfaisant à la fois pour l'AS et l'ingénieur de SdF.

CHAPITRE 2
UN MODÈLE CONCEPTUEL DE L'ISSDF

Dans le précédent chapitre nous avons explicité l'enjeu et l'intérêt de l'Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF) et les différents modèles susceptibles d'y contribuer. Ceux-ci portent autant sur le produit sûr que sur les processus à mettre en œuvre pour en assurer le bon développement. Avant d'entrer dans les détails et de présenter des modèles de processus ou de produit précis, il convient d'abord de définir et d'agencer les concepts du domaine de l'ISSdF. Pour ce faire, nous proposerons un modèle conceptuel de l'ISSdF dont la sémantique est commune aux domaines de l'Ingénierie Système (IS) et à celui de la Sûreté de Fonctionnement (SdF). Le présent chapitre exposera la nécessité d'un tel modèle, avant de décrire comment il a été élaboré, puis de présenter et de commenter celui que nous proposons.

2.1 UN MODÈLE CONCEPTUEL POUR COUPLER DEUX DOMAINES

Rappelons que l'objectif de la présente thèse est d'aboutir à des architectures de systèmes sûrs de fonctionnement dans le secteur automobile. Pour atteindre cet objectif, notre hypothèse est d'intégrer les activités de SdF dans le cadre de référence de l'IS. Ce qui suppose de redéfinir le contenu de certaines activités mentionnées dans le référentiel de l'IS afin d'aboutir à des livrables conformes à l'ISO 26262:2011. Comme les référentiels de l'IS et de la SdF ne coïncident pas du fait d'une histoire parallèle de ces deux domaines, plusieurs approches peuvent être imaginées pour les coupler. Celle retenue dans cette thèse repose sur le modèle. L'approche par le modèle a pour principal mérite d'obliger à définir aussi rigoureusement que possible les concepts, les variables, les relations causales, etc., manipulées.

De plus, un modèle conceptuel partagé est intéressant au sens où il oblige à tenir compte, pour certains concepts ou variables, de leur aspect dual, c'est-à-dire à la fois fonctionnel et dysfonctionnel. L'architecte système est focalisé sur ce premier aspect ; l'ingénieur en SdF, sur le second. Une telle différence cognitive modifie les sens donnés aux concepts de base. Ainsi, si nous prenons le concept de système, nous constatons le hiatus suivant : l'ISO 26262:2011 le définit comme un « *set of elements that relates at least a sensor, a controller and an actuator with one another* », alors que l'IS le définit de façon extensive comme un « *ensemble d'éléments en interaction, organisés pour atteindre un ou plusieurs résultats déclarés* » dans l'ISO

15288:2008. De même, l'ISO 26262:2011 définit une situation opérationnelle comme un « *scenario that can occur during a vehicle's life* » alors qu'en IS, ce concept décrit un macro-état dans lequel se déroule un scénario. Il y a également le cas du « *technical safety requirement* », défini par l'ISO 26262:2011 comme « *requirement derived for implementation of associated functional safety requirements* », alors que l'IS suppose qu'une exigence technique est de niveau système, si bien qu'elle ne situe pas qu'au niveau organique. Pour conclure sur ce point, on peut citer le concept « *d'item* » dans l'ISO 26262:2011, qui signifie un système, un ensemble de systèmes, une fonction, voire un composant sur lesquels on applique les méthodes de SdF. Le diagramme de structure suivant déploie les différentes occurrences de concept.

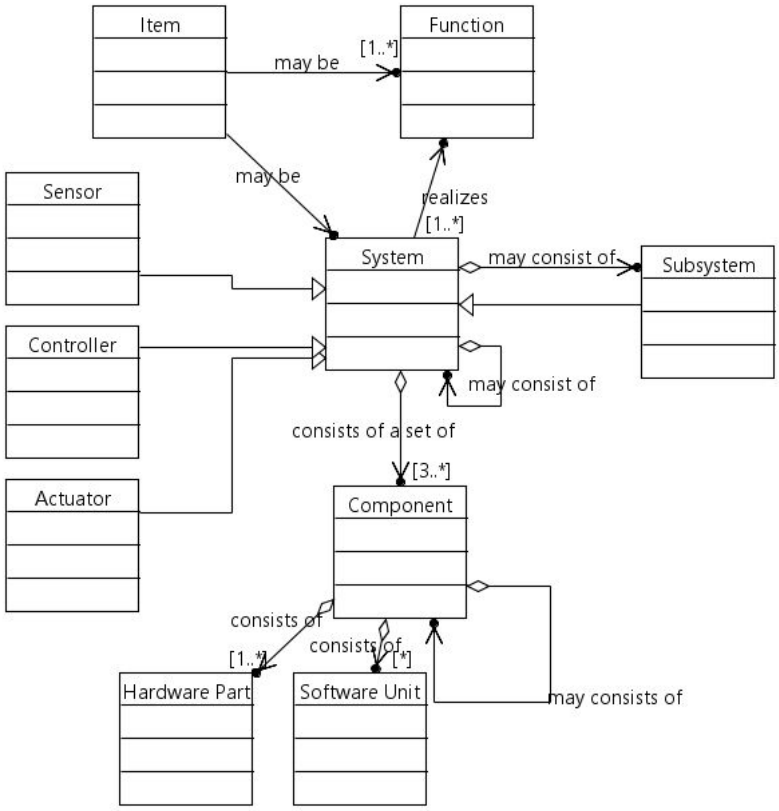


Figure 8 : Concept d'item pour l'ISO 26262:2011 (ISO 26262, 2011)

Le concept d'item n'est pas utilisé en IS alors qu'il présente d'évidentes caractéristiques systémiques. Il faudra donc trouver un concept équivalent ou bien l'expliciter à l'aide de concepts reconnus dans ce domaine. Fort de cette nécessité de modélisation, cette première section permettra d'exposer la démarche de modélisation retenue, avant de présenter la liste des exigences du modèle conceptuel de l'ISSdF.

2.1.1 La démarche de modélisation adoptée

Nous venons d'expliciter les attentes du modèle conceptuel de l'ISSdF ; un métamodèle ou une ontologie formelle étant moins adaptés à ce cadre (*Cf.* ANNEXE 1). Pour élaborer ce modèle conceptuel, nous avons choisi d'appliquer un principe réflexif et donc de le développer en suivant le canon de l'IS. Nous rédigerons sous forme d'exigences les attentes du modèle conceptuel de l'ISSdF. Les critères dérivés de ces exigences permettront de comparer l'état de l'art en matière de complétude, concision, respect de la problématique, etc. Une solution sera ensuite développée, qui concerne le modèle conceptuel proprement dit. Cette solution composite associera des modèles conceptuels existants, les normes de l'IS et de la SdF, ainsi que des articles académiques dont les auteurs se sont penchés sur le rapprochement entre ces deux domaines.

De plus, le modèle proposé doit suivre certaines règles. Tout concept qui nous semblera majeur sera ainsi défini comme une classe à part entière. Tout concept secondaire sera défini comme un attribut du concept majeur auquel il est associé. Par exemple, la définition du phénomène dangereux (*hazard*) dans l'ISO 26262:2011 est « *potential source of harm caused by malfunctioning behaviour of the item* ». Pour des raisons de clarté, nous avons limité le concept d'item à un système pour le phénomène dangereux. Cette définition permet donc de définir certains concepts et relations comme le montre la figure ci-dessous. Ensuite, les relations entre ces concepts pourront évoluer à l'aide d'autres relations provenant d'autres définitions.

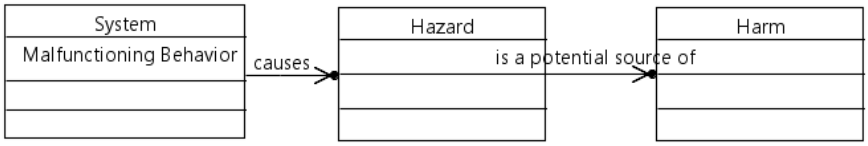


Figure 9 : Exemples de relations entre concepts

Contrairement à d'autres modèles conceptuels mentionnés dans les pages suivantes, le choix a été fait de ne pas prendre en compte les concepts englobants comme celui d'architecture fonctionnelle. En revanche, tous les concepts permettant de la définir seront pris en compte. Les concepts du modèle seront ensuite mis en forme afin que celui-ci soit lisible et aide à élaborer le processus d'ISSdF. Pour cela, il est possible de distinguer, parmi les concepts retenus, ceux qui serviront d'entrées d'un processus et ceux jouant le rôle de sorties.

2.1.2 Exigences du modèle conceptuel de l'ISSdF

Il est maintenant possible de définir les exigences pour le modèle conceptuel de l'ISSdF souhaité. Par commodité, après avoir défini les exigences, nous les avons classées par type : des exigences ontologiques, qui portent sur la nature, l'essence du modèle, des exigences normatives, de conformité aux normes, et des exigences pragmatiques, d'usage du modèle. Cependant, nous avons conservé la numérotation des exigences avant classification.

2.1.2.1 Exigences ontologiques

Exigence 1 — Le modèle conceptuel doit présenter les concepts clés de l'ISSdF.

Pour cela, il devra présenter les concepts de l'IS, ceux de la SdF et les liens permettant de coupler ces deux domaines, ce qui implique trois autres exigences que sont les exigences 1.1, 1.2 et 1.3

Exigence 1.1 — Le modèle conceptuel doit présenter les concepts d'Ingénierie Système.

Exigence 1.2 — Le modèle conceptuel doit présenter les concepts de sûreté de fonctionnement et couvrir au moins la sûreté (*safety*).

Exigence 1.3 — Le modèle conceptuel doit présenter les liens entre l'Ingénierie Système et la sûreté de fonctionnement.

Exigence 3 — Le modèle conceptuel doit être vérifié par la définition d'un processus d'ISSdF.

Exigence 4 — Le modèle conceptuel doit fournir un langage partagé et une cohérence sémantique entre les différents intervenants d'un projet de développement d'une architecture sûre de fonctionnement : architectes systèmes, ingénieurs en SdF, etc.

Exigence 5 — Le modèle conceptuel doit être articulé aux modèles et méthodes habituelles de la SdF. Les ingénieurs en SdF utilisent de nombreuses méthodes d'analyse fondées sur des concepts déjà bien définis. Si le modèle conceptuel proposé ne doit pas intégrer tous ces concepts ou toutes ces méthodes, il doit cependant être suffisamment pertinent pour un ingénieur en SdF pour faire en sorte qu'il l'utilise pour initier ensuite des analyses plus approfondies.

2.1.2.2 Exigences normatives

Exigence 1.4 — Le modèle conceptuel doit être cohérent avec la norme ISO 15288:2008.

Exigence 1.5 — Le modèle conceptuel doit être cohérent avec la norme ISO 26262:2011 (secteur automobile)[8].

2.1.2.3 Exigences pragmatiques

Exigence 2.1 — Le modèle conceptuel doit être multivue. Cette exigence dérive de l'état de l'art en matière de modélisation de systèmes. Par ailleurs, dans la pratique de l'ingénierie, un architecte système s'occupant d'une activité bien précise cherchera à connaître à la fois les concepts de son périmètre et ceux qui sont connexes.

Exigence 2.2 — Le modèle conceptuel doit être concis et clair.

Exigence 2.3 — Le modèle conceptuel doit être spécifique à l'automobile et en particulier à PSA ; il doit être facilement rendu générique.

2.1.2.4 Récapitulatif des exigences

Le tableau 4 récapitule et numérote à l'aide d'une classification décimale les exigences (*Exi.*) du modèle conceptuel de l'ISSdF.

[8] *Le modèle conceptuel pourrait également être cohérent avec l'IEC 61508:2010 avec des modifications mineures étant donné le lien de filiation entre ces deux référentiels.*

Tableau 4 : Exigences du modèle conceptuel de l'ISSdF

Référence	Exigence
Exi. 1	Il doit présenter les concepts de l'ISSdF.
Exi. 1.1	Il doit présenter les concepts d'Ingénierie Système.
Exi. 1.2	Il doit présenter les concepts de sûreté de fonctionnement et couvrir au moins la sûreté (*safety*).
Exi. 1.3	Il doit présenter les liens entre l'Ingénierie Système et la sûreté de fonctionnement.
Exi. 1.4	Il doit être cohérent avec la norme ISO 15288:2008.
Exi. 1.5	Il doit être cohérent avec la norme ISO 26262:2011.
Exi. 2	Il doit être compréhensible.
Exi. 2.1	Il peut être multivue en respectant une logique entre les vues (par domaine, par raffinement, par activités, etc.) avec 20 concepts maximum par vue.
Exi. 2.2	Il doit être concis et clair.
Exi. 2.3	Il doit être spécifique à l'automobile et en particulier à PSA ; il doit être facilement rendu générique.
Exi. 3	Il doit être vérifié par la définition d'un processus d'ingénierie de systèmes sûrs de fonctionnement.
Exi. 4	Il doit fournir une cohérence sémantique entre les différents intervenants (Experts, Architectes Systèmes, Ingénieurs SdF), un langage commun partagé.
Exi. 5	Il doit être conforme avec les méthodes classiques et aux règles de l'art de l'IS et de la SdF.

2.2 ÉTAT DE L'ART

Les exigences énoncées précédemment vont permettre de guider l'état de l'art que nous entendons faire en matière de modèles conceptuels du système. Après avoir succinctement dressé une liste de huit modèles, nous déterminerons leur contexte d'usage, leur contenu, leur intérêt et leurs limites. Cela permettra ensuite de réaliser un bilan en s'appuyant sur des critères chiffrés associés à chaque exigence respectée totalement, partiellement ou non. Un tel bilan nous permettra de conclure à la nécessité de créer un nouveau modèle dont une partie associera les modèles étudiés.

2.2.1 Modèles conceptuels existants

Plusieurs travaux existent concernant la création de modèles traitant de la sémantique de l'Ingénierie Système en lien ou non avec la sûreté de fonctionnement. Nous avons distingué huit modèles qui semblent pertinents par rapport à l'objet de notre thèse. Il s'agit d'A2PNum (Deniaud, Bonjour, Micaëlli, & Loise, 2010), d'ATESST2 Concept (ATESST, 2010), du modèle de Chapurlat & Pfister (Pfister, Chapurlat, Huchard, Nebut, & Wippler, 2011), de deux modèles de Taofefinua (Chalé, et al., 2011) et (Taofifenua, 2012), du modèle de Chapurlat & Cornu (Cornu, 2012), du modèle d'Adedjouma (Adedjouma, 2012) et du modèle de Faisandier (Faisandier, 2014).

2.2.1.1 Modèle A2PNum (2010)

Comme l'expose Deniaud et al. (Deniaud, Bonjour, Micaëlli, & Loise, 2010), A2PNum, pour « *Architecture Produit / Process dans un environnement collaboratif et NUMérique* », a été élaboré entre 2007 et 2010. Ce modèle avait une vocation interne ; il ne concernait que PSA. Il devait servir de préalable, de base, pour construire une architecture système en prenant en compte la SdF. Ce modèle conceptuel distingue deux niveaux de développement : la spécification et la conception. Il permet de bien lire les distinctions et les liens entre les concepts de l'IS et ceux de la SdF, et ce à l'aide de couleurs.

Une des premières limites que nous pouvons évoquer concerne le choix des concepts de SdF. Ceux-ci proviennent uniquement de la norme ISO 26262:2011. Ils ne traitent donc que de la sûreté fonctionnelle. Certains concepts, comme celui de *Controllability*, sont spécifiques à l'automobile et ne peuvent être que difficilement rendus génériques. De plus, bien qu'une distinction entre les concepts de spécification et de conception soit faite, il ne sera pas aisé de se baser dessus pour définir un processus ISSdF. Comme en sont conscients ses créateurs, A2PNum ne se voulait qu'un premier pas vers l'ISSdF, et non un modèle complet, prêt à l'emploi.

2.2.1.2 Le métamodèle ATESST2 (2010)

Ce modèle présenté dans (ATESST, 2010) est issu du projet européen ATESST2 (*Advancing Traffic Efficiency and Safety through Software Technology*), fruit d'une collaboration entre des fabricants automobiles (Fiat, Volvo et Volkswagen), des équipementiers (Delphi, Mentor Graphics et Continental) et des institutions académiques (CEA LIST, The Royal Institute of Technology KTH Stockholm, Technische Universität Berlin, University of Hull). Le but d'ATESST2 est de traduire dans le langage de description d'architectures EAST-ADL les concepts de l'ISO 26262:2011. Pour ce faire, un métamodèle a été développé par les partenaires du projet. Ce métamodèle prend en compte de façon satisfaisante certains concepts d'IS utilisables pour mener à bien une analyse de sûreté. Cependant, il manque des concepts (*Mission,*

Function, Component,…), si bien qu'il est impossible de recourir à ATESST2 pour définir un processus ISSdF ou même disposer d'un modèle répondant aux exigences précédemment énoncées (Exigences 1.1, 1.4, 3 non respectées).

2.2.1.3 Le métamodèle de Chapurlat & Pfister (2011)

Le modèle développé dans (Pfister, Chapurlat, Huchard, Nebut, & Wippler, 2011) a été élaboré pour définir un patron de conception (*pattern*) d'un système en IS. Il s'agit donc d'un métamodèle intégrant tous les concepts nécessaires pour définir un système en IS. Cependant, il ne contient aucun concept relatif au processus de *Stakeholder Requirements Definition*, et aucun lien entre l'IS et la SdF. De la sorte, le métamodèle de Chapurlat et Pfister ne répond pas aux exigences présentées. Par contre, compte tenu de son exhaustivité pour ce qui concerne le domaine de l'IS, il pourra procurer une aide réelle pour vérifier si le modèle conceptuel de l'ISSdF que nous proposons est valide ou pas.

2.2.1.4 Les deux modèles de Taofefinua (2011)

Les modèles conceptuels évoqués ci-dessous résultent de la thèse CIFRE d'Ofaina Taofefinua adossée à un partenariat entre le CNAM et Renault (Taofifenua, 2012). La finalité des travaux de Taofefinua est d'avoir un processus de conception reposant sur une ontologie formelle. Celle-ci devrait faciliter la production de modèles et de documents de conception assurant la bonne coordination entre les processus d'IS et ceux de SdF.

Le premier modèle de Taofefinua est constitué d'une seule vue présentant les différents concepts clefs. Cette version du modèle conceptuel élaboré par (Chalé, et al., 2011) est très concise. S'il traite bien des relations entre l'IS et la SdF, les concepts de ce dernier domaine sont toutefois détachés des concepts d'IS. Par exemple, le concept de *Safety Goal* est dans un groupe *dependability* et non lié aux concepts permettant de définir la vue opérationnelle du système.

Le second modèle de Taofefinua (Taofifenua, 2012) développe l'ontologie formelle mentionnée ci-dessus. Ce second modèle a un objectif différent du nôtre, puisqu'il vise à formaliser les connaissances des experts en SdF. Il est composé de nombreuses vues montrant les concepts et les relations relatifs à un concept englobant comme celui d'architecture fonctionnelle. Toutefois, le niveau de précision du modèle n'est pas celui souhaité. En effet, alors que le modèle développé dans (Chalé, et al., 2011) est trop succinct, celui développé dans (Taofifenua, 2012) est très, voire trop, détaillé et touffu. Enfin, le modèle proposé ne permet pas de définir un processus ISSdF (Exigence 3).

2.2.1.5 Le métamodèle de Chapurlat & Cornu (2012)

Clémentine Cornu présente dans (Cornu, 2012) comment un langage est utile pour déployer des processus d'IS chez un constructeur aéronautique comme Airbus Helicopter. Ce langage a été modélisé par un « métalangage » composé de plusieurs vues renvoyant aux « Attentes et Exigences », « Processus », « Ressources », « Information », « Vérification et Validation » et « Évaluation et Comparaison ». L'utilisation des vues permet d'avoir un modèle plus lisible, car concis et clair. Cependant, le modèle proposé étant focalisé seulement sur l'IS, il n'intègre aucun concept de SdF.

2.2.1.6 Le métamodèle d'Adedjouma (2012)

Le modèle conceptuel d'Adedjouma a été élaboré dans une thèse intitulée *Requirements engineering process according to automotive standards in a model-driven framework*, élaborée en partenariat entre le CEA et l'équipementier automobile Delphi (Adedjouma, 2012). L'objectif d'Adedjouma est de rapprocher l'ISO 26262:2011 avec l'ISO 15504:2004 (ISO 15504, 2004); norme également connue par la dénomination SPICE (*Software Process Improvement and Capability dEtermination*). Ce dernier référentiel vise l'évaluation des processus de développement informatique. S'il reprend bien plusieurs concepts clefs de l'ISO 26262:2011, ce modèle se focalise sur l'informatique et la norme SPICE, c'est-à-dire sur un niveau trop détaillé de l'architecture d'un système.

2.2.1.7 Le modèle de Faisandier (2014)

Alain Faisandier présente dans un modèle conceptuel publié dans (Faisandier, 2014) les concepts majeurs de l'IS et leurs liens avec la sûreté. Celui-ci est découpé en plusieurs vues, elles-mêmes découpés en catégories. Ces vues sont alignées sur les processus d'IS. Ce modèle pourrait donc bien convenir à nos attentes. Cependant, il y a de nombreux concepts importants manquant en SdF (*Failure Mode*, ASIL…) et ceux-ci sont séparés des autres concepts de l'IS, ce qui a pour conséquence que les exigences 1.2, 1.3 et 1.5 ne sont pas complètement satisfaites.

2.2.2 Bilan et comparaison des modèles présentés

Afin de déterminer si un des modèles de l'état de l'art répond aux exigences que nous avons énoncées, il convient d'associer à chacune d'elle des critères, comme cela est présenté dans le tableau ci-dessous. Cela permettra de définir une moyenne pour chaque modèle conceptuel existant en pondérant les différents critères (3 pour les exigences principales, 2 pour les exigences secondaires, 1 pour les exigences de forme).

Tableau 5 : Critères associés à chaque exigence

Référence	Exigence	Critère associé
Exi. 1	Il doit présenter les concepts de l'ISSdF.	Liens
Exi. 1.1	Il doit présenter les concepts d'Ingénierie Système.	Couverture des concepts IS
Exi. 1.2	Il doit présenter les concepts de sûreté de fonctionnement et couvrir au moins la sûreté (safety).	Couverture des concepts SdF
Exi. 1.3	Il doit présenter les liens entre l'Ingénierie Système et la sûreté de fonctionnement.	Présence des liens entre IS et SdF
Exi. 1.4	Il doit être cohérent avec la norme ISO 15288:2008.	Cohérence avec ISO 15288:2008
Exi. 1.5	Il doit être cohérent avec la norme ISO 26262:2011.	Cohérence avec ISO 26262:2011
Exi. 2	Il doit être compréhensible	Compréhensibilité
Exi. 2.1	Il peut être multivue en respectant une logique entre les vues (par domaine, par raffinement, par activités, etc.) avec 20 concepts maximum par vue.	Modèle multivue
Exi. 2.2	Il doit être concis et clair	Concision
Exi. 2.3	Il doit être spécifique à l'automobile et en particulier à PSA ; il doit être facilement rendu générique	Généricité
Exi. 3	Il doit être vérifié par la définition d'un processus d'ingénierie de systèmes sûrs de fonctionnement.	Possibilité de vérifier par processus
Exi. 4	Il doit fournir une cohérence sémantique entre les différents intervenants (Experts, Architectes Systèmes, Ingénieurs SdF), un langage commun partagé.	Cohérence sémantique
Exi. 5	Il doit être conforme avec les méthodes classiques et aux règles de l'art de l'IS et de la SdF	Alignement avec modèles et méthodes de SdF et d'IS

En utilisant la liste des critères définis précédemment, il a été possible de comparer les différents modèles conceptuels présentés ci-dessus. Comme on peut le visualiser sur la dernière colonne de conclusion globale du tableau ci-dessous, aucun modèle ne satisfait tous les critères et aucun modèle n'est dominant (aucun ne surclasse les autres sur tous les critères).

Tableau 6 : Comparaison des différents modèles conceptuels

Nom du Projet (ou des auteurs par défaut)	IS	SdF	Liens IS&SdF	Conforme à norme IS	Conforme à norme SdF	Conclusion Liens	Multi-vues	Concision	Généricité	Conclusion Compréhensibilité	Vérification processus	Cohérence sémantique	Aligner avec méthodes SdF	Conclusion Globale
Projet A2PNum	80	80	80	50	100	78	80	80	80	80	55	85	10	62
Modèle Taofefinua (article)	60	50	50	0	90	50	0	100	20	40	0	10	0	20
Modèle Taofefinua (thèse)	75	75	50	0	100	60	100	25	30	52	70	75	0	52
ATESST2 Concept	30	80	30	0	100	48	100	75	40	72	60	80	50	62
Modèle Chapurlat/Pfister	90	0	0	80	0	34	0	50	50	34	50	70	0	38
Modèle Chapurlat/Cornu	90	0	0	80	0	34	50	60	50	54	50	60	0	40
Modèle Adedjouma	20	90	0	0	100	42	0	80	40	40	50	60	0	39
Modèle Faisandier	75	75	50	50	0	50	50	90	80	74	70	50	75	64

La comparaison effectuée ci-dessus permet de déterminer les deux modèles les plus pertinents : il s'agit d'A2PNum et de celui d'Alain Faisandier. Bien que répondant moins aux critères, ATESST2 peut être pris comme référence pour la modélisation des concepts de l'ISO 26262:2011.

Le panorama des modèles que nous venons de dresser a montré que si des solutions existent pour ce qui concerne le couplage de l'IS et de la SdF, celles-ci ne satisfont toutefois pas aux exigences énoncées. Il convient donc de développer un nouveau modèle conceptuel de l'ISSdF.

2.3 PROPOSITION D'UN MODÈLE CONCEPTUEL DE L'ISSDF

Cette section permettra de décrire l'une des contributions de ce mémoire de thèse, à savoir un modèle conceptuel de l'ISSdF. Celui-ci repose sur des vues intégrant des concepts. Aussi, après avoir explicité ces vues, nous présenterons les concepts des différentes vues, puis nous définirons leurs relations. Pour chaque vue, des exemples des différents concepts seront donnés afin de faciliter la compréhension du modèle conceptuel proposé.

2.3.1 Choisir & construire un modèle multivue

Avant de rentrer dans le détail de l'exposé du modèle conceptuel de l'ISSdF, il est nécessaire de définir les différentes vues afin de respecter l'exigence 2.2. Le choix réalisé dans (Chalé, et al., 2011) n'est pas pertinent dans notre cas. Le modèle de Chalé rassemble tous les concepts de SdF dans une vue unique, ce qui ne clarifie pas les relations entre IS et SdF. Dès lors, quelles vues retenir ? Dans le précédent chapitre, nous avons mentionné le fait que l'IS repose sur quatre processus génériques : *Stakeholder Requirements Definition* (SRD)*, Requirements Analysis* (AR)*, Functional Architecture Design* (FAD), et *Physical Architecture Design* (PAD).

Les deux premiers processus portent sur la vue externe du système. On peut donc les regrouper en une seule vue appelée *Operational and System Requirements Definition* (O&SRD). Le terme *Operational* se réfère à l'analyse des besoins des parties prenantes (*stakeholders*) en situation opérationnelle. Le terme *System Requirements* est, quant à lui, conforme à la définition énoncée dans l'ISO 15288:2008 ou le *SEBoK*. Dès lors, les concepts de la SdF doivent être distribués selon les vues retenues. Étant donné le nombre de concepts différents dans les deux domaines, nous avons préféré faire une vue séparée pour chacun des processus d'architecture (FAD et PAD). À chaque processus d'IS correspondra ainsi une vue du modèle conceptuel de l'ISSdF. Celles-ci ne sont pas isolées ; il est possible de les associer de différentes façons. Nous avons choisi de les associer par la traçabilité des exigences, qui est une propriété importante autant en IS qu'en SdF. Une fois ce choix fait, il a fallu suivre une démarche d'élaboration d'une vue. Cette procédure est décrite dans le diagramme suivant.

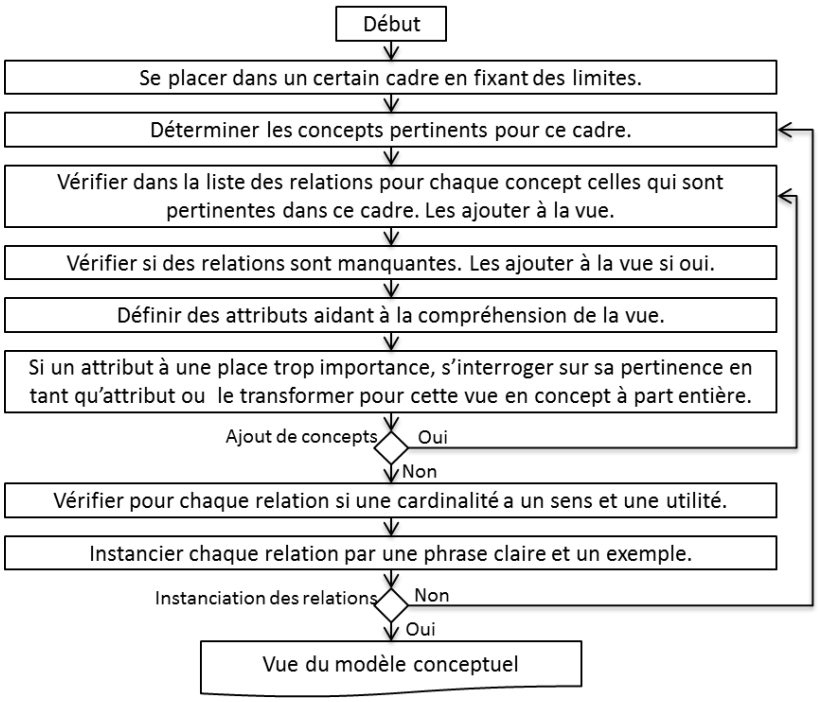


Figure 10 : Proposition d'une démarche d'élaboration d'une vue

Les vues et leur mode d'élaboration précisés, nous pouvons maintenant détailler celles-là une par une. Pour l'exposé de chacune d'elles nous suivrons toujours le même plan, avec une présentation panoramique de la vue présentant le cadre et les limites, puis une définition des différents concepts constitutifs, une explicitation de leurs relations. Enfin, si possible, l'ensemble de la vue sera illustré à l'aide d'un exemple.

2.3.2 La première vue, centrée sur L'O&SRD (Operational and System Requirements Definition)

2.3.2.1 Panorama de la vue

La première vue du modèle conceptuel de l'ISSdF peut être représentée à l'aide du diagramme structurel suivant.

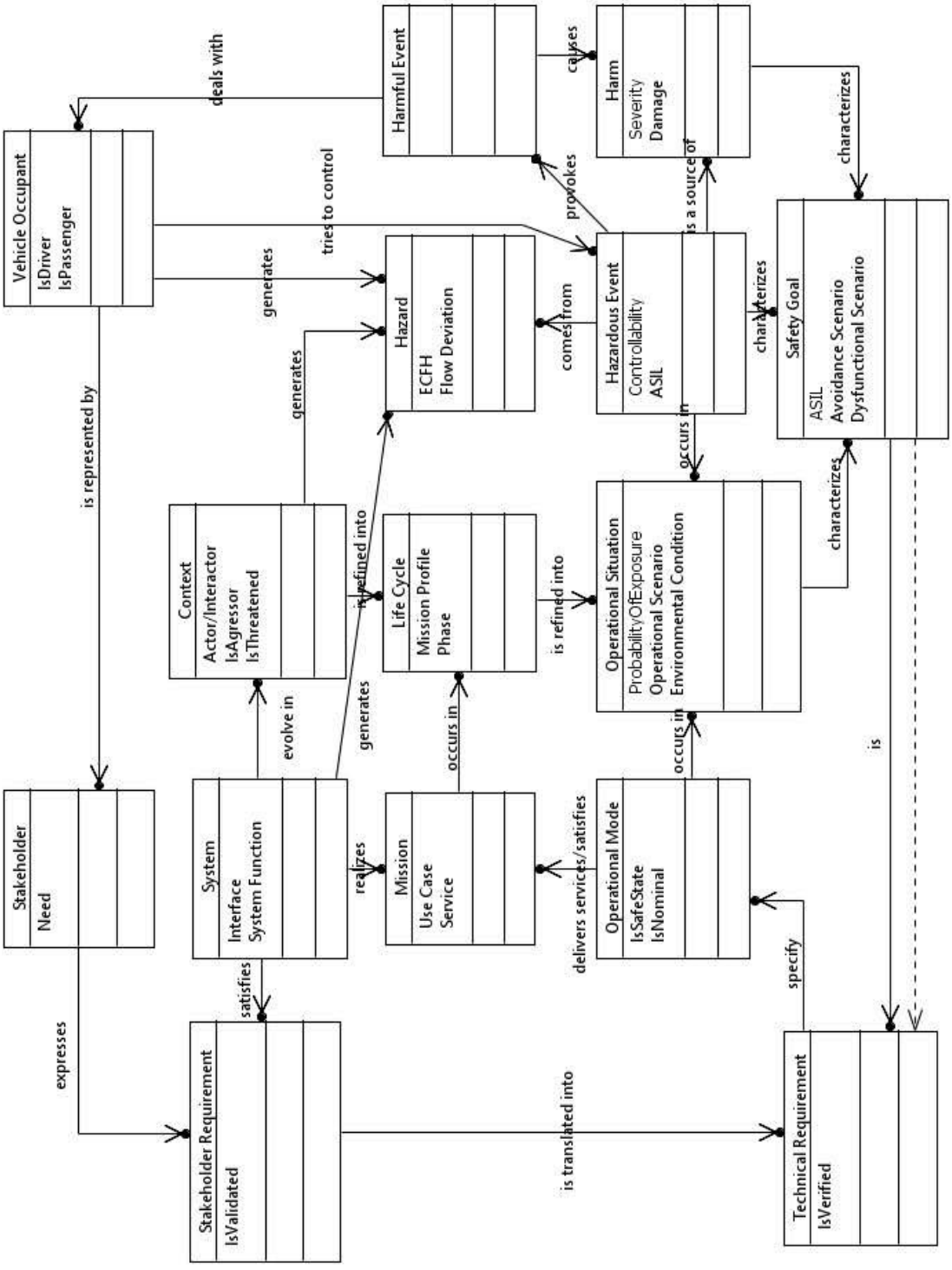


Figure 11 : Modèle conceptuel de l'ISSdF : vue centrée sur l'O&SRD

2.3.2.2 Description des concepts constitutifs[9]

Parties prenantes (Stakeholder) — Selon l'ISO 15288:2008, une partie prenante est une « *partie ayant un droit, une part ou une prérogative qui fait que le système ou certaines de ses propriétés doivent satisfaire les besoins ou les attentes de cette partie* ». Les parties prenantes peuvent aussi exprimer des peurs quant au comportement dégradé ou dysfonctionnel du système. Dans le cas de l'automobile, le conducteur ou le passager sont des parties prenantes. Nous avons distingué ce premier car ses actions sont importantes en matière de SdF. Par une faute ou une erreur de conduite, il peut provoquer un dysfonctionnement ou un dommage. Grâce à ses compétences, il peut assurer la contrôlabilité du véhicule en cas de problème. Du fait de ce rôle prééminent, le conducteur est donc intégré dans le périmètre du système étudié. Enfin, une exigence particulière de sûreté peut être énoncée explicitement par une partie prenante ou traduite de son besoin par le concepteur. La SdF raffine une telle exigence. Dans ce domaine, le substantif "sûreté" renvoie ainsi à un ensemble de propriétés dérivées caractérisant un système sûr. Comme cela est énoncé dans (Faisandier, 2014), ce système doit ainsi résister à des agressions de l'environnement (immunité), ne pas s'altérer du fait de son fonctionnement (intégrité), et garantir la protection de l'environnement dans lequel il est intégré (innocuité).

Exigence des parties prenantes (Stakeholders' Requirement) — Selon l'ISO 15288:2008, une exigence des parties prenantes est la formalisation d'un besoin des parties prenantes.

Système (System) — Selon la norme ISO 15288:2008, le système est un « *ensemble d'éléments en interaction, organisés pour atteindre un ou plusieurs résultats déclarés* ». Cette définition est plus large que celle proposée par l'ISO 26262:2011. De plus, il est préférable d'utiliser le concept de système à celui d'item. Pour le SEBoK, « *A function is defined by the transformation of input flows to output flows, with defined performance* ». Une fonction système assure une transformation de flux (à travers le système) de Matière, Énergie, Information provenant d'un élément de l'environnement vers un autre élément de l'environnement, avec un niveau (ou objectif) de performance défini.

Contexte (Context) ou Environnement (Environment) — Nous considérons ces deux termes comme équivalents. L'environnement est constitué de l'ensemble des interacteurs du système.

[9] *Nous avons choisi de reprendre la définition en Anglais dans notre mémoire quand la source est écrite en Anglais pour éviter de propager des ambiguïtés ou approximations de traduction. À titre d'exemple, a safe state se traduit en Français par mode sûr, mais le terme Anglais fait référence.*

Cycle de vie (Life Cycle) — Selon la norme NF EN ISO 14040:2006 (NF EN ISO 14040, 2006), le cycle de vie désigne les « *phases consécutives et liées d'un système de produits, de l'acquisition des matières premières ou de la génération des ressources naturelles à l'élimination finale* ». Nous définissons le cycle de vie d'un produit comme l'ensemble des phases par lesquelles il passe. Notons que l'IEEE 1220:2005 définit ce cycle de vie non par rapport au produit, mais par rapport aux parties prenantes. Il est ainsi défini comme « *the system or product evolution initiated by a perceived stakeholder need through the disposal of the products* ».

Mission — Étant donné que ce concept n'est pas défini dans l'ISO 15288:2008, nous avons repris celle indiquée dans le glossaire de l'AFIS (AFIS, 2004), la mission est définie comme « *the specific task, duty or function defined to be accomplished by a system* ».

Situation, Scénario, Mode Opérationnel (Operational Situation, Scenario, Operational Mode) — Que ce soit dans les normes d'IS ou de SdF, les concepts d'événement, de situation, de scénario et de mode présentent d'évidentes proximités sémantiques. Avant de proposer une définition pour ces concepts clefs, nous proposons de dresser un bref état de l'art. Dans l'IEEE 1220:2005, les scénarios opérationnels « *define the range of the anticipated uses of system product(s). For each operational scenario, the project defines expected interactions with the environment and other systems; human tasks and task sequences; and physical interconnections with interfacing systems, platforms, or product*s ». Pour l'EIA 632:2003, le scénario opérationnel est « *a sequence of events expected during operation of system products. Includes the environmental conditions and usage rates as well as expected stimuli (inputs) and responses (outputs)* ». Le *SEBoK*, quant à lui, définit le concept générique de scénario de la façon suivante : « *(1) Description of an imagined sequence of events that includes the interaction of the product or service with its environment and users, as well as interaction among its product or service components. (ISO/IEC 2011), (2) A set of actions or functions representing the dynamic of exchanges between the functions allowing the system to achieve a mission or a service […], (3) Stories which describe the expected utilization of the future system in terms of actions* ». Le scénario opérationnel est un scénario spécifique, défini comme « *an operational situation/configuration of a system characterized by its active functions. Inside an operational mode the system can perform specific operational scenarios, and so activate the corresponding functions. (same as state)* » (Faisandier, 2014). Enfin, dans l'ISO 26262:2011, nous trouvons la définition non du scénario opérationnel, mais de la situation opérationnelle comme « *a scenario that can occur during a vehicle's life* ». Cette dernière norme admet donc un équivalent entre un scénario et une situation, ce qui ne nous convient pas car ce sont deux concepts différents.

Vu le foisonnement des définitions, il convient de lever certaines ambiguïtés. Aussi proposons-nous, dans le modèle conceptuel de l'ISSdF, les définitions suivantes :

1. le *Mode Opérationnel* correspond à un état stationnaire du système dans lequel celui-ci fonctionne. Certains paramètres du système peuvent évoluer, mais globalement, il reste dans un état identique. Ainsi, le véhicule peut être en roulage (mode). Sa vitesse peut changer, mais il sera toujours dans le même mode, qui est le roulage. Un mode opérationnel principal peut être raffiné en plusieurs modes opérationnels dérivés. Le mode roulage peut être ainsi raffiné en roulage autoroutier (vitesse élevée), roulage en ville (vitesse basse), en départementale sinueuse, etc. ;
2. les *Conditions Environnementales* sont l'ensemble des conditions stationnaires de l'environnement. Il peut également y avoir plusieurs niveaux de raffinement de ces conditions environnementales ;
3. le S*cénario Opérationnel* est une description des échanges entre le système et son environnement. Il est donc constitué d'un ensemble d'actions et d'événements. Il peut être décrit en traçant les échanges cycliques entre le système et son environnement. L'association du mode opérationnel du système et des conditions environnementales est une *situation opérationnelle*. Le changement du mode opérationnel du système ou la modification des conditions environnementales provoquent un changement de la situation opérationnelle.

Exigence Technique (Technical Requirement) — Ce concept est aussi nommé exigence système. Une exigence est « *a statement that identifies a system, product or process characteristic or constraint, which is unambiguous, clear, unique, consistent, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability.* » (INCOSE, 2010).

Phénomène Dangereux (Hazard) — D'après l'ISO 26262:2011, « *le phénomène dangereux est une source potentielle de préjudice due à un mauvais fonctionnement du système* ». Pour la norme ARP 4761:1996, il est « *a potentially unsafe condition resulting from failures, malfunctions, external events, errors, or a combination thereof* ». De leur côté, (Haworth & Symmons, 2002) considèrent que : « *in terms of hazards to road users, any object, situation, occurrence or combination of these that introduce the possibility of the individual road user experiencing harm should be included. Hazards may be obstructions in the roadway, a slippery road surface, merging traffic, weather conditions, distractions, a defective vehicle, or any number of other circumstances* ». En revanche, comme le précise le guide 51 (Guide 51, 1999): « *the term hazard can be qualified in order to define its origin or the nature of the expected harm (e.g. electric shock hazard, crushing hazard, cutting hazard, toxic hazard, fire hazard, drowning hazard)* ». Le concept de phénomène dangereux (*Hazard*) admet donc différentes acceptions. Bien qu'il soit fréquemment traduit par *Danger*, il existe une nuance entre ces deux termes. Aussi le traduisons nous comme *Phénomène Dangereux*, et ce conformément à ce qui

est indiqué dans le guide 51 de l'ISO. Par ailleurs, nous distinguons deux types de phénomènes dangereux : ceux provenant du système et ceux externes. Par la suite, nous nommerons *Phénomène Dangereux (Hazard ou System Hazard)* ce premier type. Le second type sera considéré comme l'apparition de conditions environnementales favorables (ou propices) au danger (ECFH). De plus, dans le cadre de l'ISO 26262:2011, les phénomènes dangereux pris en considération proviennent uniquement d'un mauvais fonctionnement du système. Il est important d'y inclure les phénomènes dangereux provenant d'un fonctionnement non prévu, également appelés **Menaces** (*Threat*) ainsi que les possibles **agressions** (*Attack*) de l'environnement sur le système. Cela permettra d'avoir un meilleur périmètre d'étude de la sûreté de fonctionnement que celui prévu par la norme ISO 26262:2011.

Préjudice (Harm) — Pour l'ISO 26262:2011, « *le préjudice est une blessure physique ou une atteinte à la santé des personnes* ».

Événement Dangereux et Événement Critique (Hazardous Event, Harmful Event) — L'IEC 61508:1998 propose deux définitions : « *hazardous situation: circumstance in which a person is exposed to hazard(s) […] hazardous event: hazardous situation which results in harm* ». L'ISO 26262:2011, quant à elle, énonce : « *Hazardous Event is a combination of a hazard and an operational situation* ». Enfin, selon le guide 51 (Guide 51, 1999) : « *harmful event: occurrence in which a hazardous situation results in harm* » (traduit en évènement dangereux) « *hazardous situation: circumstance in which people, property or the environment are exposed to one or more hazards* ». Afin de lever l'ambiguïté entre ces deux concepts, nous allons les interpréter en reprenant les définitions du modèle conceptuel de l'ISSdF retenues dans les pages précédentes. Nous avons évité le terme d'évènement redouté dont l'interprétation peut être ambigü. En effet, selon les cas dans la littérature, il recouvre les concepts de phénomène dangereux (*Hazard*), évènement dangereux (*Hazardous Event*) ou évènement critique (*Harmful Event*), concepts que nous avons clairement distingués dans ce mémoire.

Un phénomène dangereux (*Hazard*) est un événement permettant de passer d'une situation opérationnelle dite nominale à une situation opérationnelle potentiellement dangereuse. Comme l'indique l'ISO 26262:2011, le phénomène dangereux résulte d'un mauvais comportement du système, et ceci quelles que soient les conditions environnementales. La présence de certaines conditions environnementales ou la modification de celles-ci conduit à une situation dangereuse. L'apparition de ces conditions environnementales favorables au danger (*Environmental Conditions Favorable to Hazard - ECFH*) permet de passer d'une situation opérationnelle potentiellement dangereuse à une situation opérationnelle dangereuse. Si celles-ci étaient présentes dans la situation opérationnelle nominale, un phénomène dangereux permet alors de passer de cette situation à la situation opérationnelle dangereuse (*Hazardous Situation*). Les propositions que nous venons de faire nous permettent de définir l'événement dangereux (*Hazardous Event*) comme l'apparition d'un phénomène dangereux avec certaines conditions environnementales favorables à ce phénomène. Une fois que nous

sommes dans la situation dangereuse, le danger est de passer dans une situation avec préjudice (*Harmful Situation*). L'événement provoquant ce passage est nommé événement critique (*Harmful Event*).

Pour clarifier l'enchaînement proposé, de prime abord peu évident, de concepts et d'occurrences, nous pouvons recourir à la machine à états (*state machine*) du système étudié présentée dans la Figure 12. Une autre représentation de cet enchaînement peut être réalisée à l'aide de plusieurs machines à états, comme le montre la Figure 13.

Une machine décrit le comportement du système étudié et les évolutions entre les différents modes possibles. L'environnement peut être décrit par plusieurs machines à état. Pour un phénomène dangereux, nous pouvons définir des conditions favorables à celui-ci. Nous pouvons donc définir une machine à états dont la modification de l'état est due à la modification d'états des machines de l'environnement. Le concept de *Safety Mechanism* sera défini dans la partie suivante car il est construit au niveau organique. Cependant, afin d'avoir un diagramme complet, il nous a semblé nécessaire de l'indiquer.

Sur cette dernière figure, les différentes machines à états des éléments du contexte situés en bas à droite permettent de modifier l'état en haut à droite des conditions environnementales. Cela va influer sur les modes et états du système et permet donc de reconstituer les différentes situations opérationnelles. La première figure facilite la spécification des différentes situations opérationnelles. La seconde aide à simuler le contexte du système en s'appuyant sur le profil de mission.

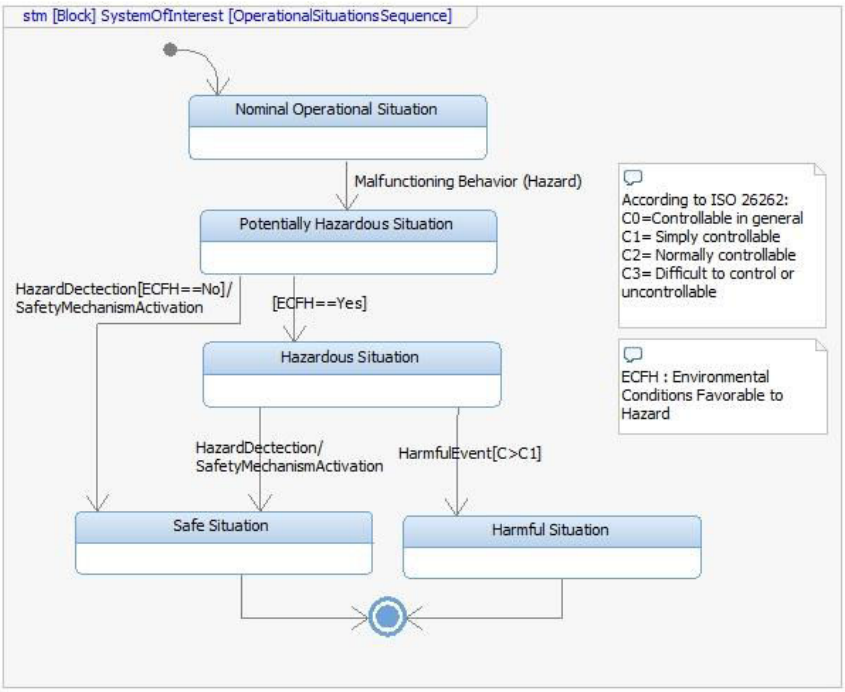


Figure 12 : Description par des machines à état – Une machine à état pour le système

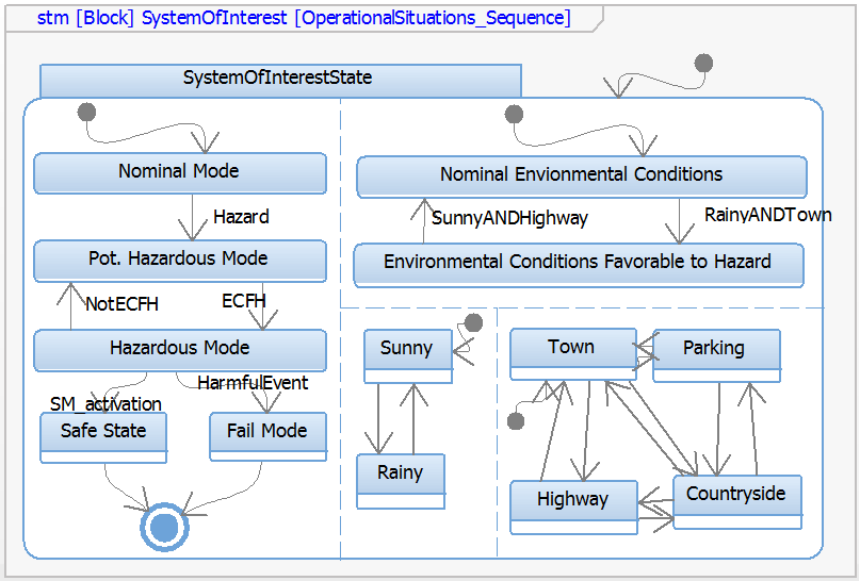


Figure 13 : Description par association de plusieurs machines à état

Scénario d'évitement (Avoidance Scenario) — Nous définissons le scénario d'évitement comme la description d'un événement (ou d'une succession d'événements) permettant d'éviter un préjudice lorsqu'un danger apparaît.

Safety Goal — Selon l'ISO 26262:2011, le *Safety Goal* (terme non traduit) est une exigence de sûreté de niveau système résultant de l'activité *Hazard Analysis and Risk Assessment*. Le *Safety Goal* permet de spécifier les scénarios d'évitement d'une situation opérationnelle avec préjudice. Cette exigence va donc permettre de spécifier le comportement souhaité du système en cas d'apparition du phénomène dangereux. Cette spécification peut être une exigence textuelle et/ou une extraction de modèles.

2.3.2.3 Explicitation des relations entre concepts de la vue

Les occupants du véhicule (conducteur et passager) éprouvent des besoins (*Needs*) relatifs au système (exemple : avoir besoin d'une automobile pour se déplacer les fins de semaines). Ces contraintes peuvent être explicitées et traduites par le concepteur en exigences des parties prenantes (*Stakeholder Requirement*), qui seront ensuite traduites en exigences techniques (*Technical Requirement*). Le système étudié remplira ainsi des missions dans un certain environnement afin de satisfaire les exigences des parties prenantes retenues par le concepteur. L'environnement potentiellement présent tout au long de la vie du système peut être explicité en un cycle de vie auquel on attache un profil de mission. Le système comporte des modes opérationnels qui satisferont ses missions. Ceux-ci seront activés dans certaines conditions environnementales que l'on associera par des situations opérationnelles décrites par des scénarios opérationnels.

Concernant le côté dysfonctionnel de la première vue du modèle conceptuel de l'ISSdF proposé, on peut tout d'abord dire que des phénomènes dangereux (*Hazards*) proviennent d'un mauvais fonctionnement interne du système, d'un occupant du véhicule, ou de l'environnement. Un phénomène dangereux associé à certaines conditions environnementales est propice à un événement dangereux (*Hazardous Event*). Cet événement dangereux pourra causer un événement critique (*Harmful Event*) qui provoquera un préjudice (*Harm*) pour les occupants du véhicule. Afin de spécifier l'absence d'événements dangereux, il faut spécifier, puis caractériser, des *Safety Goals*. L'ASIL, métrique associé à un *Safety Goal*, est déterminé à partir de trois critères : la probabilité d'être dans une situation opérationnelle potentiellement dangereuse, la contrôlabilité de cet événement dangereux par un occupant du véhicule et la sévérité des préjudices en cas de survenue de l'événement critique.

Les concepts et les relations de la première vue (vue dite O&SRD) du modèle de l'ISSdF proposé étant décrits, il est possible de les illustrer par un cas particulier.

2.3.2.4 Illustrations de la vue O&SRD

L'exemple choisi est celui d'une défaillance d'un véhicule roulant sur autoroute entraînant une accélération intempestive. Les occurrences des concepts présentés ci-dessus sont listées dans le tableau ci-dessous.

Tableau 7 : Exemple de la vue O&SRD du modèle conceptuel de l'ISSdF

Concept	Niveau Véhicule
Stakeholder	Conducteur, constructeur, …
Vehicle occupant	Conducteur, Passagers
Stakeholder Requirement	Le conducteur souhaite pouvoir contrôler la vitesse de son véhicule.
System	Véhicule
Context	Route, Signalisation, Conducteur
Mission	Le véhicule doit pouvoir transporter des passagers et /ou des charges d'un point A à un point B en toute sécurité.
Life Cycle	Utilisation Client
Hazard	Accélération intempestive du véhicule
Harm	blessure du conducteur et des passagers
Harmful Event	Choc du véhicule contre un véhicule suivi
Hazardous Event	Accélération intempestive du véhicule en roulage
Operational Mode	Roulage
Operational Situation	Le conducteur conduit sur route en gérant son allure.
Technical Requirement	Le véhicule doit suivre la consigne de vitesse demandée par le conducteur.
Safety Goal	L'accélération intempestive du véhicule (> xx m/s²) due à une défaillance interne doit être extrêmement improbable

Afin de faciliter la compréhension de certains concepts de SdF présents dans la première vue du modèle conceptuel de l'ISSdF, nous avons décrit l'accélération intempestive à l'aide de la machine à états suivante. Elle décrit la séquence redoutée de situations opérationnelles. Alors que le véhicule est dans une situation opérationnelle nominale de roulage (*driving*), un phénomène dangereux (*hazard*) lié à un mauvais comportement du véhicule (accélération intempestive), place le véhicule dans une situation potentiellement dangereuse. Si des conditions environnementales favorables au danger, comme la présence d'un obstacle, sont présentes, le véhicule passe dans une situation dangereuse correspondant à une accélération intempestive pendant roulage avec présence proche d'un obstacle. Si l'évènement dangereux est détecté, un scénario d'évitement sera défini afin de placer le véhicule dans une situation sûre. Dans le cas contraire, l'évènement critique (*harmful event*) peut survenir, ici sous la forme d'un choc, menant à une situation avec préjudice.

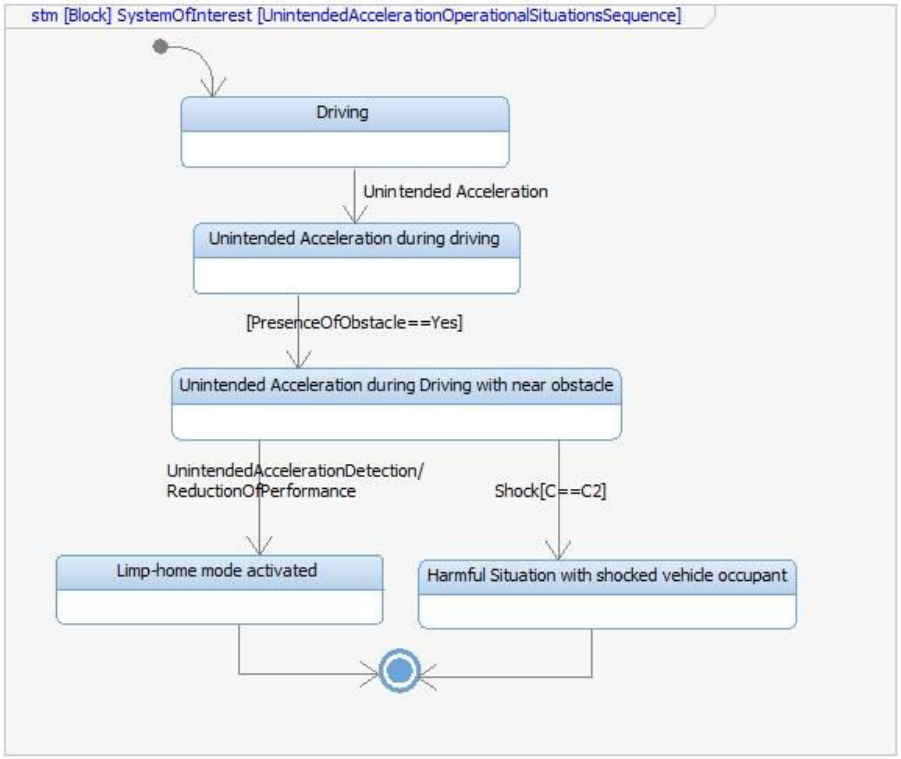


Figure 14 : Application de la Figure 11 pour l'accélération intempestive

2.3.3 La deuxième vue, centrée sur la FAD (*Functional Architecture Design*)

2.3.3.1 Panorama de la vue

La deuxième vue du modèle conceptuel de l'ISSdF, centrée sur la *Functional Architecture Design* (FAD), peut être représentée à l'aide du diagramme structurel suivant. Cette vue rassemble les concepts fonctionnels et dysfonctionnels relatifs à la conception de l'architecture fonctionnelle du système. Cependant, comme énoncé auparavant, le concept d'architecture fonctionnelle ne sera pas présent dans la vue car il s'agit d'un concept englobant. Le trait en pointillé distingue les concepts provenant de la vue précédente (première vue, dite O&SRD) de ceux propres à la vue associée à la FAD. Ces premiers concepts ayant été définis, ils ne seront donc pas explicités dans les pages suivantes.

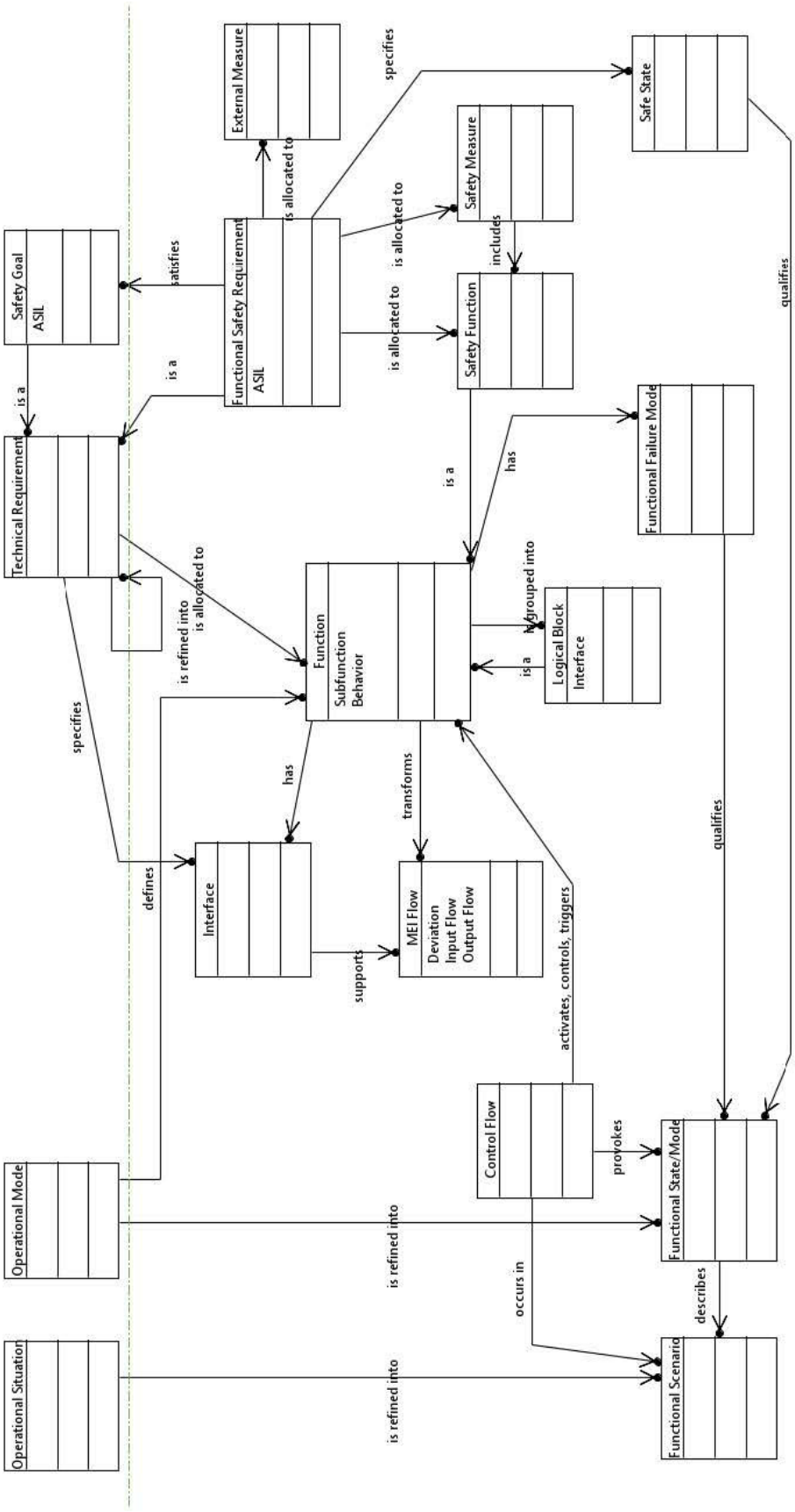


Figure 15: Modèle conceptuel de l'ISSdF : vue centrée sur la FAD

2.3.3.2 Description des concepts constitutifs

Interface, Flux MEI (Interface, MEI Flow) — Selon Alain Faisandier (Faisandier, 2014), « *An interface includes generally both functional and physical elements. Functional and physical interfaces are distinct. A functional interface is constituted of an input flow or an output flow between two functions so that they may exchange material, energy, and/or information*. ». Selon l'ISO/IEC 2382:1993 (ISO/IEC 2382, 1993), une interface est définie comme « *a shared boundary between two functional units, defined by various characteristics pertaining to the functions, physical signal exchanges, and other characteristics* ». Sa fonction est de permettre l'échange des *Flux MEI* (Matière, Énergie, Information) transitant entre les fonctions du système. Dans le modèle conceptuel proposé, le concept d'interface se réfère uniquement au niveau fonctionnel comme l'explicite la norme ISO/IEC 2382:1993 (ISO/IEC 2382, 1993). Par contre, elle sera matérialisée par des ports et des liens au niveau physique.

Flux de contrôle (Control Flow) — Nous le définissons comme un flux spécifique (de contrôle ou de commande) entre les fonctions de contrôle-commande et les fonctions techniques du système.

Fonction (Function) — Selon la norme EIA 632:2003 (EIA 632, 2003), « *une fonction est une tâche ou activité exécutée pour atteindre un résultat attendu* ».

Scénario Fonctionnel (Functional Scenario) — Pour le *SEBoK*, *« a scenario of functions is a chain of functions that are performed as a sequence and synchronized by a set of control flows to work to achieve a global transformation of inputs into outputs, […]. A scenario of functions expresses the dynamic of an upper level function* » (Pyster & Olwell, 2013). Un scénario fonctionnel devient pour un sous-système (ou un bloc logique) son scénario opérationnel. On obtient ainsi une structure imbriquant les scénarios opérationnels qui est construite de façon récursive.

Mode Fonctionnel (Functional Mode) — Nous proposons la définition suivante : dans un mode fonctionnel, la fonction sera activée avec un certain niveau de performance (donc si ce niveau = 0, la fonction n'est pas active). Par extension, dans un mode fonctionnel, on retrouve les fonctions du système activées (ou non) avec un certain niveau de performance, ce qui nous permet de caractériser les différents modes fonctionnels (nominal, dégradé, de protection ou sûr, etc.). Le mode spécifié est le mode nominal. Lorsque le niveau de performance est acceptable mais différent de celui spécifié, la fonction est alors dans un mode dégradé acceptable. En cas contraire, elle est dans un mode défaillant. Pour revenir à un niveau acceptable, la fonction peut aller dans un *safe state* qui est un mode nominal ou un mode dégradé acceptable. Bien que souvent confondus et qu'ils représentent un aspect similaire d'un système (d'où la représentation dans un même bloc dans le modèle conceptuel), nous faisons la

distinction entre les concepts de mode et état. Pour un système ou une fonction, 4 états sont possibles : ON, OFF, FAILED et SUSPEND. Quand la fonction est active (dans un état ON), plusieurs modes peuvent être possibles selon le type de fonctionnement souhaité.

Bloc Logique (Logical Block) — Nous le définissons comme un regroupement de fonctions afin de pouvoir les allouer à un constituant.

Exigence Fonctionnelle de Sûreté (*Functional Safety Requirement*) — Pour l'ISO 26262:2011, le *Functional Safety Requirement* permet une « *specification of implementation-independent safety behaviour, or implementation-independent safety measure, including its safety-related attributes* ». Ces exigences dérivent des *Safety Goals*. Ils spécifient une solution à un danger possible.

Mesure de Sécurité, Mesure Externe, Solutions de sûreté, Fonction de Sécurité (Safety Measure, External Measure, Safety Mechanism, Safety Fonction) — Ces quatre concepts proviennent de l'ISO 26262:2011. Ainsi, la *Mesure de Sécurité* est définie par : « *activity or technical solution to avoid or control systematic failures and to detect random hardware failures or control random hardware failures, or mitigate their harmful effects* », alors que la Mesure Externe décrit la « *measure that is separate and distinct from the item which reduces or mitigates the risks resulting from the item* ». Elle est une solution satisfaisant les *Exigences Fonctionnelles de Sûreté* (*Functional Safety Requirement*). Certaines d'entre elles sont définies dans des normes, des lois, des codes, comme dans le cas des panneaux routiers. En cas d'absence de mesure externe suffisante, le concepteur devra alors mettre en œuvre des mesures internes (*Safety Measures*). Lorsque celles-ci prennent la forme de solutions techniques, alors ces mesures sont appelées par l'ISO 26262:2011 *Safety Mechanism*. Pour éviter des confusions entre le domaine (ou niveau, vue) fonctionnel et le domaine (ou niveau, vue) organique, nous avons choisi de conserver le terme *Safety Mechanism* pour le domaine organique et d'utiliser le concept de *Safety Function* pour le domaine fonctionnel, concept qui est présent dans la norme IEC 61508:2010.

Mode de Défaillance Fonctionnel (Functional Failure Mode, FFM) — L'ISO 26262:2011 définit le mode de défaillance comme une « *manner in which an element or an item fails* ». Il est habituel, en SdF, de retenir cinq modes de défaillance « *génériques* », qui sont : la perte, l'absence, le maintien, l'intempestif et la dégradation. Le schéma ci-dessous compare le mode fonctionnel avec ces cinq occurrences du mode de défaillance fonctionnel (Desinde, 2006). Notons qu'il ne faut pas confondre le mode de défaillance fonctionnel avec une déviation de flux (M,E,I) de sortie d'une fonction. En effet, cette déviation peut être causée par ce mode ou par la propagation d'un flux dysfonctionnel en entrée de la fonction. Le mode de défaillance est constaté *ex post* au niveau d'une fonction étudiée ; l'analyse de la cause de la déviation peut nécessiter d'étudier d'autres fonctions auxquelles ladite fonction est liée.

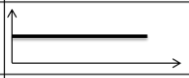
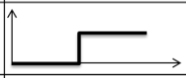
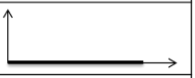
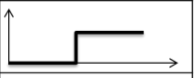
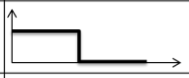
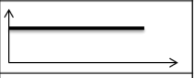
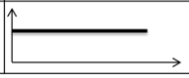
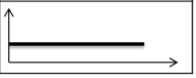
Modes de défaillance génériques	Fonctionnement	
	attendu	réel
Perte de la fonction		
Absence de la fonction à la sollicitation		
Fonction intempestive		
Maintien de la fonction sur ordre d'arrêt		
Dégradation de la fonction		

Figure 16: Modes de défaillances fonctionnels typiques (Desinde, 2006)

Mode Sûr (Safe State) — Pour l'ISO 26262:2011, le Mode Sûr (ou *safe state*) est un « *operating mode of an item without an unreasonable level of risk* ». Cette norme ne distingue donc pas mode et état. Il s'agit d'un mode du système dans une situation opérationnelle dans laquelle le conducteur est sauf. Nous pouvons donc aussi dire : un *Mode Sûr* est un *Mode Fonctionnel* nominal ou dégradé acceptable du système (ou d'une fonction) associé à certaines conditions extérieures. Celui-ci a également comme dénomination *État Sauf* ou *Mode de protection*.

2.3.3.3 Explicitation des relations entre concepts de la vue

L'architecture fonctionnelle, concept englobant, représente la logique et l'enchaînement des fonctions qui peuvent être activés par des flux de contrôle, qui peuvent être internes ou externes. Chaque fonction transforme des flux MEI et les transmet ou les reçoit par des interfaces avec le contexte ou d'autres fonctions. L'analyse de ces flux et de ces interfaces permet de définir des blocs logiques. D'un point de vue comportemental, l'analyse des situations opérationnelles (et donc des scénarios opérationnels) permet de définir les scénarios fonctionnels. Afin de déterminer quand ces fonctions doivent être activées pour échanger des flux, les modes et états fonctionnels sont définis à partir des modes opérationnels. Par ailleurs, à l'aide des *Safety Goals*, il est possible de déterminer l'aspect dysfonctionnel qui correspond à des déviations de flux MEI ou des modes de défaillance fonctionnels. Afin que cela n'ait pas de conséquence critique, des concepts de sûreté tels que les *Safety Functions*, les *Safety Measures* ou les *External Measures* sont mis en place. Leurs objectifs sont de diminuer l'apparition de ces défaillances ou de placer la fonction ou le système dans un mode sûr. L'architecte système définit et raffine les exigences techniques, ce qui correspond, d'un point de vue dysfonctionnel, à définir et à raffiner les *Functional Safety Requirements* à partir des *Safety Goals*.

2.3.4 La troisième vue, centrée sur la PAD (*Physical Architecture Design*)

2.3.4.1 Panorama de la vue

La troisième vue du modèle conceptuel de l'ISSdF, centrée sur l'architecture organique (*Physical Architecture Design* - PAD), peut être représentée à l'aide du diagramme suivant. Étant donné le périmètre de la, cette vue n'a pas été détaillée. La proposition ci-dessous pourra être cependant un point de départ pour définir de futures méthodes d'ISSdF dédiées à l'architecture organique.

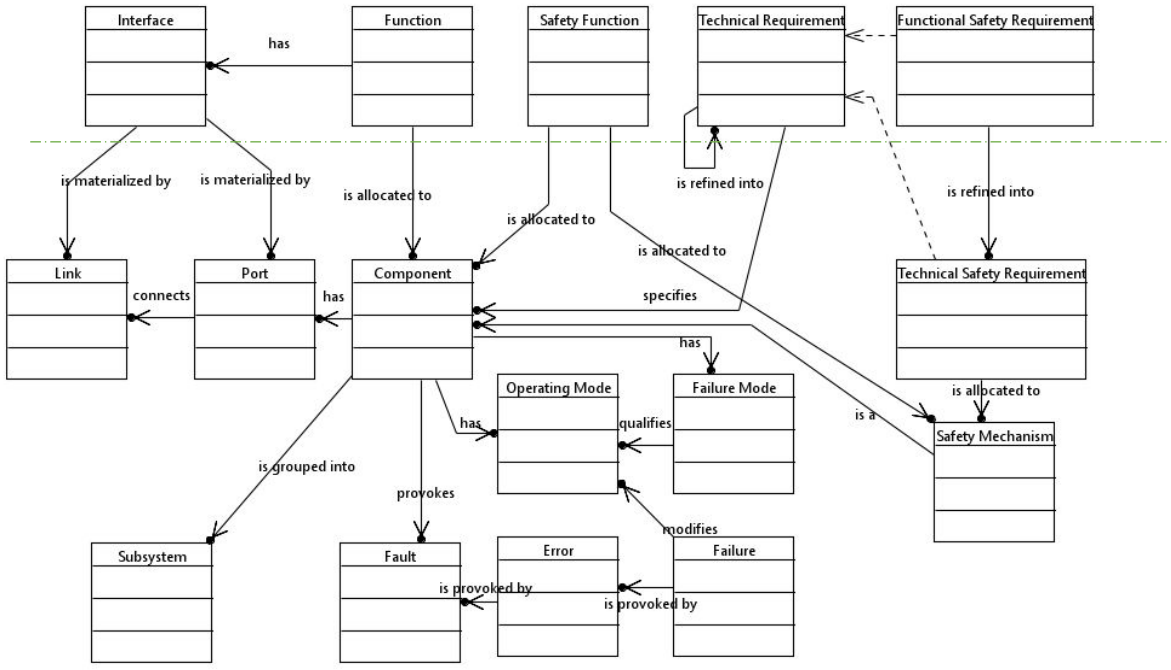


Figure 17 : Modèle conceptuel de l'ISSdF : vue centrée sur la PAD

2.3.4.2 Description des concepts constitutifs

Constituant (Component) — Selon l'ISO 15026:2013 (ISO/IEC 15026, 2013), un constituant est défini comme étant « *an entity with discrete structure, such as an assembly or software module, within a system considered at a particular level of analysis*. »

Port et Lien (Port, Link) — Le **Port** et le **Lien** sont les deux concepts qui matérialisent le concept d'interface fonctionnelle. Les ports appartiennent à un constituant et un lien relie au moins deux ports. Selon Alain Faisandier (Faisandier, 2014), « *A physical interface is a System Element that binds physically two System Elements. A port is an element of a System Element that allows binding this System Element to another one with a Physical Interface.* » Le concept d'interface organique englobe les deux concepts de Port et de Lien.

Faute, Erreur, Défaillance (Fault, Error, Failure) — L'ISO 26262:2011 distingue les concepts, qu'on a parfois tendance à confondre, de **Faute**, d'**Erreur** et de Défaillance. Ainsi, la Faute est une « *abnormal condition that can cause an element or an item to fail* », l'Erreur, une « *discrepancy between a computed, observed or measured value or condition, and the true, specified or theoretically correct value or condition* », et la Défaillance : une « *termination of the ability of an element, to perform a function as required* ».

Mode de Défaillance (Failure Mode) — Le précédent diagramme comprend une occurrence du *Mode de Défaillance* (*Failure Mode*), qui concerne le constituant. Le concept de *Mode de Défaillance* a déjà été défini dans la deuxième vue du modèle conceptuel de l'ISSdF relative à l'architecture fonctionnelle (FAD), comme l'écart entre la sortie souhaitée et la sortie réelle d'une fonction. Cette définition est reprise dans le cas du *Mode de Défaillance* organique, relatif à un constituant. Un mode de défaillance d'un constituant caractérise pour une certaine entrée la différence entre la sortie souhaitée et la sortie réelle. Ce qui change, c'est la connotation du concept. Par exemple, pour la fonction accélérer, nous avons mentionné en illustrant la deuxième vue du modèle conceptuel de l'ISSdF l'accélération intempestive. Dans la troisième vue de ce même modèle, relative à l'architecture organique, nous mentionnerons nécessairement un constituant. Nous parlerons alors de fonctionnement intempestif du moteur.

Exigence Organique de Sûreté (Technical Safety Requirement) — L'exigence organique de sûreté est l'allocation des *Functional Safety Requirements* aux constituants. Cela permet de constituer le *Technical Safety Concept* qui guidera le développement, aval, d'une architecture organique de système sûr.

2.3.4.3 Explicitation des relations

Les fonctions peuvent être allouées à un constituant. Ces constituants interagissent entre eux à l'aide de liens connectés à leurs ports, ce qui correspond à l'allocation des interfaces de la vue fonctionnelle. Ces constituants peuvent également avoir des modes de défaillances. Ces modes de défaillances peuvent être liées à une faute qui s'exprimera par une erreur. Cette erreur pourra provoquer une défaillance. Afin de ne pas produire de conséquences critiques, il est possible, pour le concepteur, de créer et de de mettre en place des *Safety Mechanisms*.

De plus, l'analyse de ces composants va permettre de raffiner les exigences techniques, ce qui se traduit d'un point de vue dysfonctionnel par le raffinement des *Functional Safety Requirements* en *Technical Safety Requirements*.

2.3.5 Intégrer les vues par les exigences

Les trois vues du modèle conceptuel de l'ISSdF (O&SDR, FAD, PAD) ont été présentées sans avoir montré qu'il existe des entités intégratives. En IS, l'une d'elles est l'exigence. Elle est définie dès que des parties prenantes sont identifiées, puis dès que leurs besoins sont traduits en exigences. Elle se retrouve ensuite raffinée, dérivée, complétée, etc., par le concepteur à mesure qu'il progresse dans son activité. Cependant, compte tenu de la profusion des définitions et des termes associés à l'exigence, une vue intégrative nous a semblé nécessaire afin de clarifier certaines relations du modèle conceptuel de l'ISSdF proposé. En IS, le concept d'exigence technique est englobant et peut être spécifié au niveau fonctionnel. Par contre, en SdF, et dans l'ISO 26262:2011, le *Technical Safety Requirement* ne s'applique qu'à un seul constituant. Pour garantir la cohérence entre les différentes vues du système, il faut donc créer des relations spécifiques, comme le montre le diagramme de structure suivant.

Dans ce schéma, nous retrouvons les exigences des parties prenantes qui sont la formalisation, par le concepteur, de leurs besoins. Ces exigences sont ensuite raffinées et traduites en exigences de niveau système ou, ce qui est synonyme, en exigences techniques. Ces exigences peuvent être fonctionnelles, d'interface, etc. Pour ce qui concerne plus particulièrement la sûreté, le diagramme précédent montre que les *Safety Goals* sont des exigences de haut niveau de sûreté spécifiant le comportement souhaité du système en cas de phénomène dangereux. Elles sont raffinées en exigences fonctionnelles de sûreté, puis en exigences techniques de sûreté lors de la définition de l'architecture organique. Lorsque l'architecte alloue les exigences techniques à des constituants ou des applications logicielles, il les dérive en exigences de sûreté matérielles ou logicielles.

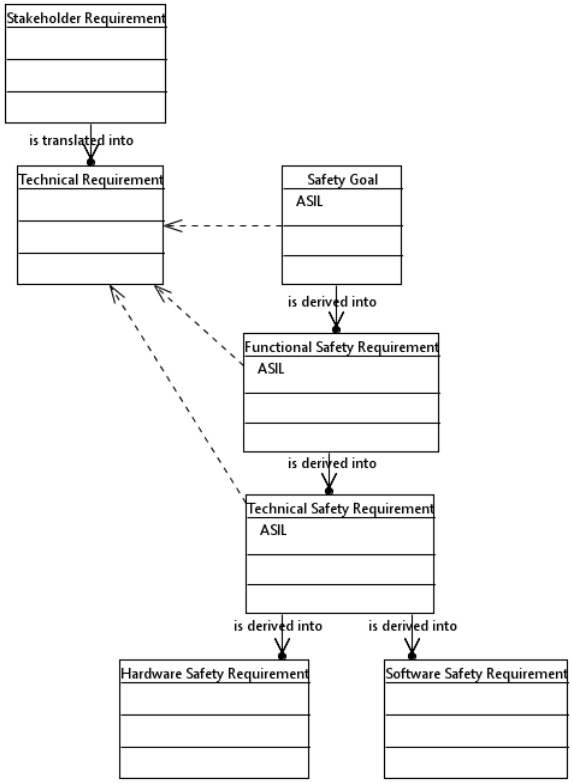


Figure 18 : Proposition de déclinaison des exigences en se basant sur l'ISO 26262:2011

2.4 BILAN DU MODÈLE CONCEPTUEL PROPOSÉ

Nous venons de proposer un modèle conceptuel de l'ISSdF. Pour rester cohérent avec la démarche d'IS, il convient maintenant de le vérifier. Sa validation pourrait être effectuée par le biais de son utilisation par des experts, ce qui n'a pas pu être effectué compte tenu du temps imparti à l'élaboration de ce mémoire. À défaut, le modèle proposé peut être comparé aux modèles présentés de l'état de l'art, puis vérifié d'après les exigences d'un modèle conceptuel de l'ISSdF listées en début de chapitre. Ce qui consiste en une auto-évaluation.

Comme nous l'avons suggéré dans l'état de l'art, le modèle conceptuel de Pfister et Chapurlat qui couvre correctement le domaine de l'Ingénierie Système comporte 17 concepts. Notre modèle contient 10 concepts (même s'ils ne sont pas forcément exprimés avec le même terme). Les 7 concepts restants sont 2 concepts englobants (*Functional Architecture* et *Physical Architecture*), qui ne sont pas présents dans notre modèle par hypothèse. 4 concepts génériques pour nommer un élément de l'architecture (*Model Element, Item, Functional Item* et *Physical Item*) et le concept de *Parameter* ne nous ont pas semblé essentiel dans notre modèle conceptuel.

De plus, comme nous l'avons expliqué dans ce chapitre, plusieurs concepts étaient ambigus pour déterminer le danger ou l'évènement redouté. Mortureux (Mortureux, 2002) ou PSA

Peugeot Citroën[10] (Ozouf, 2009) ont défini des termes qui ne lèvent pas totalement l'ambigüité. De même, la norme ISO 26262:2011 propose un lexique avec des confusions entre les concepts génériques d'évènement et de situation. Avec le tableau suivant, nous mettons en avant l'apport de notre modèle par rapport aux terminologies existantes.

Tableau 8: Comparaison des différentes terminologies autour du danger

Termes ambigus	Techniques de l'Ingénieur (Mortureux, 2002)	ISO 26262:2011	PSA Peugeot Citroën (Ozouf, 2009)	Proposition Français	Proposition Anglais
	Phase	*Operational Situation*	Phase	Situation Opérationnelle Nominale	*Nominal Operational Situation*
Danger, Évènement redouté	Évènement causant une situation dangereuse	*Hazard*	Évènement initiateur	Phénomène Dangereux	*Hazard*
		Operational Situation		Situation potentiellement dangereuse	*Potentially Hazardous Situation*
Danger, Évènement redouté		*Hazardous Event*		Évènement Dangereux	*Hazardous Event*
				Conditions environnementales favorables au danger	*Environmental Conditions Favorable to Hazard*
	Situation Dangereuse	*Operational Situation*		Situation Dangereuse	*Hazardous Situation*
Évènement redouté	Évènement causant un accident potentiel		Évènement redouté	Évènement critique	*Harmful Event*
	Accident	*Harm*	Conséquence Système	Situation avec préjudices	*Harmful Situation*

Comme nous l'avions énoncé, le modèle proposé contient quasiment tous les concepts nécessaires de l'IS et de la SdF tels que définis dans différentes normes. Toutefois, il manque les concepts englobants (architecture fonctionnelle, par exemple). Cette absence se justifie par le fait qu'il y aurait redondance entre le concept englobant et ces concepts englobés.

Le modèle conceptuel proposé est conforme à l'esprit de normes de l'IS ou de la SdF, voire à la lettre, quand des définitions normatives ont été reprises en l'état. De plus, ce modèle est assez concis au sens où chaque vue comporte peu d'entités ou de relations. Notons à ce sujet qu'un travail d'identification de modules au sein de ces vues permettrait sans doute de rendre les vues encore plus concises. En revanche, comme nous l'avons énoncé en introduction de ce mémoire (section sur les restrictions), il ne sert à rien d'intégrer dans le modèle conceptuel de l'ISSdF les concepts associés à l'analyse probabiliste ou stochastique des architectures fonctionnelles et organiques. Leur compréhension requiert, en effet, une expertise certaine, que seuls les ingénieurs en SdF possèdent.

Fort de ces éléments, il devient possible de dresser le tableau comparatif ci-dessous.

[10] Une terminologie plus récente est existante qui est proche de celle proposée mais elle n'a pas été diffusée à l'extérieur du groupe PSA Peugeot Citroën.

Tableau 9 : Comparatif des modèles de l'ISSdF.

Nom du Projet (ou des auteurs par défaut)	IS	SdF	Liens IS&SdF	Conforme à norme IS	Conforme à norme SdF	Conclusion Liens	Multi-vues	Concision	Généricité	Conclusion Compréhensibilité	Vérification processus	Cohérence sémantique	Aligner avec méthodes SdF	Conclusion Modèles
Projet A2PNum	80	80	80	50	100	78	80	80	80	80	55	85	10	**62**
Modèle Taofefinua (article)	60	50	50	0	90	50	0	100	20	40	0	10	0	**20**
Modèle Taofefinua (thèse)	75	75	50	0	100	60	100	25	30	52	70	75	0	**52**
ATESST2 Concept	30	80	30	0	100	48	100	75	40	72	60	80	50	**62**
Modèle Chapurlat/Pfister	90	0	0	80	0	34	0	50	50	34	50	70	0	**38**
Modèle Chapurlat/Cornu	90	0	0	80	0	34	50	60	50	54	50	60	0	**40**
Modèle Adedjouma	20	90	0	0	100	42	0	80	40	40	50	60	0	**39**
Modèle Faisandier	75	75	50	50	0	50	50	90	80	74	70	50	75	**64**
Modèle proposé	90	90	90	100	100	94	100	90	100	97	100	90	60	**89**

En reprenant la liste énoncée au début de ce chapitre, nous allons maintenant évaluer si le modèle proposé est conforme aux exigences d'un modèle conceptuel de l'ISSdF.

Ce modèle doit présenter les concepts de l'ISSdF — Les concepts du modèle renvoient à des entités et des relations clefs de l'ISSdF, au moins pour ce qui concerne les processus de définition des exigences (O&SRD), d'architecture fonctionnelle (FAD) et d'architecture organique (PAD). En effet, les concepts utilisés ont été puisés dans les normes de l'IS ou de la SdF. En cas d'absence ou d'incohérence, des définitions cohérentes avec l'esprit de ces normes ont été proposées.

Ce modèle doit être compréhensible, donc concis et clair — L'approche multivue et l'alignement des vues sur les processus d'IS permettent de respecter cette exigence. Il y a certes un grand nombre de relations, mais le nombre de concepts intégrés à une vue demeure limité. D'autres concepts mineurs ont été ajoutés comme attributs pour permettre d'avoir un modèle complet mais lisible. Ajoutons qu'un retour sur le modèle permettrait sans doute d'identifier des modules, des paquets, etc., simplifiant encore plus le panorama associé à chaque vue.

Ce modèle doit être vérifié par la définition d'un processus d'ingénierie de systèmes sûrs de fonctionnement — Cette exigence ne peut être vérifiée à ce stade du mémoire, le processus d'ISSdF n'ayant pas encore été défini. Nous reviendrons donc sur elle à la fin du prochain chapitre. Celui-ci montrera que cette exigence est satisfaite.

Ce modèle doit fournir une cohérence sémantique entre les différents intervenants, un langage commun partagé — Cette exigence a été respectée. Chaque vue associe les aspects fonctionnels et dysfonctionnels du système. Pour le dire autrement, il existe bien des relations connexes entre les concepts relevant de ces deux domaines. Lors de la présentation des concepts constitutifs d'une vue, nous avons énoncé ceux équivalents à celui retenu dans le modèle proposé.

Ce modèle doit être conforme avec les méthodes classiques d'analyse de SdF et d'IS selon les règles de l'art — Nous reviendrons sur cette exigence à la fin des chapitres 4 et 5, qui montreront que le modèle proposé la satisfait.

2.5 CONCLUSION

L'objectif de cette thèse est de proposer un modèle et un processus d'ISSdF permettant d'aligner la SdF et l'IS. La voie choisie a été l'élaboration d'un modèle conceptuel à même de faciliter le partage de connaissance entre l'AS et l'ingénieur de SdF. Ce deuxième chapitre a mis en évidence une première contribution de cette thèse, à savoir un modèle conceptuel reposant sur des vues à la fois cohérentes avec les concepts d'IS ou ceux de la SdF présents dans des normes de ces domaines. Le modèle conceptuel proposé se réfère principalement à l'ISO 15288:2008 (domaine de l'IS) et à l'ISO 26262:2011 (domaine de la SdF). Cependant, le modèle proposé n'est qu'une étape intermédiaire. Pour le rendre effectif, et assurer l'alignement pratique de la SdF sur l'IS, il faut l'intégrer dans un processus d'ISSdF. C'est ce point qu'élaborera le prochain chapitre. Notons que le modèle conceptuel proposé pourra être également utile, dans les quatrième et cinquième chapitres du présent mémoire, lorsqu'il s'agira de définir des méthodes de SdF pour supporter des activités du processus ISSdF présenté dans les pages suivantes.

CHAPITRE 3
PROPOSITION D'UN PROCESSUS D'ISSDF

3.1 INTRODUCTION

Le précédent chapitre a proposé un modèle conceptuel de l'Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF). Ce modèle a décrit ce qu'est une architecture de produit sûre. En revanche, rien n'a été explicité concernant la façon de pratiquer l'ISSdF. L'objet du présent chapitre est de combler ce manque et de proposer un processus d'ISSdF cohérent avec le référentiel de l'Ingénierie Système (IS), et plus précisément avec les principes de l'IS basée sur les modèles (MBSE). Pour élaborer un tel processus, nous nous appuierons sur un état de l'art, puis nous définirons un macro-processus qui sera ensuite décliné sur les quatre niveaux de conception qui ont retenu notre attention au chapitre précédent, à savoir : *Stakeholder Requirements Definition* (SRD), *Requirements Analysis* (RA), *Functional Architecture Design* (FAD), et *Physical Architecture Design* (PAD) ; ce dernier n'ayant été qu'abordé partiellement. En effet, il existe déjà de nombreuses méthodes existantes et efficaces au niveau organique, ce qui n'est pas le cas pour les autres processus. Le processus d'ISSdF proposé dans ce chapitre enrichira les processus d'IS, soit en complétant le contenu d'activités déjà présentes dans l'ISO 15288:2008, soit en ajoutant des activités dédiées à la sûreté de fonctionnement (SdF), et ce afin d'être cohérent avec les normes de la SdF et plus particulièrement avec l'ISO26262:2011.

3.2 EXIGENCES DU PROCESSUS D'ISSDF

Cette thèse entend proposer un ensemble de modèles, processus et méthodes de l'ISSdF cohérent avec l'ISO 15288:2008 et l'ISO 26262:2011. Par conséquent, le processus d'ISSdF élaboré doit être conforme à ces normes. Un processus d'ISSdF abouti devrait :

1. être validé par les industriels par des tests concernant le développement d'un véhicule ;
2. intégrer les activités de SdF tout au long du processus de conception, et ce, même si nous nous sommes focalisés sur la conception amont (SRD, RA, FAD, et de façon mineure : PAD) ;
3. être interfaçable avec un processus d'expertise de sûreté et limiter les allers-retours entre les architectes systèmes et les ingénieurs de SdF ;
4. être structuré en cohérence avec l'approche multivue du modèle conceptuel de l'ISSdF détaillé dans le chapitre précédent et assurer la cohérence entre les différentes vues ;

5. pouvoir être piloté et évalué à l'aide d'indicateurs opérationnels.

Les exigences principales du processus d'ISSdF sont regroupées dans le tableau suivant.

Tableau 10 : Exigences du processus d'ISSdF

Référence	Exigence
Exi. 1	Il doit être conforme à la norme ISO 15288:2008.
Exi. 2	Il doit être conforme à la norme ISO 26262:2011.
Exi. 3	Il doit pouvoir être validé par l'expérimentation sur un exemple concret.
Exi. 4	Il doit intégrer la sûreté de fonctionnement dans les différentes activités de conception.
Exi. 5	Il doit être interfaçable avec un processus d'expertise de sûreté.
Exi. 6	Il doit assurer la cohérence entre les différentes vues des différents points de vue.
Exi. 7	Il doit être pilotable et évaluable à l'aide d'indicateurs opérationnels

Pour reprendre la démarche suivie dans le précédent chapitre, nous allons maintenant dresser un bref état de l'art des processus d'IS intégrant la SdF existants, en apprécier la performance eu égard aux exigences listées dans le tableau ci-dessus, avant de proposer un processus jugé plus satisfaisant.

3.3 FONDEMENTS D'UN PROCESSUS D'ISSDF

3.3.1 Processus, de quoi parle-t-on ?

Le processus décrit l'ensemble des activités, des tâches − avec des ressources dédiées, des flux de données entrant et sortant − qui doivent être enchaînées pour obtenir un résultat jugé performant d'un point de vue opérationnel (conformité, efficacité, délai, coût, etc.). Un processus peut être générique, opérationnel ou instancié. Les processus listés dans l'ISO 15288:2008 sont génériques : tout responsable d'un bureau d'étude souhaitant aligner sa pratique sur les principes de l'IS doit se mettre en cohérence avec leur contenu. Un processus opérationnel est spécifique : il concerne par exemple un bureau d'études, un secteur, etc., donnés. Ainsi, dans cette thèse, le processus d'ISSdF présenté est spécifique mais généralisable à d'autres secteurs. Enfin, un processus instancié est un processus mis en œuvre dans un projet de conception donné. S'ajoute à cette typologie des processus (générique, opérationnel, instancié), une typologie reposant sur leur périmètre. Il est ainsi possible de distinguer un macroprocessus, qui présente de façon panoramique ce qui doit être fait, des processus opérationnels, qui sont plus détaillés.

Plusieurs sources peuvent alimenter la modélisation d'un processus. Il peut s'agir d'un travail académique, d'une norme comme l'ISO 15288:2008, d'un recueil de connaissances comme le

SEBoK (Pyster & Olwell, 2013), d'un mode d'emploi d'une suite logicielle dédiée au MBSE, etc. Certaines sources ont une vocation ontologique, comme les travaux académiques ou l'ISO 15288:2008 ; d'autres, comme le mode d'emploi, ont une vocation pratique : son objectif est de bien utiliser un logiciel. Pour ce qui concerne cette dernière source, de plus en plus importante à mesure que l'IS est instrumentée par du logiciel, on peut par exemple mentionner, et comme expliqué dans (INCOSE, 2008), la méthodologie *Harmony SE*. Cette méthodologie, rédigée par Hans-Peter Hoffmann, est associée au logiciel *Rhapsody,* outil proposé par IBM. La part consacrée à la SdF est très faible dans cette suite logicielle. À suivre le *Deskbook* d'IBM (IBM, 2011), un modèle exécutable ne s'appuie que sur des AMDEC, ce qui est insuffisant pour développer des architectures sûres. Un tel manque se constate aussi dans les produits concurrents, que ce soit la méthodologie associée à *Object-Oriented Systems Engineering Method* (OOSEM) de (Lykins, Friedenthal, & Meilich, 2000) ou au *Rational Unified Process for SE* de (Kruchten, 2004). Si les modes d'emploi des logiciels dédiés au MBSE n'intègrent pas la problématique de la SdF, qu'en est-il des processus proposés par le monde académique ?

3.3.2 Les processus de conception prenant en compte la SDF

Dans (Cressent, David, Idasiak, & Kratz, 2012), Robin Cressent présente un processus d'IS prenant en compte la SdF sous la forme d'un diagramme d'activités avec deux travées (*swimlanes*) ; l'une pour l'IS et l'autre pour la sûreté. Il détaille chacun de ces types de processus en précisant les activités à réaliser et les livrables attendus. Si cette approche est un premier pas vers l'ISSdF, elle présente toutefois des inconvénients. Elle ne repose sur aucune norme établie en IS ou en SdF. Les activités sont séquentielles et non interactives, si bien qu'il est impossible d'apprécier les tâches, les modèles, les livrables, etc., qui peuvent être partagés entre l'architecte système et l'ingénieur spécialiste en SdF. Une telle problématique se rencontre heureusement dans le travail de Faida Mhenni (Mhenni, 2014) intitulé SafeSysE, pour *Safety Analysis Integration In Systems Engineering*. La sûreté est mieux traitée que dans le modèle de Crescent, mais elle ne se limite qu'aux activités d'analyse. De plus, s'il est possible de lire et de bien comprendre les liens de l'IS vers la SdF, la réciproque n'est pas vraie.

La méthode STAMP de Nancy Leveson définie dans (Leveson, 1995), puis développée dans (Leveson, 2011), satisfait mieux les exigences énoncées précédemment. Dans (Weiss, 2006), Kathrin Anne Weiss utilise STAMP pour proposer un processus à vocation intégrative. Celle-ci détaille uniquement les étapes du processus de sûreté système, comme le montre le tableau suivant. De plus, la SdF est vue comme faisant l'objet d'activités réalisées *ex post*, c'est-à-dire après la conception. Le concepteur développe ainsi une solution, puis applique des critères issus de la SdF pour apprécier si elle est sûre. L'auteure ne traite donc pas des questions relatives aux exigences de SdF, à la proposition d'un concept de SdF, etc. Enfin, elle suppose que la seule

solution organique possible pour garantir la sûreté est le contrôle / commande du système réside dans l'électronique embarquée.

Tableau 11 : Étapes du System Safety Process *basé sur STAMP (Weiss, 2006)*

Étape	*System Safety Process*
1	*Identify System Hazards*
2	*Identify System-Level Safety-related Requirements and Constraints*
3	*Define the System Safety Control Structure*
4	*Identify Inadequate Control Actions leading to a Hazardous State*
5	*Determine how the Constraint could be Violated and Implement Mitigation*

Si nous reprenons les exigences énoncées ci-dessus, nous constatons que les processus à vocation intégrative proposés présentent des limites. Aucun ne se réfère explicitement à l'ISO 15288:2008 et à l'ISO 26262:2011 (Exi. 1, Exi. 2). La prise en compte de la SdF est faite *ex post*, une fois la conception réalisée (Exi. 4). D'où la possibilité, constatée dans la pratique, de modifications coûteuses des solutions proposées par le concepteur du fait des analyses menées par l'ingénieur de SdF (Exi. 6). Enfin, les travaux mentionnés couvrent insuffisamment le domaine de la SdF. Des activités habituelles comme l'APR ou l'AMDEC sont présentes, mais il n'y a pas de lien avec l'ISO 26262:2011 (Exi. 1, Exi. 2, Exi. 5). Il n'a pas été possible de vérifier les exigences 3 et 7 dans cet état de l'art. Un tel bilan motive donc le choix de définir un nouveau processus d'ISSdF susceptible de mieux respecter les exigences définies précédemment.

3.4 PROPOSITION D'UN PROCESSUS D'ISSDF

La démarche suivie pour proposer un nouveau processus d'ISSdF est la même que celle conduite pour élaborer le modèle conceptuel de l'ISSdF du précédent chapitre. Aussi, après avoir explicité notre démarche de modélisation, un macroprocessus représenté sous la forme d'une matrice regroupant plusieurs vues et phases du projet sera détaillé. Enfin, la couverture des concepts du modèle conceptuel de l'ISSdF sur le processus ISSdF sera vérifiée.

3.4.1 Démarche de modélisation suivie

Le macroprocessus d'ISSdF proposé reprend à son compte différentes propriétés du modèle conceptuel du domaine présenté au chapitre précédent. L'une des caractéristiques majeures est l'approche multivue. Ainsi, les points de vue du processus d'ISSdF doivent être alignés sur les vues du modèle conceptuel de l'ISSdF. Nous aurons donc les points de vue opérationnel et fonctionnel externe alignés avec la vue *Operational & System Requirements Definition*, le point de vue fonctionnel interne aligné avec la vue *Functional Architecture Design* et le point de vue organique aligné avec la vue *Physical Architecture Design*. Cet ensemble de points de vue ne suffit toutefois pas. Un processus a vocation à être instancié, c'est-à-dire mis en œuvre dans des projets donnés. Or, il y a un consensus, parmi les praticiens et les théoriciens de la conception, selon lequel, à mesure qu'on progresse dans le projet de conception, les vues du système à concevoir sont élaborées, raffinées, complétées, enrichies, etc. Depuis les travaux de Pahl et Beitz dans les années 1960, il est habituel de distinguer en ingénierie mécanique ces différentes phases : *Clarification of the task*, *Conceptual Design*, *Embodiment Design*, *Detailed Design* (Pahl, Beitz, Feldhusen, & Grote, 2007). Si on s'en tient au domaine de l'IS, tout projet cohérent avec l'IS comporterait ainsi sept phases de projet selon les experts de la NASA (NASA, 2013). Celles-ci sont listées dans le tableau suivant. Seules les phases A à C nous intéressent.

Tableau 12 : Phases d'un projet d'Ingénierie Système selon (NASA, 2013)

Phase	Contenu
Pre-phase A	*Concept Studies*
Phase A	*Concept & Technology Development*
Phase B	*Preliminary Design & Development Completion*
Phase C	*Final Design & Fabrication*
Phase D	*System assembly, integration & Test. Launch*
Phase E	*Operations & Sustainment*
Phase F	*Closeout*

Si les modèles présentés décrivent le projet de conception, le lien avec les processus d'IS n'est pas explicite. En se basant sur les travaux de la NASA (NASA, 2013), l'alignement suivant est proposé, qui associe la vue qu'a le service planification de projet (*planning*) avec les processus techniques (cycle en V) définis par l'ISO 15288:2008, donc les vues du modèle conceptuel de l'ISSdF proposées précédemment. Un tel alignement est présenté Figure 19.

Le véhicule étant un système ayant plusieurs strates (véhicule, puis GMP, puis moteur, puis injection, etc.), le cycle en V suivant est reproduit à tous les niveaux ou échelles du produit. Enfin, cette figure peut être lue horizontalement, par le chef de projet ou toute personne s'occupant de planification de projets, ou verticalement, par l'expert en IS spécialiste de tel ou tel processus. Il serait donc judicieux d'utiliser les approches multivues explicitées dans l'état de l'art de la section 1.2.2 . Cette double lecture permettra de comprendre l'approche multivue retenue dans la description du macroprocessus d'ISSdF proposé dans la section suivante.

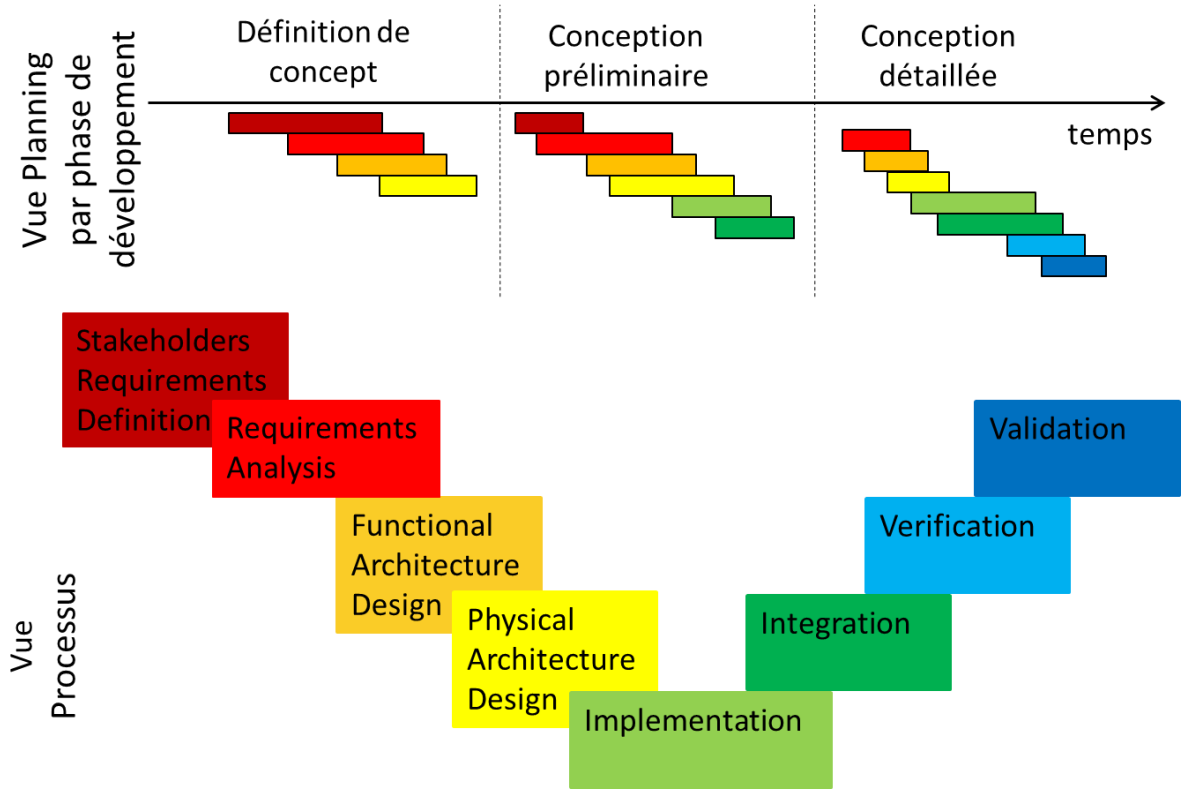


Figure 19 : Cycle de développement pour une strate système

3.4.2 Matrice du macroprocessus d'ISSdF

Les lignes de la matrice (Tableau 13) décrivant le macroprocessus d'ISSdF reprennent les vues du modèle conceptuel de l'ISSdF, qui lui-même est cohérent avec l'ISO 15288:2008 et l'ISO 26262:2011. Les colonnes, quant à elles, représentent les phases habituellement identifiées dans un projet de conception. Elles correspondent à un avancement du projet en maturité. Dans le cas qui nous intéresse, la phase de définition de concept permet de définir des architectures candidates. La phase de conception préliminaire permet de définir l'architecture de la solution retenue et de s'assurer qu'elle est réalisable avec un bon niveau de confiance. La phase de conception détaillée permet de consolider l'architecture réalisable retenue en détaillant tous les constituants. À l'intersection entre les lignes et les colonnes, on retrouve les processus de l'ISSdF[11]. Comme nous le verrons par la suite, la plupart des activités intègrent un aspect dysfonctionnel. L'architecte peut lire la matrice verticalement, afin d'identifier ses activités lors de chaque phase. Le macroprocessus d'ISSdF étant défini et représenté matriciellement, il est désormais possible de le détailler en suivant le point de vue de l'architecte, donc de lire verticalement la matrice suivante.

[11] Nous avons décidé de placer ce tableau récapitulatif à cet emplacement de la thèse pour permettre au lecteur de s'y référer et de vérifier progressivement sa cohérence avec le détail des processus fournis par la suite. Un autre emplacement aurait pu être dans la section 3.5 « bilan du processus proposé » de ce chapitre.

Tableau 13: Description tabulaire du processus ISSdF

	Définition de Concept	**Conception Préliminaire**	**Conception Détaillée**
Point de vue Opérationnel - **StakeHolders Requirements Definition**	Collecter les besoins déclinables en termes de SdF Modéliser les exigences des Parties prenantes Définir le contexte du système Identifier les services requis Définir les interactions du système Analyser et maintenir les exigences des parties prenantes	Raffiner le contexte du système Finaliser les exigences des parties prenantes Approuver les exigences des parties prenantes Analyser et maintenir les exigences des parties prenantes	Mettre à jour le concept opérationnel et le référentiel des exigences des parties prenantes
Point de vue Fonctionnel Externe - **Requirements Analysis**	Définir les exigences système clefs Définir les fonctions et les déviations de flux Définir les frontières, interactions et les dommages potentiels Définir les modes opérationnels, les scénarios opérationnels et les séquences de situations opérationnelles Spécifier les exigences de niveau système dont les *Safety Goal's* Analyser et maintenir les exigences système	Raffiner les exigences système Raffiner les fonctions systèmes et les déviations de flux Raffiner les frontières, interactions et les dommages potentiels Raffiner les modes opérationnels, les scénarios opérationnels et les séquences de situations opérationnelles Raffiner les exigences de niveau système dont les Safety Goals Analyser et maintenir les exigences système	Maintenir à jour le référentiel des exigences système
Point de vue Fonctionnel Interne - **Functional Architecture Design**	Définir la vue structurelle de l'architecture fonctionnelle Définir les fonctions techniques Identifier les modes de défaillance fonctionnels Définir les interfaces fonctionnelles et les flux de contrôle Définir les chemins critiques dysfonctionnels Raffiner la vue par l'ajout de concepts de sûreté Dériver et allouer les exigences dimensionnantes aux fonctions techniques dont les *Safety Goals* Déterminer la vue comportementale de l'architecture fonctionnelle Définir les modes et états des différentes fonctions Définir les scénarios fonctionnels Regrouper les éléments fonctionnels en blocs logiques Spécifier puis raffiner les exigences techniques dont les *FSR* Évaluer la satisfaction des exigences techniques et des *FSR* Évaluer les architectures candidates et en sélectionner une Mettre à jour l'architecture fonctionnelle en fonction des choix de composants	Raffiner la vue structurelle de l'architecture fonctionnelle Raffiner les fonctions et intégrer les fonctions techniques induites Raffiner les interfaces fonctionnelles et les flux de contrôle Raffiner les chemins critiques dysfonctionnels Raffiner la vue par l'ajout de concepts de sûreté Dériver et allouer les exigences aux fonctions techniques dont les *Safety Goals* Raffiner la vue comportementale de l'architecture fonctionnelle Raffiner les modes et états des différentes fonctions Raffiner les scénarios fonctionnels Raffiner les exigences techniques dont les Functional Safety Requirements pour chaque bloc logique	Consolider la vue structurelle de l'architecture fonctionnelle Consolider les fonctions techniques Consolider les interfaces fonctionnelles et les flux de contrôle Consolider la vue comportementale de l'architecture fonctionnelle Consolider les modes et états des différentes fonctions Consolider les scénarios fonctionnels Analyser et valider l'architecture fonctionnelle et le *Functional Safety Concept*
Point de vue Organique - **Physical Architecture Design**	Partitionner et allouer les fonctions à des constituants réalisables Établir les architectures organiques candidates Regrouper les constituants en sous-systèmes si besoin Identifier les modes de défaillance organiques Définir les interfaces organiques Définir les chemins critiques dysfonctionnels Définir les Safety Mechanisms sur les concepts nouveaux ou modifiés Représenter les architectures organiques Élaborer un modèle de décision afin d'évaluer les architectures organiques candidates Évaluer la satisfaction des exigences techniques incluant les *TSR*	Raffiner l'architecture organique de la solution retenue Raffiner la définition des constituants et leurs modes de défaillance organique Raffiner les interfaces organiques Mettre à jour les chemins critiques dysfonctionnels Raffiner les Safety Mechanisms	Consolider l'architecture organique Consolider la définition des constituants et leurs modes de défaillance organiques Consolider les interfaces organiques mettre à jour les chemins critiques dysfonctionnels Définir les constituants élémentaires Analyser et valider l'architecture organique et le *Technical Safety Concept*

3.4.3 Le processus D'ISSdF en phase de définition de concept

La phase de définition de concept consiste à définir des architectures possibles afin de pouvoir en sélectionner une qui satisfait au mieux aux exigences. Nous proposons dans cette section des processus correspondant aux trois vues retenues : opérationnel, fonctionnel et organique. Ces processus sont en partie repris des processus génériques de l'ISO 15288:2008, auxquels ont été ajoutées les caractéristiques propres au MBSE, ainsi que des activités relatives à la sûreté (et non présentes dans cette norme). Pour les activités de définition de l'architecture, nous avons pris comme référence le *SEBoK* étant donné que le processus de l'ISO 15288:2008 est trop succinct pour ces points de vue. Comme l'ISO 15288:2015 (ISO 15288, 2015) a été mise en cohérence avec le SEBoK, ce choix nous paraît tout à fait acceptable.

3.4.3.1 Point de vue Opérationnel : Stakeholder Requirements Definition

Pour ce qui concerne la *Stakeholder Requirements Definition* (SRD), qui consiste à déterminer les exigences des parties prenantes (*stakeholders*), les activités associées sont listées dans le Tableau 14. Réaliser de façon satisfaisante une SRD requiert ainsi :

1) d'éliciter les exigences des parties prenantes, ce qui suppose d'identifier ceux-ci, de recueillir leurs besoins, demandes, souhaits, attentes, etc., puis d'homogénéiser cet ensemble disparate en termes de contenu et de forme ;
2) de modéliser les exigences des parties prenantes. Cette activité peut être vue comme un sous-processus qui se décompose à son tour en différentes activités telles que la contextualisation du système (cycle de vie, etc.), l'identification de ses services, de ses interactions, etc.
3) d'analyser et maintenir ces exigences.

À mesure que le processus SRD se déroule, l'aspect dysfonctionnel des activités peut être mené. On peut ainsi collecter les besoins, attentes, souhaits, mais aussi les contraintes non souhaitées. Ces besoins sont également déclinables en termes de SdF. En effet, faute de pouvoir de supprimer toutes les défaillances, avoir un système avec un certain nombre de défaillances peu probables peut être souhaité. Par ailleurs, la contextualisation du système peut prendre en compte les phases de vie non-nominales du cycle de vie, c'est à dire dégradées, dangereuses, etc. La prise en compte des interacteurs comprenant les différents occupants du véhicule permettra d'identifier des utilisations fonctionnelles et dysfonctionnelles, qui seront raffinées et enrichies en scénarios opérationnels nominaux et des scénarios opérationnels dysfonctionnels lors de la phase *Requirements Analysis*. Enfin, il faut vérifier, valider et documenter les besoins retenus par le concepteur ; également ceux associés à la SdF.

Tableau 14 : Processus de SRD – Définition de Concept

Activités selon l'ISO 15288	Activités selon l'ISO 15288 enrichies par l'ISSdF
Éliciter et définir les exigences des parties prenantes	Collecter les souhaits, besoins, etc., déclinables également en termes de SdF
Modéliser les exigences des parties prenantes	Définir le contexte du système (phases de vie nominales et dégradées, acteurs, dont agresseurs et menacés)
	Identifier les services du système requis par les parties prenantes
	Définir les interactions du système (utilisations fonctionnelles et dysfonctionnelles)
Analyser et maintenir les exigences des parties prenantes	Analyser et maintenir les exigences des parties prenantes

3.4.3.2 Point de vue Fonctionnel Externe : Requirements Analysis

Une fois définies les exigences des parties prenantes, l'ISO 15288:2008 demande au concepteur de procéder à l'analyse des exigences système. La définition des fonctions dites système répondant aux services attendus permettra de définir les exigences de niveau système. Il est possible de définir les fonctions système et de les relier par des flux de matière, d'énergie ou d'information à l'environnement, ce qui permet d'appréhender les modes opérationnels du système. Fort d'une connaissance accrue du fonctionnement opérationnel du système, le concepteur peut raffiner les exigences de niveau système. Comme dans le processus précédent, le normalisateur lui demande d'analyser et de maintenir les exigences de niveau système, puis de les vérifier, de les valider, et de les documenter.

Dans le cas de l'ISSdF, différentes activités visant la sûreté du système sont conduites à mesure de l'analyse des exigences. L'ISSdF suppose ainsi de prendre en compte les déviations des flux transitant par ces interfaces et considérées comme des phénomènes dangereux. De telles déviations serviront à définir ultérieurement les *Safety Goals*. En phase de définition de concept, et afin de ne pas se surcharger, le concepteur se focalise sur les fonctions dimensionnantes, les nouvelles fonctions (dont la maîtrise peut générer des risques au niveau de la réussite du projet) ou à forte évolution.

Le chapitre 4 reviendra sur ce processus, le détaillera et proposera une méthode pour supporter ses activités spécifiques à l'ISSdF.

Tableau 15 : Processus de Requirements Analysis – Définition de Concept

Activités selon l'ISO 15288	Activités selon l'ISO 15288 enrichies par l'ISSdF
Définir les exigences système[12]	Définir les fonctions du système et les déviations de flux (en particulier, les phénomènes dangereux)
	Définir les frontières (interfaces) et interactions fonctionnelles du système avec son contexte et les dommages potentiels
	Définir les modes opérationnels, les scénarios opérationnels (nominaux et dysfonctionnels), les séquences de situations opérationnelles
	Spécifier puis raffiner les exigences de niveau système dont les Safety Goals
Analyser et maintenir les exigences système	Analyser et maintenir les exigences système

3.4.3.3 Point de vue Fonctionnel Interne : Functional Architecture Design (FAD)

Comme son nom l'indique, la FAD permet de définir l'architecture fonctionnelle du système. Comme il s'agit d'un processus technique clef de l'IS, les activités associées ont été détaillées.

La première activité du FAD consiste à identifier les éléments de l'architecture fonctionnelle que sont les fonctions et leurs interfaces. Du point de vue de l'ISSdF, cette identification aide à dériver les modes de défaillance fonctionnels. En effet, si une fonction a un mode de défaillance, alors au moins une sous-fonction associée en aura un. La définition des interfaces et des flux MEI d'entrée-sortie permet en parallèle de définir les chemins critiques dysfonctionnels en se basant sur les *Safety Goals* définis précédemment. Ces activités d'analyse dysfonctionnelle permettront de déterminer des concepts de sûreté. Ceux-ci seront des exigences de sûreté sur la robustesse d'une fonction vis-à-vis d'une défaillance, des *Safety Functions* prises en compte dans l'architecture fonctionnelle, sauf si des mesures ont été prises dans le contexte du système (*External Measure*).

La deuxième activité du FAD consiste à dériver et à allouer les exigences dites dimensionnantes [13], dont les *Safety Goals,* aux fonctions retenues dans les alternatives d'architecture fonctionnelle. Ce qui facilite la troisième activité qui est la définition, pour chaque fonction technique, des modes et états fonctionnels, mais aussi des modes dégradés et sûrs. Ensuite, il faudra déterminer l'enchaînement des fonctions de l'architecture. Cela permettra de définir la vue comportementale de l'architecture fonctionnelle.

[12] Pour rappel, ce concept est équivalent à celui d'exigence technique

[13] Les exigences dimensionnantes ont une importance forte sur les choix d'architecture qui pourront être réalisés en phase de définition de concept. D'autres exigences pourront être prises en compte dans la phase de conception préliminaire sans remettre en cause les choix architecturaux précédents. L'identification de ces exigences dimensionnantes est liée à l'expertise des concepteurs formalisée sur des projets antérieurs.

Il est ensuite possible de regrouper les éléments fonctionnels en blocs logiques ou sous-systèmes selon plusieurs critères possibles (similarités, imbrication, dépendance…), ce qui permet de spécifier puis raffiner les exigences système dont les *Functional Safety Requirements*. Il sera donc possible d'extraire un premier *Functional Safety Concept* pour les concepts nouveaux ou modifiés.

L'activité suivante vise à évaluer les architectures candidatures au regard de la satisfaction des exigences techniques (notamment des exigences associés à la SdF) et ce, afin de pouvoir sélectionner une architecture. Enfin, l'architecture fonctionnelle est mise à jour en parallèle de la définition de la solution organique.

Le cinquième chapitre sera l'occasion de détailler le contenu de ces activités, ainsi que les modèles et méthodes requis pour les assister.

Tableau 16 : Processus de FAD – Définition de Concept

Processus selon le SEBoK	Activités selon le SEBoK enrichies par l'ISSdF
Définir la vue structurelle de l'architecture fonctionnelle	Définir les fonctions techniques par décomposition & Identifier les modes de défaillance fonctionnels
	Définir les interfaces fonctionnelles et les flux de contrôle
	Définir les chemins critiques dysfonctionnels
	Raffiner la vue par l'ajout de concepts de sûreté dont les *Safety Functions*
Dériver et allouer les exigences dimensionnantes aux fonctions techniques	Dériver et Allouer les exigences techniques du niveau supérieur dont les *Safety Goals*
Définir la vue comportementale de l'architecture fonctionnelle	Définir les modes et états des différentes fonctions (nominaux, dégradés, sûrs)
	Définir les scénarios fonctionnels (nominaux et dysfonctionnels) caractérisant la dynamique d'enchaînement
Regrouper les éléments fonctionnels en blocs logiques	Regrouper les éléments fonctionnels en blocs logiques
Spécifier (ou raffiner) les exigences techniques	Spécifier (ou raffiner) les exigences techniques dont les *Functional Safety Requirements* pour chaque bloc logique
Évaluer les architectures candidates et en sélectionner une	Évaluer la satisfaction des exigences techniques et des *Functional Safety Requirements*
	Évaluer les architectures candidates et en sélectionner une
Mettre à jour l'architecture fonctionnelle en fonction des choix de composants	Mettre à jour l'architecture logique lorsque les choix organiques sont effectués

3.4.3.4 Point de vue Organique : Physical Architecture Design (PAD)

Avec le *Physical Architecture Design* commence le développement de la solution organique, ce qui n'est pas le focus de cette thèse. Toutefois, la conception d'une architecture organique sera d'autant mieux intégrée à l'ISSdF qu'elle sera cohérente avec les processus définis ci-dessus.

Pour les normalisateurs, la première activité liée à la PAD consiste à définir des architectures organiques candidates, puis à allouer les fonctions techniques du système aux constituants réalisables. Enfin, il est possible de spécifier puis de raffiner les exigences de niveau système.

Dans le cadre de l'ISSdF, des activités spécifiques à la SdF doivent être combinées avec celles évoquées au paragraphe précédent. Ainsi, le concepteur peut définir des modes de défaillance organiques possibles. La vue structurelle de l'architecture organique pourra être enrichie par la définition de chemins critiques dysfonctionnels. La vue comportementale pourra l'être par celle des modes de défaillance organiques. Ils enrichissent les modes de fonctionnement (*operating mode*) en se basant sur ceux déterminés au point de vue fonctionnel. Les chemins critiques dysfonctionnels pourront se traduire au niveau physique par des fautes, des erreurs ou des défaillances. Les enrichissements dysfonctionnels pourront se traduire par l'ajout de *Safety Mechanisms* réalisant les *Safety Functions*. Enfin, pour ce qui concerne l'évaluation des architectures organiques candidates, il convient de prendre en compte des critères de sûreté.

Tableau 17 : Processus de PAD – Définition de Concept

Processus selon le SEBoK	Activités selon le SEBoK enrichies par l'ISSdF
Partitionner et allouer les éléments fonctionnels à des constituants réalisables	Partitionner et allouer les éléments fonctionnels à des constituants réalisables
Établir les architectures organiques candidates	Regrouper les constituants en sous-systèmes si besoin
	Identifier les modes de défaillance organiques
	Définir les interfaces organiques (Ports et Liens)
	Définir les chemins critiques dysfonctionnels
	Définir les *Safety Mechanisms* sur les concepts nouveaux ou modifiés
	Représenter les architectures organiques
Évaluer les architectures organiques candidates et en sélectionner une	Élaborer un modèle de décision avec des critères fonctionnels ou non-fonctionnels afin d'évaluer les architectures organiques candidates
	Évaluer la satisfaction des exigences techniques incluant les *Technical Safety Requirements*

3.4.4 Le processus D'ISSdF en phase de conception préliminaire

La conception préliminaire permet de définir l'architecture de la solution retenue en raffinant l'architecture issue de la définition de concept. Cette phase du projet de conception a été moins développée dans le cadre de cette thèse. Aussi définirons-nous qu'à grands traits les processus et activités associés au processus d'ISSdF en conception préliminaire.

Le premier processus concerne le *Stakeholder Requirements Definition*, qui vise à finaliser et valider l'ensemble des exigences des parties prenantes. Le contexte du système sera raffiné afin de prendre en compte des acteurs ou des phases du cycle de vie induites par le choix de la solution. Pour ce qui concerne l'ISSdF, et comme l'indique le tableau suivant, il s'agira de raffiner les phases du cycle de vie non-nominales, puis les agresseurs et les parties prenantes menacées. Le second processus, qui concerne le *Requirements Analysis*, vise à raffiner les exigences techniques de niveau système afin de pouvoir le définir complètement. Dans l'optique de l'ISSdF, il convient alors de raffiner la définition des *Safety Goals* en prenant en compte tous les scénarios dysfonctionnels avec un ASIL différent de QM (*Quality Management*). Les deux premiers processus ont été regroupés dans un même tableau.

Tableau 18 : Processus liés aux exigences – Conception Préliminaire

Activités selon l'ISO 15288	Activités selon l'ISO 15288 enrichies par l'ISSdF
Stakeholder Requirements Definition	
Finaliser les exigences des parties prenantes	Raffiner le contexte du système
	Finaliser les exigences des parties prenantes
Approuver les exigences des parties prenantes	Approuver les exigences des parties prenantes
Analyser et maintenir les exigences des parties prenantes	Analyser et maintenir les exigences des parties prenantes
Requirements Analysis	
Définir les exigences système	Raffiner les fonctions systèmes et les déviations de flux
	Raffiner les frontières et interactions fonctionnelles du système et les dommages potentiels
	Raffiner les modes opérationnels, les scénarios opérationnels (nominaux et dysfonctionnels)
	Raffiner les exigences de niveau système dont les Safety Goals
Analyser et maintenir les exigences système	Analyser et maintenir les exigences système

Maintenant que l'architecture fonctionnelle a été choisie à la fin de la phase de définition de concept, il faut la raffiner et intégrer les fonctions techniques induites. Comme le concepteur connaît mieux les constituants auxquels sont allouées les fonctions, il peut mener à bien une analyse, fonctionnelle ou dysfonctionnelle, plus détaillée de l'architecture et concrétiser une vue extraite qu'est le *Functional Safety Concept* sur toute l'architecture. L'architecture fonctionnelle ainsi réalisée précède l'architecture organique, qui raffine les modifications apportées à celle-là en conception préliminaire. Ceux-ci permettent de concrétiser le *Technical*

Safety Concept. Comme précédemment, les activités *Functional Architecture Design* et *Physical Architecture Design* ont été regroupées dans un même tableau.

Tableau 19 : Processus liés aux architectures – Conception Préliminaire

Processus selon le SEBoK	Activités selon le SEBoK enrichies par l'ISSdF
Functional Architecture Design	
Raffiner la vue structurelle de l'architecture fonctionnelle	Raffiner les fonctions techniques et intégrer les fonctions techniques induites
	Raffiner les interfaces fonctionnelles et les flux de contrôle
	Mettre à jour les chemins critiques dysfonctionnels
	Raffiner la vue par l'ajout de concepts de sûreté dont les Safety Functions
Dériver et allouer les exigences aux fonctions techniques	Dériver et allouer les exigences techniques du niveau supérieur dont les *Safety Goals*
Raffiner la vue comportementale de l'architecture fonctionnelle	Raffiner les modes et états des différentes fonctions (nominaux, dégradés, sûr)
	Raffiner les scénarios fonctionnels caractérisant la dynamique d'enchaînement
Raffiner les exigences techniques	Raffiner les exigences techniques dont les Functional Safety Requirements pour chaque bloc logique
Physical Architecture Design	
Raffiner l'architecture organique	Raffiner la définition des constituants et leurs modes de défaillance organique
	Raffiner les interfaces organiques et mettre à jour les chemins critiques dysfonctionnels
	Raffiner les *Safety Mechanisms*

3.4.5 Le processus D'ISSdF en phase de conception détaillée

La conception détaillée permet de consolider l'architecture retenue en conception préliminaire et la définition des constituants élémentaires par des analyses provenant de différents experts focalisés sur la réalisabilité, les performances opérationnelles, etc., de la solution. De telles analyses reposent sur des retours d'expérience conséquents.

La conception détaillée étant hors du champ de cette thèse, la présentation des processus d'ISSdF associés sera encore plus succincte que celle faite à la section précédente. Ainsi, pour ce qui concerne la *Stakeholder Requirements Definition*, les activités d'ISSdF sont courtes puisqu'il faut juste mettre à jour le concept opérationnel et le référentiel des exigences des parties prenantes en cas de possible modification mineure. Il en est de même pour le référentiel des exigences système pour l'activité *Requirements Analysis*. Lors de la *Functional Architecture Design*, l'architecture retenue en conception préliminaire est consolidée. Suite à des analyses d'expert, des modifications mineures peuvent être réalisées. Cette configuration

se retrouvera dans le cas de la *Physical Architecture Design* et amènera à la définition des constituants élémentaires.

Tableau 20 : Processus d'ISSdF en conception détaillée

Processus selon le SEBoK	Activités selon le SEBoK enrichies par l'ISSdF
Stakeholder Requirements Definition	
Mettre à jour le concept opérationnel et le référentiel des exigences des parties prenantes	
Requirements Analysis	
Mettre à jour le référentiel des exigences systèmes	
Functional Architecture Design	
Consolider la vue structurelle de l'architecture fonctionnelle	Consolider les fonctions techniques
	Consolider les interfaces fonctionnelles et les flux de contrôle
	Mettre à jour les chemins critiques dysfonctionnels
Consolider la vue comportementale de l'architecture fonctionnelle	Consolider les modes et états des différentes fonctions (nominaux, dégradés, sûrs)
	Consolider les scénarios fonctionnels caractérisant la dynamique d'enchaînement
Analyser l'architecture fonctionnelle	Analyser et valider l'architecture fonctionnelle et le *Functional Safety Concept*
Physical Architecture Design	
Consolider l'architecture organique	Consolider la définition des constituants et leurs modes de défaillance organiques
	Consolider les interfaces organiques et mettre à jour les chemins critiques dysfonctionnels
	Définir les constituants élémentaires
Analyser l'architecture organique	Analyser et valider l'architecture organique et le *Technical Safety Concept* (AMDEC, etc.)

Le macroprocessus d'ISSdF vient d'être défini et décomposé en différents processus alignés sur les phases du projet de conception (définition de concept, conception préliminaire, conception détaillée). Le processus d'ISSdF devant être cohérent avec les référentiels de l'IS et de la SdF, mais aussi avec le modèle conceptuel du domaine proposé au précédent chapitre, il convient maintenant d'en tirer un bilan.

3.5 BILAN DU PROCESSUS PROPOSÉ

3.5.1 Respect des exigences

Tout d'abord, il est nécessaire de revenir sur l'exigence 3 du modèle conceptuel (Le modèle conceptuel doit être vérifié par la définition d'un processus d'ISSdF). Le processus proposé dans ce chapitre a été élaboré en se basant sur le modèle conceptuel de l'ISSdF. Il permet donc de vérifier cette exigence.

Le modèle proposé peut d'abord être comparé aux références d'IS retenues dans cette thèse, à savoir l'ISO 15288:2008, l'IEEE 1220:2005, l'EIA 632:2003 ou le *SEBoK*. Comme le montre le tableau suivant, la première exigence énoncée (Exi. 1 – Il doit être conforme à la norme ISO 15288:2008) est respectée, ce qui est logique vue la démarche adoptée. Ce tableau indique également que nos activités sont bien alignées avec les autres standards. En outre, le processus d'ISSdF suggéré propose des processus plus globaux, donc moins nombreux. D'une phase à l'autre (Phase de définition de concept, Conception Préliminaire, Conception Détaillée), les mêmes activités sont répétées avec des niveaux de raffinements différents. L'ingénierie du système se réalise avec quatre processus reliés ; les sorties d'un processus donné étant les entrées d'un autre processus. De cette façon, nous avons assuré la cohérence entre les différentes vues du modèle conceptuel et les activités du processus (Exi. 6 - Il doit assurer la cohérence entre les différentes vues des différents points de vue).

Tableau 21 : Comparaison de notre proposition avec les processus d'IS classiques

<table>
<tr><th>ISO 15288:2008</th><th>IEEE 1220:2005</th><th colspan="3">EIA 632:2003</th><th>SEBoK v1.4</th><th>Processus ISSdF</th></tr>
<tr><td></td><td></td><td colspan="3"></td><td>Business or Mission Analysis</td><td></td></tr>
<tr><td>Stakeholder Requirements Definition</td><td></td><td rowspan="4">System Design</td><td rowspan="2">Requirements Definition Process</td><td>Acquirer Requirements & Other Stakeholder Requirements</td><td>Stakeholder Needs and Requirements</td><td>Stakeholder Requirements Definition</td></tr>
<tr><td>Requirements Analysis</td><td>Requirements Analysis & Requirements Validation</td><td>System Technical Requirements</td><td>System Requirements</td><td>Requirements Analysis</td></tr>
<tr><td rowspan="2">Architectural Design</td><td>Functional Analysis & Functional Verification</td><td rowspan="2">Solution Definition Process</td><td>Logical Solution Representations</td><td>Logical Architecture Model Development</td><td>Functional Architecture Design</td></tr>
<tr><td>Synthesis & Design Verification</td><td>Physical Solution Representations</td><td>Physical Architecture Model Development</td><td>Physical Architecture Design</td></tr>
<tr><td></td><td>System Analysis</td><td colspan="3">System Analysis</td><td>System Analysis</td><td></td></tr>
</table>

Il convient maintenant de dresser un bilan plus détaillé de chaque processus et de déterminer le degré de couverture de chacun de ces processus avec le modèle conceptuel de l'ISSdF correspondant (Exi. 4). Cela permettra également de répondre à l'exigence Exi. 5 (interface possible avec les processus d'expertise de sûreté) en vérifiant si tous les concepts d'entrée nécessaire à une étude détaillée de sûreté sont définis dans le processus.

La première vue du modèle conceptuel de l'ISSdF regroupe les processus de définition des exigences des parties prenantes et d'analyse des exigences systèmes (*Stakeholder Requirements Definition*, *System Requirements Analysis*). La couverture du processus proposé par le modèle conceptuel de l'ISSdF n'est donc possible qu'en prenant ces deux processus (SRD, RA) d'ISSdF simultanément. Le tableau suivant précise ce point ; les lignes représentant les différentes activités du processus d'ISSdF ; les colonnes, la première vue du modèle conceptuel de l'ISSdF. Une croix est indiquée dès qu'un concept est manipulé lors de l'activité. Nous nous sommes focalisés sur la phase de définition de concept, les concepts étant repris, enrichis ou raffinés lors des phases suivantes.

Tableau 22 : Couverture pour les processus SRD & RA

Modèle Conceptuel Vue O&SRD / Processus ISSdF - SRD & RA	Stakeholder	Vehicle Occupant	Stakeholder Requirement	System	Context	Mission	Life Cycle	Hazard	Harmful Event	Operational Mode	Operational Situation	Hazardous Event	Harm	Technical Requirement	Safety Goal
Collecter les souhaits, besoins, etc.,	X	X	X	X											
Définir le contexte du système		X		X	X	X	X								
Identifier les services du système requis par les parties prenantes	X			X		X									
Définir les interactions du système	X	X		X	X										
Analyser et Maintenir les exigences des parties prenantes			X												
Définir les fonctions du système et les déviations de flux			X	X				X							
Définir les frontières et interactions fonctionnelles du système avec son contexte et les dommages potentiels			X	X	X				X				X		
Définir les modes opérationnels, les scénarios opérationnels et les séquences de situations opérationnelles			X	X				X	X	X	X	X			
Spécifier puis raffiner les exigences de niveau système dont les *Safety Goals*			X											X	X
Analyser et maintenir les exigences système														X	X

Une grille similaire a été utilisée pour apprécier la couverture du processus d'ISSdF associé à l'architecture fonctionnelle (FAD) et la deuxième vue du modèle conceptuel du domaine. Cette couverture est décrite par le tableau suivant.

Tableau 23 : Couverture pour le processus FAD

Modèle Conceptuel Vue FAD Processus ISSdF - FAD	Operational Situation	Operational Mode	Context	Technical Rqt	Safety Goal	Interface	Functional Safety Rqt	External Measure	Function	MEI Flow	Control Flow	Safety Function	Safety Measure	Functional Scenario	Functional Mode	Functional Failure Mode	Logical Block	Safe State
Définir les fonctions techniques & Identifier les modes de défaillance fonctionnels			X	X	X				X							X		
Définir les interfaces fonctionnelles et les flux de contrôle			X	X		X			X	X	X							
Définir les chemins critiques dysfonctionnels					X	X			X	X								
Raffiner la vue par l'ajout de concepts de sûreté dont les *Safety Functions*			X	X	X	X		X	X	X	X	X	X					
Dériver et Allouer les exigences techniques du niveau supérieur dont les *Safety Goals*			X	X	X		X											
Définir les modes et états des différentes fonctions	X	X							X		X	X			X	X		X
Définir les scénarios fonctionnels caractérisant la dynamique d'enchaînement	X								X		X	X		X				
Regrouper les éléments fonctionnels en blocs logiques						X			X								X	
Spécifier puis raffiner les exigences techniques dont *les Functional Safety Requirements*				X	X		X											
Évaluer la satisfaction des exigences techniques et des *Functional Safety Requirements*				X	X		X											
Évaluer les architectures et en sélectionner une						X			X	X	X	X		X	X	X	X	
Mettre à jour l'architecture logique en fonction des choix de composants				X			X		X	X	X	X	X	X	X	X	X	X

Enfin, la grille de la page suivante présente la couverture du processus d'ISSdF associé à l'architecture organique (PAD).

Tableau 24 : Couverture pour le processus PAD

Processus ISSdF - PAD / Modèle Conceptuel Vue PAD	Interface	Function	Safety Function	Technical Requirement	Functional Safety Requirement	Link	Port	Component	Technical Safety Requirement	Operating Mode	Failure Mode	Safety Mechanism	Failure, Error & Fault	Subsystem
Partitionner et allouer les éléments fonctionnels à des constituants réalisables	X	X	X	X	X			X						
Regrouper les constituants		X	X	X				X						X
Identifier les modes de défaillance		X	X	X	X			X			X			X
Définir les interfaces organiques	X					X	X	X						X
Définir les chemins critiques dysfonctionnels	X	X				X	X	X	X		X		X	
Définir les *Safety Mechanisms*						X	X	X	X		X	X	X	
Représenter les architectures organiques				X		X	X	X	X	X		X	X	X
Élaborer un modèle de décision avec des critères fonctionnels ou non-fonctionnels				X		X	X	X	X			X		X
Évaluer la satisfaction des exigences techniques incluant les *Technical Safety Requirements*				X	X				X					

Le processus d'ISSdF proposé satisfait à l'exigence Exi. 4 (prise en compte de la sûreté dans les différentes activités de conception) puisqu'à tout processus d'ISSdF présenté dans les tableaux de la section 3.4, il est possible d'associer des activités liées à la SdF. En revanche, le manque de mise en œuvre du processus présenté ne permet pas de valider totalement l'exigence Exi. 3 (Il doit pouvoir être validé par l'expérimentation sur un exemple concret.) : seule l'étude, longue, d'une mise en œuvre du processus d'ISSdF dans le cadre d'un projet de véhicule pourrait la valider, ce qui n'a pas été réalisable pendant la durée d'une thèse.

3.5.2 Intégration des méthodes de SdF

L'un des objectifs de cette thèse était de produire les livrables prescrits par l'ISO 26262:2011. Dans le premier chapitre, nous avons explicité notre problématique et rappelé les méthodes classiques de SdF que sont l'APR, l'AMDEC et l'analyse des arbres de défaillance. Cela a permis de mettre en avant l'objectif de ces méthodes et les différentes activités que leur mise en œuvre suppose. Dans un processus de sûreté de fonctionnement traditionnel, ces méthodes sont décorrélées de la conception de l'architecture système et réalisées par des spécialistes de la sûreté de fonctionnement. Notre proposition consiste en un processus d'ISSdF permettant de traiter ces aspects de sûreté de fonctionnement lors de la conception du système. Il est donc logique que les méthodes classiques de SdF n'apparaissent pas en tant qu'activités spécifiques dans le processus d'ISSdF. Néanmoins, comme nous le détaillerons dans les chapitres 4 et 5, il

est possible, avec le processus d'ISSdF proposé, d'aboutir à des livrables conformes à l'ISO 26262:2011. De plus, tous les éléments utiles pour faire une expertise en sûreté de fonctionnement sont définis dans ledit processus. Il sera donc plus simple pour un expert en SdF de s'approprier ces livrables pour mener des analyses quantitatives et pointues.

3.6 CONCLUSION

Le présent chapitre a permis de montrer comment opérationnaliser dans un processus spécifique le modèle conceptuel de l'ISSdF proposé au chapitre précédent. Pour ce faire, a été défini un macroprocessus d'ISSdF aligné sur les processus techniques de l'IS. Des activités liées à la SdF ont été associées à chacun des processus d'IS relatifs à la conception du système. Le macroprocessus proposé repose sur des vues dépendant à la fois du phasage des projets de conception (définition de concept, conception préliminaire, conception détaillée) et de la structure du modèle conceptuel (vue opérationnelle (externe au système), vue fonctionnelle, vue organique). Ce macroprocessus a défini le "pour quoi faire ?" de l'ISSdF. Il est désormais possible de préciser le "comment faire ?". Pour cela, nous nous focaliserons sur deux méthodes fortement liées à la SdF basés sur ces processus, à savoir la définition des exigences de système sûr et donc des *Safety Goals*, puis la conception d'une architecture fonctionnelle de système sûr dont le *Functional Safety Concept* pourra être extrait.

CHAPITRE 4
DÉFINITION DES EXIGENCES DE SYSTÈME SÛR

4.1 INTRODUCTION

Le second chapitre a présenté une première approche de l'ingénierie de systèmes sûrs de fonctionnement (ISSdF) en montrant les associations possibles entre les concepts du domaine de l'Ingénierie Système (IS) et ceux relevant de la sûreté de fonctionnement (SdF). Le troisième chapitre a proposé un premier modèle intégrant certains concepts des deux domaines mentionnés. Ces deux chapitres ont consisté en une première étape, proprement conceptuelle, vers l'ISSdF. Il convient désormais de s'intéresser aux dimensions pragmatiques du modèle conceptuel proposé, c'est-à-dire aux façons dont celui-ci peut être utilisé dans le cadre d'activités industrielles. Pour ce qui nous concerne, nous nous focaliserons sur la conception de systèmes et sur ce que l'ISO 26262:2011 appelle « *Hazard Analysis and Risk Assessmen*t ». Nous avons choisi de définir une méthodologie permettant de réaliser de façon rigoureuse la définition des exigences de système sûr. Cela vise à générer les exigences système, ce qui inclut les *Safety Goals*. Habituellement, l'APR est une activité propre à la SdF, non intégrée aux activités de définition des exigences. Pour pallier ce manque, ce quatrième chapitre présentera une méthodologie permettant de réaliser une analyse opérationnelle prenant en compte les aspects dysfonctionnels. Celle-ci sera cohérente avec l'ISO 26262:2011 et l'ISO 15288:2008.

La présentation de la méthodologie se fera en trois temps. Nous montrerons d'abord la nécessité d'une telle méthodologie en pointant les insuffisances des travaux existants en ce qui concerne les liens entre IS et SdF pour les points de vue opérationnel et fonctionnel externe (vues externes). Nous détaillerons ensuite la méthodologie proposée permettant de déterminer les exigences techniques incluant les *Safety Goals* à partir d'une analyse opérationnelle. Dans une troisième partie, nous montrerons la pertinence industrielle de notre méthodologie en l'illustrant à partir d'un cas tiré de la conception automobile. Nous nous intéresserons enfin à l'intérêt pragmatique d'une telle méthodologie.

4.2 ÉTAT DE L'ART

La problématique de ce chapitre fait l'objet d'une littérature établie. Aussi, avant d'exposer notre méthodologie, il convient de dresser un état de l'art pour vérifier si des méthodologies associant IS et SdF existent pour les points de vue opérationnel et fonctionnel externe. Pour ce faire, nous avons distingué les travaux centrés sur la sûreté de ceux focalisés sur le système.

4.2.1 Approches centrées sur la sûreté

L'Analyse Préliminaire des Risques (APR) est une méthodologie classique d'analyse de la sûreté d'un produit en conception amont. L'APR aide à construire des scénarios dysfonctionnels (Desroches, Leroy, & Vallée, 2003). Elle permet de les catégoriser à l'aide de facteurs comme la gravité ou le niveau d'intégrité (SIL, ASIL, DAL). Selon Vincoli (Vincoli, 2014) et Desroches et al. (Desroches, Leroy, & Vallée, 2003), l'APR suit trois étapes, à savoir : l'identification des scénarios dysfonctionnels, leur évaluation, et la proposition de couverture (ou d'atténuation) des risques comme des scénarios d'évitement.

On retrouve une grande partie de l'APR dans des normes anglo-saxonnes ou internationales. On peut citer le « *Hazard and Risk Analysis* » tel que défini dans l'IEC 61508 (IEC 61508, 2010), l'activité « *Aircraft (or System) Hazard Analysis* » dans la norme aéronautique ARP 4761 (ARP 4761, 1996), l'activité « *Hazard Analysis and Risk Assessment* » dans la norme automobile ISO 26262:2011, « *Risk assessment* » dans les normes sur les machines, comme le précise Hiettiko (Hiettiko, Malm, & Alanen, 2011), etc. Toutefois, comme le notent Jang et al. (Jang, Kwon, Hong, & Lee, 2015), même si les normes évoquées vont plus loin que l'APR classique, notamment pour ce qui concerne la définition explicite d'exigences de sûreté, il n'en demeure pas moins qu'elles ne proposent pas de méthodologie pour élaborer les scénarios dysfonctionnels. Par exemple, l'ISO 26262:2011 spécifie uniquement les entrées (« *item definition report* »), un sous-processus et des livrables. À suivre cette norme, on peut inférer le fait que l'analyse préliminaire des risques se déroule après l'Analyse Fonctionnelle Externe (AFE) du système étudié, et ce, même si les liens entre la SdF et l'AFE ne sont pas clarifiés dans le référentiel. L'élaboration des scénarios dysfonctionnels critiques demeure donc tacite. À ce jour, elle est analogique et repose sur le retour d'expérience des spécialistes de SdF.

Jang et al. (Jang, Kwon, Hong, & Lee, 2015) complètent l'APR en présentant une méthodologie déterminant l'ASIL du *Safety Goal.* Toutefois, les liens avec les processus et les activités de l'IS ne sont pas explicités. Kemmann (Kemmann, 2015) propose une approche structurée du « *Hazard Analysis and Risk Assessment* » focalisée sur la formalisation et l'analyse des situations opérationnelles. Sinha (Sinha, 2011), lui, revisite l'APR et suggère un processus intégrant les activités de conception. Le processus proposé par l'auteur est cohérent avec l'ISO 26262:2011. De ce fait, les travaux de (Sinha, 2011) peuvent être considérés comme

une base possible pour élaborer des scénarios dysfonctionnels. Cependant, une telle base n'est pas suffisante. Dans les travaux mentionnés, il n'y a, en effet, ni méthodologie complète d'élaboration des scénarios dysfonctionnels, ni le moyen d'éviter une analyse détaillée, et chronophage, de scénarios dysfonctionnels non critiques. Ce qui en réduit fortement l'intérêt pratique.

4.2.2 Approches focalisées sur le système

Si les approches centrées sur la sûreté ne permettent pas de créer un lien explicite entre l'activité de SdF et celle de conception concernant la définition des exigences d'un système sûr, d'autres travaux ont pour objectif de créer une telle association. Pour faire simple, deux approches peuvent même être distinguées : celle centrée sur les modèles (Yakymets, Jaber, & Lanusse, 2012), (Papadopoulos & McDermid, 1999) *vs*. celle centrée sur les processus (David, Idasiak, & Kratz, 2010). La première est focalisée sur l'objet à concevoir, alors que la seconde est centrée sur ce que le concepteur doit faire.

4.2.2.1 Approche centrée sur les modèles

Pour ce qui concerne l'approche centrée sur les modèles, on peut d'abord citer les travaux de Oliveira et al. (de Oliveira, et al., 2014), qui ont suggéré un processus d'analyse de sûreté fondée sur l'outil HipHops mentionné en premier chapitre. Ce processus tient compte des activités d'architecture. Toutefois, les activités de sûreté sont réalisées séparément des activités de conception, si bien qu'il n'y a pas de couplage explicite entre la SdF et l'IS. On peut ajouter les travaux de Yakymets et al. (Yakymets, Jaber, & Lanusse, 2012), qui ont sélectionné de nombreux langages de sûreté comme AltaRica (Kloul, Prosvirnova, & Rauzy, 2013) ou NuSMV (Cimatti, Clarke, Giunchiglia, Giunchiglia, & Pistore, 2002) pour mener à bien des études de sûreté. Après avoir annoté des diagrammes SysML (Block Definition Diagram, ou BDD, Internal Block Diagram, ou IBD, and State Machines, ou STM), les auteurs montrent comment transformer ces modèles SysML en un modèle écrit dans un langage de sûreté.

4.2.2.2 Approche centrée sur les processus

Pour ce qui concerne l'approche centrée sur les processus, il convient de noter que les normalisateurs à l'origine de l'ISO 15288:2008 (ISO 15288, 2008) ou de l'IEEE 1220:2005 (IEEE 1220, 2005) mentionnent explicitement la sûreté. Toutefois, les activités afférentes sont réalisées en parallèle des activités de conception prescrites dans les processus standardisés.

Leveson (Leveson, 2011) fait un pas de plus, en proposant un processus de sûreté fondé sur la théorie des systèmes, donc compatible avec l'IS. STAMP, ou *Systems-Theoretic Accident*

Model and Processes, repose sur des contraintes et des boucles de contrôle, ce qui est une vue partielle du modèle du système. STAMP ne traite pas des déviations de flux.

David et al. (David, Idasiak, & Kratz, 2010) ont élaboré une méthodologie pour réaliser une AMDEC en déterminant les informations nécessaires soit dans la vue fonctionnelle du modèle, soit dans une base de données spécifique, contenant par exemple les modes de défaillance génériques. Cressent et al. (Cressent, David, Idasiak, & Kratz, 2012) ont intégré cette méthodologie dans un processus de conception cohérent avec l'IS. Leur travail a pour mérite de clarifier les liens entre la sûreté et l'IS. Néanmoins, le processus qu'ils ont élaboré n'intègre pas les activités de sûreté qui peuvent être réalisées par des architectes systèmes. Or, comme l'écrit Clifton Ericson (Ericson, 2005), « *experience with, or a good working of, the particular type of system and subsystem is necessary in order to identify and analyze all hazards* »[14]. L'individu qui a le plus de connaissances sur le système lorsque celui-ci est conçu reste l'architecte système. Il est donc celui qui devrait réaliser l'analyse des phénomènes dangereux. De plus, quand Ericson écrit : « *The PHA methodology is uncomplicated and easily learned* »[15], l'auteur souligne deux points importants. D'abord, le fait qu'un niveau d'expertise en sûreté n'est pas nécessaire pour réaliser l'APR. Ensuite, que cette analyse peut reposer non seulement sur des connaissances tacites, mais aussi et surtout sur une véritable méthodologie pouvant être formalisée, généralisée, enseignée, etc.

4.2.3 Quel bilan tirer de l'état de l'art ?

L'état de l'art présenté dans les pages précédentes a montré qu'il n'existe pas à ce jour de solution permettant de bien coupler la sûreté à l'**architecture opérationnelle**, la SdF à l'IS, donc de mener à bien une véritable ISSdF. L'APR habituelle n'est pas assez liée aux activités de conception. Les résultats de l'APR ne sont pas réinterprétés par le concepteur, notamment l'architecte système, et restent au final dans le seul domaine de la SdF. Quant aux travaux conceptuellement plus proches de l'IS, ils laissent dans l'ombre certains aspects clefs de l'APR, comme la classification des scénarios dysfonctionnels. Or, un tel manque est dommageable. En effet, la norme ISO 26262:2011 demande aux bureaux d'études d'élaborer des exigences de sûreté de haut-niveau, ce qui suppose de réaliser l'analyse des risques lors de l'élaboration de l'architecture opérationnelle du système.

Les manques que nous venons de souligner permettent de bien cerner la problématique qui se pose à nous dans ce quatrième chapitre. Elle s'énonce de la façon suivante : comment définir les exigences de système sûr ? La voie choisie sera celle de l'enrichissement des modèles de

[14] Notre interprétation : l'expérience et/ou une bonne connaissance d'un type particulier de système ou de sous-système est nécessaire afin d'identifier et d'analyser tous les phénomènes dangereux
[15] Notre interprétation : la méthodologie APR n'est pas compliquée et facilement acquise

sûreté ou d'IS actuellement disponibles. La méthodologie proposée visera à définir les *Safety Goals* à partir de l'architecture opérationnelle, puis enrichir celle-ci à l'aide de l'analyse des risques.

4.3 UNE MÉTHODOLOGIE D'ISSDF DÉDIÉE À LA DÉFINITION DES EXIGENCES DE SYSTÈME SÛR

Avant de détailler une méthodologie satisfaisant la problématique exposée ci-dessus, il est nécessaire de spécifier la qualité des livrables attendus. Ainsi, la méthodologie proposée a vocation à aider l'architecte système. Elle doit donc à la fois réutiliser les concepts définis dans les précédents chapitres et être conforme aux processus d'IS et aux normes de la SdF. Comme nous l'avons indiqué, la voie choisie, dans cette méthodologie, est l'enrichissement de modèles existants dans le domaine de l'IS. Enfin, la méthodologie doit aboutir aux exigences de système sûr dont les *Safety Goal*, exigences qui seront ensuite traitées en conception aval, par exemple en architecture du système.

4.3.1 Processus de détermination des exigences de système sûr

La *vue d'ensemble* de la méthodologie proposée présentée ci-dessous permet d'avoir un panorama des étapes, des activités et des livrables d'une méthodologie d'ISSdF définissant les exigences de systèmes sûrs, ce qui inclut donc les *Safety Goals*. Dans la suite de ce chapitre, nous nous concentrerons uniquement sur les activités nouvelles que nous proposons ou que nous adaptons à partir des méthodes classiques, à savoir A4 (*Définir les exigences système*) et A5 (*Spécifier les exigences de niveau système dont les Safety Goals*). Les autres activités ne présentant pas de nouveautés majeures ont été définies dans le chapitre précédent et ne seront pas détaillées (A1, A2, A3 et A6).

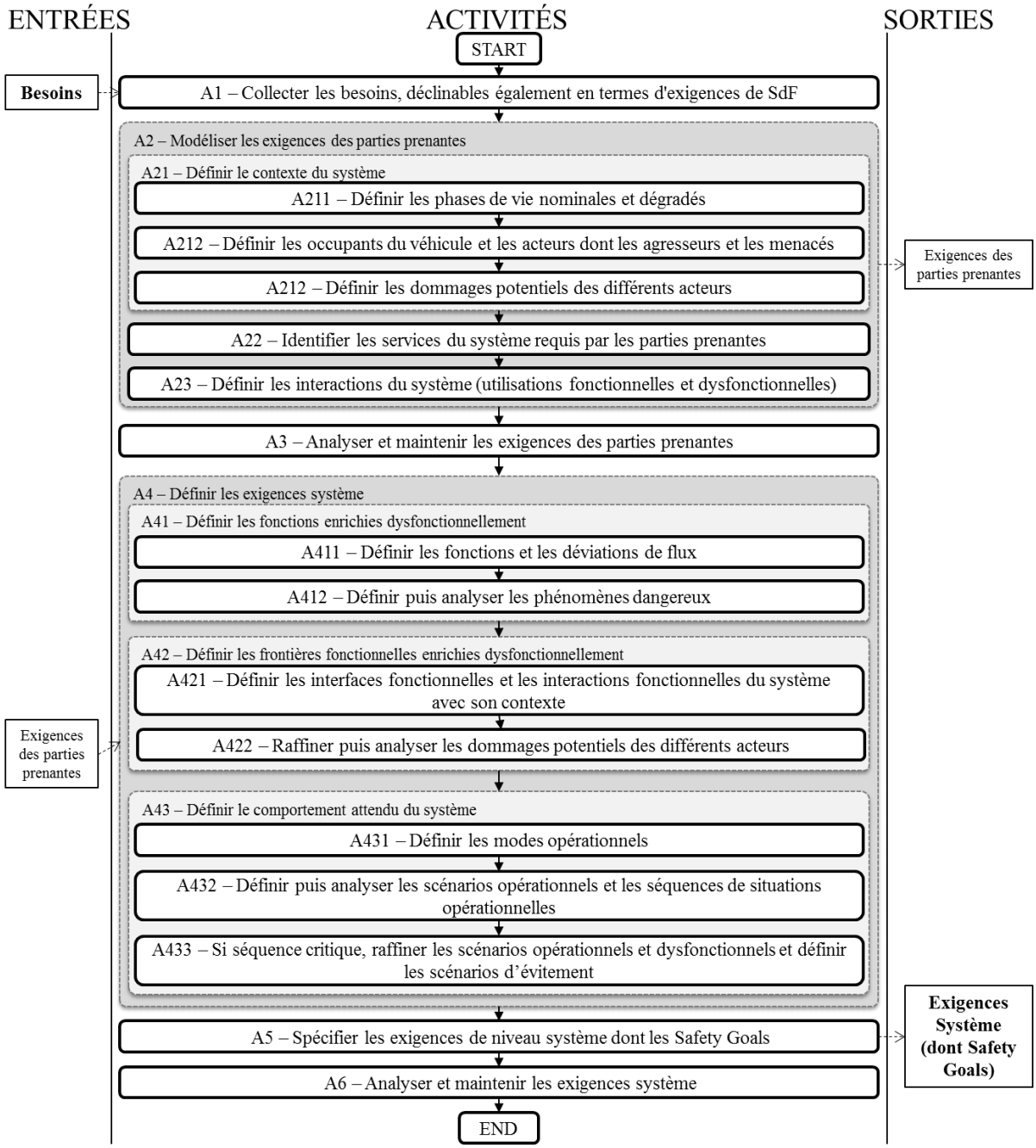


Figure 20 : Vue d'ensemble de la méthodologie proposée

4.3.1.1 Activité A41 – Définir les fonctions enrichies dysfonctionnellement

Cette activité se décompose en deux sous-activités que sont *Définir les fonctions et les déviations de flux* (A411) et *Définir puis analyser des phénomènes dangereux* (A412).

Partant de l'analyse des exigences des parties prenantes, il s'agit d'identifier les fonctions système qui permettent de rendre les services attendus. Pour chaque fonction, les déviations de flux possibles sont identifiées. Celles qui semblent être critiques seront les phénomènes dangereux (hazard) à éviter. Il est nécessaire de définir l'élément déclenchant le scénario

dysfonctionnel et l'évènement concluant le scénario dysfonctionnel, que sont les phénomènes dangereux et les préjudices potentiels à l'environnement. Comme Vincoli le mentionne (Vincoli, 2014), « *The PHA may be preceded with the preparation of a Preliminary Hazard List (PHL)* ». Cette liste répertorie tous les phénomènes dangereux du système. En ISSdF, celle-ci dérive du modèle de système vu que nous considérons que tout phénomène dangereux est conceptualisé comme une déviation anormale d'un flux de sortie d'une fonction système.

4.3.1.2 Activité A42 – Définir les frontières fonctionnelles enrichies dysfonctionnellement

Cette activité se décompose en deux sous activités que sont *Définir les interfaces fonctionnelles et les interactions fonctionnelles du système avec son contexte* (A421) et *Raffiner puis analyser les dommages potentiels des différents acteurs* (A422).

Une fois les phénomènes dangereux analysés, il convient d'identifier les états de chaque interacteur présent dans le diagramme de contexte, diagramme présentant les interactions entre le système, les occupants du véhicule et les autres éléments de l'environnement externe. Afin de s'assurer qu'un interacteur ou un état d'un interacteur existant n'ait pas été oublié, une piste pourrait être de s'appuyer sur une ontologie comme OASIS (*Ontology-based Analysis of Situation Influences on Safety*). Elle modélise différentes situations opérationnelles et a été récemment proposée par Kemmann (Kemmann, 2015). Dans le cas du véhicule, les états possibles pour un piéton sont : sauf, choqué, blessé, décédé. Les préjudices potentiels sont obtenus par combinaison de ces états et des évènements les induisant. Enfin, il est possible d'évaluer ces combinaisons à l'aide d'une table comprenant des classes de sévérité allant de *S0 non sévère* à *S3 critique*.

4.3.1.3 Activité A43 – Définir le comportement attendu du système

Cette activité se décompose en trois sous-activités que sont *Définir les modes opérationnels* (A431), *Définir puis analyser les scénarios opérationnels et les séquences de situations opérationnelles* (A432) et *Si Séquence critique, raffiner les scénarios opérationnels et dysfonctionnels et définir les scénarios d'évitement* (A433).

Une fois réalisées les activités A1 à A42, il est alors possible de définir un certain nombre de situations opérationnelles et d'évènements permettant de transiter entre les différentes situations opérationnelles. Pour rappel, comme nous l'avons défini dans le troisième chapitre, nous considérons une situation opérationnelle comme un mode opérationnel du système et un état de son environnement. L'enchaînement des situations opérationnelles permet de définir des séquences redoutées de situation opérationnelles prenant en compte l'aspect dysfonctionnel. Ces séquences peuvent être construites en instanciant le patron présenté ci-dessous, et ce, pour chaque déviation de flux.

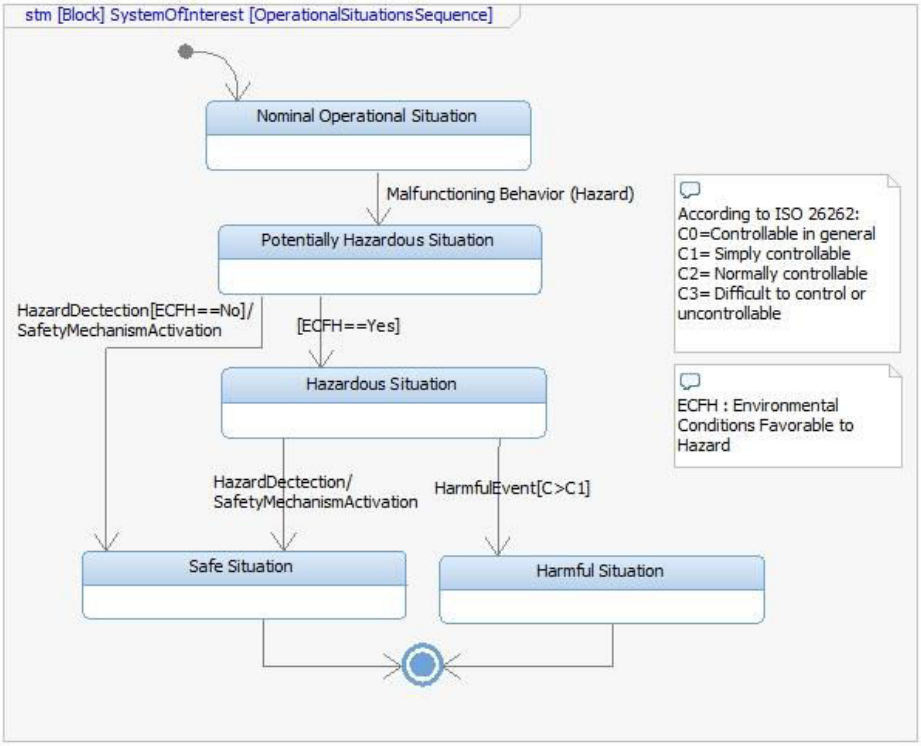


Figure 21 : Patron d'une séquence redoutée de situations opérationnelles[16]

En analysant les séquences redoutées de situations opérationnelles, il devient possible de définir le troisième critère de l'ASIL, à savoir la contrôlabilité. Ce critère est le plus subjectif. Il dépend à la fois de la compétence de l'utilisateur à faire fonctionner le système, mais aussi de la représentation que l'expert en SdF ou l'architecte système se font de cette compétence. Le choix de ce critère se fait en deux passes afin d'éliminer rapidement les scénarios non critiques, dont le traitement est chronophage et dispendieux. Premièrement, tous les phénomènes dangereux qui conduisent à une situation dangereuse (*Hazardous Situation*) contrôlable sont éliminés. Ils auront un critère C0, qui conduit à un ASIL inférieur à B. Dans une seconde phase, la contrôlabilité des scénarios dysfonctionnels est examinée par analyse du scénario ou retour d'expérience (C1 ➔ C3). Il est possible de définir l'ASIL des scénarios à l'aide d'une table comme celle présente ci-dessous. Enfin, les scénarios dysfonctionnels critiques dont l'ASIL est supérieur à B sont identifiés et sélectionnés pour des analyses plus détaillées.

[16] *Nous avons déjà présenté ce diagramme dans le chapitre 2 pour montrer les relations entre différentes situations opérationnelles. Nous le présentons dans ce chapitre comme un patron de conception.*

Tableau 25 : Table des ASIL.

Sévérité	Exposition	Contrôlabilité		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Après avoir défini les scénarios dysfonctionnels critiques en se basant sur la séquence redoutée de situations opérationnelles, il convient d'élaborer les scénarios d'évitement et d'identifier les modes sûrs. Pour rappel, nous reprenons la définition de la norme ISO 26262:2011, qui explicite qu'un mode sûr du système est un mode dans lequel les éléments humains (conducteur, passagers, piétons …) sont saufs.

4.3.1.4 Activité A5 – Spécifier les exigences de niveau système dont les Safety Goals

Il ne reste alors plus qu'à définir les *Safety Goals*. Ces exigences résument toutes les activités réalisées le long du processus exposé ci-dessus. Les *Safety Goals* forment une exigence d'entrée pour les processus de conception aval, par exemple la définition de l'architecture du système.

4.3.2 Adaptation de la méthodologie proposée

Selon le contexte organisationnel et la maturité en ISSdF, il est possible que cette méthode ne soit pas directement applicable ou qu'elle ne couvre pas totalement le périmètre désiré. En effet, certaines organisations intègrent des ingénieurs en sûreté dans les équipes de conception alors que dans d'autres, ils interviennent *ex post*, en aval, dans les dernières phases de conception. De plus, en raison de plusieurs paramètres (taille de l'entreprise, réactivité, adoption du changement, etc.), il est plus ou moins simple de passer de l'IS à l'ISSdF. Nous proposons donc dans cette section deux adaptations possibles. La première consiste en une étape

intermédiaire qu'est la définition des *Safety Goals* en s'appuyant sur l'architecture opérationnelle. La seconde adaptation consiste en la prise en compte des menaces et des agressions.

4.3.2.1 Définition des Safety Goals liée à la définition de l'architecture opérationnelle

Comme cela a été présenté dans (Mauborgne, et al., 2015), il est possible de réaliser une étape intermédiaire, utile pour une entreprise dissociant IS et SdF. Il est possible de prendre en compte un processus qui détermine les *Safety Goals* en se basant sur l'architecture opérationnelle. Dans ce cas, les activités sont liées, mais l'intégration entre fonctionnel et dysfonctionnel est moins forte. Comme le montre la figure ci-dessous, le résultat sera identique, mais il sera plus aisé de le réaliser avec un processus intégré comme celui exposé précédemment. De plus, il sera plus simple pour le concepteur de prendre en compte les retours d'analyse de sûreté dans un processus intégré, ce qui n'est pas négligeable.

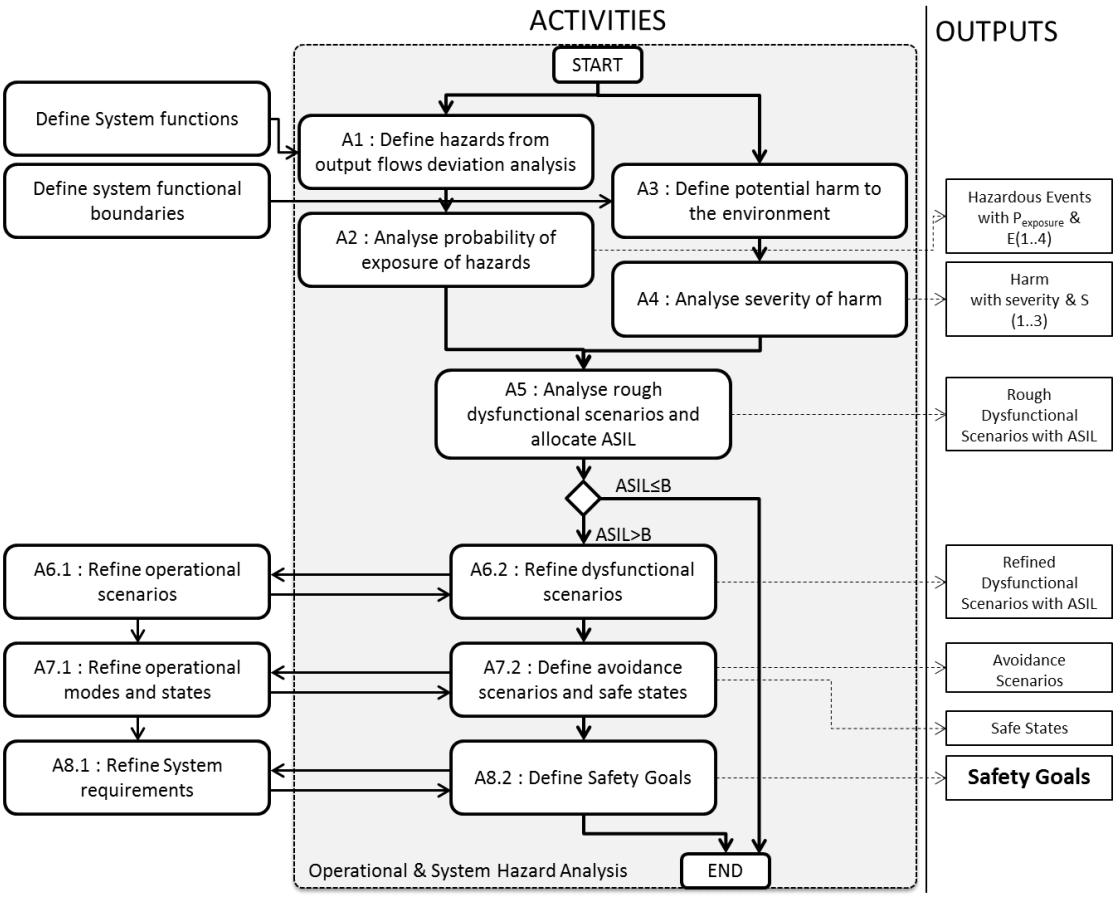


Figure 22 : Adaptation du processus avec intégration plus faible

4.3.2.2 Extension : vers la prise en compte des menaces et des agressions

Comme défini dans le deuxième chapitre, une menace est, pour le système, une agression potentielle venant de son environnement externe. L'agression, quant à elle, est ce que peut produire effectivement l'environnement extérieur sur le système. De plus, bien que l'ISO 26262:2011 ne traite que de sûreté fonctionnelle, il serait intéressant, voire judicieux, de s'intéresser à la menace et à l'agression, de sorte à couvrir un grand nombre de cas relevant de la sûreté. La méthodologie exposée ci-dessus ne prend pas en compte ces deux éléments puisqu'elle est focalisée sur la sûreté fonctionnelle, périmètre de la thèse.

La définition des exigences de système sûr relative aux menaces du système sur l'environnement est proche de la méthodologie proposée ci-dessus. La principale distinction concerne la détermination du phénomène dangereux (activité A312). Dans le cas du traitement des menaces, cette détermination n'est pas systématique et ne peut s'appuyer sur les fonctions système. Les phénomènes dangereux dus aux menaces correspondent à de nouveaux flux anormaux (du système vers l'environnement) qui n'ont pas d'existence fonctionnelle. Par exemple, une fuite d'huile n'est pas analysée, car il n'y a pas de flux fonctionnel entre le véhicule et la route. Il faut également faire attention aux interacteurs qui peuvent être impactés et qui n'ont pas nécessairement été pris en compte dans le diagramme de contexte. A ces ajouts près, la suite de la méthodologie demeure inchangée.

Pour ce qui concerne la prise en compte des agressions de l'environnement sur le système, va être modifiée l'activité *Définir les interfaces fonctionnelles et les interactions fonctionnelles du système avec son contexte* (activité A421). De nouvelles interactions fonctionnelles seront examinées et définies comme étant la dégradation par un interacteur. En revanche, avant de réaliser une analyse quant à la conséquence de ces dégradations, il faut vérifier si elles n'ont pas déjà été traitées par une analyse traitant une menace. En effet, la cause d'une dégradation pourra être l'effet d'une menace et il n'est pas utile de la spécifier deux fois. Par exemple, une menace de la partie du système contenant l'huile aura été traitée lors de la fuite d'huile. Il suffira donc de raffiner l'exigence de sûreté pour prendre en compte l'agression associée.

4.4 APPLICATION DE LA MÉTHODOLOGIE PROPOSEE

Les concepts et les activités de notre méthodologie d'ISSdF ayant été présentés, il devient désormais possible de l'illustrer. Comme pour tout ce mémoire de thèse, l'exemple choisi concerne l'accélération intempestive du véhicule. Nous traiterons donc la définition des exigences de système sûr pour ce cas. L'exemple choisi est simple à comprendre. Pour diverses raisons (confidentialité, compréhension de la méthode, etc.), nous avons fait le choix de simplifier les scénarios opérationnels et dysfonctionnels, les scénarios réels étant relativement plus complexe. De plus, nous avons repris uniquement les parties montrant l'originalité de la

méthodologie proposée. Enfin, dans l'exemple choisi, nous nous sommes placés dans une phase de vie d'utilisation normale.

4.4.1 Activité A41 – Définir les fonctions enrichies dysfonctionnellement

Un véhicule a pour fonction système *Accélérer*. Une fois celle-ci identifiée, il est ensuite nécessaire de définir les phénomènes dangereux (*hazards*), ce qui correspond à l'activité A412 de notre méthodologie. Pour l'exemple retenu, le flux pris en compte est celui entre le véhicule et la route pour la fonction système *Accélérer*. On peut identifier plusieurs déviations de flux pour cette fonction comme *Perte d'accélération* ou *Accélération intempestive*. Une des déviations du flux de sortie pour cette fonction qui semble critique est *Accélération intempestive*. Il s'agira donc du phénomène dangereux. En ANNEXE 3, nous expliquons à l'aide de variables empiriques en quoi ce choix a été judicieux.

L'ISO 26262:2011 contient des échelles pour définir un niveau de probabilité d'exposition. Dans le cas présent, la phase de conduite est une des phases de vie de base du cycle de vie pour un véhicule et *Accélérer* est une de ses principales fonctions. Grâce à la table définie dans la norme, le critère correspondant à l'accélération intempestive du véhicule est E4. En effet, E1 correspond à des situations exceptionnelles, E2 des situations rares, E3 des situations communes et E4 des situations régulières.

4.4.2 Activité A42 – Définir les frontières fonctionnelles enrichies dysfonctionnellement

La deuxième étape consiste définir les frontières fonctionnelles enrichies dysfonctionnellement. Pour cela, il est possible de déterminer les interfaces entre le système et les différents interacteurs et occupants du véhicule. Cela est effectué par le diagramme de contexte ci-dessous. Il aide à analyser quels sont les dommages potentiellement provoqués par le phénomène dangereux. Il est ainsi possible de rechercher quels sont les pires dommages potentiels pour les différents interacteurs, comme les passagers ou les piétons. Comme le suggère la norme ISO 26262:2011, un dommage sévère pour les passagers ou les piétons étant possible, le niveau de sévérité est S3.

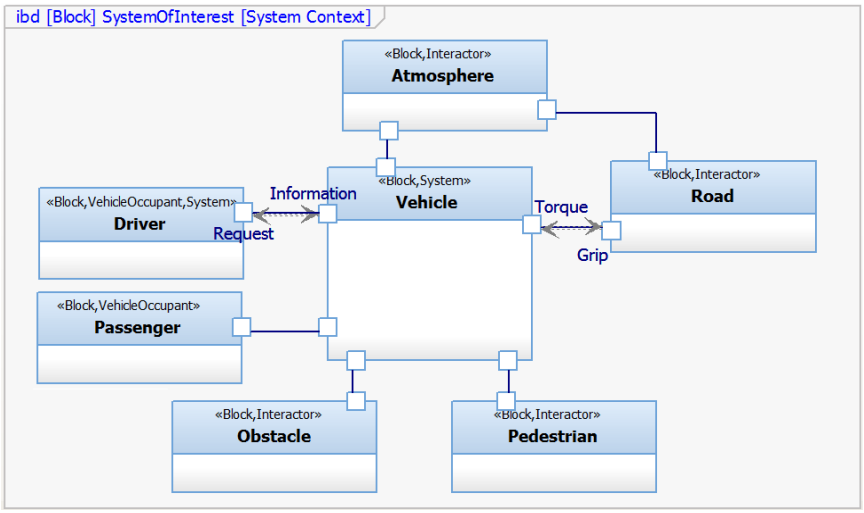


Figure 23: Diagramme de contexte

4.4.3 Activité A43 – Définir le comportement attendu du système

Une séquence redoutée de situations opérationnelles peut être élaborée à l'aide des informations précédemment produites. Afin d'avoir un résultat pertinent, ce scénario est une instanciation du patron présenté précédemment. Une condition environnementale favorable au danger pertinente pour cet exemple est la présence d'un piéton ou plus généralement d'un obstacle. Il existe d'autres conditions qui aboutiront aux mêmes conclusions, par exemple le fait d'être en virage. Il est donc inutile de traiter ces cas qui vont créer des modèles supplémentaires sans intérêt car aboutissant au même *Safety Goal*.

En analysant les situations opérationnelles associées à la fonction *Accélérer*, la séquence redoutée de situations opérationnelles peut être associée à certaines situations opérationnelles. Dans ce cas, la situation opérationnelle à prendre en compte est l'évitement d'obstacle en phase de conduite. Il est donc possible de considérer qu'au moins 90% des conducteurs pourront réagir en freinant s'il y a une accélération intempestive. De ce fait, le niveau de contrôlabilité s'élève à C2. L'ASIL associé est ASIL C (Combinaison de E4, S3 et C2). Nous pouvons donc considérer qu'il s'agit d'un scénario critique, qui doit être de ce fait défini.

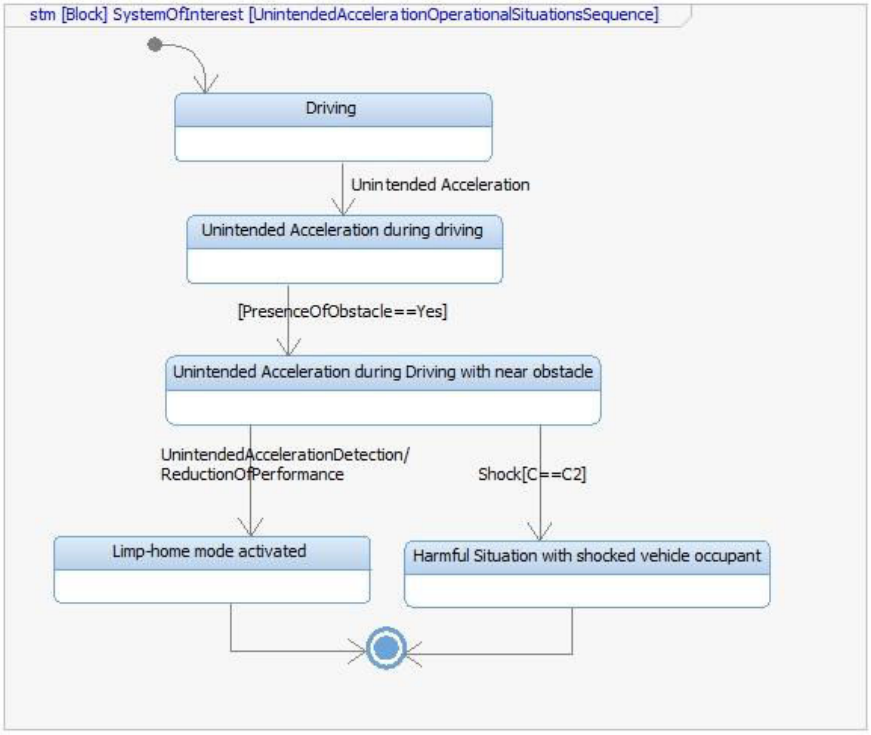


Figure 24: Séquence redoutée de situations opérationnelles pour l'exemple

Le choix du piéton comme ECFH est le plus contraignant. Afin d'avoir une séquence redoutée de situations opérationnelles la plus englobante possible, nous avons pris le cas général d'un obstacle. D'où la situation critique qui ne contient pas le piéton choqué. Comme nous l'avons déjà indiqué, l'analyse des scénarios opérationnels associés à la situation opérationnelle nominale est nécessaire. Ainsi, le scénario opérationnel de l'évitement d'obstacle est décrit par le diagramme de séquence ci-dessous. Ce type de diagramme a été choisi car il représente bien les interactions entre le système et son environnement.

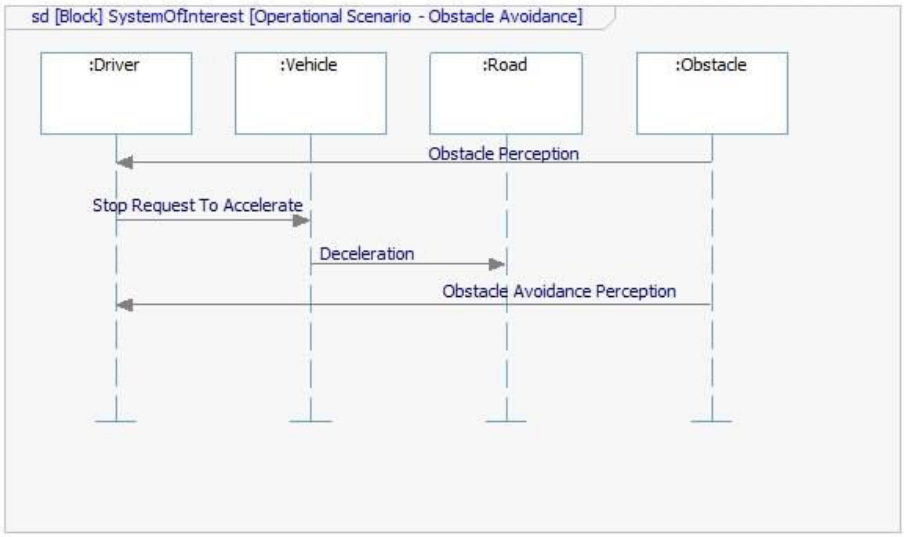


Figure 25: Scénario opérationnel de l'évitement d'obstacle

Grâce aux retours d'expérience des experts de SdF, il est possible d'élaborer le scénario dysfonctionnel en se basant sur la séquence redoutée de situations opérationnelles. Par exemple, il est possible de définir le délai pour percevoir l'accélération intempestive, ce qui permet de préciser le scénario dysfonctionnel en introduisant des variables physiques. Une fois encore, nous avons retenu un diagramme de séquence pour représenter ce scénario affiné.

À l'aide du scénario dysfonctionnel, des données plus détaillées sont disponibles pour élaborer le scénario d'évitement. Un scénario d'évitement possible est la limitation ou la réduction de la performance d'accélération du véhicule. Ce qui peut être représenté par un diagramme de séquence. Le scénario d'évitement permet aux occupants d'être saufs et donc de placer le système dans un *safe state* (mode sûr: véhicule arrêté avec conducteur sauf), *« avec un niveau raisonnable de risque* » comme l'énonce la norme ISO 26262:2011.

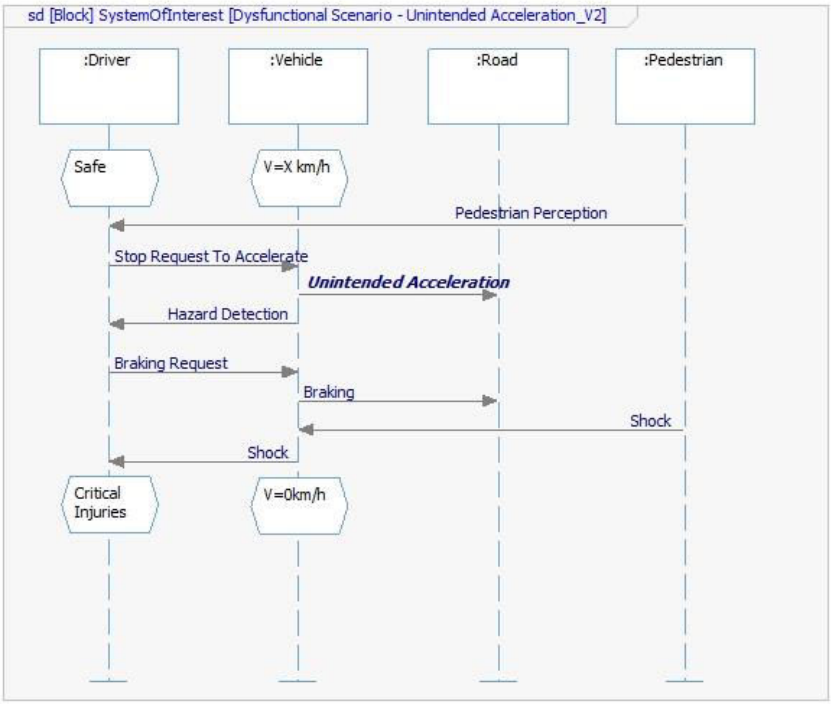


Figure 26 : Scénario dysfonctionnel de l'accélération intempestive

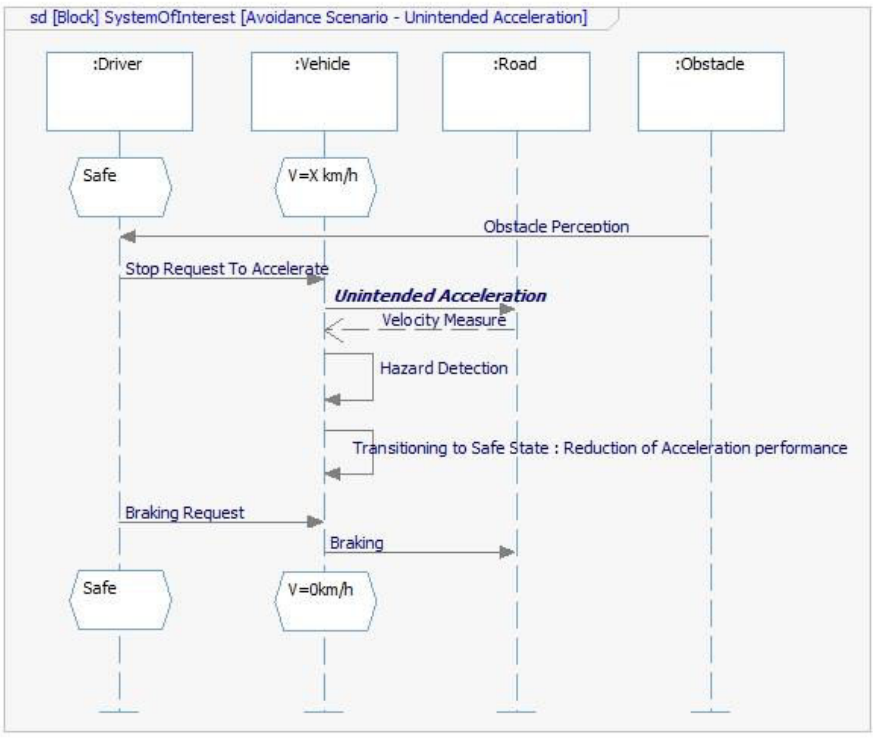


Figure 27: Scénario d'évitement

4.4.4 Activité A5 – Spécifier les *safety goals*

Les résultats obtenus peuvent être synthétisés sous forme tabulaire pour le *Safety Goal* correspondant. Un tel tableau présente une forme cohérente avec celle utilisée dans l'APR habituelle. Pour des raisons de lisibilité, ce tableau est présenté en deux parties. La partie haute comporte la description du scénario dysfonctionnel. La partie basse est composée de l'évaluation et de la proposition de la couverture du risque. Si nécessaire, la quantification du *Safety Goal* peut être réalisée par une analyse raffinée de sûreté impliquant l'expert en sûreté de fonctionnement. Par exemple, la simulation des scénarios dysfonctionnels et d'évitement permet de définir la valeur du Y% dans le *Safety Goal*.

Tableau 26 : Exigence de système sûr pour l'accélération intempestive[17]

System	Phase	Considered Flow	Hazard	Hazardous Event	Harmful Event	Operational Situation with Harm
Vehicle	Driving	Mechanical Energy to Road	Unintended Acceleration	Unintended Acceleration during driving	Shock	Critical Injuries of the driver

Probability of Exposure		Severity	Controllability	ASIL	Possible Avoidance Scenario	Safety Goal
X %	E4	S3	C2	C	Reduction of performances	the difference between driver's intend and the acceleration of the vehicle must be less than Y%

L'ensemble de notre méthodologie ayant été exposé, il convient maintenant d'en discuter l'intérêt pragmatique.

4.5 INTÉRÊT PRATIQUE DE LA MÉTHODOLOGIE PROPOSÉE

Dans le contexte de l'industrie automobile, et plus généralement en ce qui concerne la conception de systèmes complexes, il est pertinent d'utiliser l'Ingénierie Système basée sur les modèles (MBSE) ainsi que les analyses de sûreté basées sur les modèles (MBSA), ce qui a été la raison de ce mémoire sur l'Ingénierie de Systèmes Sûrs de Fonctionnement. Un avantage principal des approches MBSE est la réduction du temps de conception, comme l'énoncent Friedenthal et al. (Friedenthal, Moore, & Steiner, 2011). Les différents patrons proposés peuvent être réutilisés d'un projet à l'autre. Par ailleurs, notre méthodologie étant calée sur des processus d'IS, elle peut être reproduite à différents niveaux du système étudié, c'est-à-dire du système global à ses constituants élémentaires. Il est possible d'analyser des scénarios dysfonctionnels et donc de réaliser cette méthodologie pour tout sous-système ayant une interface avec l'environnement extérieur du système, voire des interfaces avec d'autres sous-systèmes. Dans un cadre de sous-traitance et co-développement d'un véhicule, cela signifie que le constructeur, responsable de l'ensemble, définit les exigences de système sûr pour un niveau système, et les fournisseurs de rang 1, 2, etc., celles relatives à des sous-systèmes de périmètres de plus en plus étroits.

La méthodologie proposée a un deuxième intérêt pratique, qui est de ne pas proposer des systèmes surprotégés. En effet, il sera nécessaire de répondre aux exigences de sûreté mais également aux exigences de performance et de coût. La protection contre des défaillances

[17] La valeur Y% placée dans le tableau et concernant le Safety Goal est en réalité une valeur absolue : de 1,5 m/s2 durant 1 seconde. Cette valeur sera supprimée dans la version finale pour des raisons de confidentialité.

pouvant avoir un impact négatif sur la performance ou le coût, ces exigences seront un garde-fou contre la proposition de systèmes surprotégés. Les activités de processus comme la définition des exigences de systèmes sûrs ne nécessitent pas de connaissances, de compétences spécifiques ou d'expertises rares et pointues en termes de SdF. Par exemple, les analyses quantitatives ou stochastiques ne sont pas dans le périmètre du processus d'ISSdF envisagé. Ces analyses demeureront le fait d'experts en sûreté ayant toutes les compétences requises. Comme indiqué dans le chapitre 2, tous les concepts nécessaires à ce type d'analyse sont présents dans ces modèles. Il sera donc aisé de les récupérer afin de réaliser ces analyses.

Le troisième intérêt pratique de notre méthodologie concerne la connaissance partagée entre l'expert en SdF et l'architecte système, si bien que leur collaboration peut s'en trouver facilitée, donc le déroulement de futurs projets amélioré.

Le quatrième intérêt pratique concerne l'économie des modèles. En effet, il y a un risque, avec le MBSE, de voir exploser le nombre de modèles à utiliser, donc leur combinatoire. L'heuristique proposée consiste à réduire le nombre de scénarios dysfonctionnels (donc de modèles) en recourant à un critère sélectif, ici l'ASIL. Ce qui a pour conséquence de prendre en compte tous les scénarios, dont ceux improbables, puis d'analyser uniquement les scénarios critiques. Pour le phénomène dangereux étudié dans ce chapitre, il y a au moins cinq dommages possibles (blessures mineures ou majeures pour le conducteur ou les passagers ou pour le piéton, destruction d'un élément de l'environnement) et au moins sept conditions environnementales favorables au danger (présence d'un obstacle, véhicule dans un virage à faible/moyenne/forte vitesse, véhicule sur route humide à faible/moyenne/forte vitesse), ce qui induit au moins 35 scénarios dysfonctionnels, donc au moins 35 *Safety Goals* ! En appliquant notre heuristique, il n'y a plus qu'un scénario dysfonctionnel et un seul *Safety Goal* couvrant tous les cas listés dans la phrase précédente. Concernant le choix de ce critère pour déterminer la criticité d'un scénario, il est vrai que la détermination de l'ASIL peut être considérée comme trop tacite ou empirique. Avec la définition du scénario dysfonctionnel en utilisant le patron proposé dans ce mémoire, il est simple de déterminer la sévérité. Concernant l'exposition, plusieurs tableaux sont déjà proposés en annexe de la partie 3 de la norme ISO 26262:2011. La méthodologie que nous proposons peut être en outre utile pour aider un expert à vérifier la détermination de l'ASIL. De plus, grâce à la description précise du scénario dysfonctionnel, il peut aussi vérifier la traçabilité entre le scénario opérationnel et l'ASIL.

Le cinquième et dernier intérêt pratique de notre méthodologie concerne sa cohérence avec l'IS et son intégration dans ce cadre de référence. Comme cela avait été montré en bilan du processus précédent, il y a une excellente couverture du processus et donc de la méthodologie par rapport au modèle conceptuel ISSdF proposé dans le chapitre 2. De plus, les différentes activités enrichissent fonctionnellement ET dysfonctionnellement le modèle du système.

4.6 CONCLUSION

Ce quatrième chapitre a permis de proposer une méthodologie d'ISSdF consacrée à la définition des exigences de systèmes sûrs. Elle permet de produire les exigences systèmes, dont les *Safety Goals*, à partir de différents modèles associés à l'architecture opérationnelle. Et ce, en suivant un processus conforme à l'IS. Notre proposition a été aussi construite en cohérence avec l'ISO 26262:2011. La méthodologie a été illustrée à l'aide du cas de l'accélération intempestive du véhicule, puis son intérêt pratique a été discuté.

Dans la méthodologie proposée, les éléments du scénario dysfonctionnel sont élaborés dans le modèle du système, donc au sein du domaine de l'IS. Il n'y a donc pas recours à un modèle relevant de la seule SdF. Une fois les *Safety Goals* définis, il convient de développer une architecture fonctionnelle sûre. Pour ce faire, un autre principe de la SdF va être utilisé, à savoir la propagation de défaillance. C'est grâce à lui qu'il devient possible de proposer le « *Functional Safety Concept* » attendu dans la norme ISO 26262:2011. Ce point va être explicité dans le prochain chapitre.

CHAPITRE 5
CONCEPTION FONCTIONNELLE DE SYSTÈME SÛR

5.1 INTRODUCTION

Dans une approche classique de la conception et de la Sûreté de Fonctionnement (SdF), le concepteur réalise une analyse fonctionnelle externe, puis l'ingénieur spécialiste en SdF mène une analyse préliminaire des risques (APR) pour définir les scénarios et les événements redoutés. À l'étape suivante, le concepteur réalise une analyse fonctionnelle interne et définit une architecture fonctionnelle (structure et comportement) dont l'ingénieur en SdF évalue la sûreté en recourant à des méthodes telles que l'HAZOP, l'AMDEC, etc.

L'ISO 26262:2011 donne un cadre conceptuel à partir duquel on peut mieux intégrer la conception et la SdF. Partant d'un *item definition*, le *Hazard Analysis and Risk Assessment* définit des scénarios dysfonctionnels et les *Safety Goals* avec leur niveau d'ASIL. La norme recommande de traduire les *Safety Goals* en *Functional Safety Requirements* associés aux différents constituants de l'architecture fonctionnelle. Les tâches peuvent être assistées par les méthodes de SdF classiques. Toutefois, la norme n'explicite pas comment ces tâches doivent être réalisées. Il convient donc maintenant de traiter ce point et d'expliciter comment élaborer une architecture fonctionnelle de système sûr (AFSS). L'idée de base est d'enrichir une architecture fonctionnelle d'une vue dysfonctionnel qui serait le dual de la vue fonctionnelle. Un tel enrichissement peut être réalisé par l'architecte système (AS).

Pour traiter cette question, nous commencerons par dresser un panorama des méthodes permettant d'analyser et d'enrichir l'architecture fonctionnelle d'un point de vue dysfonctionnel. Celles-ci ne répondent pas à toutes les exigences du processus d'Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF). Aussi proposerons-nous d'abord un processus d'élaboration d'une AFSS cohérent avec le processus de *Functional Architecture Design (FAD)*, avant de présenter des modèles inspirés à la fois des domaines de la SdF et de l'Ingénierie Système (IS). L'enrichissement dysfonctionnel proposé concernera à la fois l'aspect structurel et statique, puis comportemental et dynamique, des architectures fonctionnelles. Enfin, un exemple sera donné, qui concerne la génération d'énergie et la combustion. Cet exemple permettra de juger de l'intérêt pratique des résultats méthodologiques proposés.

La démarche suivie dans ce chapitre peut être décrite à l'aide du diagramme en ''nœud papillon'' de Kemmann (Kemmann, 2015). Dans le chapitre précédent, nous avons identifié et

classifié les phénomènes dangereux menant à un évènement critique et leurs effets potentiels (partie droite du nœud papillon). Dans le présent chapitre, nous nous focaliserons sur les causes amenant à ces phénomènes (partie gauche du nœud papillon). Pour ce faire, et afin de délimiter notre périmètre d'étude, nous définirons le point de départ - une architecture fonctionnelle définie par un processus de *Functional Architecture Design* - et les attendus, à savoir une AFSS porteuse du *Functional Safety Concept*. Nous prendrons alors en compte les limites et les lacunes des méthodes existantes, ce qui permettra de mettre en avant les particularités de notre contribution. Puis, après avoir défini celle-ci, nous l'appliquerons à un exemple afin de vérifier son application. Enfin, nous terminerons ce chapitre en rappelant l'originalité de l'approche proposée.

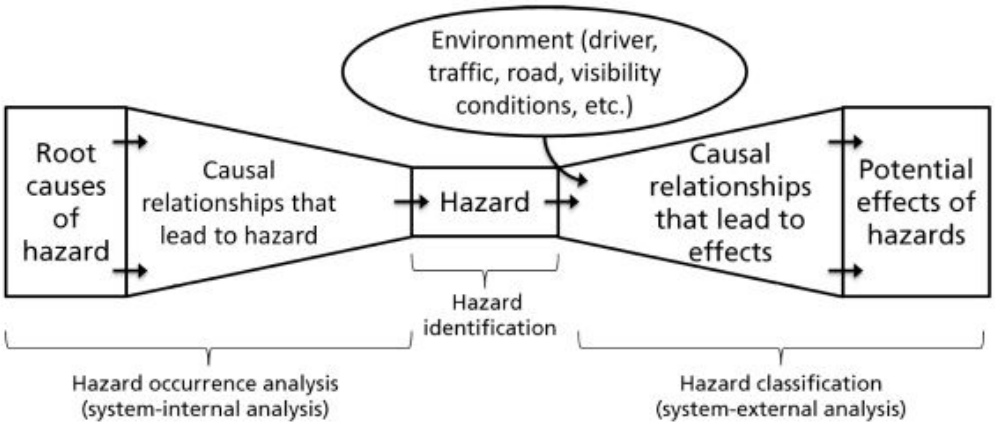


Figure 28 : diagramme nœud papillon pour l'analyse des risques (Kemmann, 2015)

5.2 ÉLABORER UNE AFSS : BREF ÉTAT DE L'ART

Comme énoncé en introduction, l'ISO 26262:2011 recommande la définition d'un *Functional Safety Concept* comportant la spécification des *Functional Safety Requirements*, de leur allocation aux éléments de l'architecture fonctionnelle et des interactions nécessaires afin de satisfaire les *Safety Goals*[18]. Il est donc essentiel de déterminer comment cet ensemble peut être conçu et s'il existe des méthodes permettant de réaliser intégralement ou partiellement le *Functional Safety Concept*.

Les référentiels de l'IS sont peu précis en matière d'élaboration de l'aspect dysfonctionnel dans l'architecture fonctionnelle (ou logique). L'ISO 15288:2015 (ISO 15288, 2015) se contente de mentionner une seule sous-activité d'analyse des risques, réalisée *ex post*, en fin de

[18] *functional safety concept : specification of the functional safety requirements, with associated information, their allocation to architectural elements, and their interaction necessary to achieve the safety goals*

conception de l'architecture, pour obtenir une solution réalisable. Le *SEBoK* suggère de réaliser une AMDE (Analyse qualitative des modes de défaillance et de leurs Effets), puis de mettre à jour l'architecture si nécessaire. Une telle mise à jour peut consister en l'ajout de barrières si l'architecture contient une fonction pouvant causer un phénomène dangereux critique. L'IEEE 1220:2005 est plus complète. En plus de prescrire la réalisation d'une AMDE, elle recommande de modéliser la causalité dysfonctionnelle et les défaillances afin d'avoir une vision complète du système [19]. On le constate aisément : la détermination de l'aspect dysfonctionnel d'une architecture fonctionnelle ou d'un *Functional Safety Concept* ne peut se faire en recourant seulement aux référentiels de l'IS existants. Heureusement, plusieurs méthodes utilisées par l'ingénieur spécialiste en SdF peuvent être utiles. Afin d'en dresser le panorama, nous partirons d'une typologie distinguant les méthodes statiques *vs*. les méthodes dynamiques ; l'aspect dysfonctionnel d'une architecture fonctionnelle ayant à la fois un impact sur la structure et sur le comportement du système.

5.2.1 Approches structurelles

Nous ne reviendrons pas sur l'AMDEC ou les arbres de défaillance, outils classiques mentionnés dans le premier chapitre. Nous avons pu identifier quels étaient les intérêts de ces outils et comment il était possible d'atteindre les objectifs souhaités (sans pour autant nécessairement produire les mêmes modèles[20]). Pierre Mauborgne (Mauborgne, Labe, Smouts, Stojanovic, & Delgado, 2013) a proposé une méthode permettant de définir des arbres de défaillance à partir d'un modèle d'IS. D'autres approches existent. Par exemple, Jean-Baptiste Léger (Leger, 1999) et Pierre Cocheteux (Cocheteux, 2010) ont exploré, pour modéliser des systèmes industriels sûrs, les équations logiques des déviations de flux d'une fonction technique en se référant à l'HAZOP. Cela permet d'avoir les effets possibles d'une déviation de flux d'entrée, ou d'un mode de défaillance, sur une déviation de flux de sortie d'une fonction. Comme nous l'avons évoqué en premier chapitre, Yannis Papadopoulos et al. (Papadopoulos & McDermid, 1999) se sont focalisés sur la génération d'arbres de défaillance et d'AMDEC à partir de diagrammes structurels. Les modèles proposés relèvent du niveau organique ; ils sont toutefois facilement utilisables au niveau fonctionnel. La modélisation d'un composant enrichie par la modélisation de son comportement dysfonctionnel et la création de liens fonctionnels entre les composants permet de propager certaines défaillances et d'obtenir un arbre de défaillance. Cependant, le résultat obtenu par (Papadopoulos & McDermid, 1999) est un arbre

[19] « *Failures* […] *are modeled to completely understand system impacts*. »
[20] *À titre d'exemple, un arbre de défaillance n'est pas forcément un modèle cohérent avec une approche d'ingénierie système, qui passe d'une strate système aux strates sous-systèmes et constituants par la définition d'architecture.*

des coupes minimales alors que nous souhaitons avoir les différents chemins de propagation critiques. Il est vrai que les coupes minimales représentent les combinaisons d'évènements les plus courtes menant à l'évènement redouté. Toutefois, il peut être assez complexe de retrouver, au sein d'une vue structurelle de l'architecture fonctionnelle, d'où proviennent certaines combinaisons. Les chemins de propagation critiques permettraient de mieux intégrer une *Safety Function* dans une telle architecture. Il serait alors plus aisé de déterminer (1) des flux qui ont en amont plusieurs modes de défaillances ou (2) des déviations de flux critiques. Pour conclure, on peut donc dire que de nombreuses approches reposent sur la vue structurelle de l'architecture, mais que leurs résultats ne correspond pas au livrable prescrit par l'ISO 226:2011, à savoir le *Functional Safety Concept*.

5.2.2 Approches comportementales

D'autres approches reposent sur la vue comportementale de l'architecture fonctionnelle. On peut d'abord citer le langage AltaRica, dont la dernière version développée dans (Batteux, Prosvirnova, Rauzy, & Kloul, 2013) permet de réaliser un lien entre les langages de spécification et les modèles classiques de sûreté. Cette version repose sur les *Guarded Transition Systems*, formalisme de modèle état/transition. Pour chaque composant, il est possible de dresser un modèle comportemental incluant la sûreté. En s'appuyant sur les flux échangés entre différents composants, un modèle dysfonctionnel d'un ensemble de composants est réalisé par composition de ces différents modèles ou en s'appuyant sur les flux échangés entre les différents composants. Cette solution est proche du résultat souhaité par l'ISO 26262:2011. Cependant, la modélisation AltaRica est disjointe de la modélisation fonctionnelle. De plus, il n'y a pas de méthodologie précisant comment élaborer ce type de modèle.

Un autre formalisme état/transition, classique dans les études de sûreté, concerne les chaînes de Markov. Celles-ci peuvent être liées avec AltaRica comme cela est montré dans (Brameret, Roussel, & Rauzy, 2013) et plus spécifiquement avec les *Safety Mechanisms* comme dans (Cherfi, Leeman, Meurville, & Rauzy, 2014). AltaRica et les chaînes de Markov permettraient de faire une vérification probabiliste du respect d'une exigence de sûreté pour une *Safety Function* ou un *Safety Mechanism*. D'autres formalismes état/transition sont utilisés en SdF. Nous pouvons citer les réseaux de Petri, les *Stochastic activity networks* ou les Automates stochastiques hybrides. Ces différents formalismes reprennent les concepts de modes et d'états avec des transitions dépendant d'évènements dûs à des défaillances. Ils incluent également les concepts de taux de défaillance (ou d'autres concepts ayant trait à la probabilité de défaillir). Ces concepts sont toutefois spécifiques à l'activité des experts en SdF. Comme notre thèse est focalisée sur l'architecte système, il est donc logique qu'ils ne soient pas présents dans notre modèle conceptuel ou dans les méthodologies proposées. Cependant, l'intérêt d'un modèle à

états pour spécifier le comportement souhaité d'une *Safety Function* est vérifié. Enfin, dans (Bouissou & Bon, 2003), les auteurs proposent d'étendre le périmètre des arbres de défaillance en introduisant des aspects dynamiques, et ce, à l'aide des BDMP (*Boolean logic Driven Markov Process*). Cependant, le formalisme proposé n'est pas d'un abord facile pour qui n'est pas un expert en SdF. Il n'aide pas l'AS à enrichir une architecture d'un point de vue dysfonctionnel.

En plus de leur difficulté d'emploi pour les Béotiens en SdF, le principal inconvénient des différentes approches dynamiques évoquées réside dans le fait qu'elles ne sont effectuées qu'une fois le modèle fonctionnel arrêté. Elles ne permettent donc pas d'entretenir le dialogue que mène l'AS entre l'architecture fonctionnelle et son analyse à l'aune du critère de sûreté. Par contre, ces approches permettent de mener des analyses de SdF quantitatives et détaillées.

5.2.3 Synthèse

Qu'ils soient statiques ou dynamiques, les modèles cités issus de la SdF ne permettent pas de répondre à notre exigence 2 du processus d'ISSdF à propos de la conformité avec l'ISO 26262:2011, en particulier, du besoin de définition du *Functional Safety Concept*. De plus, ces modèles, qui ne sont pas liés aux activités du processus de *Functional Architecture Design*, sont utilisés *ex post*, en aval de la conception. Enfin, ils dépendent de l'expérience et de l'expertise de l'ingénieur spécialiste de SdF, si bien que leur partage avec l'AS paraît difficile à mener. Malgré ces limites, les modèles mentionnés permettent de souligner un point important. Élaborer un *Functional Safety Concept* consistant à définir les *Functional Safety Requirements* et spécifier des *Safety Functions* suppose de combiner deux vues, structurelle et comportementale, comme nous l'expliciterons ensuite.

La première vue repose sur la logique booléenne. En établissant et en formalisant des relations causales entre les déviations des entrées et les déviations des sorties pour chaque fonction de l'architecture, il devient possible de déterminer les chemins critiques dysfonctionnels de la causalité d'un phénomène dangereux. Grâce à eux, l'AS peut dériver les exigences de sécurité et définir éventuellement des fonctions de sécurité (*Safety Function*). L'approche comportementale, quant à elle, l'aide à déterminer les modes et les états dysfonctionnels des fonctions concernées. Ce qui permet de déterminer à la fois le comportement en cas de défaillance de chaque fonction, un éventuel mode sûr, ainsi que les échanges entre cette fonction et les fonctions de sécurité associées.

Ces deux approches, structurelle et comportementale, seront prises en compte dans les modèles proposés dans ce chapitre. Toutefois, avant de les détailler, il convient de définir précisément ce qu'est une architecture fonctionnelle de système sûr (AFSS).

5.3 UNE AFSS

Nous venons dresser un rapide panorama des méthodes existantes qui pourraient permettre d'enrichir dysfonctionnellement une architecture fonctionnelle. Certaines répondent partiellement à notre problématique. Pour mieux satisfaire aux exigences du processus d'ISSdF, nous proposerons un patron (*pattern*) associant les deux vues, structurelle et comportementale, d'une AFSS. Cela permettra de cadrer les méthodes proposées dans la suite de ce chapitre. De plus, la proposition d'un patron est un premier pas vers la réalisation d'une architecture fonctionnelle de système sûr. En effet, en cadrant la conception fonctionnelle, on définit intrinsèquement des bonnes pratiques de conception.

5.3.1 Fonctions et patrons fonctionnels

Il est admis que la fonction est un concept clef de l'ingénierie. Quel que soit le niveau de raffinement avec lequel l'architecture fonctionnelle est décrite, la plupart des auteurs supposent ou définissent explicitement le fait qu'il existe des patrons de conception fonctionnelle (patron fonctionnel). En tant que patrons, ils peuvent être reproduits d'un système à l'autre, d'une strate d'un système donné à l'autre, etc. Dans l'approche SADT, on a une séparation entre fonctions et données, si bien qu'on considère que la fonction transforme un flux d'entrée en flux de sortie à l'aide de ressources dédiées (Ross, 1977). Son activation ou son déclenchement est réalisé par le biais d'un flux de contrôle / commande. Ce *patron* fonctionnel est désormais classique. Il a donné lieu à différents raffinements. Sans prétendre à l'exhaustivité, nous pouvons mentionner les travaux :

1. de Claude Feliot et de Jean-Louis Le Moigne dans (Feliot, 1997) et (Le Moigne, 1990) qui ont proposé une typologie des flux finalisants et des fonctions. Les flux peuvent prendre la forme de flux de matière, d'énergie, d'information, et les transformations de flux opérées par la fonction concernent le temps, l'espace, la forme ;
2. de Frédérique Mayer (1995), qui a associé des modalités aux flux. La fonction transforme ainsi un flux principal nommé *Devoir Faire Entrant* en flux principal de sortie nommé *Devoir Faire Sortant*. Des flux secondaires entrants ou sortants sont nommés *Pouvoir Faire Entran*t et *Pouvoir Faire Sortant*. L'intérêt de cette approche modale est qu'une fonction donnée peut avoir un spectre plus large que celui défini à partir de SADT. Ainsi, des sorties secondaires de ladite fonction peuvent avoir une utilité pour une autre fonction du système, par exemple le chauffage, la récupération d'énergie, pour le groupe motopropulseur automobile. De plus, des connaissances sont nécessaires pour réaliser la fonction étudiée, d'où la définition d'un flux de *Savoir Faire*. Enfin, le contrôle et la commande de la fonction sont réalisés par les flux de *Vouloir Faire*.

Les patrons fonctionnels SADT et modaux sont mis côte à côte dans la figure suivante.

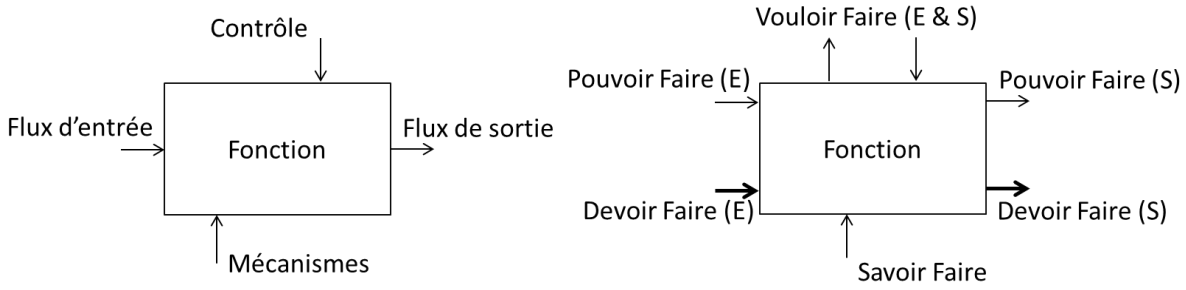


Figure 29 : Patrons fonctionnels

5.3.2 L'architecture fonctionnelle comme association de fonctions

5.3.2.1 À la recherche de patrons fonctionnels

L'AFIS (2007) définit une architecture fonctionnelle comme la « *description du système sous forme d'un arrangement de fonctions, de leurs sous-fonctions ainsi que de leurs interactions* ». Le *SEBoK* est plus précis : « a *functional architecture is a set of functions and their sub-functions that defines the transformations of input flows into output flows performed by the system to achieve its mission* »[21]. Pour les auteurs du Le *SEBoK,* l'architecture fonctionnelle est décomposable en deux vues :

1. la vue statique ou structurelle, qui recense et agence les fonctions développées par le concepteur ;
2. des vues comportementales, définissant les modes et les états fonctionnels, l'enchaînement des fonctions, voire la logique de commande de ces mêmes fonctions.

L'approche du *SEBoK* est d'une indéniable actualité. Avec le développement de solutions automatisées, des fonctions spécifiques de contrôle / commande sont nécessaires pour activer les fonctions techniques vues comme des entités opérantes. Une alternative s'offre alors à l'AS. L'approche dualiste considère les fonctions de contrôle / commande comme ontologiquement distinctes des fonctions techniques. L'approche intégrée, quant à elle, admet que celles-là font totalement partie des fonctions techniques, qu'elles n'ont de raison d'être, de sens, qu'en rapport à celles-ci. Comme le montre la figure suivante, ces deux approches donnent lieu à deux architectures fonctionnelles différentes. Notre choix se portera sur l'approche intégrée car le contrôle / commande peut coordonner plusieurs fonctions.

[21] *Une architecture fonctionnelle est un ensemble de fonctions et de leurs sous fonctions qui définit les transformations d'entrées en sorties exécutées par le système pour accomplir sa mission.*

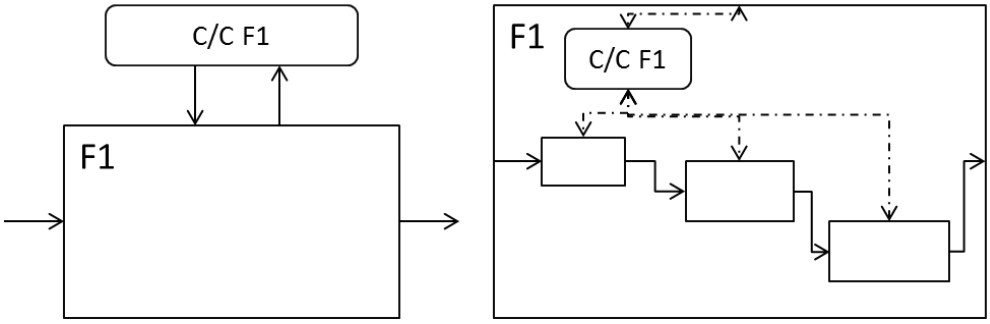


Figure 30 : Fonctions de contrôle/commande par rapport aux fonctions techniques

Des auteurs dualistes ont raffiné les sous-fonctions de la fonction de contrôle / commande. Ainsi, au moins deux *patrons* architecturaux sont proposés dans la littérature. Le premier est une application de la boucle de rétroaction de la Cybernétique. Il décompose chaque fonction en quatre sous-fonctions que sont *Opérer*[22], *Actionner*, *Observer*, et *Contrôler,* comme cela est exposé dans (Dobre, 2010). Pétin, Iung, et Morel (Pétin, Iung, & Morel, 1998) ont réalisé des travaux complémentaires. Leur proposition associe la validation des objectifs, l'élaboration des rapports, la validation des observations et l'exécution des actions. Les fonctions *Observer* et *Actionner* réalisent une transformation de flux supplémentaire, celle de nature. Une fonction *Observer* peut prendre en entrée un flux de matière ou énergie, et produire en sortie un flux d'informations. Inversement, une fonction *Actionner* transforme un ordre (provenant de *Contrôler*) en un flux physique, de matière ou énergie. La figure suivante place ces deux patrons fonctionnels évoqués en vis-à-vis.

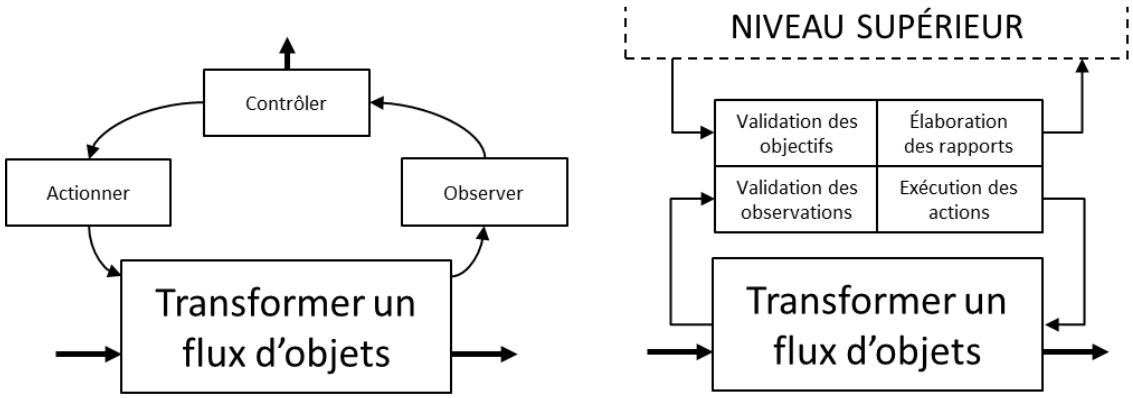


Figure 31 : Patrons de fonctions techniques

[22] *C'est-à-dire "Transformer un flux d'objets"*

Cependant, il serait possible d'adapter ces patrons en définissant une fonction de contrôle / commande par fonction technique. Par exemple, chez PSA Groupe, lors de la décomposition d'une fonction technique, les concepteurs ont choisi le patron suivant : une fonction de contrôle / commande pouvant piloter plusieurs fonctions transformant des flux d'objets.

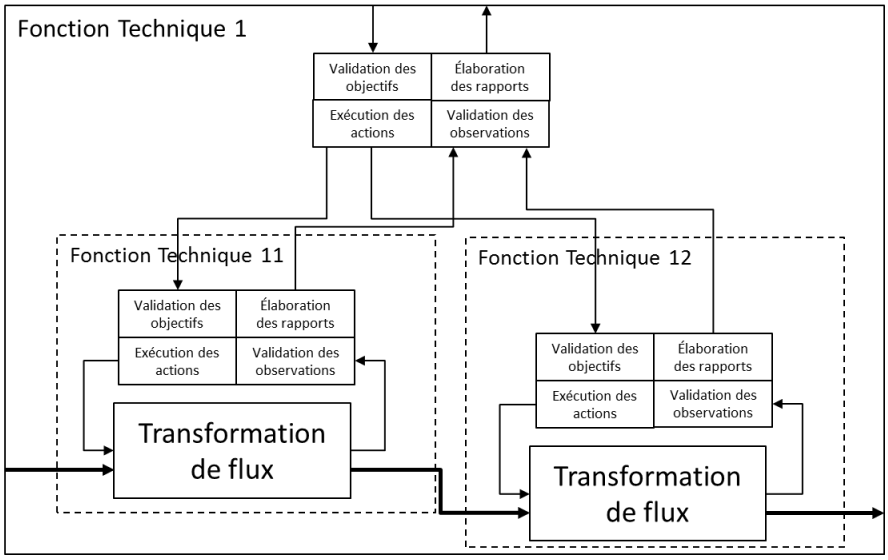


Figure 32 : Adaptation des travaux de Dobre pour notre cadre

D'autres patrons de conception fonctionnelle existent. En s'inspirant du management de projet (Project Management Institute, 2001), il est ainsi possible de décomposer une fonction technique en *Initialiser, Préparer, Réaliser, Clôturer*. Ce patron demeure spécifique. Il s'applique surtout à la conception de chaînes de production ou pour la conception de pièces ; il n'est donc pas dans le périmètre de nos travaux. Si on regarde les programmes d'enseignement technologique français, on constate que l'Éducation Nationale propose deux, voire trois, chaînes fonctionnelles (Ministère de l'Education Nationale, 2011). Il s'agit de la chaîne d'énergie, la chaîne d'information, et de la chaîne d'action. La chaîne d'information regroupe les fonctions *Acquérir*, *Traiter*, *Communiquer,* voire *Restituer*. La chaîne d'énergie regroupe les fonctions *Convertir*, *Alimenter*, *Distribuer* (ou *Moduler*), *Transmettre*, *Agir*. La chaîne d'action, proche ou confondue avec la chaîne d'énergie, regroupe les fonctions *Supporter*, *Contenir*, *Agir* et *Transmettre*. Une représentation classique des deux premières chaînes est représentée ci-dessous (Inspection Générale de l'Education Nationale, 2012). De telles chaînes sont pertinentes pour l'analyse de systèmes didactisés et mécaniques qui transforment de la matière. La chaîne d'information peut être perçue comme étant une description de la fonction de contrôle / commande. Elle peut cependant sembler trop simpliste dans un contexte industriel dans la mesure où les chaînes fonctionnelles ne peuvent pas forcément être décrites de cette manière.

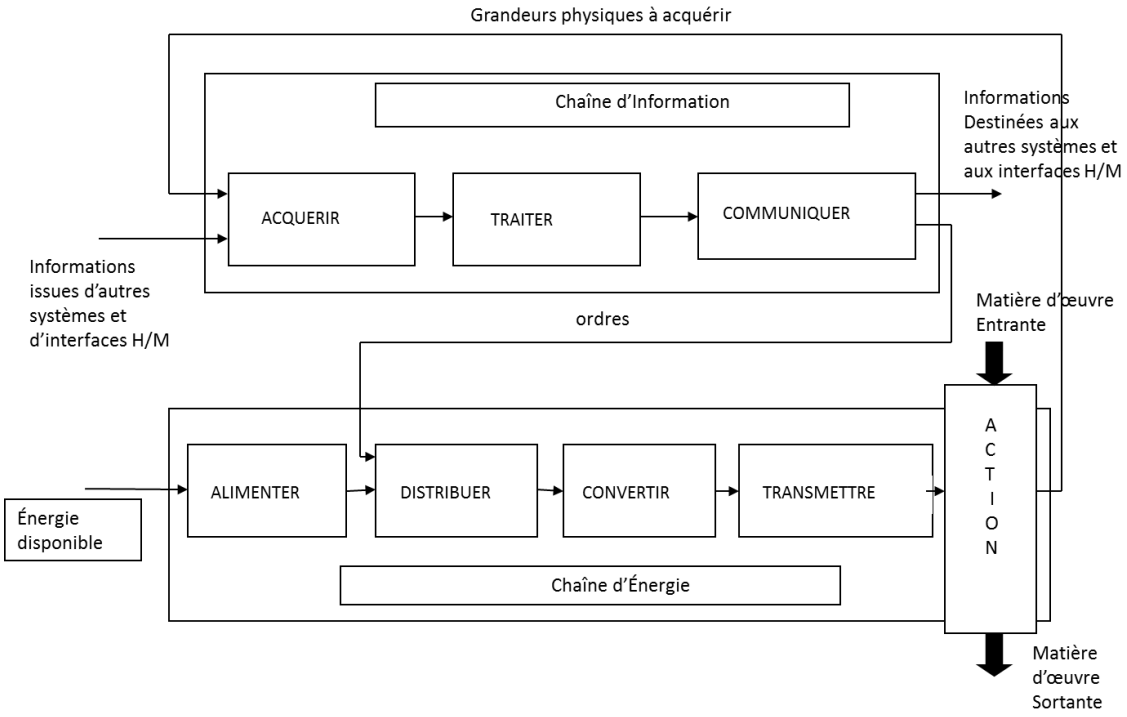


Figure 33 : Représentation des chaînes d'énergie et d'information

Nous concluons donc qu'une fonction technique peut être décomposée en un ensemble de fonctions transformant des flux de matière, d'énergie ou d'information échangés à travers des interfaces pilotées par des fonctions de contrôle / commande. Il existe différents moyens pour représenter ces patrons de conception fonctionnelle. Étant donné l'orientation de ce mémoire vers le langage SysML, nous nous concentrerons sur leur représentation à l'aide des diagrammes proposés par ce langage. Les différents patrons de conception pourront être ainsi représentés sous forme d'un IBD (*Internal Block Diagram*), voire d'un diagramme d'activité si l'on s'intéresse plus au comportement qu'à la structure.

5.3.2.2 Des patrons fonctionnels sous SysML

Toute fonction peut être modélisée dans SysML par un bloc éventuellement décomposable. Elle est alors liée à d'autres fonctions par des interfaces modélisées par des liens et des ports[23]. Le patron de conception d'une architecture fonctionnelle représenté à l'aide de SysML peut ainsi être représenté à l'aide de la figure ci-dessous. Nous distinguons donc les fonctions techniques (rectangles bleus) et les fonctions de contrôle / commande (rectangles verts) ; les

[23] *Le concept de Bloc dans SysML utilise les concepts de Lien et de Port pour modéliser une interface. Dans le cadre de la modélisation d'une architecture organique, cette terminologie est cohérente avec notre modèle conceptuel (Figure 17). Nous sommes conscients que cet usage n'est pas en cohérence avec notre choix dans le modèle conceptuel (Figure 15) lorsqu'il s'agit de modéliser une architecture fonctionnelle.*

flux MEI (Matière, Énergie, Information, traits gras) et les flux de contrôle / commande (traits fins). Afin de bien comprendre l'élaboration du patron fonctionnel, nous nous appuierons sur une légende qui sera complétée au fur et à mesure du chapitre.

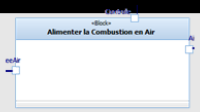
Concept	Modélisation
Fonction Technique	
Flux MEI	
Fonction de Contrôle/Commande	
Flux de Contrôle/Commande	

Figure 34 : Légende pour patron d'architecture fonctionnelle

Dans ce chapitre, nous suivrons l'exemple de la fonction *Générer l'Énergie Mécanique* qui est une sous-fonction de la fonction système *Assurer la dynamique longitudinale du véhicule*. Cette fonction produit un flux d'énergie mécanique.

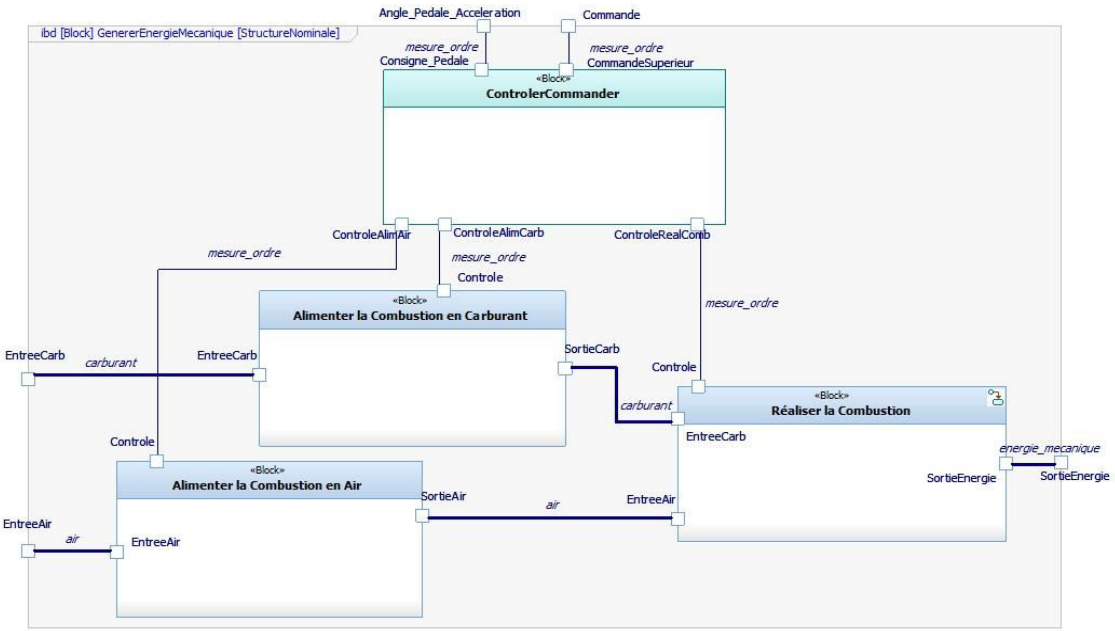


Figure 35 : Exemple d'application du patron de la vue structurelle

La Figure 35 est une illustration du patron fonctionnel proposé. La fonction *Générer l'énergie mécanique* se décompose en une fonction de contrôle / commande et trois fonctions techniques qui sont *Contrôler Commander*, *Alimenter la combustion en carburant*, *Alimenter la combustion en air* et *Réaliser la combustion*. La fonction de contrôle / commande échange avec les fonctions techniques des flux d'ordres et de compte-rendu. Dans le cas présenté, les

différentes fonctions techniques échangent entre elles des flux de matière (air, carburant) ou d'énergie. Nous exploiterons ensuite ce patron dans la section 5.3.4 afin de l'enrichir dysfonctionnellement, pour en extraire le *Functional Safety Concept*.

Les patrons fonctionnels présentés concernent la vue structurelle de l'architecture fonctionnelle. Qu'en est-il de sa vue comportementale ?

5.3.3 La vue comportementale

Comme l'expose le *SEBoK*, la vue comportementale permet de définir l'enchaînement des fonctions et leur activation dans un scénario. Deux étapes sont à parcourir et plusieurs types de modèles sont à utiliser pour ce faire :

1. utiliser un diagramme eFFBD, comme cela a été proposé dans le *SEBoK* ou dans (Seidner, 2009). Ce qui permet de définir l'enchaînement des fonctions et leur activation par des flux de contrôle. Des travaux ont montré qu'un eFFBD peut être représenté aisément par un diagramme d'activité en SysML (Bock, 2006) ;
2. déterminer les différents modes et états des fonctions, ainsi que leur activation, à l'aide d'un *statechart* en SysML reposant sur les *statemachines* d'Harel (Harel, 1987).

eFFBD et *statemachines* sont complémentaires. Ce premier facilite l'élaboration du diagramme d'état. Pour ce qui concerne celui-ci, comme l'énonce Bernard Rygaert dans (Rygaert, 2009), il est possible de distinguer 4 états pour une fonction :

1. l'état ON, qui est l'état dans lequel la fonction est activée et transforme des flux ;
2. l'état OFF, qui est l'état dans lequel la fonction ne transforme pas de flux, suite à un arrêt ;
3. l'état SUSPEND, qui est l'état dans lequel la fonction est activée mais ne transforme pas de flux, en attente ;
4. l'état FAIL, qui est l'état dans lequel la fonction ne produit pas de transformation de flux suite à une défaillance.

Ces quatre états peuvent être ensuite enrichis par des modes fonctionnels. Comme nous l'avons défini en section 2.3.4, un mode fonctionnel correspond à l'activation de la fonction avec un certain niveau de performance. Ainsi, l'état ON peut comporter plusieurs modes nominaux correspondant à plusieurs niveaux de performance, ainsi que plusieurs modes sûrs comme nous l'expliquerons en section 5.3.5. Selon le niveau de performance souhaité d'un flux, il sera possible de modifier le mode fonctionnel.

Maintenant que les vues nominales ont été explicitées, il est possible de déterminer comment des architectures fonctionnelles peuvent être enrichies dysfonctionnellement.

5.3.4 Proposition d'un patron structurel dysfonctionnel

Le chapitre précédent a présenté une méthode permettant d'aboutir à une définition d'exigences système dont les *Safety Goals*. Ils ont été identifiés à partir de phénomènes dangereux critiques (comportement anormal du système pouvant mener à une situation avec préjudices) et des scénarios d'évitement associés. L'objectif de la conception architecturale est d'éviter que ces phénomènes dangereux ne se produisent. Pour cela, l'AS doit pouvoir identifier, dans une architecture fonctionnelle candidate, les chemins de propagation critiques menant à ce phénomène. La méthode et le patron proposés dans cette section visent à répondre à ce besoin.

Le patron proposé doit permettre de visualiser, dans une architecture fonctionnelle enrichie dysfonctionnellement, les chemins de propagation critiques. Pour cela, il faut pouvoir modéliser les modes de défaillance et les causalités logiques à l'aide desquelles déterminer les combinaisons amenant à un phénomène dangereux critique. L'idée retenue est d'enrichir la vue structurelle avec des items relevant de l'univers dysfonctionnels. À cet effet, deux options sont possibles. La première est de l'enrichir par des éléments graphiques stéréotypés permettant de visualiser les chemins critiques. Cela se fera par l'intermédiaire d'un IBD si l'on se réfère à SysML. L'autre option est d'enrichir la vue structurelle en rattachant une équation logique à chaque sous-fonction. Cela peut se faire par l'intermédiaire de diagrammes paramétriques si l'on se réfère à SysML.

5.3.4.1 Enrichissement dysfonctionnel à l'aide d'un IBD et d'un stéréotype

Pour ce qui concerne la première option, nous avons intégré dans la vue structurelle des opérateurs logiques et des modes de défaillance, qui sont les éléments utiles pour définir un chemin critique dysfonctionnel. Afin de ne pas trop dérouter l'ingénieur, la symbolique des arbres de défaillance a été reprise, comme c'est le cas dans l'application Hiphops développée par Papadopoulos et McDermid (1999). En ce qui concerne les déviations de flux, nous avons repris les mots guides de l'HAZOP (cf.

Tableau 3). En réalisant un profil SysML combinant un IBD et des stéréotypes reprenant une représentation de type Hiphops, nous proposons ainsi un patron de conception représenté par la figure suivante dans le cas de la fonction *Générer l'Énergie Mécanique*. Ce patron vise à produire une équation logique d'un système (ou d'une fonction système) intégrant d'éventuels déviations de flux et modes de défaillances de ses fonctions composantes. Ce patron repose sur celui de l'architecture fonctionnelle proposé ci-dessus. La légende associée à ce premier patron est présentée ci-dessous.

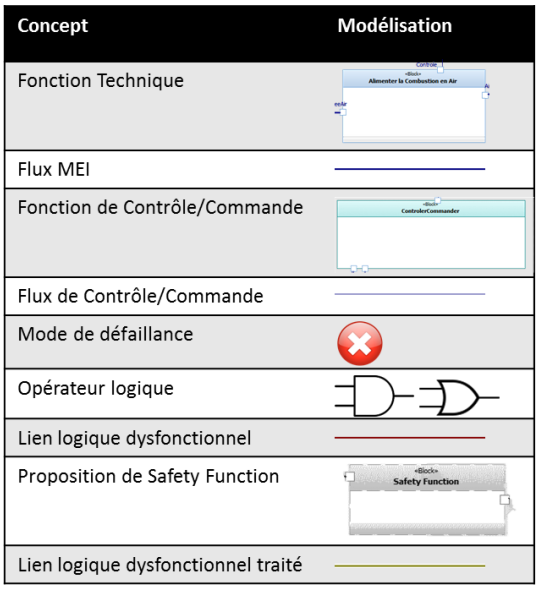


Figure 36: Légende pour la vue structurelle des architectures fonctionnelles

Comme nous pouvons le noter dans l'exemple concernant l'accélération intempestive, la fonction *Alimenter la combustion en air* ne comprend pas d'élément dysfonctionnel car ce n'est pas une source de la déviation du flux de sortie de la fonction étudiée.

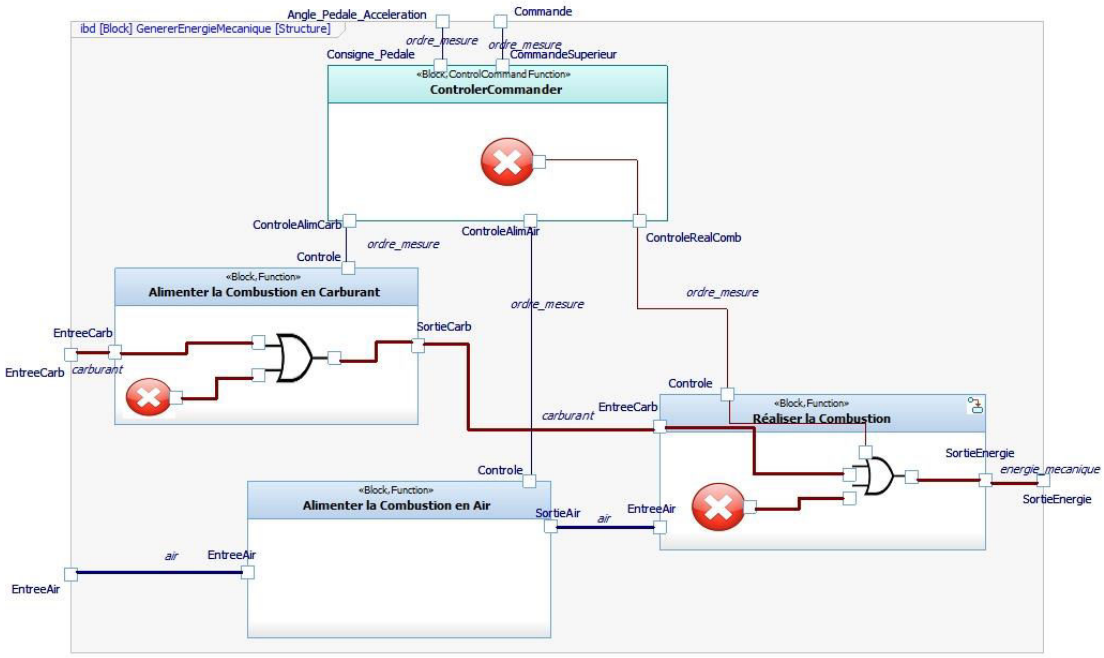


Figure 37 : Illustration pour la vue structurelle enrichie dysfonctionnellement.

L'équation logique d'une fonction pour une déviation de flux est une relation liant la déviation de flux en sortie, les déviations de flux en entrée, les modes de défaillance de la fonction, voire d'autres déviations de flux en sortie (dans le cas de sorties couplées). Une telle équation peut être visualisée à l'aide d'une table de vérité. En effet, si l'on reprend l'exemple pour la fonction *Réaliser la Combustion*, on étudie alors le flux d'énergie et la déviation intempestive, ce qui nous donne la table de vérité ci-dessous. Nous pouvons ainsi avoir les différents cas pour lesquels la sortie de la fonction *Réaliser la Combustion* est intempestive.

Tableau 27 : Représentation tabulaire d'une équation logique associée à une fonction technique

Sortie	Entrée Carburant	Entrée Air	Mode de défaillance intempestif
Nominale	Nominale	Nominale	Non
Intempestive	Nominale	Nominale	Oui
Nominale	Nominale	Intempestive	Non
Intempestive	Nominale	Intempestive	Oui
Intempestive	Intempestive	Nominale	Non
Intempestive	Intempestive	Nominale	Oui
Intempestive	Intempestive	Intempestive	Non
Intempestive	Intempestive	Intempestive	Oui

La table précédente permet de déterminer tous les cas, ce qui est trop extensif et donc trop complexe pour des modèles aidant l'activité de l'AS. Par exemple, le cas 1 ne nous intéresse pas d'un point de vue dysfonctionnel. De même, le cas 3 n'est pas exactement nominal, mais il n'est pas possible de le considérer comme étant intempestif et critique. De plus, *l'Entrée d'Air* n'a pas d'influence dans ce cas. Nous l'avons laissée car elle a été utile pour la construction de l'équation logique.

Une version améliorée de la précédente table consisterait en un ensemble de règles (avec priorités même si cela ne s'applique pas dans ce cas) exprimées sous forme propositionnelle :

Si j'ai une entrée carburant intempestive alors j'ai une sortie carburant intempestive ;

Si j'ai un mode de défaillance intempestif alors j'ai une sortie carburant intempestive, etc.

Ces propositions peuvent également se modéliser par des équations logiques avec des opérateurs classiques (OU, ET, OU exclusif, NON). Ce qui donne dans notre cas :

SortieCarburantIntempestive

= EntréeCarburantIntempestive

OU ModeDeDéfaillanceIntempestif.

Une seconde étape est de produire une AFSS. Celle-ci doit donc comporter les chemins de propagation et les *Safety Functions* susceptibles d'éliminer certains des chemins critiques dysfonctionnels. Comme le montre la figure ci-dessous, il faut donc pouvoir distinguer les chemins traités à l'aide d'un principe de sûreté et ceux qui demeurent critiques. Cela permet d'aider la conception sûre et de pouvoir justifier certaines *Safety Functions* lors de l'activité

Évaluer la satisfaction des exigences techniques et des Functional Safety Requirements en *Functional Architectural Design* ou lors des phases aval de conception.

Nous avons illustré ce patron de conception pour la vue structurelle avec les *Safety Functions*. Comme nous le montrerons ensuite, cette architecture n'est pas réaliste pour la fonction *Générer l'énergie mécanique* pour des raisons de perte excessive de performance.

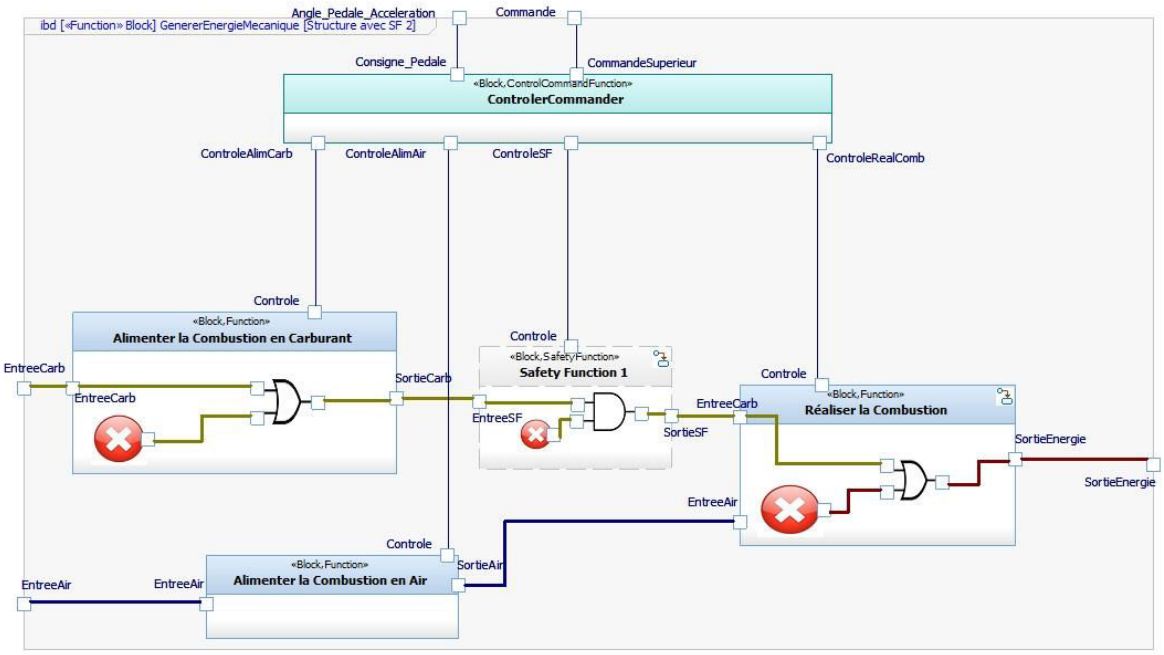


Figure 38 : Illustration du patron pour la vue structurelle avec les Safety Functions

Nous proposons maintenant un stéréotype SysML de la vue structurelle d'une AFSS. Ce stéréotype est représenté ci-dessous. Ce stéréotype permet d'avoir une première identification des éléments utiles d'un point de vue dysfonctionnel.

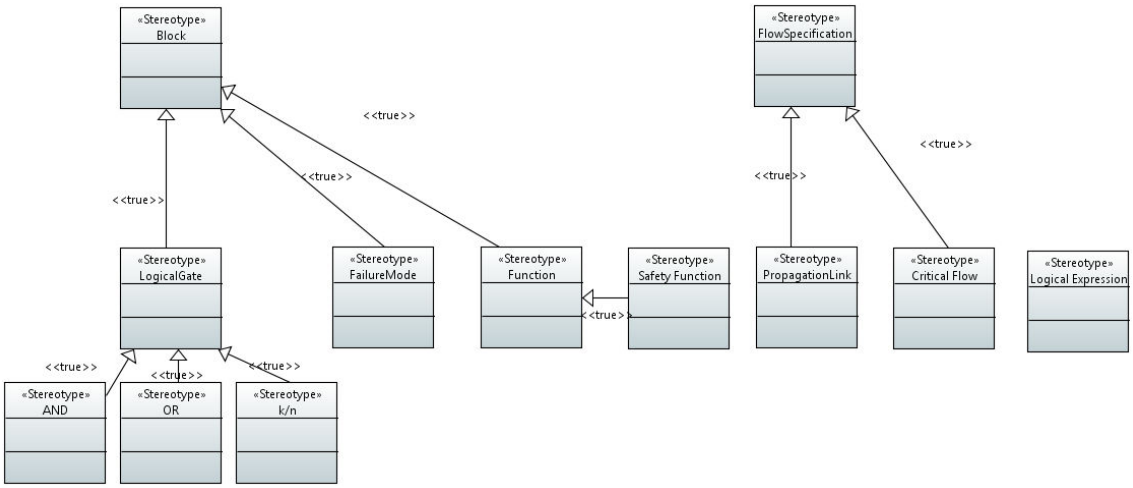


Figure 39 : Stéréotype SysML de la vue structurelle dysfonctionnelle

Le *patron* et le stéréotype proposés seront vérifiés dans la dernière section du présent chapitre. Une fois leur définition faite, il convient de les mettre en œuvre dans le cadre du processus d'ISSdF exposé précédemment.

5.3.4.2 Enrichissement dysfonctionnel à l'aide d'un diagramme paramétrique

Une autre option pour représenter l'équation logique d'une fonction est de recourir aux diagrammes paramétriques de SysML (OMG, 2012). Nous proposons donc de définir l'équation logique par ce diagramme qui sera lié au bloc de la fonction. Nous proposons deux manières non exclusives d'utiliser les diagrammes paramétriques. La première consiste à indiquer l'expression logique dans la même *constraint property*. La seconde prend des *constraint property* sur étagère. Ceux-ci sont les opérateurs logiques classiques et des modes de défaillance paramétrables.

Si l'on reprend l'exemple de la fonction *Générer l'énergie mécanique*, une des sous-fonctions dont on va souhaiter l'équation logique est la fonction *Réaliser la combustion*. Les ports de l'IBD sont liés à ceux du diagramme paramétrique. Les déviations de flux sont déterminées en fonction d'une liste prédéfinie. Les deux options pour la fonction *Générer l'énergie mécanique* sont illustrées ci-dessous.

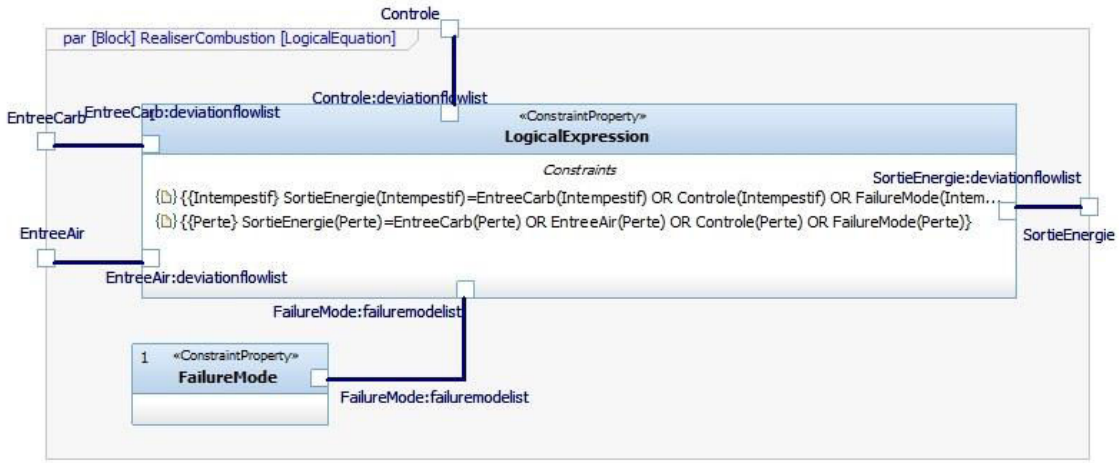


Figure 40: Diagramme paramétrique pour définir une équation logique

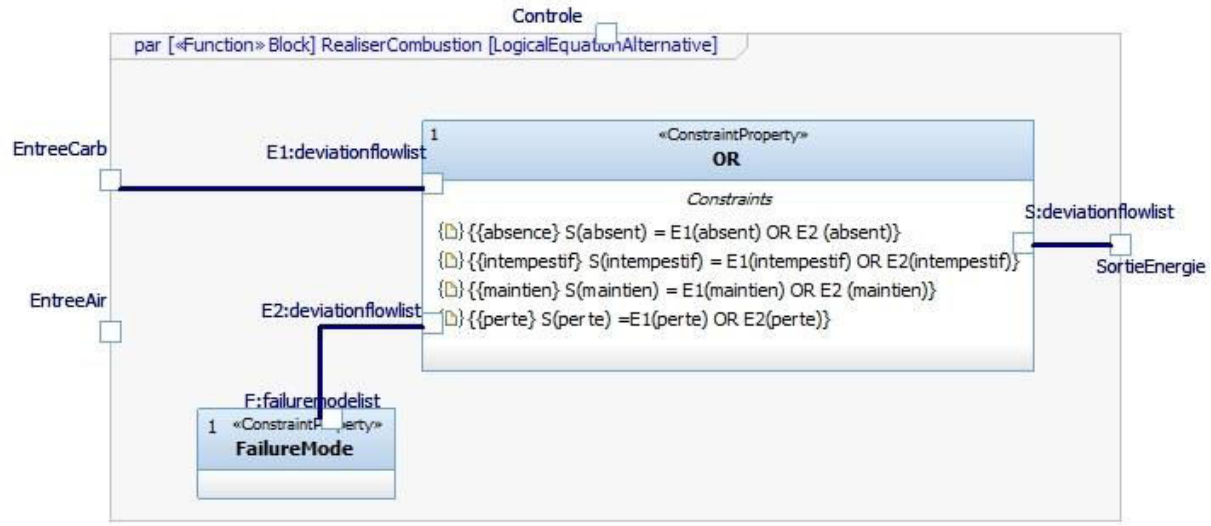


Figure 41 : Option de diagramme paramétrique pour l'équation logique

Dans cette partie, nous avons donc fait deux propositions pour décrire la vue structurelle d'une architecture fonctionnelle de système sûr. La première repose sur l'enrichissement des IBD. Elle est très graphique et se comprend aisément. Toutefois, sa simulation sera complexe à réaliser. La seconde basée sur la définition de diagrammes paramétriques devrait être plus simple pour simuler la recherche de causalité.

5.3.5 Proposition d'un patron comportemental dysfonctionnel

Nous venons de définir le patron pour la vue structurelle d'une AFSS, il est également nécessaire de le définir pour la vue comportementale.

Comme cela a déjà été explicité, l'objectif de cette thèse n'est pas de définir un nouveau langage de sûreté — comme l'est AltaRica (Prosvirnova & Rauzy, 2012) — ou un méta-langage dédié à l'ISSdF, mais de proposer un processus d'ISSdF aboutissant à une AFSS. Ainsi, le patron que nous allons proposer a été élaboré à l'aide d'un *statechart* réalisé avec le langage SysML. Tous les concepts présents dans ce patron sont toutefois indépendants du choix du langage ou de l'outil de modélisation. En conséquence, il est aussi possible de le modéliser sous AltaRica, devenu un langage incontournable en matière de SdF.

En reprenant les modes et états énoncés par Bernard Rygaert, nous pouvons les détailler de cette façon. L'état ON peut être raffiné en plusieurs modes de fonctionnement. La fonction peut comporter plusieurs modes nominaux dans lesquels elle réalise totalement les performances souhaitées et plusieurs modes dégradés dans lesquelles elle les réalise partiellement. Ces modes peuvent être qualifiés de *Safe* si à un niveau global, la fonction permet d'assurer une performance du système dans lequel les passagers sont saufs.

De plus, nous proposons d'intégrer les états des sorties de la fonction et ceux des entrées[24]. D'un point de vue fonctionnel, cette intégration n'est quasiment jamais faite. Lorsque l'on développe un modèle à états, on s'occupe uniquement des modes et états de la fonction dont les entrées et sorties sont connus. Cependant, cette modélisation peut permettre d'explorer plus amplement son comportement dysfonctionnel. Notons qu'elle est utilisée dans certains modèles de sûreté comme ceux basés sur AltaRica.

L'une des principales différences entre un *statechart* SysML et AltaRica concerne le traitement des entrées et des sorties. Celles-ci sont nommées dans AltaRica *variables de flux*. Le choix effectué dans les exemples suivants sera de les considérer comme des variables booléennes. Une variable d'entrée est "OK" ou "KO". Nous avons fait le choix d'avoir une liste d'états possibles pour les entrées et sorties qui dépend du cas traité. Celle-ci sera en lien avec les déviations de flux au niveau structurel.

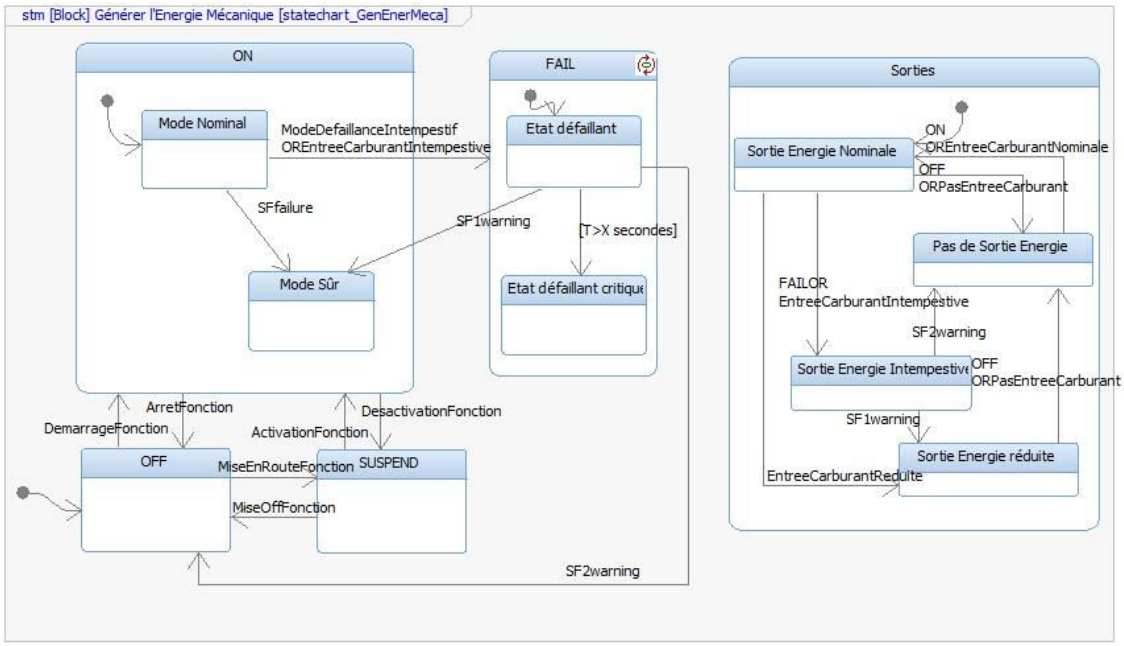


Figure 42 : Illustration du patron de la vue comportementale

Cette figure est une illustration du patron comportemental dysfonctionnel. Il concerne le comportement d'une fonction. Nous avons repris les quatre états *ON*, *OFF*, *SUSPEND* et *FAIL*. Dans l'état *ON*, différents modes de fonctionnement sont instanciés c'est-à-dire les modes nominaux et les modes sûrs. Dans l'état *FAIL*, les différents états défaillants de la fonction sont instanciés. Le fait d'avoir une défaillance provoque un changement d'un mode nominal vers un état défaillant. L'activation de la *Safety Function* (*SF1Warning*) permet de revenir dans un mode sûr, voire dans l'état *OFF* (*SF2Warning*). Si celle-ci n'est pas activée, la fonction peut

[24] *En effet, la sortie d'une fonction étant l'entrée d'une autre fonction lors d'une décomposition fonctionnelle.*

passer dans un état plus critique. Ces différentes transitions vont avoir un impact sur l'état du flux de sortie qui pourra être nominal, dégradé ou intempestif. Par ailleurs, à l'aide de la partie droite de la figure, il est possible de relier les différents modèles comportementaux. Cela permettra de simuler et de vérifier la spécification de la *Safety Function*, c'est-à-dire de déterminer si la sortie en énergie est nominale ou intempestive.

5.4 PROPOSITION D'UN PROCESSUS D'AFSS

Une fois les différents patrons de la conception d'une AFSS définis, il convient de se pencher sur le processus d'élaboration d'une telle architecture et sur la façon d'extraire le *Functional Safety Concept*[25], livrable principal de l'ISO 26262:2011 pour l'architecture fonctionnelle.

5.4.1 Structure générale du processus

Nous proposons le processus suivant, décrit par la figure ci-dessous. Les entrées et sorties ont été indiquées en gras. Les livrables peuvent être des vues extraites qui vont autant intéresser l'AS que l'expert en SdF. L'enchaînement d'activités proposé est cohérent avec le macro-processus d'ISSdF du troisième.

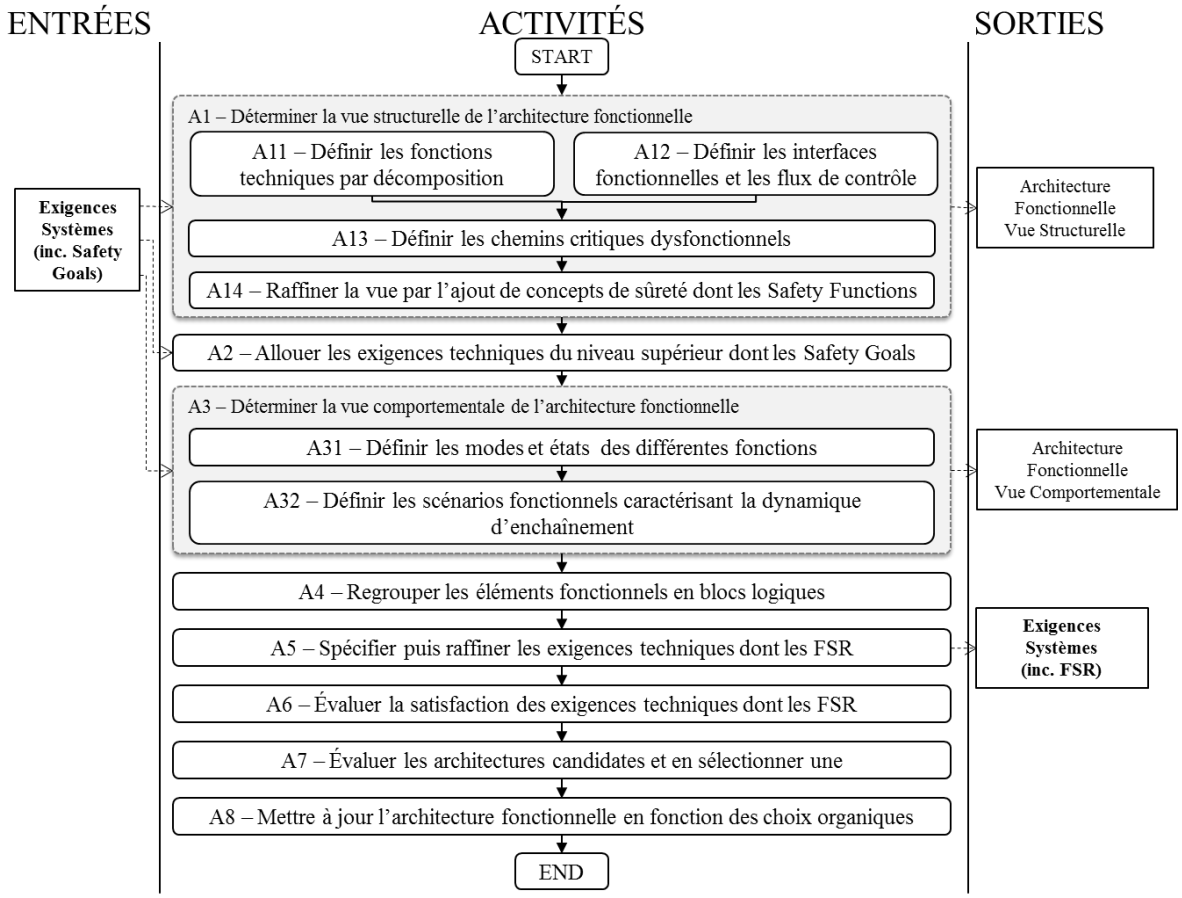


Figure 43 : Élaboration d'une architecture fonctionnelle de système sûr

[25] *L'ISO 26262:2011 utilise le même terme pour l'activité et le livrable d'où la précision.*

L'élaboration d'une architecture fonctionnelle de système sûr, représenté en Figure 43, comprend plusieurs activités. Pour des raisons de lisibilité, les bouclages liés aux enrichissements successifs entre les différentes vues n'ont pas été représentés.

La première activité consiste à déterminer la vue structurelle de l'architecture fonctionnelle (Activité A1). Il faut d'abord déterminer les différentes fonctions, les interfaces, et les flux entre les fonctions. Il est ensuite possible de déterminer les chemins critiques dysfonctionnels en se basant sur la causalité des phénomènes dangereux - choix que nous justifierons dans la section suivante -. Ceci permet : (1) d'identifier les fonctions ayant un impact dans la production de phénomènes dangereux et de ne pas retenir les chemins de propagation ayant une conséquence non critique, et (2) de proposer des alternatives de concepts de sûreté dont les *Safety Functions*.

La deuxième activité consiste en l'allocation des exigences techniques (Activité A2). Elle permet ensuite de définir la vue comportementale de l'architecture fonctionnelle de système sûr (Activité A3) en élaborant les modes et états des différentes fonctions (diagramme *statechart* en SysML) ou éventuellement les scénarios fonctionnels (diagramme d'activité en SysML) caractérisant la dynamique d'enchaînement des fonctions. Ceux-ci pourront être nominaux ou dysfonctionnels. Après un éventuel regroupement des éléments fonctionnels en blocs logiques (Activité A4), il est possible de spécifier les exigences techniques dérivées, dont les FSR (Activité A5) et d'évaluer leur satisfaction, et ce, pour différentes architectures candidates afin d'en sélectionner une (Activité A6). Enfin, une fois l'architecture organique élaborée, il deviendra possible de mettre à jour l'architecture fonctionnelle (Activité A7) en tant compte des choix arrêtés lors de l'activité *Physical Architecture Design*, comme cela a été indiqué dans le chapitre 3.

Les modèles souhaités et le processus étant spécifiés, il est maintenant possible de définir comment aboutir à ces modèles en raffinant les activités énoncées dans le processus. Pour rendre plus synthétique la présentation, nous n'allons détailler que certaines activités définies précédemment, à savoir celles nécessitant à titre prioritaire une méthodologie.

5.4.2 A13. Définition des chemins dysfonctionnels critiques

Lors du premier chapitre, nous avons distingué deux méthodes de SdF susceptibles de s'appliquer à la conception fonctionnelle, à savoir l'AMDEC et l'HAZOP. Alors que ces méthodes reposent sur une propagation des entrées vers les sorties (Leger, 1999), nous avons fait le choix de déterminer la causalité des évènements dangereux critiques. La première option implique une analyse exhaustive partant des défaillances élémentaires et aboutissant à aux chemins de propagation. Ceux-ci ne seraient pas forcément critiques. Tous les prendre en compte alourdirait donc l'analyse des risques assurée par l'architecte système. La recherche de causalité a pour mérite de ne retenir que les chemins critiques dysfonctionnels pour les

phénomènes dangereux choisis. On a alors affaire à une méthode très discriminante. Néanmoins, nous avons distingué deux types de méthodes pour aider l'activité A13.

5.4.2.1 Méthode proposée en cas d'architecture fonctionnelle reconduite

Dans le cas d'une architecture fonctionnelle en grande partie reconduite, le *flowchart* suivant montre les entrées et les étapes d'une méthode d'ISSdF permettant de définir les chemins de propagation. C'est ce qui se passe en conception routinière.

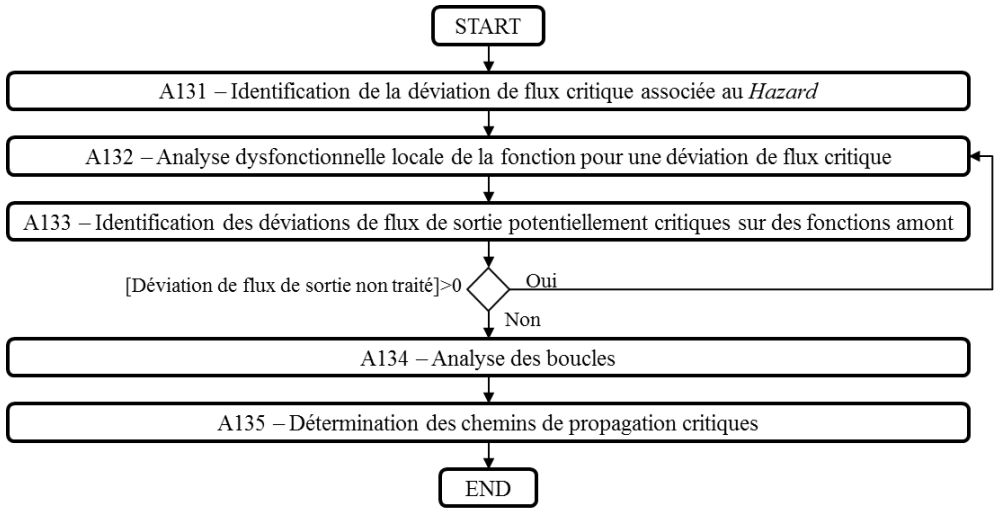


Figure 44 : Vue d'ensemble de la méthode

En entrée de cette méthode, nous avons un *Safety Goal* auquel est associé un phénomène dangereux (*hazard*) dont les conséquences peuvent être critiques. L'ayant modélisé dans le chapitre précédent comme la déviation d'un flux d'une fonction système, il est possible d'identifier la fonction de l'architecture qui produit cette déviation de flux (Activité A131). De la sorte, on peut mener une analyse dysfonctionnelle locale de la fonction concernée (Activité A132). Cette analyse consiste à déterminer la relation entre la déviation de la sortie, les modes de défaillance de la fonction, et les déviations des entrées. Les entrées pouvant causer le phénomène dangereux peuvent être ensuite identifiées. Une fois l'analyse locale réalisée, il convient de refaire la même analyse sur les fonctions amont, en reliant les déviations des entrées de la fonction étudiée aux déviations de sorties d'autres fonctions de l'architecture (Activité A133). Nous ne mettons pas en place un ordre entre les différents types de déviations car cela n'est pas pertinent. Selon les cas, en effet, la perte peut être soit plus, soit moins critique que l'absence ou le maintien. Les activités de la méthode (Activités A132 & A133) doivent être réalisées tant qu'il reste des déviations de sortie non étudiées.

Dans un deuxième temps, il faut s'occuper des boucles qui peuvent apparaître lors de la détermination des chemins critiques dysfonctionnels. Ces boucles peuvent provenir de boucles du contrôle / commande ou de retour de flux, par exemple un retour d'excès de carburant pompé

vers le réservoir. Il est possible de reprendre ce qu'a fait Martin Walker dans (Papadopoulos, et al., 2011) en mettant des points d'arrêt lorsque la même déviation de sortie a déjà été parcourue. Cela risque cependant d'entrainer des problèmes de stabilité lors de l'exploration des chemins critiques dysfonctionnels. Le passage à la vue comportementale pourra être une solution pour traiter cela. Enfin, il devient possible de déterminer les chemins de propagation critiques et de visualiser les fonctions critiques, c'est-à-dire celles qui sont sur les chemins de propagation critiques et qui nécessitent de ce fait une exigence de sûreté supplémentaire ou l'ajout d'une *Safety Function* afin de rendre le système sûr.

5.4.2.2 Méthode proposée en cas de nouvelle conception d'une fonction.

Lorsque l'on réalise une nouvelle conception d'une fonction, les données d'entrée se présentent sous forme d'une ''boîte noire'' avec des interfaces et des déviations de flux non souhaitées. Cela est spécifié par un certain nombre d'exigences techniques dont des *Safety Goals* pour les déviations de flux non souhaitées pouvant causer un phénomène dangereux. La méthode proposée consiste alors à repartir du ou des flux de sortie et de déterminer quelle sous-fonction va produire ce ou ces flux. Cette sous-fonction va également produire la (ou les) déviation(s) de flux non souhaitée(s). Il va donc être alors possible de mener à bien l'analyse locale de cette sous-fonction. Cette analyse permet de mieux spécifier les entrées de cette sous-fonction car il sera possible de déterminer le flux souhaité et les limites acceptables.

5.4.2.3 Instances de la méthode proposée

La méthode proposée est générique. Il convient de l'instancier et de l'adapter à différentes phases de conception :

1. en définition du concept, il s'agit d'explorer des architectures alternatives décrites de façon sommaire, mais aussi d'identifier et de détailler les concepts critiques. Il est alors possible d'analyser les modes de défaillance du concept critique. La méthode proposée permet surtout de vérifier les chemins les plus critiques et à plus fort impact, ce qui est caractérisé par un ASIL fort (supérieur à B) pour le *Safety Goal* ;
2. en conception préliminaire, l'objectif est de vérifier la faisabilité de l'ensemble de l'architecture choisie. Il faut donc prendre en compte tous les *Safety Goals* dont l'ASIL est strictement supérieur à A comme cela a été évoqué au chapitre précédent. De plus, il est possible d'avoir une connaissance plus détaillée de l'architecture fonctionnelle, si bien qu'il est alors possible d'identifier et de raffiner les chemins de propagation ;

3. en conception détaillée, l'objectif est de faire une analyse de SdF elle aussi détaillée. Il faut recourir à la méthode proposée pour vérifier que des chemins de propagation amenant à un phénomène dangereux n'ont pas été oubliés. Pour ce faire, l'expert en SdF peut réaliser des AMDEC au niveau de l'architecture organique et vérifier la bonne application de la méthode au niveau fonctionnel lors des phases précédentes.

Dans le secteur automobile, l'utilisation de la méthode proposée diffère aussi selon les contextes de conception. Dans le cas de la conception routinière, fondée sur des retours d'expérience conséquents, l'architecture fonctionnelle est quasi-reconduite à l'identique, si bien qu'il suffit de mettre à jour les chemins de propagation en fonction des modifications constatées dans le projet de conception courant. Dans le cas d'une conception externalisée auprès d'un équipementier, une fonction peut être appréhendée comme une "*Function On The Shelf",* par analogie avec les COTS. Il faut alors, pour le constructeur en charge de l'intégration, récupérer auprès de l'équipementier les modèles fonctionnels et dysfonctionnels de la fonction sous-traitée. Enfin, une fonction d'une architecture donnée peut être d'un niveau d'abstraction plus ou moins élevé. Ainsi, les fonctions *Assurer le déplacement longitudinal*, *Réaliser la combustion* et *Injecter le carburant*, relèvent de différents niveaux d'abstraction ; le détail des informations dysfonctionnelles variant de façon inverse à ce niveau. Moins la fonction est abstraite, plus il devient possible de la saisir à partir d'une solution technique connue. Il est alors possible d'identifier précisément ses modes de défaillances et d'en déterminer les causes potentielles.

5.4.3 A14. Proposer des alternatives de concepts de sûreté comprenant les *Safety Functions*

Une fois les chemins critiques dysfonctionnels identifiés, l'activité suivante consiste à proposer des alternatives de concepts de sûreté parmi lesquelles on trouve les *Safety Functions*. Nous détaillerons donc dans un premier temps le principe de détermination des *Safety Functions,* puis nous proposerons plusieurs patrons possibles de *Safety Functions*.

5.4.3.1 Principe de détermination des Safety Functions

Une fois l'architecture fonctionnelle enrichie dysfonctionnellement définie, c'est-à-dire une fois les chemins critiques dysfonctionnels identifiés, il est possible de créer des alternatives de définition de *Safety Functions* ou de spécifier d'autres concepts de sûreté. Ceux-ci peuvent avoir un impact sur les performances des différentes fonctions. Il est donc nécessaire de réaliser une évaluation de ces différents concepts en prenant en compte des critères relevant de la SdF, de la performance de l'architecture, etc. Pour chaque sous-fonction source d'un chemin de propagation identifiée précédemment, c'est-à-dire pour chaque sous-fonction comprenant un

mode de défaillance pouvant causer la déviation de flux de sortie non souhaitée, il faut envisager de mettre en place une *Safety Function* en aval. Selon le mode de défaillance étudié et la structure de l'architecture fonctionnelle, plusieurs *Safety Functions* sont possibles. Si cela n'est pas possible pour des raisons de performance globale de l'architecture, de coût, etc., il est possible de placer une *Safety Function* à un niveau plus global de l'architecture fonctionnelle, entre la source de la défaillance et la sortie du système provoquant un phénomène dangereux.

5.4.3.2 Propositions de patrons de Safety Functions

Le but de cette section n'est pas de proposer une liste exhaustive de patrons de *Safety Functions*, ce qui ne semble pas envisageable. En s'inspirant des travaux d'Aurélie Léger dans (Léger, 2009), il est toutefois possible de proposer deux patrons en déterminant les fonctions supportant les barrières proposées.

Tout d'abord, le patron de type *Composant de sécurité* est une fonction dédiée à un flux, et dont le but est de limiter, réduire ou supprimer un flux donné en cas de défaillance, comme le montre la figure ci-dessous. Un des inconvénients possibles provient de la diminution de la performance nominale souhaitée si la *Safety Function* est en contradiction avec une exigence de performance.

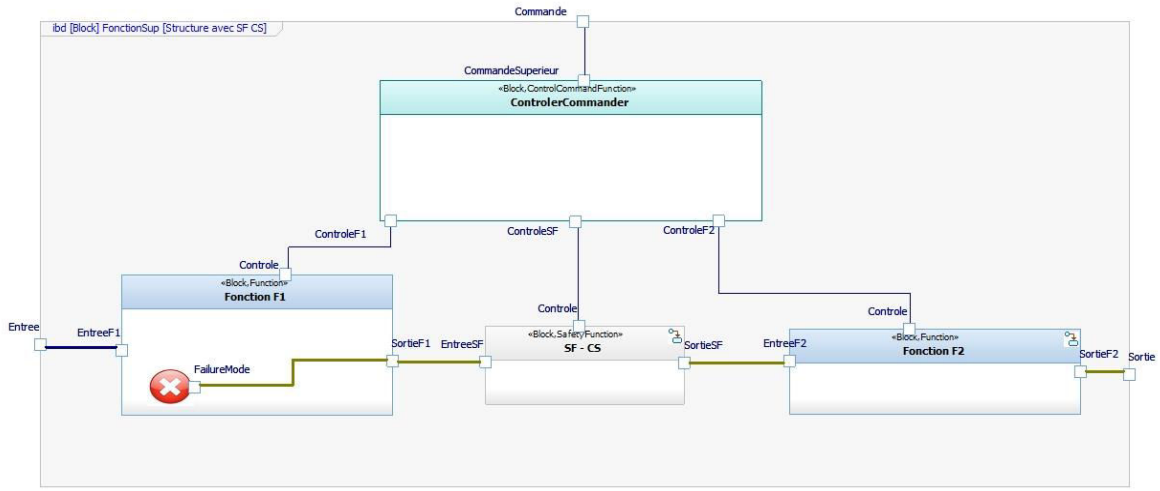


Figure 45 : Exemple de Safety Function de type Composant de Sécurité

Selon le mode de fonctionnement de la fonction globale, la *Safety Function* pourra être paramétrée pour que la limitation de flux diffère selon les cas. Enfin, par le lien noué avec le contrôle / commande, cette *Safety Function* pourra également remonter l'information que la fonction 1 est dans un état FAIL. Et ce, afin de prendre les mesures adéquates, voire de prévenir le conducteur, pour passer dans un mode sûr au niveau système.

Le patron de type *Systèmes instrumentés de sécurité* est proche de la boucle de la Cybernétique rappelée en Figure 31. Elle permet d'observer la sortie et/ou la comparer avec la

consigne afin de détecter la défaillance ou la déviation d'un flux amenant à l'évènement redouté. S'il y a défaillance, il y a une remontée de l'information à un plus haut niveau fonctionnel, voire au conducteur, et ce, de sorte à agir pour modifier le comportement du système et le faire passer dans un mode sûr.

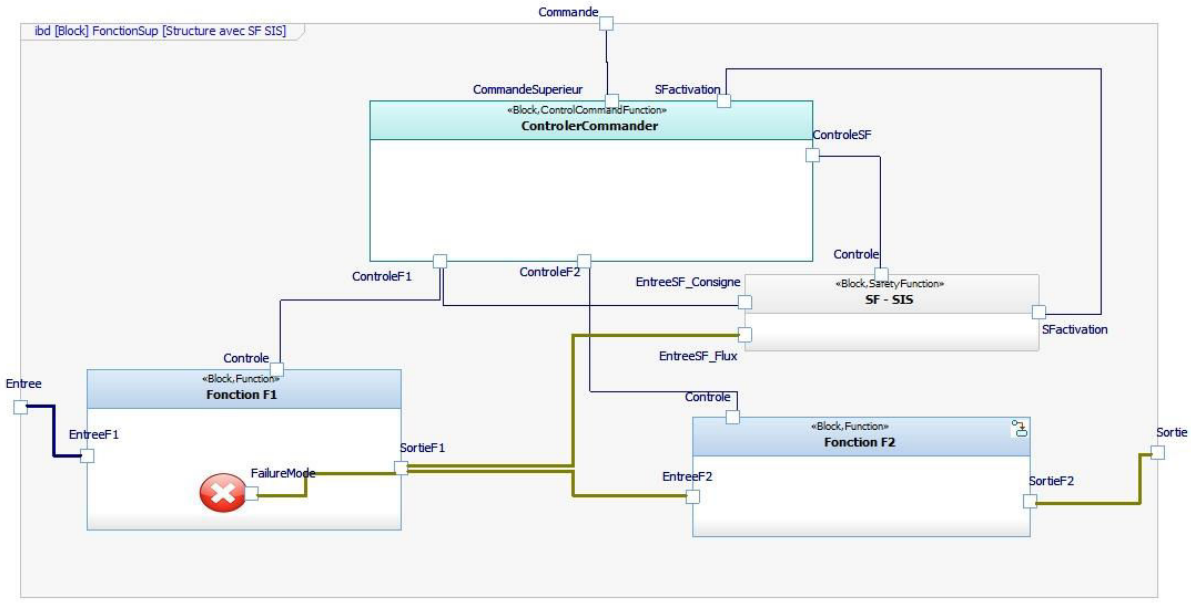


Figure 46 : Exemple de Safety Function Type Système Instrumenté de Sécurité

Quand l'écart entre la consigne et la propriété du flux de sortie est supérieure à une valeur de sécurité, la *Safety Function* remonte cette information au niveau du contrôle / commande afin que la fonction 1 passe dans un mode sûr et que cela ne se propage pas jusqu'au niveau système.

Les pistes de *Safety Functions* sont confirmées par Abraham Cherfi dans (Cherfi, Leeman, Meurville, & Rauzy, 2014). Les auteurs proposent deux types de *Safety Functions* : ceux-ci sont fondés sur la transition vers un mode de protection (syn. mode sûr) suite à la détection d'une erreur ou ceux fondés sur l'inhibition du système quand celui-ci est dans un état potentiellement dangereux. Dans le premier cas, la *Safety Function* est proche de la proposition type *Système Instrumenté de Sécurité*. En effet, lorsqu'il y aura un écart trop important entre la consigne et la sortie de la fonction surveillée, la *Safety Function* remontera l'information afin qu'un mode de protection soit activé. Dans le second cas, la *Safety Function* est proche de la proposition type *Composant de Sécurité*. Lorsque la *Safety Function* détecte un comportement anormal de la fonction surveillée, alors elle inhibera le flux de sortie dévié pour éviter une propagation de cette déviation. Une autre proposition de *Safety Function* proche de la *Safety Function* de type *Système Instrumenté de Sécurité* est une fonction de comparaison suivie d'une fonction permettant de commuter suite à la redondance de la fonction étudiée (Schätz & Voss, 2014).

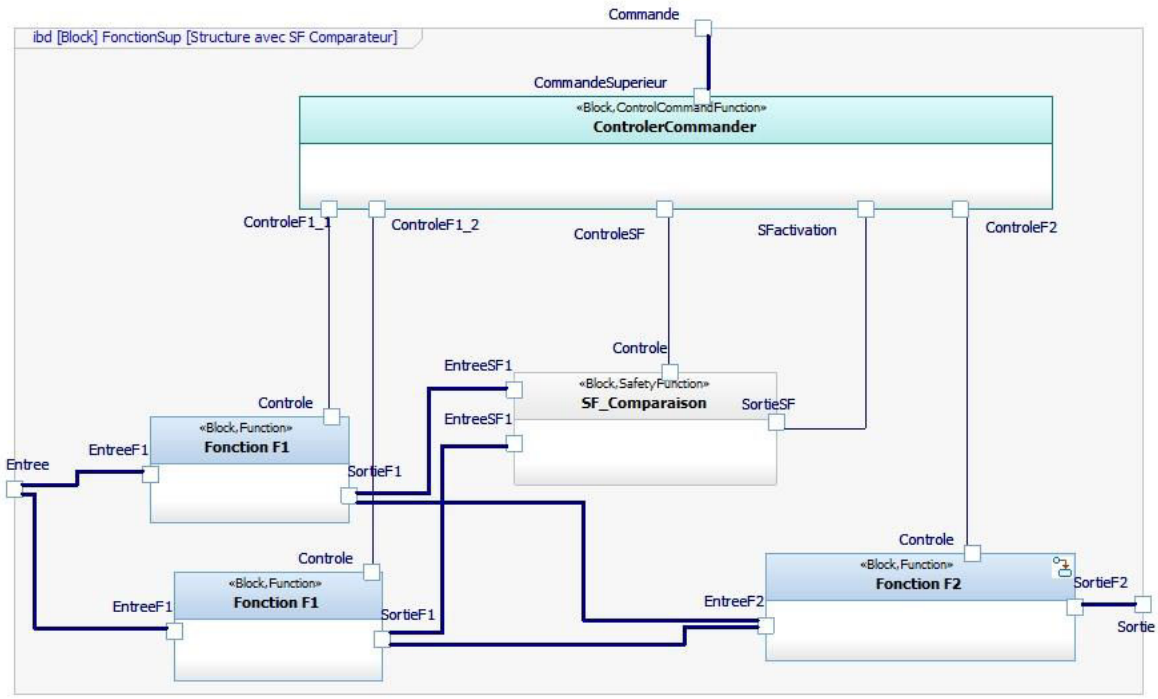


Figure 47 : Exemple de Safety Function type Comparaison et Commutation

Comme l'illustre la figure précédente, la première *Safety Function* compare chaque flux de sortie des fonctions redondantes avec la consigne. En cas de défaillance d'une des deux fonctions, l'alerte est transmise au contrôle / commande, ce qui va permettre de modifier les modes et états des deux fonctions F1 afin de réaliser la commutation. La fonction défaillante F1 passera dans un mode sûr alors que celle encore fonctionnelle passera dans un mode de fonctionnement de surrégime afin d'assurer la mission.

5.4.4 A31. Définir les modes et états des différentes fonctions

Le *flowchart* de la méthode proposée permet d'avoir un panorama des activités d'une méthodologie d'ISSdF permettant de définir la vue comportementale de l'AFSS. Afin de garder un *flowchart* lisible, les itérations et les rebouclages n'ont pas été indiqués. En effet, nous sommes conscients que suite à des enrichissements successifs, il est nécessaire de re-simuler la vue comportementale. Nous prenons comme patron celui défini en partie 5.3.5.

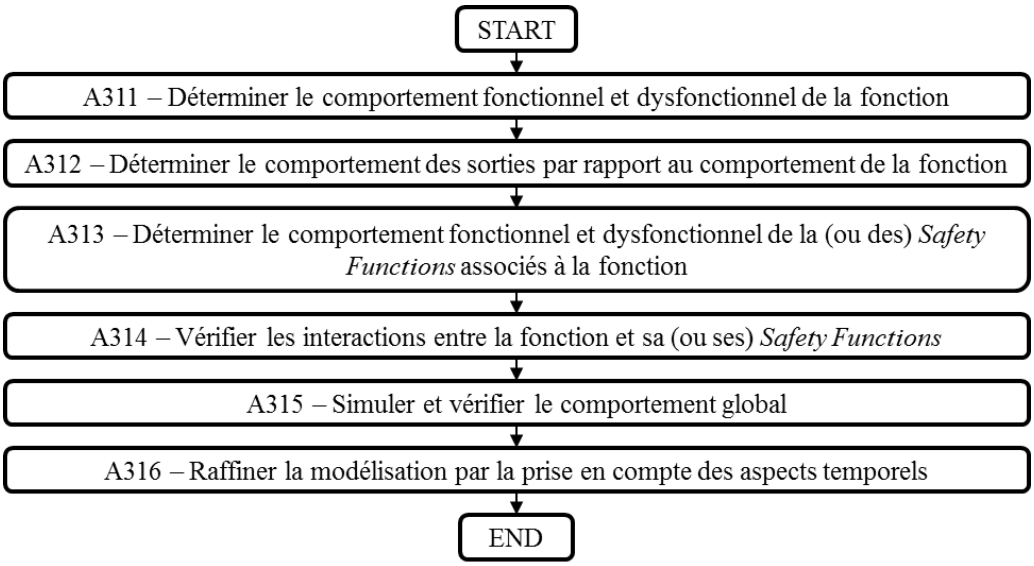


Figure 48 : Processus d'enrichissement des modes et états fonctionnels

Tout d'abord, il convient de déterminer le comportement fonctionnel et dysfonctionnel de la fonction (Activité A311) étudiée en respectant le patron énoncé dans la section 5.3.5. Cela consiste tout d'abord à définir les différents états, les modes fonctionnels qu'ils soient nominaux, dysfonctionnels ou sûrs. Ensuite, il est possible de déterminer les transitions entre ces différents modes et états. Ces transitions peuvent dépendre d'évènements internes ou d'évènements externes liées aux entrées de la fonction. Informer ces éléments conduit à définir la partie gauche du patron présenté Figure 42.

Il devient ensuite possible d'identifier le comportement de sortie de la fonction et de définir les transitions entre les différents états de la sortie présentés dans la vue structurelle (Activité A312). Celles-ci seront basées sur des évènements externes provenant des entrées de la fonction ou des changements de ses modes et états. Les informations obtenues permettent ensuite de définir le comportement souhaité de la (ou des) *Safety Function*(s) déterminée(s) dans la vue structurelle (Activité A313). Cette *Safety Function* étant une fonction, son comportement pourra être basé sur le patron défini Figure 42. Ce qui implique donc qu'il est possible de modéliser le comportement dysfonctionnel d'une telle fonction lors d'une seconde passe de modélisation, lorsqu'il s'agira d'apprécier le comportement du système en cas de défaillance de la *Safety Function*. Celle-ci peut se mettre active en cas de défaillance de sa part, ne pas signaler sa défaillance, ne pas signaler la défaillance de la fonction, etc. Une surveillance du bon fonctionnement de la *Safety Function* est souvent nécessaire.

Il est ensuite nécessaire de vérifier que les transitions entre la fonction et sa *Safety Function* associée sont bien prises en compte (Activité A314). Il est enfin possible de simuler et vérifier le comportement global de l'architecture fonctionnelle (Activité A315). D'un point de vue dysfonctionnel, il convient de vérifier si les *Safety Functions* permettent bien de ne pas aboutir à un état non souhaité d'une sortie des fonctions étudiées. Si nécessaire, il sera possible d'ajuster les aspects temporels et les transitions pour avoir le comportement souhaité. Ces ajustements

pourront être effectués suite à une simulation du modèle ou une analyse spécifique de sûreté conduite en phase de conception détaillée.

5.5 ILLUSTRATIONS

Afin de montrer la pertinence de la méthode et des modèles permettant l'enrichissement dysfonctionnel d'une architecture fonctionnelle, nous proposons un exemple relatif à la fonction *Générer l'énergie mécanique*. Afin de montrer la récursivité et l'applicabilité de la méthode et des outils à différents niveaux d'une architecture fonctionnelle, celui-ci sera traité sur deux niveaux fonctionnels, à savoir *Générer l'énergie mécanique* et *Alimenter la combustion en carburant*. Deux phénomènes dangereux critiques, un couple intempestif et une perte de couple en sortie, seront pris en compte.

5.5.1 Illustration des vues structurelles

Le langage graphique utilisé pour décrire une architecture fonctionnelle comprend les primitives suivantes, tirées de l'IBD de SysML et de Hiphops. L'IBD suivant reprend l'architecture fonctionnelle associée à la fonction *Générer l'énergie mécanique.* Cette architecture est enrichie dysfonctionnellement en déterminant la causalité d'une déviation d'un flux de sortie, à savoir une sortie d'énergie mécanique intempestive.

La figure de base est la Figure 35. Une analyse dysfonctionnelle a été schématisée par la Figure 37. Deux options d'architecture contenant deux *Safety Functions* différentes sont proposées. Sur la Figure 49, la solution introduisant la *Safety Function* 1 n'est pas envisageable car celle-ci a un impact sur la performance de la fonction : elle limite les performances à un niveau en dessous ce qui est souhaité. Une autre option est présentée Figure 50. Celle-ci décrit une *Safety Function* envoyant un message au contrôle / commande lorsque le flux de sortie de la fonction *Réaliser la combustion* est intempestif. Le contrôle / commande agit sur la fonction *Alimenter la combustion en carburant* par le flux de contrôle. Cette solution est jugée acceptable.

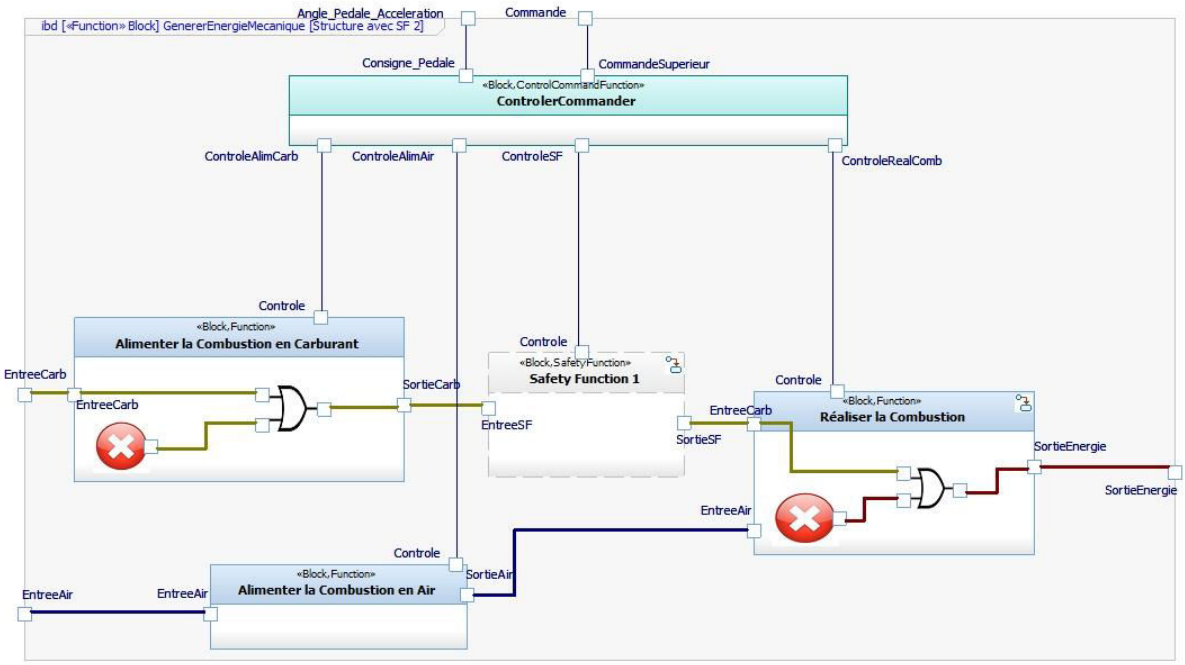


Figure 49 : Énergie Mécanique Intempestive – Vue structurelle – Première option

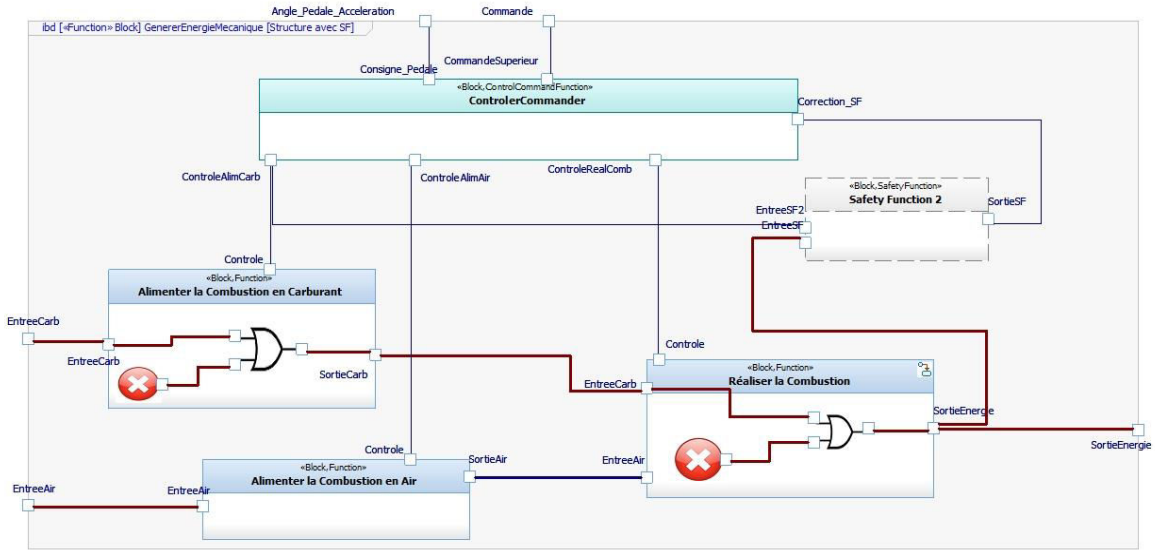


Figure 50 : Énergie Mécanique Intempestive – Vue structurelle – Seconde option

Les précédents diagrammes aident à déterminer les fonctions impactées et les chemins de propagation. De plus, des *Safety Functions* peuvent être placées sur certains flux afin de ne pas permettre la déviation de flux critique. Afin de montrer une autre vue dysfonctionnelle de *Générer l'énergie mécanique*, le choix s'est porté sur une déviation de flux critique qu'est l'absence de couple en sortie de cette fonction. Ce second exemple permet de vérifier que les fonctions impactées peuvent être différentes selon la déviation de flux prise en compte par le concepteur. De même, des *Safety Functions* peuvent être propres à une déviation de flux ou mutualisées, ce qui traduit, chez l'AS, la recherche d'une certaine économie ou parcimonie fonctionnelle.

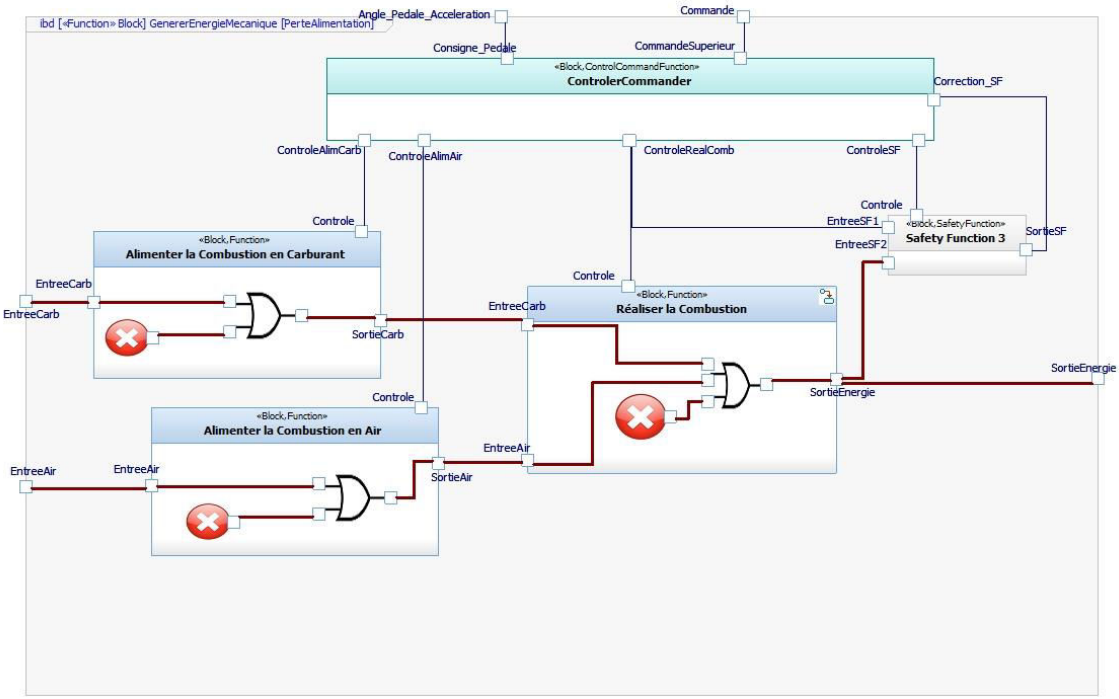


Figure 51 : Perte Énergie Mécanique – Vue structurelle

Comme pour la Figure 50, nous avons repris la Figure 37 pour laquelle nous avons défini les chemins critiques dysfonctionnels pour la déviation de flux *Perte d'énergie mécanique*. Dans ce cas, la perte d'air en entrée de la fonction *Réaliser la combustion* est également critique. Plusieurs *Safety Functions* sont possibles. Par exemple, deux sont au même emplacement que celles définies pour l'accélération intempestive. Une troisième pourrait également être sur le flux placé entre les fonctions *Alimenter la combustion en air* et *Réaliser la combustion*. Elle aurait alors les mêmes défauts que celle définie sur la Figure 49. Une autre option est représentée ci-dessus, qui permet de réaliser une surveillance globale de la chaîne fonctionnelle et d'alerter le contrôle / commande.

Afin de vérifier la récursivité de la méthode et des modèles proposés, nous allons maintenant passer de la fonction *Générer l'énergie mécanique* à la sous-fonction *Alimenter la combustion en carburant*. Le résultat auquel nous parvenons est présenté dans la Figure 52. Pour des raisons de lisibilité, nous n'avons pas fait apparaître les *Safety Functions* dans cette vue. Il est possible de déterminer une fois de plus les chemins de propagation et les modes de défaillance. Enfin, grâce à ce diagramme, il est possible de vérifier l'équation dysfonctionnelle de la strate supérieure ou de la modifier si nécessaire. Nous proposerons dans la section suivante une approche basée sur une modélisation en Prolog pour procéder à cette vérification d'une façon plus rigoureuse.

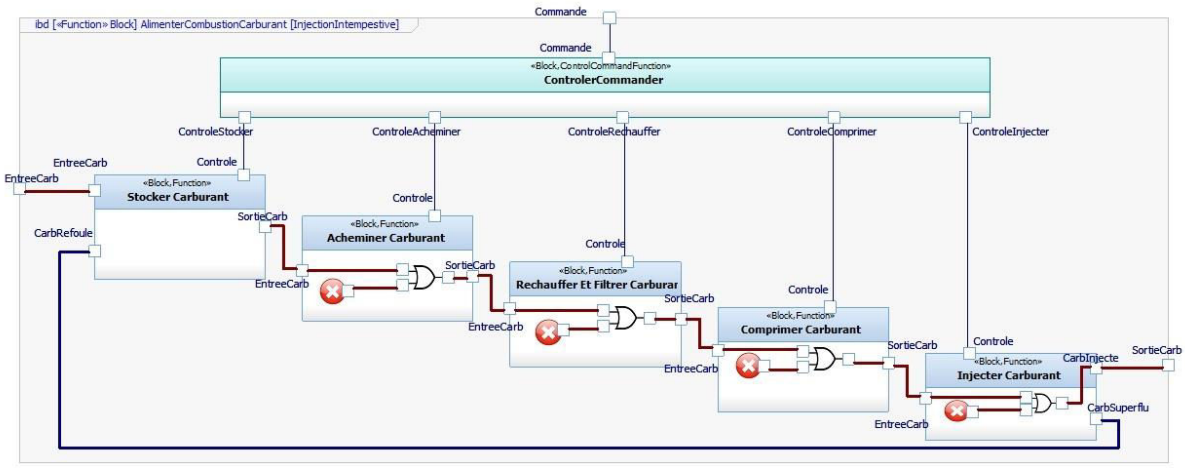


Figure 52 : Injection en Carburant Intempestive – Vue structurelle

L'injection en carburant intempestive a plusieurs causes possibles, dues à différents modes de défaillance, voire à une mauvaise entrée en carburant. Les différents IBD présentés correspondent à une approche structurelle et statique de l'AFSS. En traitant chaque phénomène dangereux, il est possible de déterminer des options en matière de *Safety Functions*. Une étape de synthèse est présente à la fin de l'activité *Proposer des alternatives de concepts de sûreté comprenant les Safety Functions*. Elle permet de réduire le nombre de *Safety Functions* qui pourraient être mutualisées selon les niveaux de décomposition de l'architecture ou les types de phénomène dangereux.

5.5.2 Vues comportementales

Nous allons montrer maintenant comment traiter les aspects comportementaux de la fonction *Générer l'énergie mécanique*. Pour ce faire, nous avons tout d'abord construit le comportement fonctionnel et dysfonctionnel de cette fonction, comme le montre la *state machine* SysML présentée Figure 42. Afin d'avoir un diagramme lisible, nous avons seulement pris en compte le cas de la déviation intempestive du flux de sortie.

La fonction étudiée est dans un mode nominal. Si un mode de défaillance se manifeste ou s'il y a une déviation de flux en amont, elle passe dans un état défaillant. Sans aucune action, au bout d'un certain temps (en accord avec les scénarios opérationnels du niveau supérieur), elle passe dans un état défaillant critique. Si des *Safety Functions* ont été mises en place (*SF1warning* et *SF2warning*), elles vont permettre à la fonction étudiée de passer dans un mode de fonctionnement sûr (couple réduit) ou dans l'état OFF.

Les modes et états de la fonction mentionnés vont influer sur l'état de sa sortie. Pour la sortie de fonction *Générer l'énergie mécanique*, nous avons pris en compte quatre états que sont *Sortie Énergie Nominale*, *Sortie Énergie Intempestive*, *Sortie Énergie réduite* et *Pas de Sortie*

Énergie. Les transitions entre eux dépendent des états de la fonction et des états de l'entrée de la fonction.

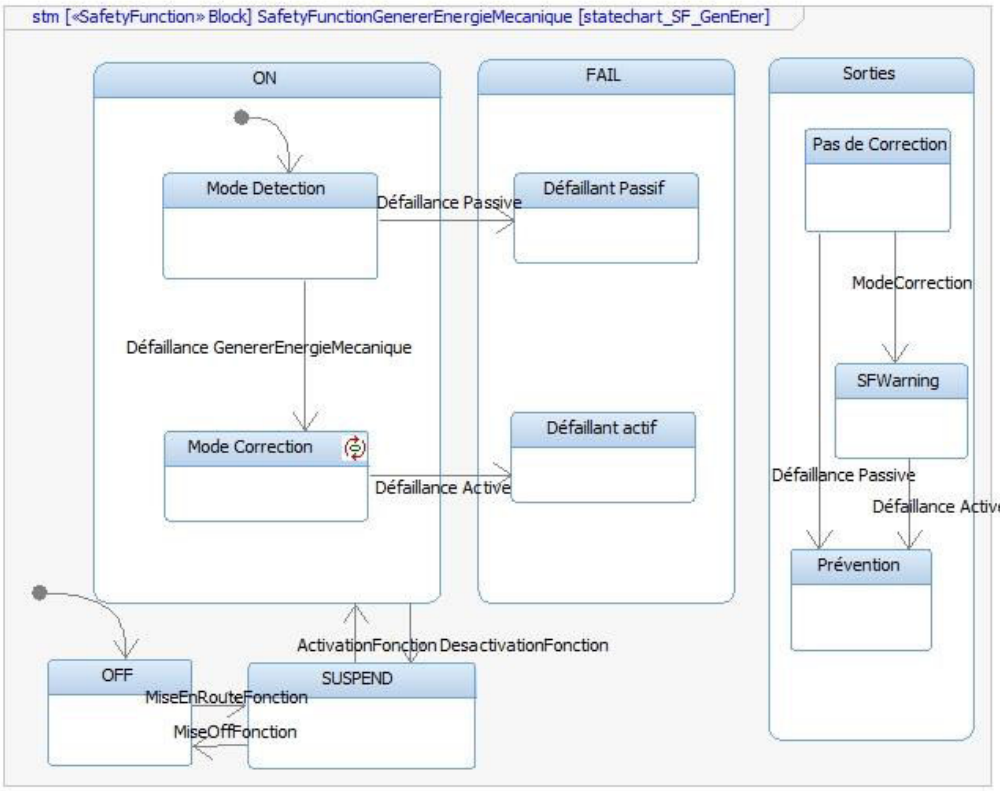


Figure 53 : Modes et états d'une Safety Function

Nous avons repris le patron de conception fonctionnelle que nous avons appliqué à la *Safety Function* de la fonction *Générer l'énergie mécanique*. Cette *Safety Function* a donc deux modes de fonctionnement que sont les *Mode Détection* et *Mode Correction*. Dans le premier, elle surveille s'il y a une défaillance de la fonction étudiée. Dans le second, elle agit suite à une défaillance de ladite fonction. Nous avons pris en compte les possibilités que les *Safety Functions* peuvent connaître une défaillance passive (défaillance lorsque la *Safety Function* n'est pas active) et une défaillance active (défaillance lorsque la *Safety Function* est active).

5.6 VÉRIFICATION DES DIFFÉRENTES VUES

Ce chapitre a permis de montrer plusieurs méthodes enrichissant les vues structurelle et comportementale de l'AFSS. Un apport supplémentaire est le développement d'un programme informatique permettant de vérifier que les choix de conception permettent bien de satisfaire les exigences dont celles de sûreté. Pour ce faire, une application sous Prolog a été développée.

5.6.1 Vérification de la vue structurelle par une approche

Nous avons souhaité réaliser un prototype afin de vérifier la faisabilité de l'identification de chemins critiques dysfonctionnels et le choix de mise en place de *Safety Functions*, permettant de supprimer, ou de limiter leurs effets. Pour ce faire, nous formalisons par une équation logique la valeur d'un flux de sortie pouvant conduire à un phénomène dangereux, à partir des valeurs des entrées et des modes de défaillances des fonctions de l'architecture fonctionnelle. Partant d'une valeur de flux de sortie critique, nous mettons en œuvre un principe de rétropropagation. Il permet de retrouver les valeurs possibles de variables d'entrée et les modes de défaillances menant à ce flux critique. Ce principe permet donc de construire le chemin critique dysfonctionnel.

Le problème et le principe évoqués sont bien traités par la programmation logique[26]. Pour contribuer à une vérification de la vue structurelle, nous avons collaboré avec Christophe Perrard, enseignant-chercheur à l'Université de Franche-Comté, qui nous a aidé à traiter cette vérification à l'aide de Prolog. Nous avons pu tester plusieurs fonctions que nous détaillons ensuite et qui sont illustrées par l'exemple décrit ci-dessous.

5.6.2 Définition et modélisation d'une fonction sous Prolog

Chaque fonction est considérée comme une transformation de flux d'entrée en flux de sortie. Une transformation est composée d'un identifiant, d'une description qui peut être le nom de cette fonction, d'une liste de flux d'entrée, d'une liste de flux de sortie et d'un comportement. Elle est programmée de la façon suivante :

"transformation: id,description, liste_flux_entrée, liste_flux_sortie,comportement"

À titre d'exemple

transformation(1,"générer l'énergie mécanique",1.2.nil,3.nil,1) ->[27] *;*

[26] *Principe de cette programmation expliquée en annexe de ce mémoire.*
[27] *Ce symbole permet de déclarer un fait en fin de ligne ou une règle lorsqu'il est suivi d'une deuxième partie en Prolog II.*

indique que la transformation 1 décrite comme étant la fonction *Générer l'énergie mécanique* transforme les flux 1 et 2 en un flux 3 par le comportement 1.

Les flux sont composés d'un identifiant, d'un type (matière, énergie ou information), d'un nom, d'une valeur possible et de ses extrema mini et maxi. Ces champs ont été ajoutés au cas où on souhaite ultérieurement travailler sur des variables quantitatives (et non plus discrètes) pour quantifier les flux et les borner par des valeurs mini et maxi.

Les comportements peuvent être décrits de différentes façons. Dans l'exemple, nous avons défini deux types de comportements. Le premier est une description par extension à l'aide d'une table de vérité. Le second est une description en compréhension par un ensemble de règles. La table de vérité pour la fonction *Alimenter la combustion en carburant* est définie ci-dessous. Nous avons émis l'hypothèse que deux modes de défaillance ne peuvent être présents simultanément. C'est pour cela que nous renverrons un « *non considéré* » quand ce sera le cas

Tableau 28: Table de vérité pour la fonction Alimenter la combustion en carburant

Entrée Carburant	Déf. Absent	Déf. Intempestif	Sortie Carburant
Nominal	Oui	Oui	Non considéré
Nominal	Oui	Non	Absent
Nominal	Non	Oui	Intempestif
Nominal	Non	Non	Nominal
Absent	Oui	Oui	Non considéré
Absent	Oui	Non	Absent
Absent	Non	Oui	Absent
Absent	Non	Non	Absent
Intempestif	Oui	Oui	Non considéré
Intempestif	Oui	Non	Absent
Intempestif	Non	Oui	Intempestif
Intempestif	Non	Non	Intempestif

Selon l'état du flux en entrée et selon la présence d'un des deux modes de défaillance, il est possible de déterminer l'état du flux de sortie, donc les combinaisons pour lesquelles on souhaite (ou non) un certain état pour le flux de sortie. Ce qui est décrit par cet ensemble de règles :

S'il y a le mode de défaillance absent, alors le flux de sortie sera absent.
Sinon Si le flux d'entrée est absent, alors le flux de sortie est absent.
Sinon, si le flux d'entrée en carburant intempestif ET/OU si le mode de défaillance intempestif est présent, alors le flux de sortie est intempestif.

Les flux sont décrits de la façon suivante :

"flux: id,type,nom,valeur si connue,mini,maxi"[28]
"type : matiere,information,energie"
flux(1,matiere,"carburant",_,mini,maxi))->;

[28] *Une ligne entre guillemets permet d'indiquer un commentaire sous Prolog II.*

Par exemple, le flux 1 (qui était un flux d'entrée pour la fonction 1) est un flux de type matière dont la valeur n'est pas connue, ce qui est symbolisée par _. Par défaut, on lui indique une valeur « mini » et « maxi ».

Prenons l'exemple de la Figure 37. Pour une première modélisation, nous nous sommes concentrés sur les fonctions techniques en n'introduisant pas de modélisation pour les fonctions de contrôle / commande.

version(1,1.nil)->;
version(2,<1,2.3.4>.nil)->;
version(3,<1,2.3.5.6>.nil)->;

transformation(1,"générer l'énergie mécanique",1.2.nil,3.nil,1)->;
transformation(2,"alimenter la combustion en carburant",1.nil,4.nil,2)->;
transformation(3,"alimenter la combustion en air",2.nil,5.nil,3)->;
transformation(4,"réaliser la combustion",4.5.nil,3.nil,4)->;

flux(1,matiere,"carburant",_,mini,maxi)->;
flux(2,matiere,"air",_,mini,maxi)->;
flux(3,matiere,"énergie mécanique",_,mini,maxi)->;
flux(4,matiere,"carburant injecté",_,mini,maxi)->;
flux(5,matiere,"air",_,mini,maxi)->;

La décomposition fonctionnelle se fait par l'intermédiaire de la « version[29] ». Dans cet exemple, nous représentons par la version 1 le fait que la fonction « générer l'énergie mécanique » est la fonction de haut niveau. La version 2 nous indique qu'elle est décomposée en 3 sous-fonctions qui sont : « alimenter la combustion en carburant », « alimenter la combustion en air » ainsi que « réaliser la combustion ». Les différents comportements sont pour l'instant non définis. Ils pourront être enrichis ensuite par une table de vérité, une équation logique ou un autre type de description logique du comportement de la fonction.

La structure Prolog utilisée permet également de déterminer des alternatives. Par exemple, les versions 2 et 3 sont deux décompositions fonctionnelles. La première décompose la fonction 1 en fonctions 2, 3 et 4, alors que la seconde décompose la fonction 1 en fonctions 2, 3, 5 et 6.

[29] *Fait de notre modélisation permettant d'indiquer le contenu d'un niveau fonctionnel pour notre approche.*

5.6.3 Prototypage réalisé sous Prolog

L'objectif de ce prototypage est de déterminer les chemins critiques dysfonctionnels, puis d'aider le concepteur à choisir des emplacements de *Safety Functions*.

Pour cela, avant de déceler les chemins critiques dysfonctionnels, il convient les comportements en programmation logique. Comme nous l'avons expliqué précédemment, nous avons commencé par une expression en extension à l'aide d'une table de vérité, puis par une expression en compréhension à l'aide de règles. Il a été possible de comparer les deux approches pour plusieurs cas tests. Ensuite, nous avons vérifié l'adéquation des comportements des sous-fonctions avec le comportement de la fonction composée. Cela a en effet permis de vérifier les lois de composition entre le comportement logique d'une fonction et les comportements logiques des sous-fonctions. Le fait qu'il y ait des différences n'est pas forcément une erreur, mais il faut vérifier les cas divergents pour déterminer si les comportements des sous-fonctions ont bien été spécifiés. Après, un troisième programme a concerné la recherche des chemins critiques dysfonctionnels conduisant à un phénomène dangereux. En programmation logique, les modes de défaillances des sous-fonctions et/ou leurs entrées sur le chemin critique sont identifiés. En sortie, on obtient l'ensemble des chemins critiques dysfonctionnels pour un phénomène dangereux choisi. Enfin, après analyse des chemins critiques dysfonctionnels et ajout d'une *Safety Function*, on peut refaire la vérification de l'adéquation (deuxième programme) et la recherche des chemins critiques afin de déterminer quels sont ceux qui ont été supprimés. Cela permet de vérifier l'efficacité d'une *Safety Function*. On analyse les résultats en comparant (1) avant, puis après l'introduction de la *Safety Function* et (2) plusieurs options comprenant des *Safety Functions* différentes. En procédant de la sorte, il est possible de déterminer si une solution n'affecte pas les performances de la fonction et si son effet est bien pris en compte.

Le détail de ces trois différents programmes est présenté en Annexe.

5.7 INTÉRÊT DE LA MÉTHODE & DES MODÈLES PROPOSÉS

Dans cette section, nous allons revenir sur plusieurs aspects de ce chapitre. Après avoir explicité l'originalité de l'approche proposée, nous reviendrons sur les liens de cette méthode avec ces congénères de SdF, puis sur les liens entre les différentes vues de l'architecture fonctionnelle.

5.7.1 Originalité de l'approche

Nous avons proposé une méthodologie permettant de construire la vue structurelle d'une architecture de système sûr avec un enrichissement dysfonctionnel. Celle-ci a ensuite été raffinée à l'aide d'une méthode permettant de visualiser les chemins de propagation et des pistes pour définir les *Safety Functions*. Cependant, on peut remarquer qu'il n'est pas possible d'en spécifier complètement le comportement attendu avec la seule vue structurelle. Il faut pour cela s'appuyer sur la vue comportementale.

À l'aide de l'approche comportementale, nous avons pu spécifier l'aspect dynamique des *Safety Functions* avec leurs interactions avec le reste de l'architecture fonctionnelle. À l'aide du patron proposé, il est possible de simuler le comportement nominal et le comportement dysfonctionnel. Il est ainsi possible d'évaluer les *Safety Functions* pour réaliser une architecture fonctionnelle de système sûre satisfaisante. À travers le processus et les différentes activités explicitées, nous avons montré comment définir et spécifier ces *Safety Functions* à l'aide des deux vues, et où elles seront mises en œuvre dans l'architecture fonctionnelle.

5.7.2 Lien avec les activités spécifiques de SdF

Comme nous l'avons explicité dans l'état de l'art de ce chapitre, de nombreuses méthodes existent pour réaliser des analyses spécifiques de sûreté. Celles-ci se reposent sur des langages comme AltaRica ou les BDMP. Si l'on reprend le langage AltaRica, comme cela est exposé dans (Prosvirnova, et al., 2013), il est possible de déterminer qu'il s'agit d'une modélisation basée sur les Systèmes à Transitions Gardées (GTS). Ces systèmes à transitions gardées sont un formalisme de type états/transitions où les états sont implicites, donnés par des assignations de variables[30]. Cela est donc proche de notre modélisation pour la vue comportementale. Par contre, nous n'avons pas pris en compte l'aspect stochastique avec les lois de défaillances contrairement aux GTS. Ce manque n'est pas rédhibitoire pour notre travail, qui est focalisé sur l'activité de l'architecte système et non de l'expert en SdF.

De plus, si l'on se réfère au projet OpenPSA (Hibti, Friedlhuber, & Rauzy, 2012), notre modélisation reprend tous les concepts nécessaires pour réaliser un arbre de défaillance qualitatif. Il manque donc également les lois de probabilité qui peuvent être déterminées par une analyse plus approfondie par un expert en SdF. Il sera possible de récupérer la structure de l'arbre de défaillance, la seule activité avant l'analyse sera donc de compléter la structure avec les lois de défaillance.

[30] *« GTS is thus a states/transitions formalism where states are implicit, i.e. given by variables assignments. »*

5.7.3 Lien entre les approches structurelle et comportementale

Si l'on reprend la Figure 37 et la Figure 42, on constate que les approches structurelle et comportementale sont liées. Chaque diagramme à états comporte ainsi un patron commun provenant du comportement de la fonction. Chaque port de sortie de cette fonction nécessitera plusieurs états liés dans le diagramme comportemental. Ces différents états liés comportent un certain nombre de règles, car il ne peut y avoir qu'un seul de ces états actif à chaque instant. Leur activation ou désactivation permet de tirer une transition pour un autre diagramme comportemental d'une fonction en aval de celle étudiée.

L'ajout d'une *Safety Function* au niveau structurel permet par ailleurs d'avoir des évènements supplémentaires dans le diagramme comportemental. Dans le cas d'une *Safety Function* type *Composant de sécurité*, l'impact de son activation ne sera pas direct. Toutefois, la réduction d'un flux amont modifiera le comportement de la fonction étudiée. Il sera ainsi possible de spécifier le comportement de la *Safety Function* et son impact sur les entrées de la fonction aval. Dans le cas d'une *Safety Function* type *Système instrumenté de sécurité*, la *Safety Function* aura un impact direct avec la création d'un flux d'entrée supplémentaire pour la fonction étudiée au niveau structurel et des évènements tels qu'*activation_SF* au niveau comportemental.

5.8 CONCLUSION

Ce chapitre a proposé une méthode et des modèles, basés sur SysML, permettant d'aboutir au *Functional Safety Concept*. L'idée a été d'enrichir dysfonctionnellement des architectures fonctionnelles décrites à l'aide d'*Internal Block Diagram* ou de *State Machine Diagram*, et de formalismes dérivés de HipHops ou AltaRica. La méthode proposée intègre quatre activités focalisées tantôt sur l'aspect structurel et statique de l'architecture fonctionnelle, tantôt sur son aspect comportemental et dynamique. La détermination des chemins critiques dysfonctionnels par rétropropagation des phénomènes dangereux et la définition des *Safety Functions* permettent d'enrichir dysfonctionnellement la vue structurelle de l'architecture fonctionnelle. La définition de *Safety Functions* enrichit les vues structurelle et comportementale. Enfin, la méthode et les modèles proposés ont été illustrés dans un cas relatif à la génération d'énergie et à la combustion. Cependant, les résultats présentent des limites. Tout d'abord, la méthode n'a pas été encodée dans un outil logiciel. Une première vérification à l'aide d'une modélisation sous Prolog a été effectuée, mais cela n'est qu'une toute première étape dans une telle voie. Une vérification couplée de la vue structurelle et de la vue comportementale permettrait de valider complètement les apports de ce chapitre.

Ce cinquième chapitre a proposé la dernière contribution de ce mémoire. Il est donc maintenant nécessaire de conclure celui-ci.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Cette thèse a abordé une problématique classique en conception, à savoir le couplage de de la sûreté de fonctionnement (SdF) et du processus de conception. Cette problématique a été traitée d'un point de vue particulier, puisque l'enjeu de ce travail de recherche était de donner un premier cadre à une Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF) à laquelle pourrait se référer l'architecte système (AS) lorsqu'il définit les exigences systèmes ou développe une architecture fonctionnelle sûre. L'état de l'art a montré qu'il existe, en matière de SdF et d'Ingénierie Système (IS), des éléments de base, mais que ceux-ci ne peuvent être réutilisés en l'état pour définir complètement l'ISSdF. De ce fait, un modèle conceptuel de ce nouveau domaine a été proposé, suivi d'un processus d'ISSdF et de la présentation de deux méthodes permettant à l'AS d'enrichir dysfonctionnellement les exigences systèmes et les architectures fonctionnelles. L'objet de cette conclusion sera de synthétiser les résultats de cette thèse, avant d'esquisser des perspectives possibles.

SYNTHÈSE DES RÉSULTATS

La présente thèse a apporté quatre contributions à l'ISSdF, résumées et discutées ci-dessous.

La première contribution consiste en un modèle conceptuel de l'ISSdF. Une démarche d'IS a été adoptée pour concevoir ce modèle. Reprenant la notion de vue, le modèle conceptuel proposé permet de définir et de relier les concepts utilisables pour définir ce qu'est un système sûr selon différents points de vue de l'AS. Le modèle proposé cherche à rendre cohérent et à systématiser ces normes majeures que sont l'ISO 15288:2008 pour l'IS, et l'ISO 26262:2011 pour la SdF automobile. Ce modèle n'est pas à ce jour exécutable informatiquement. Par contre, il peut être utilisé dans le cadre d'un processus d'ISSdF qui constitue notre deuxième contribution. De plus, raffiné par les différents points de vue, il permet de définir les concepts utiles à la définition des exigences et à l'architecture d'un système sûr, tout en respectant les livrables prescrits par l'ISO 26262:2011.

La deuxième contribution réside dans un processus d'ISSdF. La conception de ce processus s'est également fondée sur une démarche d'Ingénierie Système. Basé sur les processus d'IS de l'ISO 15288:2008 et du *SEBoK*, le processus d'ISSdF permet de définir les activités permettant d'aboutir à une architecture fonctionnelle de système sûr (AFSS). Afin de bien prendre en

compte les besoins des architectes, ce processus est décomposé en trois phases de conception et en quatre points de vue. Ces points de vue sont en relation avec les vues du modèle conceptuel et couvrent bien les différents concepts définis dans ce modèle conceptuel.

La troisième contribution consiste en une méthode pour déterminer les exigences d'un système sûr, ce qui inclut les *Safety Goals*. Pour ce faire, et afin d'améliorer l'efficacité du travail des AS, une méthodologie s'appuyant sur différents diagrammes a été proposée. En reprenant les bases et l'objectif de l'Analyse Préliminaire des Risques, la méthodologie proposée insiste en particulier sur l'enrichissement dysfonctionnel des diagrammes permettant d'aboutir à une bonne détermination des *Safety goals*. L'application sur un exemple a permis de réaliser une première validation de cette méthodologie.

La quatrième contribution est une méthodologie pour concevoir une architecture fonctionnelle de système sûr (AFSS). Celle-ci a été élaborée en s'appuyant sur les méthodes et sur le processus ISSdF élaboré dans le chapitre 3. Afin de pouvoir la mettre en œuvre, des patrons et des stéréotypes SysML ont été proposés. Comme pour le chapitre précédent, une application sur un exemple a été traitée afin de réaliser une première validation. Toutefois, nous sommes conscients que cette étude de cas ne nous permet pas de tirer des conclusions sur des gains temporels ou économiques obtenus par la mise en œuvre de la méthode. Par contre, notre méthode est en conformité avec les activités et les livrables de la norme ISO26262:2011.

Nos principales contributions ayant été recensées, quelles perspectives ouvrent-elles ? C'est ce que nous allons maintenant préciser.

PERSPECTIVES

Nous appréhendons nos travaux comme un premier pas vers l'ISSdF. Pour que celle-ci gagne en crédibilité, il nous paraît nécessaire d'apporter au moins les quatre compléments suivants.

Développement d'un outil — Un premier développement possible pourrait consister à la réalisation d'un outil reprenant les différentes méthodes proposées dans ce mémoire. En effet, nous avons proposé une méthode permettant de définir les *Safety Goals* et qui pourrait être encodée. Cela est également le cas pour la méthode permettant de déterminer et de vérifier une AFSS à partir de laquelle nous pourrions extraire automatiquement *le Functional Safety Concept.* Nous avons développé une première maquette informatique servant de preuve de concept (recherche de chemins critiques avant et après introduction d'une *Safety Function*).

Mise à l'épreuve de l'ISSdF sur un véhicule correspondant à l'état de l'art automobile — Un autre développement du travail proposé dans ce mémoire pourrait consister à apprécier si le modèle conceptuel, les processus et les outils présentés pourraient être généralisés à d'autres fonctions du véhicule, voire à l'ensemble d'un véhicule, si possible correspondant à l'état de l'art automobile. Le véhicule autonome serait à ce titre un bon candidat. Il s'agit d'une innovation de rupture, au sens où non seulement il embarque des solutions inédites, notamment logicielles ou électroniques, mais il suppose de définir les limites du domaine opérationnel sûr au-delà duquel la performance, donc la sûreté, du véhicule ne peut être garantie. Du point de vue de la SdF, un tel véhicule pose de nouvelles questions. Un des leviers concernant sa sûreté est la possibilité que le conducteur a de réagir positivement en cas de situation dangereuse, ce qui est pris en compte dans l'ISO 26262:2011 par le biais du critère de contrôlabilité. Le paradoxe est donc d'avoir, pour le concepteur, à intégrer dans la démarche d'ISSdF un Humain, le conducteur, dans un système technique supposé bien fonctionner sans lui ! À ce sujet, on peut envisager plusieurs axes de réflexion :

1. contrairement à l'aéronautique où le domaine de vol est bien défini, l'ISSdF du véhicule autonome supposera de bien définir les limites des différentes situations opérationnelles du véhicule. La méthode de définition des exigences de système sûr le prend en compte et pourrait être *a priori* adaptée sans surcoût particulier ;
2. le critère de la contrôlabilité doit être analysé de près. Il doit être affiné d'un point de vue réglementaire afin de définir l'ASIL. Actuellement, tous les scénarios dysfonctionnels pour un véhicule autonome souhaité (cas où le conducteur lit son journal par exemple) auront un critère C3 (non contrôlable), ce qui doit être revu. Cette modification aura un impact sur la méthode de définition des exigences de système sûr car il faudra revoir les différentes séquences redoutées de situations opérationnelles où l'analyse de la contrôlabilité devra être modifiée.

Extension du domaine de l'ISSdF à l'organique — Le troisième complément concerne le processus d'ISSdF proposé et les outils associés. Ceux-ci ne couvrent que l'aspect fonctionnel ; l'aspect organique, physique, géométrique, etc., n'ayant pas été traité. Il convient d'étendre le processus d'ISSdF vers la conception aval, de sorte à mieux caractériser les activités d'architecture organique associées à la SdF. De plus, des méthodes, inspirées de celles proposées pour l'aspect (dys-)fonctionnel, pourraient également être mises en place pour définir le *Technical Safety Concept*. Des outils comme HipHops ou Safety Architect sont d'ailleurs utilisables pour traiter d'une architecture organique sûre de fonctionnement. Pour ce qui concerne les modes de défaillance, il serait possible de reprendre les mêmes mots guides ou de prendre les données du fournisseur du composant selon le cas, comme le montre le tableau suivant. Il sera ainsi possible de lier les modes de défaillance organiques à ceux fonctionnels.

Tableau 29 : Adaptation des méthodes fonctionnelles à l'organique

Concept de l'ISSdF fonctionnelle	Concept de l'ISSdF organique
Fonction	Composant
(Déviation de) Flux	(Déviation de) Flux
Mode de défaillance fonctionnel	Mode de défaillance organique
Opérateur	Opérateur
Safety Function, Safety Measure, External Measure	*Safety Mechanism, Safety Measure, External Measure*
Safety Goal	*Functional Safety Requirement*
Functional Safety Requirement	*Technical Safety Requirement*

Une deuxième perspective organique intéressante à envisager est l'analyse zonale. Celle-ci vise à apprécier si la forme et le placement relatif des composants permettent de garantir structurellement la sûreté. Cette dernière perspective présente l'intérêt d'articuler les modèles, abstraits, de l'IS ou de l'ISSdF fonctionnelle, avec les modeleurs géométriques de la CAO. De la sorte, c'est à la fois la conception fonctionnelle et le développement de la solution réaliste qui seraient couverts par l'ISSdF.

Une troisième perspective organique concerne les lignes de produit. Ce thème a pour mérite de faire remonter des préoccupations de process de fabrication dans une IS encore trop focalisée sur le produit. La ligne de produit est l'un des moyens pour produire de la variété à moindre coût. Des travaux récents se sont engagés dans cette voie, que ce soit par le biais de l'IS (Friedlhuber, 2014) ou par celui de la SdF (Parmeza, 2015). Une des problématiques abordées est la réutilisation de modèles (au sens de modèles informatiques, et non de modèles d'automobiles) en utilisant des variantes. Les méthodes proposées dans ce mémoire pourraient être une piste de réponse. Au lieu de recréer le modèle dysfonctionnel pour faire les analyses de sûreté, un modèle générique du type de celui proposé serait enrichi de variantes, si bien que la création du modèle dysfonctionnel gagnerait en rapidité.

Interface avec les processus de spécialité — Si l'on reprend la Figure 1, nous nous sommes concentrés sur le processus liant Ingénierie Système et Sûreté. Nous avons également précisé dans les chapitres de contribution comment cela pourrait être lié aux activités spécifiques de sûreté. Cependant, nous n'avons pas abordé les interfaces avec les processus de spécialité. Cela pourra être plus aisé une fois la vue organique réalisée. Cela permettra de dresser un cahier des charges pour chaque spécialité comprenant des exigences fonctionnelles et des exigences de sûreté.

D'autres perspectives concernent la vérification voire la simulation des modèles proposés et le développement de personnalisation d'outils SysML pour l'ISSdF. En effet, nous avons proposé une première vérification de la vue structurelle par une modélisation sous Prolog mais celle-ci pourrait être complétée, voire réellement prototypée, par un outil informatique. De plus

il serait possible de vérifier également la vue comportementale en se basant sur des mécanismes comme la recherche d'atteignabilité.

Amélioration de la vérification sous Prolog — Nous avons réalisé une première maquette, mais celle-ci pourrait être améliorée. La première amélioration consiste en l'introduction d'un critère de sûreté. Il serait de la sorte possible de visualiser l'impact de la modification d'un paramètre de sûreté d'une fonction (exposition ou ASIL) pour la sortie d'un système selon la mise en place d'une *Safety Function*. Il serait nécessaire de déterminer la loi de composition des critères, mais cela est déjà disponible si l'ASIL est repris. Un autre programme à développer permettrait de déterminer si la *Safety Function* a un effet immédiat ou s'il faut plusieurs pas de simulation avant qu'elle fasse effet. Cela nécessite d'inclure de la dynamique dans le modèle.

Une autre amélioration serait d'aider le concepteur lorsqu'il y a des différences entre le comportement logique global et les comportements logiques des différentes sous-fonctions. Un programme pourrait indiquer des pistes de solutions pour résoudre le problème de cohérence (problème de spécification, solution non cohérente ou non valable, solution meilleure que souhaitée).

Une amélioration possible, utile en conception préliminaire et en conception détaillée, serait de prendre en compte des flux continus avec des valeurs minimales et maximales et non des valeurs discrètes. Les lois de comportement risquent d'être assez complexes et devront vérifier les lois de la physique. Cette modélisation pourrait être réalisée après une modélisation multi-physique (sous Modelica par exemple) du système étudié.

La dernière amélioration concernerait la génération d'architectures enrichies dysfonctionnellement en reprenant des FOTS de bases. En effet, à l'aide des programmes réalisés, il serait possible de déterminer les phénomènes dangereux possibles d'un ensemble de sous-fonctions. Cela permettrait également de fournir par avance des emplacements idéaux de *Safety Functions*.

Vers une vérification par recherche d'atteignabilité — Au début du chapitre 5, nous avons donc déterminé un patron pour la vue comportementale de l'architecture fonctionnelle, puis nous avons explicité comment le compléter par un processus inscrit dans le processus global de l'ISSdF défini dans le chapitre 3. Une piste que nous proposons est la recherche d'atteignabilité. En effet, à l'aide du patron proposé, la déviation non souhaitée sera modélisée. Cette vérification pourra également être effectuée par un spécialiste de SdF qui pourra prendre en compte les aspects stochastiques.

Les perspectives évoquées n'épuisent pas le domaine de l'ISSdF. Nul doute que celle-ci sera amenée à intéresser plus fortement dans un avenir proche et les industriels, et les chercheurs.

RÉFÉRENCES BIBLIOGRAPHIQUES[31]

Adedjouma, M. (2012). *Requirements engineering process according to automotive standards in a model-driven framework.* Université Pierre Mendès France: Thèse de Doctorat.

AFIS. (2004). *Glossaire de base de l'Ingénierie de Systèmes.*

AFIS. (2007). *Découvrir et Comprendre l'Ingénierie Système.* Orsay: Cépaduès.

ARP 4754. (2010). *Certification Considerations for Highly-Integrated Or Complex Aircraft Systems.*

ARP 4761. (1996). *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.*

ATESST. (2010). Dependability Modeling with EAST-ADL. *ATESST2 Concept presentation 2010.*

Aubry, J.-F., Babykina, G., Brinzei, N., Medjaher, S., Barros, A., Bérenguer, C., . . . Zhang, H. (2012). The APPRODYN project: dynamic reliability approaches to modeling critical systems. *Supervision and Safety of Complex Systems*, 181-222.

Batteux, M., Prosvirnova, T., Rauzy, A., & Kloul, L. (2013, July 29-31). The AltaRica 3.0 project for model-based safety assessment. *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, 741-746.

Belmonte, F., & Soubiran, E. (2012). A Model Based Approach for Safety Analysis. *SAFECOMP Workshop*, 50-63.

Belt, R. A. (2012). *An Electronic Cause for Sudden Unintended Acceleration*. Consulté le January 26, 2016, sur The centre for auto safety: http://www.autosafety.org/dr-ronald-belt%E2%80%99s-sudden-acceleration-papers

Belt, R. A. (2015-1). *Unintended Acceleration with a Confirmed Cause*. Consulté le January 26, 2016, sur The centre for auto safety: http://www.autosafety.org/dr-ronald-belt%E2%80%99s-sudden-acceleration-papers

Belt, R. A. (2015-2). *Sudden Unintended Acceleration in an All-Electric Vehicle*. Consulté le January 26, 2016, sur The centre for auto safety: http://www.autosafety.org/dr-ronald-belt%E2%80%99s-sudden-acceleration-papers

Bock, C. (2006). SysML and UML 2 Support for Activity Modeling. *Systems Engineering, 9*(2), 160-186.

Bonjour, É. (2008). *Contributions à l'instrumentation du métier d'architecte système : de l'architecture modulaire du produit à l'organisation du système de conception.* Université de Franche-Comté: Habilitation à Diriger des Recherches.

Bonjour, É., & Micaëlli, J.-P. (2010). Design Core Competence Diagnosis: A Case from the Automotive Industry. *IEEE Transactions on Engineering Management, 57*(2), 323-337. doi:10.1109 / TEM.2009. 2036838

Bonjour, É., Deniaud, S., & Micaëlli, J.-P. (2009). Conception complexe et ingénierie système. Dans S. Aït-el-Hadj, & V. Boly, *Les Systèmes techniques : lois d'évolution et méthodologies de conception* (pp. 83-101). Paris: Hermès.

[31] *138 Références bibliographiques sont citées dans ce mémoire.*

Bonjour, É., Deniaud, S., & Micaëlli, J.-P. (2013). A method for jointly drawing up the functional and design architectures of complex systems during the preliminary system-definition phase. *Journal of Engineering Design, 24*(4), 305-319. doi:10.1080/09544828.2012.737457

Bonjour, É., Deniaud, S., Dulmet, M., & Harmel, G. (2009). A fuzzy method for propagating functional architecture constraints to physical architecture. *Transactions of ASME, Journal of Mechanical Design, 131*(6), 061002-1 – 061002-11.

Bouffaron, F. (2016). *Co-spécification exécutable basée sur des modèles - Application à la conduite interactive d'un procédé industriel critique.* Université de Lorraine: Thèse de Doctorat.

Bouffaron, F., Dupont, J.-M., Mayer, F., & Morel, G. (2014). Integrative construct for Model-Based Human-System Integration : a case study. *19th IFAC World Congress, IFAC'14.* Cape Town, South Africa.

Bouissou, M., & Bon, J.-L. (2003). A new formalism that combines advantages of fault-trees and Markov models: Boolean logic Driven Markov Processes. *Reliability Engineering & System Safety, 82*(2), 149-163.

Bourey, J.-P. (2014). Modèles, transformations de modèles et alignements de SI. *Journée Nationale du GT EASY DIM 2014.* Paris.

Brameret, P.-A., Roussel, J.-M., & Rauzy, A. (2013). Preliminary System Safety Analysis with Limited Markov Chain Generation. *4th IFAC Workshop on Dependable Control of Discrete Systems*, (p. n°3). York.

Campan, C. (2014). Factorisation de la modélisation structurelle pour plusieurs modèles métier. *GTR Recherche méthodologique de l'IMdR.*

Chalé, H. G., Taofifenua, O., Gaudré, T., Topa, A., Lévy, N., & Boulanger, J.-L. (2011). Reducing the Gap Between Formal and Informal Worlds in Automotive Safety-Critical Systems. *21st Annual INCOSE International Symposium. 28.* INCOSE.

Chapurlat, V., & Bonjour, E. (2014). From Model Based Systems Engineering to Model Based System Realization: Role and Relevance of IVTV Plan. *Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World*, 109-116.

Chapurlat, V., & Braesch, C. (2008). Verification, validation, qualification and certification of enterprise models: Statements and opportunities. *Computers in Industry, 59*(7), 711-721.

Cherfi, A., Leeman, M., Meurville, F., & Rauzy, A. (2014). Modeling automotive safety mechanisms: A Markovian approach. *Reliability Engineering & System Safety, 130*, 42-49.

Cimatti, A., Clarke, E. M., Giunchiglia, E., Giunchiglia, F., & Pistore, M. (2002). Nusmv 2: An opensource tool for symbolic model checking. *Computer Aided Verification*, 359-364.

Cocheteux, P. (2010). *Contribution à la maintenance proactive par la formalisation du processus de pronostic des performances de systèmes industriels.* Nancy: Thèse de Doctorat.

Cook, S. C., & Ferris, T. L. (2007). Re-evaluating Systems Engineering as a Framework for Tackling Systems Issues. *Systems Research and Behavioral Science, 24*(2), 169-181.

Cori, R., & Lascar, D. (1993). *Logique mathématique. Volume 1.* Masson.

Cornu, C. (2012). *Contribution à la prise en compte de l'interopérabilité pour le déploiement de processus complexes dans une grande entreprise : proposition d'un guide méthodologique outillé pour les processus d'Ingénierie Système.* Ecole Nationale Supérieure des Mines de Paris: Thèse de Doctorat.

Cressent, R., David, P., Idasiak, V., & Kratz, F. (2012). Designing the database for a reliability aware Model-Based System Engineering process. *Reliability Engineering and System Safety, 111*.

David, P. (2009). *Contribution à l'analyse de sûreté de fonctionnement des systèmes complexes en phase de conception : application à l'évaluation des missions d'un réseau de capteurs de présence humaine.* Université d'Orléans: Thèse de Doctorat.

David, P., Idasiak, V., & Kratz, F. (2010). Reliability study of complex physical systems using SysML. *Reliability Engineering & System Safety, 95*(4), 431-450.

de Oliveira, A., Braga, R. T., Masiero, P. C., Papadopoulos, Y., Habli, I., & Kelly, T. (2014). A Model-Based Approach to Support the Automatic Safety Analysis of Multiple Product Line Products. *Brazilian Symposium on Computing Systems Engineering.* Manaus, Brazil.

Deniaud, S., Bonjour, É., Micaëlli, J.-P., & Loise, D. (2010, Novembre 11-12). How to integrate reliability into Systems Engineering framework. A case from automotive industry. *CRECOS seminar*.

Desinde, M. (2006). *Contribution à la mise au point d'une approche intégrée analyse diagnostique/analyse de risques.* Université Joseph-Fourier - Grenoble I: Thèse de Doctorat.

Desroches, A., Leroy, A., & Vallée, F. (2003). *La gestion des risques : Principes et pratiques.* Hermès Sciences Publications.

DO-178C. (2012). *Software Considerations in Airborne Systems and Equipment Certification.*

Dobre, D. (2010). *Contribution à la modélisation d'un système interactif d'aide à la conduite d'un procédé industriel.* Université Henri Poincaré, Nancy 1 : Thèse de Doctorat.

Dori, D. (2002). Object-Process Methodology: A Holistic System Paradigm. (Springer, Éd.)

Doufene, A., Hugo, G. C.-G., Dauron, A., & Krob, D. (2014). Un cadre méthodologique intégré pour l'architecture et l'optimisation de systèmes. *to be published in Revue Génie Logiciel et Systèmes*.

Dumont, J., Sadmi, F., & Vallée, F. (2011). Safety Architect©: un outil d'analyse de risques s' inscrivant dans les processus d'ingénierie de systèmes complexes. *Génie Logiciel*(98), 27-33.

EIA 632. (2003). *Processes for Engineering a System.*

Ericson, C. (2005). *Hazard Analysis Techniques for System Safety.* John Wiley & Sons.

Fagès, F. (1996). *Programmation logique par contraintes.* Ellipses.

Faisandier, A. (2014). *Notions de système et d'ingénierie de système.* Sinergy'Com.

Feliot, C. (1997). *Modélisation de systèmes complexes: intégration et formalisation de modèles.* Université de Lille 1: Thèse de Doctorat.

Fiorèse, S., & Meinadier, J.-P. (2012). *Découvrir et comprendre l'Ingénierie Système.* Cépaduès Éditions.

Friedenthal, S., Moore, A., & Steiner, R. (2011). *A Practical Guide to SysML, Second Edition: The Systems Modeling Language.* (T. M. Elsevier, Éd.) Burlington.

Friedlhuber, T. (2014). *Model Engineering in a Modular PSA.* LIX, Ecole Polytechnique: Thèse de Doctorat.

Gruber, T. (1993, Août). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies, 43*, 907-928.

Gruber, T., Liu, L., & Tamer, M. (2009). *Encyclopedia of Database Systems.* Springer-Verlag.

Guide 51. (1999). *Safety aspects - Guidelines for their inclusion in standards.*

Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of computer programming, 8*(3), 231-274.

Harmel, G. (2007). *Vers une conception conjointe des architectures du produit et de l'organisation du projet dans le cadre de l'Ingénierie Système.* Université de Franche-Comté: Thèse de Doctorat.

Haworth, N., & Symmons, M. (2002). Hazard perception by inexperienced motorcyclists. *National Conference on Developing Safer Drivers and Riders.* Brisbane, Queensland, Australia.

Hibti, M., Friedlhuber, T., & Rauzy, A. (2012). Overview of the open psa platform. *International Joint Conference PSAM.*

Hiettiko, M., Malm, T., & Alanen, J. (2011). Risk estimation studies in the contex tof a machine control function. *Reliability Engineering and System Safety, 96*, 767-774.

IBM. (2011). *Systems Engineering Best Practices with the Rational Solution for Systems and Software Engineering.*

IEC 61508. (2010). *Functional safety of electrical/electronic/programmable electronic safety related systems.*

IEEE 1220. (2005). *Standard for Application and Management of the Systems Engineering Process.*

IEEE 1471. (2000). IEEE Recommended Practice for Architectural Description for Software-Intensive Systems.

INCOSE. (2008). *Survey of Model-Based Systems Engineering (MBSE) Methodologies.*

INCOSE. (2010). *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities.* (éd. Version 3.2.1). San Diego, CA, USA.

INCOSE. (2011). *INCOSE systems engineering handbook v. 3.2.2.*

Inspection Générale de l'Education Nationale. (2012). Présentation de l'organisation des services et des enseignements . *Séminaire CPGE ATS avril 2012.*

ISO 15288. (2008). Systems Engineering - System life cycle processes.

ISO 15288. (2015). Systems and software engineering — System life cycle processes.

ISO 15504. (2004). *Information technology - Process assessment.*

ISO 26262. (2011). *Functional safety for road vehicles.*

ISO/IEC 15026. (2013). *Systems and software engineering -- Systems and software assurance.*

ISO/IEC 2382. (1993). *Technologies de l'information.*

Jang, H. A., Kwon, H. M., Hong, S.-H., & Lee, M. K. (2015, June). A Study on Situation Analysis for ASIL Determination. (M. Othman, Éd.) *Journal of Industrial and Intelligent Information, 3*(2), 152-157.

Kemmann, S. (2015). *SAHARA - A Structured Approach for Hazard Analysis and Risk Assessments.* Technische Universität Kaiserslautern: Thèse de Doctorat.

Kloul, L., Prosvirnova, T., & Rauzy, A. (2013). Modeling Systems with Mobile Components: A comparison between AltaRica and PEPA nets. *Journal of Risk and Reliability, 227*(6), 599–613.

Kruchten, P. (2004). *The rational unified process: an introduction.* Addison-Wesley Professional.

Laprie, J.-C. (1995). *Guide de la sûreté de fonctionnement.* Toulouse, France: Editions Cépaduès.

Larousse. (2016). *Définitions : Architecte*. Consulté le Janvier 31, 2016, sur Larousse: http://www.larousse.fr/dictionnaires/francais/architecte/5072

Le Moigne, J.-L. (1990). *La Modélisation des Systèmes Complexes.* Dunod.

Léger, A. (2009). *Contribution à la formalisation unifiée des connaissances fonctionnelles et organisationnelles d'un système industriel en vue d'une évaluation quantitative des risques et de l'impact des barrières envisagées.* Université Henri Poincaré-Nancy I: Thèse de Doctorat.

Leger, J.-B. (1999). *Contribution Méthodologique à la Maintenance Prévisionnelle des Systèmes Industriels de Production : Proposition d'un Cadre Formel de Modélisation.* Université de Nancy 1: Thèse de Doctorat.

Leveson, N. (1995). *Safeware: System Safety and Computers.* Boston: Addison Wesley.

Leveson, N. (2004, April). A New Accident Model for Engineering Safer Systems. *Safety Science, 42*(4), 237-270.

Leveson, N. (2011). *Engineering a safer world: Systems thinking applied to safety.* MIT Press.

Lô, M. (2013). *Contribution à l'évaluation d'architectures en Ingénierie Système : application en conception de systèmes mécatroniques.* Université Montpellier II: Thèse de Doctorat.

Lykins, H., Friedenthal, S., & Meilich, A. (2000). Adapting UML for an object oriented systems engineering method (OOSEM). *Proceedings of the Tenth Annual INCOSE Symposium, International Council on Systems Engineering.*

Mauborgne, P., Deniaud, S., Levrat, É., Bonjour, É., Micaëlli, J.-P., & Loise, D. (2015). Preliminary Hazard Analysis Generation Integrated with Operational Architecture - Application to Automobile. *Complex Systems Design & Management* (pp. 297-309). Springer Verlag.

Mauborgne, P., Labe, M., Smouts, A.-S., Stojanovic, N., & Delgado, J. (2013). Towards a transition approach from a functional model in SysML with Harmony Revisited methodology to fault trees. *IWMBSA, Versailles, 25-27 Mars 2013*.

Merle, G., Roussel, J.-M., & Lesage, J.-J. (2011). Algebraic determination of the structure function of Dynamic Fault Trees. *Reliability Engineering & System Safety, 96*(2), 267-277.

Mhenni, F. (2014). *Vers une approche intégrée d'analyse de sureté de fonctionnement des systèmes mécatroniques.* Châtenay-Malabry: Thèse de Doctorat.

Ministère de l'Education Nationale. (2011). *Ressources pour le cycle terminal - Enseignements technologiques transversaux et enseignements spécifiques (série STI2D).*

Montanari, U. (1974). Networks of constraints : fundamental properties and application to picture processing. *Information sciences, 7*, 95-132.

Morel, G., Dupont, J.-M., Lieber, R., Bouffaron, F., Méry, D., Mayer, F., & Marty, J.-L. (2013). Spécification d'exigences physico-physiologiques d'interaction homme-machine en ingénierie système. *Génie Logiciel*(104), 29-39.

Mortureux, Y. (2002). *Analyse Préliminaire des Risques* (Vol. SE4010). Techniques de l'Ingénieur.

Mortureux, Y. (2002). *Arbres de défaillance, des causes et d'événement.* Techniques de l'Ingénieur.

NASA. (2013). *NPR 7123.1B NASA Systems Engineering Processes and Requirements.*

NF EN 60812. (2006). *Techniques d'analyses de la fiabilité du système - Procédure d'analyse des modes de défaillance et de leurs effets (AMDE).*

NF EN ISO 14040. (2006). *Management environnemental - Analyse du cycle de vie - Principes et cadre.*

OMG. (2012). *OMG Systems Modeling Language (OMG SysML) V1.3.*

Ozouf, V. (2009). *Comment conserver un niveau de risques acceptable dans un contexte de conception/industrialisation de plus en plus rapide d'un produit de plus en plus complexe?* Université de Savoie: Thèse de Doctorat.

Pahl, G., Beitz, W., Feldhusen, J., & Grote, K.-H. (2007). *Engineering Design : A systematic approach.* Springer Science & Business Media.

Panetto, H. (2006). *Meta-modèles et modèles pour l'intégration et l'interopérabilité des applications d'entreprises de production.* Université Henri Poincaré - Nancy I: Habilitation à Diriger des Recherches.

Papadopoulos, Y., & McDermid, J. A. (1999). Hierarchically Performed Hazard Origin and Propagation Studies. *Computer Safety, Reliability and Security*, 139-152.

Papadopoulos, Y., Walker, M., Parker, D., Rüde, E., Hamann, R., Uhlig, A., . . . Lien, R. (2011). Engineering failure analysis and design optimisation with HiP-HOPS. *Engineering Failure Analysis, 18*(2), 590-608.

Parmeza, D. (2015). *Cost and Efforts in Product Lines for Developing Safety Critical Products – An Empirical Study.* Mémoire de Master, Mälardalen University - Volvo.

Penalva, J.-M. (1990). SAGACE : une représentation des connaissances pour la supervision de procédés continus. *10. International days on Expert Systems and their Applications.* Avignon.

Pétin, J.-F., Iung, B., & Morel, G. (1998). Distributed intelligent actuation and measurement (IAM) system within an integrated shop-floor organisation. *Computers In Industry, 37*, 197-211.

Pfister, F., Chapurlat, V., Huchard, M., Nebut, C., & Wippler, J.-L. (2011). A Proposed Meta-Model for Formalizing Systems Engineering Knowledge, Based on Functional Architectural Patterns. *Systems Engineering, 15*(3), 321-332.

Project Management Institute. (2001). *Project Management Body of Knowledge.*

Projet IMOFIS. (2011). *IMOFIS Ingénierie des MOdèles de FonctIons Sécuritaires*. Consulté le 11 27, 2014, sur http://www.imofis.org/

Prosvirnova, T., & Rauzy, A. (2012). Système de transitions gardées : Formalisme pivot de modélisation pour la sûreté de fonctionnement. *LambdaMu 18*.

Prosvirnova, T., Batteux, M., Brameret, P.-A., Cherfi, A., Friedlhuber, T., Roussel, J.-M., & Rauzy, A. (2013). The AltaRica 3.0 Project for Model-Based Safety Assessment. *DCDS.*

Pyster, A., & Olwell, D. H. (2013). The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.1.2. Hoboken, NJ: The Trustees of the Stevens Institute of Technology.

Retho, F. (2014). *Méthodologie collaborative d'aide à la construction de produits virtuels pour la conception d'aéronefs à propulsion électrique.* Centrale-Supélec: Thèse de Doctorat.

Rochet, S. (2007). *Formalisation des processus de l'Ingénierie Système : Proposition d'une méthode d'adaptation des processus génériques à différents contextes d'application.* INSA Toulouse: Thèse de Doctorat.

Ross, D. (1977). Structured analysis (SA): A language for communicating ideas. *IEEE Transactions on Software Engineering*.

Rygaert, B. (2009). Analyse Fonctionnelle avec SysML. *5ème Conférence Annuelle d'Ingénierie Système.*

Saddem, R. (2012). *Diagnosticabilité modulaire appliquée au Diagnostic en ligne des Systèmes Embarqués Logiques.* Ecole Centrale de Lille: Thèse de Doctorat.

Safety Research & Strategies, Inc. (2010). *Toyota Sudden Unintended Acceleration.*

Schätz, B., & Voss, S. (2014). A pattern-based approach towards modular safety analysis and argumentation. *ERTS.*

Seidner, C. (2009). *Vérification des EFFBDs: Model-checking en Ingénierie Système.* Nantes: Thèse de Doctorat.

Sinha, P. (2011). Architectural design and reliability analysis of a fail-operational brake-by-wire system from ISO 26262 perspectives. (Elsevier, Éd.) *Reliability Engineering and System Safety, 96*, 1349–1359.

Systematic. (2012). *VERDE | Systematic*. Consulté le 11 27, 2014, sur http://www.systematic-paris-region.org/fr/projets/verde

Taofifenua, O. (2012). *Ontology centric design process - Sharing a conceptualization.* CNAM: Thèse de Doctorat.

Tichkiewitch, S. (1996). Specifications on integrated design methodology using a multi-view product model. *The 1996 3 rd Biennial Joint Conference on Engineering Systems Design and Analysis*, 101-108.

Tichkiewitch, S., & Véron, M. (1997). Methodology and product model for integrated design using a multiview system. *CIRP Annals-Manufacturing Technology, 46*(1), 81-84.

Toyota. (2010). Certain Toyota Vehicle Accelerator Pedal Assembly Issue.

Van Caneghem, M. (1984). *L'anatomie de prolog II = PROLOG II.* Intereditions.

Vareilles, É. (2005). *Conception et approches par propagation de contraintes: contribution à la mise en oeuvre d'un outil d'aide interactif.* Albi: Thèse de Doctorat.

Vincoli, J. W. (2014). *Basic Guide to System Safety* (éd. 3e). John Wiley & Sons.

Voirin, J.-L. (2008). Method & Tools for constrained System Architecting. *INCOSE'08 Symposium.* Utrecht.

Walker, M., Bottaci, L., & Papadopoulos, Y. (2007). Compositional temporal fault tree analysis. *Computer safety, reliability, and security*, 106-119.

Weiss, K. A. (2006). Engineering Spacecraft Mission Software using a Model-Based and Safety-Driven Design Methodology. *Journal of Aerospace Computing, Information, and Communication*, 562-586.

Yakymets, N., Jaber, H., & Lanusse, A. (2012). Model-based system engineering for safety analysis of complex systems. *MBSAW.* Bordeaux.

Ye, Y. (2014). *Integrated Decision Support for Architecture & Supplier Identification in Early Complex System Design.* Châtenay-Malabry: Thèse de Doctorat.

LEXIQUE

Dans ce mémoire, plusieurs concepts ont été utilisés. Certains sont définis différemment selon les ouvrages et les standards. La définition que nous avons choisie pour ces concepts est donc présente dans le tableau ci-dessous.

Tableau 30 : Glossaire du mémoire de thèse

Terme	Définition	Source
Activité	A set of actions that consume time and resources and whose performance is necessary to achieve, or contribute to the realization of one or more outcomes.	ISO 15288
Analyse préliminaire des risques	Méthode classique de SdF pour identifier puis classifier les évènements redoutés d'un système	(Mortureux, 2002)
Arbres de défaillance	L'arbre de défaillance est une méthode qui part d'un événement final pour remonter vers les causes et conditions dont les combinaisons peuvent le produire. Il vise à représenter l'ensemble des combinaisons qui peuvent induire l'événement étudié.	(Mortureux, 2002)
Architecte système	L'architecte système concentre ces efforts sur les aspects du processus d'ingénierie système (Définition des exigences, définition des architectures…) qui servent à organiser et coordonner les autres activités d'ingénierie. Dans le processus de développement d'un système, il est la première interface avec le management, les ingénieurs de spécialité, etc…	Adapté de l'INCOSE
Architecture Fonctionnelle	Description du système sous forme d'un arrangement de fonctions, de leurs sous-fonctions et de leurs interactions définissant le séquencement de leur exécution, les flux de données et de contrôle qui le conditionnent et les performances requises pour répondre aux exigences.	IEEE 1220
Architecture Fonctionnelle	A functional architecture is a set of functions and their sub-functions that defines the transformations of input flows into output flows performed by the system to achieve its mission.	SEBoK
Architecture logique	The logical architecture of a system is composed of a set of related technical concepts and principles that support the logical operation of the system. It includes a functional architecture, a behavioral architecture, and a temporal architecture.	SEBoK

Architecture opérationnelle	Architecture définissant le contexte opérationnel du système	
Architecture Organique	Souvent utilisée par opposition à architecture fonctionnelle : les fonctions de l'architecture fonctionnelle sont réalisées par les organes de l'architecture organique (par analogie avec la biologie)	Découvrir et Comprendre l'IS
Architecture Organique	Description d'un système sous forme d'un ensemble de constituants physiques et de leurs interactions traduisant l'architecture fonctionnelle (logique) et satisfaisant les exigences	IEEE 1220
Besoin	Necessity or desire felt by a user	
Cas d'utilisation	Description de l'usage attendu d'un système vu sous l'angle de l'un de ses utilisateurs.	Découvrir et Comprendre l'IS
Complexité	En systémique, terme caractérisant la difficulté de décomposer un système en éléments définissables tout en identifiant toutes leurs interactions (ou encore le fait que le comportement d'un système ne peut être déduit de la connaissance de ses constituants). Terme souvent utilisé pour désigner les caractéristiques (facteur des systèmes telles que : complexité structurelle (nombre, variété, hétérogénéité des constituants), complexité dynamique (nombre et variété des interactions, combinatoire des parcours dans l'espace des états du système), importance du facteur humain, méconnaissance de l'environnement, incertitudes diverses concernant le système ou projet, etc...	Découvrir et Comprendre l'IS
Comportement Défectueux	failure or unintended behavior of an item with respect to its design intent	ISO 26262
Comportement d'un système	Caractéristiques logiques (réponses aux stimuli) et physiques (performances, ressources utilisées) du fonctionnement d'un système.	Découvrir et Comprendre l'IS
Conception	Activité créatrice qui consiste à élaborer un projet, ou une partie des éléments le constituant, en partant des besoins exprimés, des moyens existants et des possibilités technologiques dans le but de créer un bien ou un service.	GDT
Conception d'un système	Ensemble des activités qui, partant d'une spécification des exigences du système, conduit à la définition de son architecture et de ses constituants.	Découvrir et Comprendre l'IS
Constituant logique	Constituant décrit par ses fonctions, ses caractéristiques et ses propriétés indépendamment des solutions physiques et technologiques choisies pour sa réalisation.	Découvrir et Comprendre l'IS
Constituant physique	Constituant réalisable ou existant, possédant des fonctions, caractéristiques et propriétés données.	Découvrir et Comprendre l'IS
Cycle de vie	The system or product evolution initiated by a perceived stakeholder need through the disposal of the products.	IEEE 1220

Défaillance	termination of the ability of an element, to perform a function as required	ISO 26262
Domaine d'intérêt	Le domaine d'intérêt est composé des éléments du sursystème. Il est donc composé du système, de son environnement et des différentes parties prenantes.	
Environnement	L'environnement est constitué de l'ensemble des interacteurs du système.	
Erreur	discrepancy between a computed, observed or measured value or condition, and the true, specified or theoretically correct value or condition	ISO 26262
État	Caractéristique définissant la situation temporaire d'un système ou d'un processus pendant laquelle son comportement est considéré comme invariant relativement au niveau d'analyse temporelle où l'on se place.	Découvrir et Comprendre l'IS
Évènement Dangereux	Un évènement dangereux est selon la norme ISO 26262:2011 la combinaison d'un danger et d'une situation opérationnelle	ISO 26262
Exigence	Quelque chose qui prescrit ce qu'un produit doit faire, avec quelles performances et sous quelles conditions, pour atteindre un but donné.	EIA 632
Exigence allouée	Exigence dont la satisfaction a été attribuée à un ou plusieurs constituants de la solution logique ou physique.	Découvrir et Comprendre l'IS
Exigence dérivée	Exigence découlant d'une exigence amont. L'allocation d'une exigence d'un élément à certains de ses constituants génère des exigences dérivées propres à ces constituants.	Découvrir et Comprendre l'IS
Exigence des parties prenantes	Une exigence des parties prenantes est la formalisation d'un besoin des parties prenantes	
Exigence induite	Exigence découlant de choix techniques de solution (choix d'architecture, contrainte de réalisation, etc…)	Découvrir et Comprendre l'IS
Exigence technique	Une exigence technique est une déclaration qui identifie ce que le produit doit accomplir pour produire le comportement souhaité.	
Exposition	state of being in an operational situation that can be hazardous if coincident with the failure mode under analysis	ISO 26262
Faute	abnormal condition that can cause an element or an item to fail	ISO 26262
Fiabilité	Aspect de la sûreté de fonctionnement définissant le comportement dysfonctionnel d'une pièce	
Fonction	A task, action, or activity performed to achieve a desired outcome.	EIA 632
ilités	Désigne quelquefois les exigences opérationnelles et de soutien : fiabilité, disponibilité, maintenabilité, invulnérabilité, soutenabilité	Découvrir et Comprendre l'IS
Ingénierie de systèmes sûrs de fonctionnement	Concept prôné dans ce mémoire d'une intégration de la sûreté de fonctionnement dans le processus d'Ingénierie Système	

Ingénierie Système	An interdisciplinary collaborative approach to derive, evolve, and verify a life-cycle balanced system solution which satisfies customer expectations and meets public acceptability.	IEEE 1220
Interface	Frontière commune à deux ou plusieurs constituants (ou au système et à son environnement) au niveau de laquelle les règles d'échange, de compatibilité, d'intégrité et de non régression sont à respecter au cours de la vie du système.	Découvrir et Comprendre l'IS
Maturité (pour un métier)	Aptitude à maîtriser les processus d'un métier.	Découvrir et Comprendre l'IS
Menace	Phénomène potentiel externe au système étudié qui, s'il est avéré peut affecter les éléments fonctionnels (fonctions, flux) du système et par voie de conséquence la disponibilité, la sécurité immunité, la survivabilité et/ou la sécurité des informations du système ou de ses éléments constitutifs (constituant, lien) selon leur vulnérabilité. Il peut être la cause d'une défaillance (événement redouté) du système ou du constituant étudié.	Alain Faisandier
Méthode	Ensemble de démarches s'appuyant sur des modèles, les transformant et les intégrant, permettant de réaliser tout ou partie du passage de l'expression d'un besoin à la définition d'une solution à ce besoin.	Découvrir et Comprendre l'IS
Méthodologie	Littéralement : étude des méthodes. Le terme anglais methodology désigne souvent une méthode d'ingénierie de systèmes intégrant plusieurs outils méthodologiques traitant chacune un aspect ou une partie du problème	Découvrir et Comprendre l'IS
Mission	The specific task, duty or function defined to be accomplished by a system.	
Mode	Condition de fonctionnement du système, d'une fonction ou d'un constituant	Découvrir et Comprendre l'IS
Mode de Défaillance	manner in which an element or an item fails	ISO 26262
Mode opérationnel	Un mode opérationnel est un état stationnaire du système.	
Modèle	Représentation d'un système, d'un processus ou d'un phénomène, dans un objectif donné. Système plus ou moins homomorphe au système à étudier ou à construire, qui en donne une vue partielle plus ou moins abstraite et permet d'en étudier certaines caractéristiques.	Découvrir et Comprendre l'IS
Modèle Conceptuel	Modèle permettant de montrer les concepts et le relations entre ces concepts pour un domaine donné. (Cf. Annexe 1)	Jean-Pierre Bourey
Modélisation	Analyse et représentation simplifiées d'un phénomène ou d'un système en vue d'étudier son déroulement ou son fonctionnement par simulation.	AFNOR
Outil	Instrument d'une méthode améliorant l'efficacité d'une tâche. L'outil méthodologique (élément d'une méthode) est distingué de l'outil informatisé servant de support à la mise en œuvre d'une méthode.	Découvrir et Comprendre l'IS

Partie Prenante	A party having a right, share or claim in a system or in its possession of characteristics that meet that party's needs and expectations.	ISO 15288
Phénomène Dangereux	Le danger est selon la norme ISO 26262:2011 une source potentielle de préjudice dû à un mauvais fonctionnement du système.	ISO 26262
Phénomène Dangereux	A potentially unsafe condition resulting from failures, malfunctions, external events, errors, or a combination thereof.	ARP 4761
Préjudice	Le préjudice selon la norme ISO 26262:2011 est une blessure physique ou une atteinte à la santé des personnes.	ISO 26262
Processus	Set of interrelated or interacting activities which transforms inputs into outputs	ISO 9000
Produit	Result of activities or processes	ISO 8402
Scénario opérationnel	The sequencing of the System's (or products') states during a mission and inoperation.	AFIS
Projet	Unique process, consisting of a set of coordinated and controlled activities with start and finish dates, undertaken to achieve an objective conforming to specific requirements, including the constraints of time, cost and resources	ISO 10006
Safety Goal	Le Safety Goal est une exigence de sûreté de haut-niveau résultant de l'activité Hazard Analysis and Risk Assessment.	ISO 26262
Sécurité immunité	Aptitude d'un système à résister aux agressions extérieures naturelles, accidentelles ou involontaires.	Alain Faisandier
Sécurité innocuité	Aptitude d'un système à assurer la protection de des biens, des personnes et de l'environnement de son contexte face à ses propres actions.	Alain Faisandier
Sécurité intégrité	Aptitude du système à ne pas altérer ses caractéristiques internes par son propre fonctionnement.	Alain Faisandier
Situation Opérationnelle	Scenario that can occur during a vehicle's life	ISO 26262
Sûreté de fonctionnement	Propriété qui permet aux utilisateurs du système de placer une confiance justifiée dans le service qu'il leur délivre	Jean-Claude Laprie
Sûreté fonctionnelle	absence of unreasonable risk due to hazards caused by malfunctioning behaviour of E/E systems	ISO 26262
Système	A combination of interacting elements organized to achieve one or more stated purposes	ISO 15288

GLOSSAIRES

Terme Français	Équivalent Anglais
Accélération intempestive	*Unintended Acceleration*
AMDEC	*FMECA*
Analyse préliminaire des risques	*Preliminary Hazard Analysis*
Arbres de défaillance	*Fault Tree*
Architecte système	*System Architect*
Architecture Fonctionnelle	*Functional Architecture*
Architecture logique	*Logical Architecture*
Architecture opérationnelle	*Operational Architecture*
Architecture Organique	*Physical Architecture*
Besoin	*Need*
Cas d'utilisation	*Use Case*
Complexité	*Complexity*
Comportement Défectueux	*Malfunctioning Behaviour*
Comportement d'un système	*System Behavior*
Conception	*Design*
Conception d'un système	*System Design*
Conducteur	*Driver*
Constituant logique	*Logical Component*
Constituant physique	*Component*
Cycle de vie	*Life Cycle*
Défaillance	*Failure*
Domaine d'intérêt	*Domain of interest*
Erreur	*Error*
État	*State*
Évènement Dangereux, Redouté	*Hazardous Event*
Exigence	*Requirement*
Exigence allouée	*Assigned Requirement*
Exigence dérivée	*Derived Requirement*
Exigence des parties prenantes	*Stakeholder Requirement*
Exigence induite	*Induced Requirement*
Exigence technique	*Technical Requirement*
Exposition	*Exposure*

Terme Français	Équivalent Anglais
Faute	*Fault*
Fiabilité	*Reliability*
Finalité	*Finality*
Fonction	*Function*
ilités	*ilities*
Ingénierie de systèmes sûrs de fonctionnement	*Safe Systems Engineering*
Ingénierie Système	*Systems Engineering*
Ingénieurs en sdf	*Dependability Engineer*
Objectif de sûreté	*Safety Goal*
Maturité (pour un métier)	*Maturity*
Menace	*Threat*
Mode de Défaillance	*Failure Mode*
Mode de fonctionnement	*Operating Mode*
Mode opérationnel	*Operational Mode*
Modèle	*Model*
Modèle Conceptuel	*Conceptual Model*
Partie Prenante	*Stakeholder*
Phénomène Dangereux	*Hazard*
Préjudice	*Harm*
Processus de conception	*Design Process*
Processus de SdF	*Safety Process*
Processus d'ISSdF	*SSE Process*
Profil de Mission	*Profile Mission*
Propagation de défaillances	*Failure Propagation*
Sécurité immunité	*Immunity*
Sécurité innocuité	*Innocuousness*
Sécurité intégrité	*Integrity*
Situation Opérationnelle	*Operational Situation*
Sûreté	*Safety*
Sûreté de fonctionnement	*Dependability*
Sûreté fonctionnelle	*Functional Safety*
Système	*System*

Terme Anglais	Équivalent Français
Assigned Requirement	Exigence allouée
Complexity	Complexité
Component	Constituant physique
Conceptual Model	Modèle Conceptuel
Dependability	Sûreté de fonctionnement
Dependability Engineer	Ingénieurs en sdf
Derived Requirement	Exigence dérivée
Design	Conception
Design Process	Processus de conception
Domain of interest	Domaine d'intérêt
Driver	Conducteur
Error	Erreur
Exposure	Exposition
Failure	Défaillance
Failure Mode	Mode de Défaillance
Failure Propagation	Propagation de défaillances
Fault	Faute
Fault Tree	Arbres de défaillance
Finality	Finalité
FMECA	AMDEC
Function	Fonction
Functional Architecture	Architecture Fonctionnelle
Functional Safety	Sûreté fonctionnelle
Harm	Préjudice
Hazard	Phénomène Dangereux
Hazardous Event	Évènement Dangereux, Redouté
lities	ilités
Immunity	Sécurité immunité
Induced Requirement	Exigence induite
Innocuousness	Sécurité innocuité
Integrity	Sécurité intégrité
Life Cycle	Cycle de vie

Terme Anglais	Équivalent Français
Logical Architecture	Architecture logique
Logical Component	Constituant logique
Malfunctioning Behaviour	Comportement Défectueux
Maturity	Maturité (pour un métier)
Model	Modèle
Need	Besoin
Operating Mode	Mode de fonctionnement
Operational Architecture	Architecture opérationnelle
Operational Mode	Mode opérationnel
Operational Situation	Situation Opérationnelle
Physical Architecture	Architecture Organique
Preliminary Hazard Analysis	Analyse préliminaire des risques
Profile Mission	Profil de Mission
Reliability	Fiabilité
Requirement	Exigence
Safe Systems Engineering	Ingénierie de systèmes sûrs de fonctionnement
Safety	Sûreté
Safety Goal	Objectif de sûreté
Safety Process	Processus de SdF
SSE Process	Processus d'ISSdF
Stakeholder	Partie Prenante
Stakeholder Requirement	Exigence des parties prenantes
State	État
System	Système
System Architect	Architecte système
System Behavior	Comportement d'un système
System Design	Conception d'un système
Systems Engineering	Ingénierie Système
Technical Requirement	Exigence technique
Threat	Menace
Unintended Acceleration	Accélération intempestive
Use Case	Cas d'utilisation

ACRONYMES

Sigle	Développé
A2PNum	Architecture Produit / Process dans un environnement collaboratif et NUMérique
AdD	Arbre de Défaillance
AFE	Analyse Fonctionnelle Externe
AFIS	Association Française de l'Ingénierie Système
AFSS	Architecture Fonctionnelle de Système Sûr
AMDE	Analyse des Modes de Défaillance et de leurs effets
AMDEC	Analyse des Modes de Défaillance, de leurs effets et de leur criticité
APR	Analyse Préliminaire des risques
AS	Architecte Système
ASIL	Automotive Safety Integrity Level
ATESST	Advancing Traffic Efficiency and Safety through Software Technology
BDD	Block Definition Diagram
CNAM	Conservatoire national des arts et métiers
COTS	Commercial On The Shelf
CRAN	Centre de Recherche en Automatique de Nancy
D&C l'IS	Découvrir & Comprendre l'Ingénierie Système (ouvrage de référence de l'AFIS)
EAST-ADL	Electronics Architecture and Software Technologies - Architecture Description Language
eFFBD	Extended/Enhanced Functional Flow Block Diagram
EuroNCAP	European New Car Assessment Program
FAD	Functional Architecture Design
FMECA	Failure Modes, Effects and Criticality Analysis
FOTS	Function On The Shelf
FSC	Functional Safety Concept
FSR	Functional Safety Requirement
FTA	Fault Tree Analysis
GMP	Groupe MotoPropulseur
HAZOP	HAZard and OPerability study
IBD	Internal Block Diagram
INCOSE	International Council on Systems Engineering
IS	Ingénierie Système
ISBM	Ingénierie Système Basée sur les Modèles
ISSdF	Ingénierie de Systèmes Sûrs de Fonctionnement
IVTV	Intégration Vérification Transition Validation
IVVQ	Intégration Vérification Validation Qualification
LKA	Lane Keeping Assist
MBSA	Model-Based Safety Assessment/Analysis
MBSE	Model-Based System Engineering
OASIS	Ontology-based Analysis of Situation Influences on Safety
OOSEM	Object-Oriented Systems Engineering Method
O&SRD	Operational & System Requirements Definition
PAD	Physical Architecture Design
PHA	Preliminary Hazard Analysis
PHL	Preliminary Hazard List
QM	Quality Management
RA	Requirements Analysis
SafeSysE	Safety Analysis Integration In Systems Engineering
SdF	Sûreté de Fonctionnement
SEBoK	Systems Engineering Book of Knowledge
SRD	Stakeholder Requirements Definition
STAMP	Systems-Theoretic Accident Model and Processes
SysML	System Modeling Language

LISTE DES FIGURES ET DES TABLEAUX

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

Contexte — La Sûreté de Fonctionnement (SdF) est un domaine étudié de façon de plus en plus rigoureuse par les concepteurs. Si la SdF relative aux organes bénéficie de retours d'expériences sur lesquels le développement de solutions physiques peut s'appuyer, tel n'est pas le cas lorsqu'il s'agit de concevoir les architectures fonctionnelles d'un produit. L'architecte Système (AS) conceptualise et conçoit un système qui doit fournir un service selon certaines performances ; son cadre de référence, à savoir l'Ingénierie Système (IS) étant lacunaire en matière de SdF. De la sorte, les ingénieurs en SdF interviennent après l'architecture, réalisent leurs propres analyses fonctionnelles, perturbant le travail réalisé et obligeant à des boucles qui pourraient être évitées. Cependant, des normes comme l'ISO 26262:2011 obligent les bureaux d'études des constructeurs de passer du SdF constatée à une SdF véritablement conçue. Il convient de ce fait, pour les industriels, de s'intéresser de près à l'alignement entre la SdF et l'IS.

Hypothèse — Une telle transition peut s'appuyer sur différentes sources théoriques, relevant soit du domaine de l'IS, soit de celui de la SdF. L'idée sous-jacente à ce travail de thèse est de proposer un cadre, l'Ingénierie Système et la Sûreté de Fonctionnement (ISSdF), dans lequel la SdF se trouve guidée par les modèles conceptuels, les processus, les méthodes ou les outils de l'IS. L'idée est de proposer à l'AS un ensemble d'outils lui permettant de conceptualiser et de pratiquer l'ISSdF, notamment en conception amont, lorsqu'il lui faut architecturer les fonctions du produit.

Démarche — La démarche suivie dans cette thèse a consisté à dresser un état de l'art, à coopérer avec des experts, notamment d'IS, et à développer des modèles de différents types. Ces modèles ont été exemplifiés dans des cas simplifiés, par exemple dans celui de l'accélération intempestive du véhicule. Pour être plus précis, notre travail a synthétisé un certain nombre de normes existantes (ISO 15288, IEEE 120, ISO 26262:2011), de guides opérateurs (SEBok), et de modèles conceptuels proposés dans la littérature (A2Pnum, Attest, Taofefinua). Enfin, notre démarche d'ensemble s'est alignée sur celle de l'IS. Pour chacun des livrables attendus, nous avons défini un cahier des charges de l'attendu, puis une analyse de l'existant, une proposition de solution, une exemplification, et une validation de la solution exposée.

Résultats — Nous avons obtenu quatre résultats. Le premier est un modèle conceptuel de l'ISSdF. Ce modèle conceptuel, découpé en trois vues (vues opérationnelle, fonctionnelle, organique), permet de définir les concepts et les liens entre le domaine de l'IS et celui de la SdF. L'ensemble présente une forte cohérence sémantique. Ce modèle a permis de définir ensuite un macoprocessus d'ISSdF cohérent avec les processus prescrits par l'ISO 15288:2008 et conforme aux livrables de l'ISO 26262:2011. Le macroprocessus présenté permet de définir l'architecture d'un système sûr. Pour cela, nous l'avons scindé en trois phase de conception (concept, conception préliminaire et conception détaillée) ainsi que cinq points de vue (opérationnel, fonctionnel externe, fonctionnel interne, organique, géométrique). Plusieurs activités spécifiques sont présentes dans le macroprocessus. Il a fallu ensuite proposer des méthodes pour les réaliser au mieux. Ainsi, nous avons développé *une méthode qui concerne l'analyse des risques en vue externe*. Vu les limites de méthodes comme l'Analyse Préliminaire des Risques (APR) vis-à-vis des livrables demandés par l'ISO 26262:2011 (Hazard Analysis and Risk Assessment), nous avons proposé une nouvelle méthode qui permet de définir les Safety Goals (exigence de sûreté de haut niveau) avec un ASIL (niveau d'intégrité de sûreté) associé pour les scénarios critiques. De plus, nous avons développé une méthode *permettant l'analyse des architectures fonctionnelles d'un point de vue dysfonctionnel* pour définir les *Functional Safety Requirements*. Cette méthode repose sur des mécanismes de propagation de défaillance afin de déterminer et de visualiser le comportement dysfonctionnel pour l'architecte système. Ces deux méthodes ont été appliquées dans le cas *de l'accélération intempestive du véhicule* dont une cause provient de la fonction *Alimenter la combustion en carburant*. Nous n'avons pas développé de nouvel outil pour encoder ces méthodes. Nous avons cependant proposé les références SysML.

Perspectives — Nos résultats ouvrent au moins deux types de perspectives. Le premier type concerne la mise à l'épreuve des modèles, des processus et des méthodes de l'ISSdF sur un véhicule correspondant à l'état de l'art automobile, à savoir le véhicule autonome. Celui-ci représente une innovation de rupture, et il s'agit d'évaluer si nos résultats sont robustes lorsqu'il convient de développer un produit aussi complexe La seconde perspective concerne l'ISSdF organique. Nos travaux se sont focalisés sur la conception amont, notamment l'architecture fonctionnelle. Il convient de ce fait d'évaluer si les éléments proposés permettent d'aider le concepteur en charge de l'architecture organique, voire celui gérant les lignes de produit.

ANNEXE 1
CADRE SYSTÉMIQUE

Afin que nous soyons compris dans ce manuscrit, nous avons fait plusieurs hypothèses épistémologiques que nous allons détailler dans ce chapitre. Les définitions qui seront données dans cette partie seront considérées comme valables pour tout le manuscrit. Celles-ci ont été regroupées en branches comme cela est représenté dans cette carte mentale suivante.

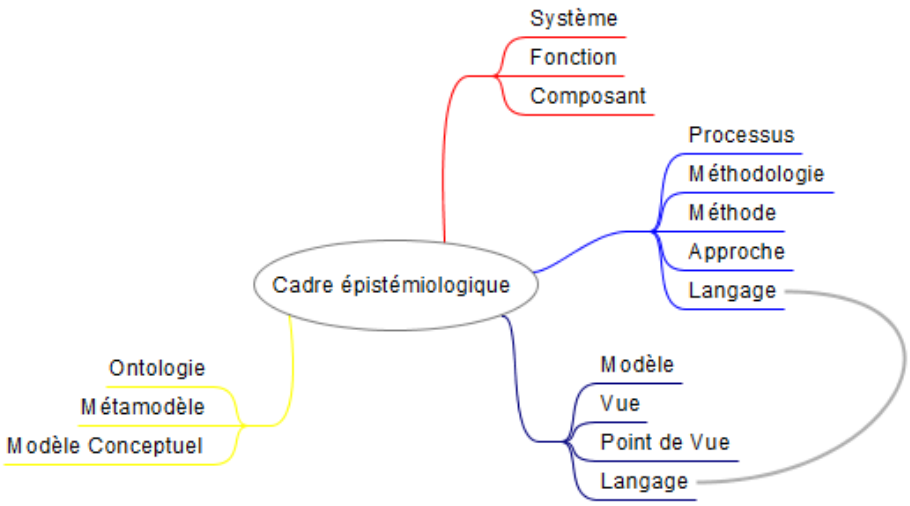


Figure 54: Cadre épistémologique

Système - Le système est, si l'on se réfère à la norme ISO 15288:2008, une combinaison d'éléments en interaction organisés pour atteindre un ou plusieurs objectifs énoncés.[32] On peut étendre cette définition à : *Un système est un ensemble d'éléments humains ou matériels en interaction entre eux et avec l'environnement selon un certain nombre de principes ou règles, intégré pour rendre à son environnement les services correspondants à sa finalité.*

Fonction & tâche - Comme l'expose l'EIA 632 (EIA 632, 2003), une fonction est une tâche, une action ou une activité effectuée pour atteindre un résultat souhaité [33]. Notre définition sera donc celle de (AFIS, 2007) : « *Action exécutée pour atteindre un résultat attendu. Dans un système une fonction transforme des flux entrants en flux sortants* ».

[32] *System: A combination of interacting elements organized to achieve one or more stated purposes.*
[33] *Function: A task, action, or activity performed to achieve a desired outcome.*

Composant - Concernant le composant, l'IEEE 1220 (IEEE 1220, 2005) propose cette définition que nous adopterons : « *Les composants complexes représentent des éléments du système qui sont composés de matériels, logiciels, et / ou les humains, qui sont reconnaissables en termes de processus de cycle de vie (comment concevoir, tester, fabriquer, soutien, etc… est connu)* [34] *(Complex components represent system elements that are composed of hardware, software, and/or humans, which are recognizable in terms of life cycle process (how to design, test, produce, support, etc., is known).* Un système sans composant n'est alors qu'un spectre et un système sans fonction est un cadavre.

Modèle - Notre définition du modèle est celle énoncée par Dori (Dori, 2002) : *Un modèle est une abstraction d'un système visant à comprendre, communiquer, expliquer, ou de concevoir différents aspects de ce système* [35].

Vues - Pour les notions de vue et de point de vue, nous nous sommes basés sur la norme IEEE 1471 (IEEE 1471, 2000). Nous avons donc comme définitions

Vue : *Une représentation de tout un système à partir de la perspective d'un ensemble connexe de préoccupations.*[36]

Point de vue: *Une spécification des conventions pour la construction et l'utilisation d'une vue. Un motif ou un patron permettant d'élaborer des vues individuelles en établissant les objectifs et le public pour une vue et les techniques de création et d'analyse.*[37]

De plus, nous prenons la définition de (Bourey, 2014) pour un langage :

Ensemble de construits (constructs) i.e. briques de base possédant une syntaxe & une sémantique pour représenter un artefact (objet, message, connaissance,…)

Pour conclure, nous considérons que nous avons un ensemble de vues cohérentes et articulées qui forme le modèle du système. Ces vues seront déterminées par la définition de point de vue. Les vues du modèle seront représentées par un ou plusieurs diagrammes exprimés selon un certain langage.

Processus, méthodes, outil - Pour le lien entre processus, méthode et outil, on peut se référer à la Figure 7. Le processus est un ensemble d'activités liées permettant de transformer des entrées en sorties[38] (ISO 15288, 2015). La méthodologie comme son étymologie l'indique est le discours sur la méthode, une réflexion. La méthode est l'art de trouver le bon chemin. Elle définit donc la manière efficace de réaliser les activités pour le processus.

---

[34] *Complex components represent system elements that are composed of hardware, software, and/or humans, which are recognizable in terms of life cycle process (how to design, test, produce, support, etc., is known).*
[35] *an abstraction of a system, aimed at understanding, communicating, explaining, or designing aspects of interest of that system*
[36] *View: A representation of a whole system from the perspective of a related set of concerns.*
[37] *Viewpoint : A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for the creation and analysis.*
[38] *set of interrelated or interacting activities that transforms inputs into outputs*

Modèle conceptuel - Selon Jean-Pierre Bourey[39], le modèle conceptuel est un modèle qui a pour objectif de caractériser et définir les relations entre ces concepts pour un domaine donné. Celui-ci peut être représenté par un diagramme de classes UML, un modèle entité-association merisien, etc. Nous proposons comme exemple celui-ci-dessous d'un établissement de golf.

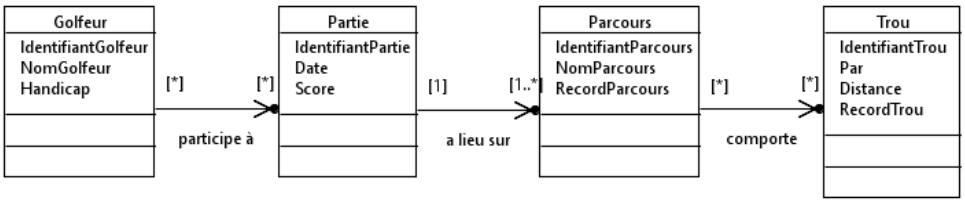


Figure 55 : Modèle conceptuel simplifié d'un établissement de golf

Selon Jean-Pierre Bourey[40], un méta-modèle est également un modèle, mais qui a pour objectif de modéliser un langage de modélisation. Le méta-modèle d'UML présenté ci-dessous décrit par exemple le langage correspondant. On retrouve dans un métamodèle, la syntaxe abstraite du langage reprenant les concepts et les relations entre ceux-ci, la syntaxe concrète définissant la représentation de ces concepts, des règles de bonne construction et la sémantique des concepts. Un langage est formel quand sa sémantique est formelle. On peut ainsi indiquer qu'un modèle est conforme à son méta-modèle. Pour faire la distinction avec le modèle conceptuel, ce dernier ne comporte pas de syntaxe concrète.

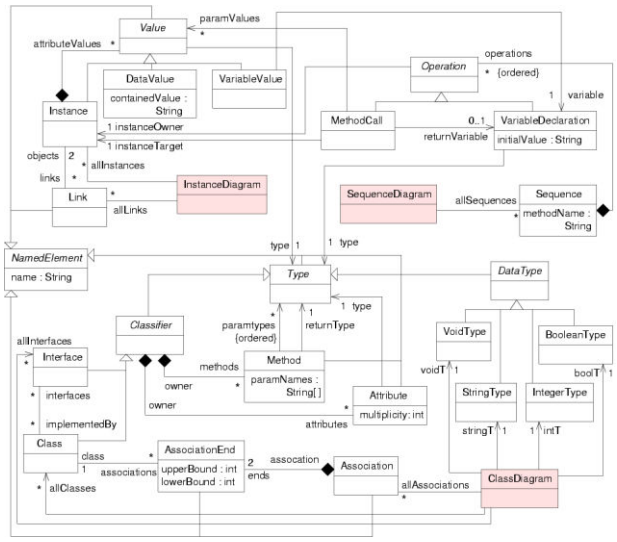


Figure 56 : Méta-modèle simplifié d'UML

[39] *Définition adaptée des supports de cours de Jean-Pierre Bourey*
[40] *Définition adaptée des supports de cours de Jean-Pierre Bourey*

Selon Gruber dans (Gruber, 1993), *An ontology is an explicit specification of a conceptualization*. Mais comme, il le precise dans (Gruber, Liu, & Tamer, Encyclopedia of Database Systems, 2009), *In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application.* De plus, selon Gruber, il existe 5 critères de conception pour une ontologie qui sont : « *Clarity, Coherence, Extendibility, Minimal encoding bias & Minimal ontological commitmen*t ». Selon Jean-Pierre Bourey[41], l'objectif d'une ontologie est formaliser les concepts, relations et instances d'un domaine. Si l'on compare l'ontologie aux deux précédents, celle-ci ne comporte pas de syntaxe concrète. Contrairement au modèle conceptuel et au méta-modèle, l'ontologie comporte toutefois des instances. Comme cela est évoqué dans l'exemple ci-dessous, il est possible d'instancier les différents concepts.

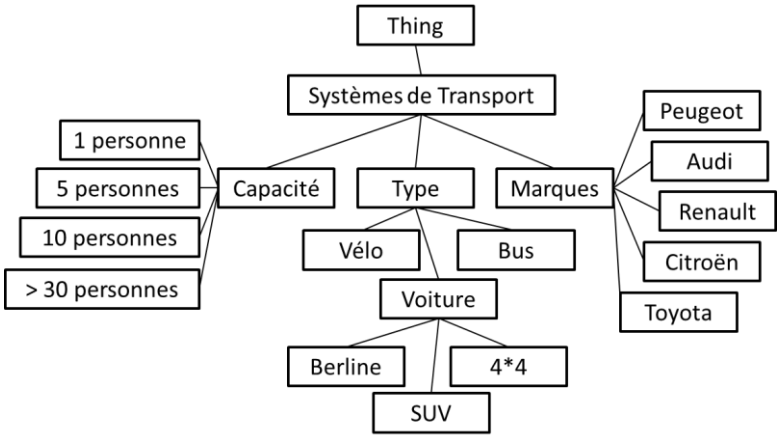


Figure 57 : Ontologie partielle des systèmes de transport

Afin de réaliser une synthèse du cadre épistémologique de la thèse, le choix a été fait d'exemplifier chacun des termes en se référant à l'automobile.

[41] *Définition extraite des supports de cours de Jean-Pierre Bourey*

Tableau 31 : Synthèse du cadre épistémologique

Concept	Exemple
Système	Véhicule Automobile
Fonction	Accélérer
Composant	Réservoir
Modèle	Modèle du véhicule
Vue	Vue Fonctionnelle Externe
Point de vue	Point de vue de l'Architecte Système
Langage	SysML
Processus	Analyse des exigences techniques
Méthodologie	ISSdF
Méthode	Utiliser un patron pour enrichir dysfonctionnellement une architecture fonctionnelle
Approche	Approche statique *vs*. Dynamique, structurelle *vs*. comportementale
Modèle Conceptuel	Modèle conceptuel de la pensée système
Méta-modèle	Méta-modèle d'UML
Ontologie	Ontologie des moyens de transport

ANNEXE 2
HISTORIQUE DE L'INGÉNIERIE SYSTÈME

Années 1940/1950 : Naissance de l'Ingénierie Système - En parallèle à l'apparition de l'informatique, le terme « Ingénierie Système » apparait pour la première fois dans les années 1940. C'était pour le premier système de défense de l'espace aérien aux États-Unis par le laboratoire Bell. C'est encore dans le domaine de la défense que l'Ingénierie Système se développe avec le Projet SAGE-ADS qui sera le premier projet avec une approche d'Ingénierie Système à grande échelle.

Années 1960 : La guerre froide et la course aux étoiles, engrais pour l'Ingénierie Système - Le programme Apollo, projet ayant pour objectif d'envoyer un homme sur la Lune, sera le projet qui mettra en lumière l'Ingénierie Système. La complexification des systèmes militaires va également avoir le besoin de formaliser l'IS sous forme de normes (MIL-STD 499) et d'ouvrages (Methodology for engineering of Systems par A. Hall). Les concepts de spécifications des exigences, de gestion des interfaces, de respects des jalons sont déjà connus.

Années 1990/2000 : Internationalisation de l'Ingénierie Système - Durant les années 1980, les systèmes se complexifient comme dans l'automobile avec l'apparition des calculateurs à injection, des bus CAN, etc. Cette complexité croissante va provoquer un besoin dans des nouvelles méthodes d'ingénierie comme l'Ingénierie Système. Celle-ci va se diffuser dans différent secteurs et s'internationaliser. Des normes d'organismes internationaux prennent la suite des standards militaires américains comme l'IEEE 1220, l'EIA 632 ou l'ISO 15288. La NCOSE, conseil faisant la promotion de l'IS aux États-Unis devient internationale en 1995 avec l'INCOSE. Cela sera suivi en France en 1998 par la création de sa branche française, l'AFIS. PSA Peugeot Citroën est l'un des fondateurs de l'AFIS et publie en interne des guides méthodologiques comme « Guide démarche d'ingénierie système automobile ». Des langages et outils sont créés afin de supporter ces nouvelles méthodes d'ingénierie comme Core ou SysML.

Aujourd'hui : où en est l'IS ? - De nos jours, l'Ingénierie Système est utilisé dans de nombreuses industries. Des projets actuels comme AGESYS regroupent autour de l'Ingénierie Système des groupes comme Airbus, Alstom, Renault, Snecma, Thales ou PSA Groupe. L'enjeu est désormais de la banaliser, par exemple en proposant des cursus didactiques adaptés.

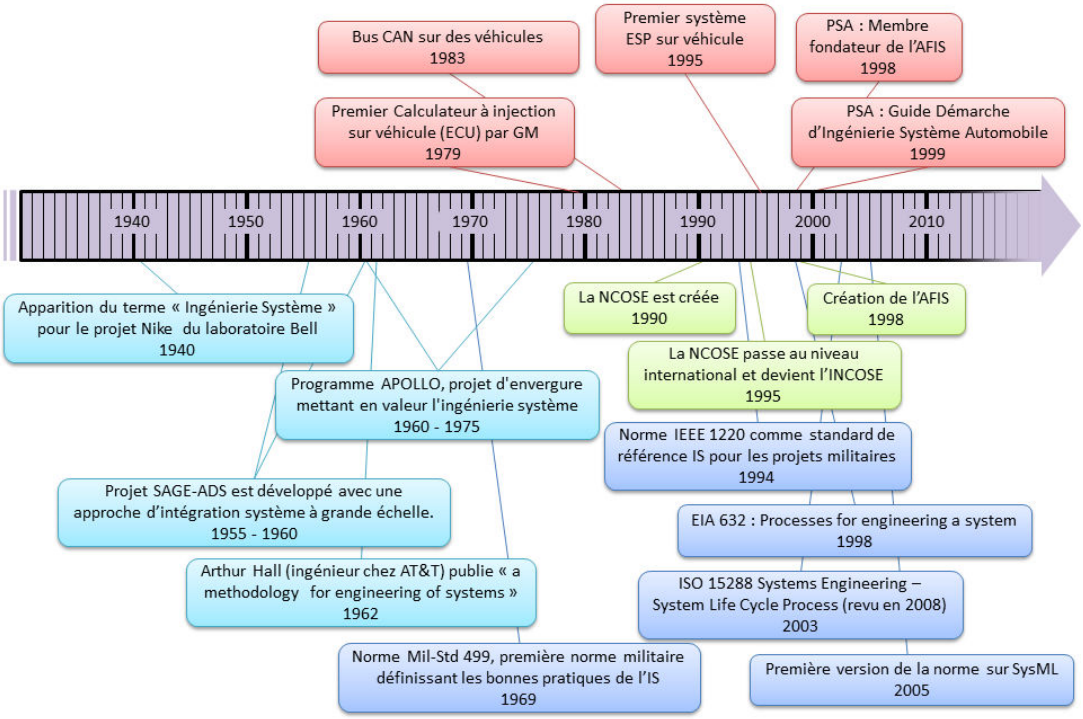


Figure 58 : Récapitulatif de l'historique de l'ingénierie système

ANNEXE 3
CADRE DE L'EXEMPLE

Dans ce mémoire, nous avons fait le choix de prendre un exemple concret afin de réaliser une première vérification de nos propositions et pour faciliter la compréhension de celles-ci lors de la lecture de ce mémoire. Nous présenterons dans cette annexe quels ont été les critères de choix, une présentation puis une justification de notre exemple par rapport à nos besoins. Cela permettra donc de contextualiser cet exemple.

Tableau 32 : Critères de choix pour l'exemple

Exigence	Libellé
Exi. 1	L'exemple doit être assez simple pour être compréhensible.
Exi. 2	L'exemple doit être assez complexe pour vérifier son application.
Exi. 3	L'exemple doit être multi physique pour aborder un cas assez général et non se limiter à de la mécanique ou du contrôle/commande.
Exi. 4	L'exemple doit être un support pour échanger avec des collaborateurs PSA.
Exi. 5	L'exemple ne doit pas être trop récent pour ne pas avoir de problèmes de diffusion.
Exi. 6	L'exemple doit être un cas qui a posé des problèmes de sûreté à une marque automobile.

L'exemple choisi concerne l'accélération intempestive d'un véhicule. Ce cas correspond à une non-diminution de l'accélération du véhicule lors du relâchement de la pédale d'accélération (à ne pas confondre avec l'absence de freinage).

L'accélération en carburant est un exemple avec un bon niveau de complexité. En effet, il s'agit d'un exemple connu en interne et également assez facilement modélisable pour échanger scientifiquement sur le sujet. Prenant en compte de la mécanique, du contrôle, du pneumatique, il permet d'aborder un exemple multiphysique. En effet, au cours de la dernière décennie, de nombreux articles de presse sont parus concernant l'accélération intempestive du véhicule (*Sudden Unintended Acceleration*) avec de dramatiques conséquences (89 morts et 52 blessures graves ont été attribués à ce cas) comme cela a été rapporté dans (Safety Research & Strategies, Inc., 2010). Des études récentes ont publié des causes potentielles. Elles proposent de possibles explications pour chaque cas :

- une défaillance de la pédale d'accélération (Toyota, 2010) ;
- une erreur de dimensionnement des pneumatiques (en raison du système de contrôle électrique de stabilité (ESP) sur un véhicule à propulsion augmentant la vitesse des roues arrières) (Belt, 2015-1) ;
- des erreurs dans les systèmes de contrôle de véhicules avec batteries (en raison de variations de l'état de charge de la batterie et d'erreurs CPU) (Belt, 2012) ;

- une condition d'emballement dans le système de contrôle du moteur dans un véhicule électrique (en raison d'un pic de tension négatif sur la ligne d'alimentation de la batterie) (Belt, 2015-2).

Cette dernière explication est similaire à une l'explication de Belt concernant une condition d'emballement dans le système de contrôle électronique de l'accélérateur des moteurs à essence, ce qui expliquerait les cas d'accélérations intempestives pour les véhicules ayant des systèmes d'accélération comportant de l'électronique. Selon cet expert, sans aucune enquête spécifique, les nouvelles fonctions électroniques vont augmenter le taux d'incident pour l'accélération intempestive dans l'avenir. Ainsi, il est utile de déterminer le *Safety Goal* de ce phénomène dangereux.

ANNEXE 4
MODÉLISATION PROLOG

La Programmation Logique permet un parcours automatisé et exhaustif d'un espace de recherche décrit sous forme arborescente par la programmation ; le résultat obtenu est donc un ensemble de solutions. Il est possible d'augmenter l'efficacité du parcours de l'espace de recherche par l'introduction de contraintes spécifiques au problème décrit, généralement sous forme de logique ensembliste. La nature du problème posé ici ne justifie pas ce dernier type de programmation, qui s'avère d'ailleurs très délicat à décoder ensuite.

La programmation par contrainte s'appuie sur les problèmes de satisfaction de contraintes ou CSPs (*Constraint Satisfaction Problems*) et sur des algorithmes de propagation de contraintes. Les CSPs permettent de modéliser de la connaissance et de raisonner sur celle-ci afin de trouver l'ensemble des solutions compatibles avec un problème courant. Les premiers problèmes de satisfaction de contraintes ont été définis par Ugo Montanari (Montanari, 1974).

Les CSPs sont définis comme un triplet (V ; D ; C) où :

$V = \{v_1, v_2, \ldots, v_k\}$ est un ensemble fini de k variables,

$D = \{d_1, d_2, \ldots, d_k\}$ est un ensemble fini de domaines de définition de ces variables,

$C = \{c_1, c_2, \ldots, c_k\}$ est un ensemble fini de contraintes portant sur ces variables.

Une solution d'un problème de satisfaction de contraintes est une instanciation de toutes les variables respectant toutes les contraintes.

Il existe différents types de CSPs (Vareilles, 2005) : les CSPs discrets (variables symboliques et entières, contraintes discrètes), les CSPs continus (variables réelles, contraintes numériques) et les CSPs mixtes (variables discrètes et numériques, contraintes mixtes). Dans notre cas, nous utilisons des CSP discrets.

Nous avons décidé d'utiliser le langage Prolog pour réaliser une maquette informatique permettant d'identifier les chemins de propagation critiques. Le langage de programmation logique Prolog, introduit au début des années 70 par A. Colmerauer et R. Kowalski, est adapté à cette modélisation car il ne manipule que des relations et permet de définir de nouvelles relations à partir de relations prédéfinies (les contraintes de base) par des clauses logiques. Prouver la consistance d'un CSP (satisfiabilité) est un problème NP-complet. Dans notre cas, la taille des problèmes traités restant limitée à chaque strate système, le temps de calcul n'est pas un critère important.

La Programmation Logique repose sur la logique des prédicats du premier ordre (Cori & Lascar, 1993). Un programme est constitué d'un ensemble de faits et d'un ensemble de règles. Les faits sont l'équivalent des données du problème que l'on cherche à résoudre. Par exemple,

le fait : *pere(Jean,Jacques) ->;* modélise le fait que Jean soit le père de Jacques. Pour plus de lisibilité, rajoutons les deux autres faits suivants à notre exemple :

pere(Jacques,Jules) ->;

pere(Jules,Julien) ->;

Les règles modélisent la logique générale du phénomène étudié en décrivant la décomposition d'un but de haut niveau en sous-buts de niveau inférieur. Par exemple, la règle :

grand-pere(X,Y)->pere(X,Z) pere(Z,Y);

permet de déduire que X est le grand-père de Y si X est le père de Z et que Z est le père de Y.

Une décomposition est localement terminée lorsque l'un des éléments de décomposition du but est un fait. La particularité de la programmation logique est qu'il peut y avoir plusieurs règles de même nom pour décrire un but dont chaque décomposition est différente. Par exemple, on pourrait rajouter une seconde règle à notre calcul de grand-père :

grand-pere(X,Y)->pere(X,Z) mere(Z,Y);

sous réserve, bien sur, de définir de nouveaux faits du type :

mere(A,B)-> ; .

La résolution d'un problème se traduit par l'interrogation du programme en lançant des buts de haut niveau ; par exemple, questionner le programme avec le but *grand-pere(Jacques , B)* produira le résultat *{B=Julien}*. En revanche, questionner le programme avec le but *grand-pere(A , B)* produira les deux résultats *{A=Jean, B=Jules}* et *{A=Jacques, B=Julien}* car il y a deux solutions au problème posé.

Ce type de programmation est généralement hautement récursif et fournit non pas une mais un ensemble de solutions (la règle indiquée ci-dessus s'appelle elle-même). Par exemple, la règle permettant de calculer les descendants X d'une personne Y serait modélisée par les deux prédicats ci-dessous :

descendant(X,Y)->pere(Y,X) ;

descendant(X,Y)->pere(Y,Z) descendant(Z,X) ;

La première règle est appelée « clause d'arrêt » tandis que la seconde traduit la récursivité du phénomène décrit.

L'explosion combinatoire, due au parcours de l'espace de recherche, est toujours un élément très délicat à maîtriser lors de la programmation. Par exemple, dans le cas présenté, le lancement du but *descendant(X,Y)* ; fournirait déjà 6 solutions ! Et il n'est pas rare qu'un programme tourne indéfiniment du fait du lancement d'un but mal défini ! Il est possible de maîtriser ce phénomène d'explosion combinatoire par une programmation judicieuse et par l'emploi du « *cut* » (couper, noté « ! ») et du « *fail* ». Le « *cut* » exprime que dès la première réussite de l'évaluation d'un sous-but (les sous-buts placés avant le cut dans une règle), le reste de l'espace de recherche correspondant n'est plus évalué (le programmeur se limite à la première solution partielle qui est trouvée). Quant à l'élément « *fail* » (échoue), qui ne fait appel à aucun but, il permet de faire échouer intentionnellement la recherche d'un but.

Ainsi, pour donner un exemple simple, le complément (généralement programmé « not »), s'écrit :

not(a)-> a ! fail ;

not(a) -> ;

Il s'explique de la manière suivante : si l'élément « a » existe, alors la première règle commence par réussir (a), puis interdit de chercher une autre solution (!) puis échoue (fail).

Si l'élément « a » n'existe pas, alors la première règle échoue (recherche de « a » impossible), et c'est la seconde règle qui est évaluée à son tour ; celle-ci réussit alors systématiquement. Donc si « a » existe, « not » échoue et si « a » n'existe pas, alors « not » réussit. Le complément est donc implémenté.

Grâce à ces deux règles de base (« cut » et « fail »), il est possible de maîtriser l'explosion combinatoire engendrée par la recherche d'un but.

Au cours des années 1980 est apparue la programmation logique sous contraintes. Aux buts ont été ajoutés des équations logiques supplémentaires permettant de réduire l'espace de recherche engendré par les buts (Fagès, 1996). Si l'intérêt de l'approche est évident, le résultat est cependant une programmation difficile à relire pour toute personne qui n'a pas écrit le programme.

Le problème qui est posé ici se prête bien à une résolution de type programmation en logique des prédicats du premier ordre car :

L'espace de recherche à parcourir se présente selon une arborescence de type « graphe ET / OU ». Le parcours de l'espace de recherche fournit généralement plusieurs solutions pour un même but à résoudre. Le langage que nous avons retenu pour nos expérimentations est le Prolog II+ de Prologia.

Comme nous l'avons expliqué dans le mémoire, le cas d'étude a été décrit par un ensemble de versions qui est une modélisation de l'exemple choisi dans le mémoire.

version(1,1.nil)->;
version(2,<1,2.3.4>.nil)->;
version(3,<1,2.3.5.6>.nil)->;

transformation(1,"générer l'énergie mécanique",1.2.nil,3.nil,1)->;
transformation(2,"alimenter la combustion en carburant",1.nil,4.nil,2)->;
transformation(3,"alimenter la combustion en air",2.nil,5.nil,3)->;
transformation(4,"réaliser la combustion",4.5.nil,3.nil,4)->;

La version 1 peut être illustrée par cette figure ci-dessous.

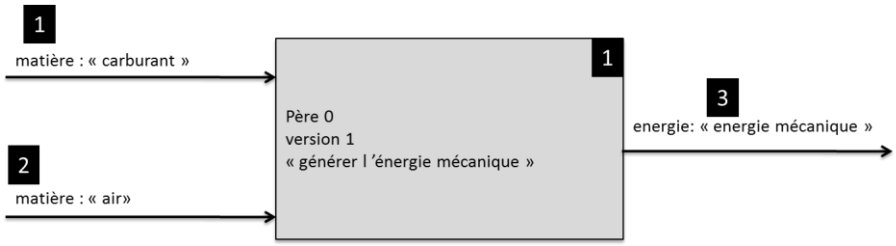


Figure 59 : Version 1 du cas d'étude

Celle-ci peut donc être raffinée par la version 2 qui peut être illustrée ci-dessous en décomposant la transformation 1 par les transformations 2, 3 et 4.

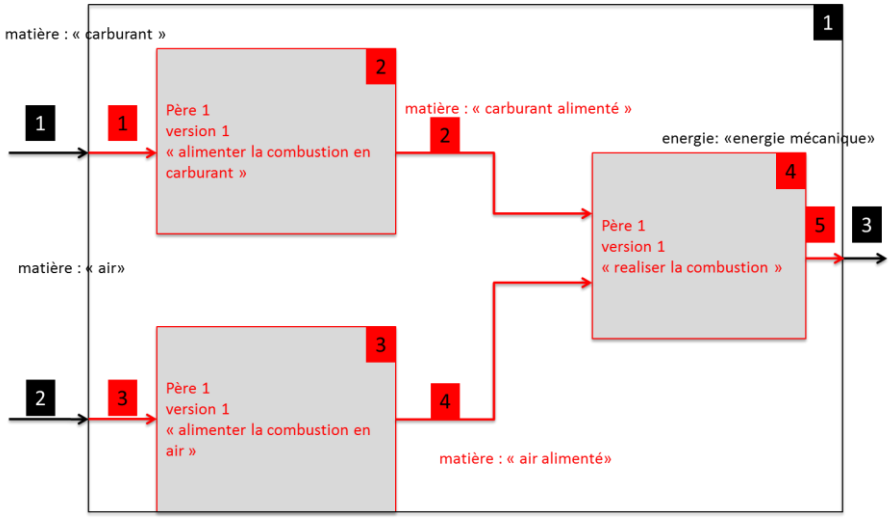


Figure 60 : Version 2 du cas d'étude

Cette version peut être également raffinée en décomposant la transformation 2 par les transformations 5 et 6.

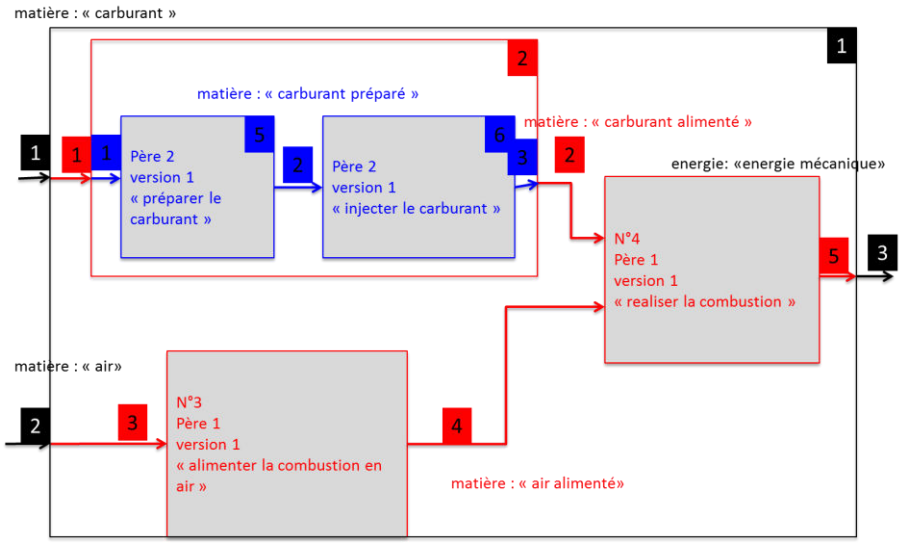


Figure 61 : Version 3 du cas d'étude

Les différents programmes réalisés avec l'aide de Christophe Perrard conformément à notre cahier des charges sont détaillés ci-dessous :

```
"============================================="
"menu fichier"
"============================================="
```
"Création de l'architecture de haut-niveau"
```
go1->
        menu_fichier_nouveau("generer l energie mecanique",standard,
                <1,matiere,"carburant">.<2,matiere,"air">.nil,
                <3,energie,"energie mecanique">.nil)
        menu_fichier_tout_lister
        ;
```
"raffinement de l'architecture par décomposition"
```
go21->
        menu_fichier_raffiner_transformation(1,1,
                <1,"alimenter la combustion en carburant",standard,1.nil,2.nil>.
                <2,"alimenter la combustion en air",standard,3.nil,4.nil>.
                <3,"realiser la combustion",standard,2.4.nil,5.nil>.nil,
                <1,_,_,1>.<2,matiere,"carburant_alimenté",-1>.<3,_,_,2>.<4,matiere,"air_alimenté",-
1>.<5,_,_,3>.nil)
        menu_fichier_tout_lister
        ;
```
"raffinement de la fonction alimenter la combustion en carburant par deux alternatives go22 et go23"
```
go22->
        menu_fichier_raffiner_transformation(2,1,
                <1,"préparer le carburant",standard,1.nil,2.nil>.
                <2,"injecter le carburant",standard,2.nil,3.nil>.nil,
                <1,_,_,6>.<2,matiere,"carburant préparé",-1>.<3,_,_,4>.nil)
        menu_fichier_tout_lister
        ;
go23->
        menu_fichier_raffiner_transformation(2,2,
                <1,"préparer le carburant",standard,1.nil,2.nil>.
                <2,"injecter le carburant",standard,2.nil,3.nil>.
                <2,"controler le carburant",safety,3.nil,4.nil>.nil,
                <1,_,_,6>.<2,matiere,"carburant préparé",-1>.<3,_,_,-1>.<4,_,_,4>.nil)
        menu_fichier_tout_lister
        ;
"---------------------------------------------"
```
"Modélisation des comportements"
```
"flux(n_umero,t_ransformation_pere,v_ersion,no_interne,no_externe,nature,description,liste_etats)->;"
flux(1,0,1,1,-1,matiere,"carburant",absent.intempestif.nominal.nil)->;

transformation(1,0,1,1,"bof bof",standard,1.2.nil,3.nil,1)->;
comportement(1,standard,nominal.nil,<0,0>,nominal.nil)->;
"---------------------------------------------"

menu_fichier_nouveau(d_escription,t_ype,f1,f2)->
        menu_fichier_tout_effacer
        fail;
menu_fichier_nouveau(_,_,l_f1,_)->
        element(f_lux,l_f1,vrai)
        idem(f_lux,<n_o_interne,t_ype,d_escription>)
        creer_flux(n_umero,0,1,n_o_interne,-1,t_ype,d_escription,absent.intempestif.nominal.nil)
        fail;
menu_fichier_nouveau(_,_,_,l_f2)->
        element(f_lux,l_f2,vrai)
        idem(f_lux,<n_o_interne,t_ype,d_escription>)
        creer_flux(n_umero,0,1,n_o_interne,-1,t_ype,d_escription,absent.intempestif.nominal.nil)
```

```
        fail;
menu_fichier_nouveau(d_escription,t_ype,l_f1,l_f2)->
        extraire_numeros_internes(l_f1,f1)
        extraire_numeros_internes(l_f2,f2)
        creer_transformation(n_umero,0,1,1,d_escription,t_ype,f1,f2,c_omportement)
        fail;
menu_fichier_nouveau(_,_,_,_)->
        incrementer_variable(nb_max_modification,n_umero)
        fail;
menu_fichier_nouveau(_,_,_,_)->
        val(nb_max_transformation,n_b_max_transformation)
        assert(max_transformation(n_b_max_transformation),nil)
        val(nb_max_flux,n_b_max_flux)
        assert(max_flux(n_b_max_flux),nil)
        val(nb_max_modification,n_b_max_modification)
        assert(max_modification(n_b_max_modification),nil)
        fail;
menu_fichier_nouveau(_,_,_,_)->
        val(nb_max_transformation,n_b_max_transformation)
        assert(historique_modifications(n_b_max_transformation,0,1.nil),nil)
        assert(version_courante(n_b_max_transformation.nil),nil)
        fail;
menu_fichier_nouveau(_,_,_,_)->;


extraire_numeros_internes(nil,nil)->!;
extraire_numeros_internes(<n,_,_>.l_f1,n.f1)->
        extraire_numeros_internes(l_f1,f1);
"-------------------------------------------"
"menu_fichier_tout_effacer"
"-------------------------------------------"
menu_fichier_tout_effacer->
        efface(transformation,9)
        efface(comportement,5)
        efface(historique_modifications,3)
        efface(max_modification,1)
        efface(version_courante,1)
        efface(flux,8)
        efface(max_transformation,1)
        efface(max_flux,1)
        fail;
menu_fichier_tout_effacer->
        assign(nb_max_flux,0)
        assign(nb_max_transformation,0)
        assign(nb_max_modification,0)
        fail;
menu_fichier_tout_effacer->;

"-------------------------------------------"
"menu_fichier_tout_lister"
"-------------------------------------------"

menu_fichier_tout_lister->
        outml("-------------------------------------------")
        outml("*** transformation : numero, transformation_pere, version, no_interne, description, type,
liste_flux_entree, liste_flux_sortie, comportement")
        list(transformation)
        list(comportement)
        list(max_transformation)
        outml("*** flux : numero, transformation_pere, version, no_interne, no_externe, nature, description,
liste_etats")
        list(flux)
```

```
        list(max_flux)
        list(historique_modifications)
        list(max_modification)
        list(version_courante)
        fail;
menu_fichier_tout_lister->;
```

“Règle permettant de raffiner les transformations”

```
"-----------------------------------------------"
"raffiner_transformation(t_ransformation_pere,v_ersion,l_iste_nouvelles_transformations,l_iste_nouveaux_flux)
->;"
"les entrées et les sorties restent les mêmes - on décompose une transformation"
"-----------------------------------------------"

menu_fichier_raffiner_transformation(t_ransformation_pere,v_ersion,_,l_iste_nouveaux_flux)->
        element(n_ouveau_flux,l_iste_nouveaux_flux,vrai)
        idem(n_ouveau_flux,<n_o_interne,n_ature,d_escription,n_o_externe>)
        idem(n_o_externe,-1)
        idem(l_iste_etats,absent.intempestif.nominal.nil)
        creer_flux(n_umero,t_ransformation_pere,v_ersion,n_o_interne,n_o_externe,n_ature,d_escription,l_iste
_etats)
        fail;
menu_fichier_raffiner_transformation(t_ransformation_pere,v_ersion,_,l_iste_nouveaux_flux)->
        element(n_ouveau_flux,l_iste_nouveaux_flux,vrai)
        idem(n_ouveau_flux,<n_o_interne,n_ature,d_escription,n_o_externe>)
        val(sup(n_o_externe,0),1)
        creer_flux(n_umero,t_ransformation_pere,v_ersion,n_o_interne,n_o_externe,n_ature,d_escription,l_iste
_etats)
        fail;
menu_fichier_raffiner_transformation(t_ransformation_pere,v_ersion,l_iste_nouvelles_transformations,_)->
        element(n_ouvelle_transformation,l_iste_nouvelles_transformations,vrai)
        idem(n_ouvelle_transformation,<n_o_interne,d_escription,t_ype,l1,l2>)
        creer_transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l1,l2,c_o
mportement)
        fail;
menu_fichier_raffiner_transformation(t_ransformation_pere,v_ersion,l_iste_nouvelles_transformations,_)->
        efface(max_transformation,1)
        efface(max_flux,1)
        efface(max_modification,1)
        fail;
menu_fichier_raffiner_transformation(t_ransformation_pere,v_ersion,l_iste_nouvelles_transformations,_)->
        val(nb_max_transformation,n_b_max_transformation)
        assert(max_transformation(n_b_max_transformation),nil)
        val(nb_max_flux,n_b_max_flux)
        assert(max_flux(n_b_max_flux),nil)
        fail;
menu_fichier_raffiner_transformation(t_ransformation_pere,v_ersion,l_iste_nouvelles_transformations,l_iste_no
uveaux_flux)->
        incrementer_variable(nb_max_modification,n_umero)
        fail;
menu_fichier_raffiner_transformation(t_ransformation_pere,v_ersion,l_iste_nouvelles_transformations,l_iste_no
uveaux_flux)->
        retrouver_numeros_globaux(t_ransformation_pere,v_ersion,l_iste_nouvelles_transformations,l_iste_nos
)
        val(nb_max_modification,n_b_max_modification)
        assert(max_modification(n_b_max_modification),nil)
        assert(historique_modifications(n_b_max_modification,t_ransformation_pere,l_iste_nos),nil)
        fail;
menu_fichier_raffiner_transformation(t_ransformation_pere,v_ersion,l_iste_nouvelles_transformations,l_iste_no
uveaux_flux)->
        val(nb_max_modification,n_b_max_modification)
        retract(version_courante(l),nil)
```

```
        assert(version_courante(n_b_max_modification.l),nil)
        fail;
menu_fichier_raffiner_transformation(t_ransformation_pere,v_ersion,l_iste_nouvelles_transformations,l_iste_no
uveaux_flux)->;

retrouver_numeros_globaux(_,_,nil,nil)->!;
retrouver_numeros_globaux(t_ransformation_pere,v_ersion,<n_o_interne,_,_,_>.l_t1,n_o_global.t1)->
        transformation(n_o_global,t_ransformation_pere,v_ersion,n_o_interne,_,_,_,_)
        retrouver_numeros_globaux(t_ransformation_pere,v_ersion,l_t1,t1);
"-------------------------------------------"
"menu_fichier_definir_comportement_transformation(t_ransformation)->;"
```
"on balaye toutes les possibilites d'entrée d une transformation donnée et on indique la sortie"

```
menu_fichier_definir_comportement_transformation(n_umero)->
        bound(n_umero)
        transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l1,l2,c_omporte
ment)
        comportement(c_omportement,t_ype,e_tat_entrees,e_tat_interne,e_tat_sorties)
        retract(comportement(c_omportement,t_ype,e_tat_entrees,e_tat_interne,e_tat_sorties),nil)
        fail;
menu_fichier_definir_comportement_transformation(n_umero)->
        bound(n_umero)
        transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l1,l2,c_omporte
ment)
        calculer_etat_entrees(n_umero,t_ransformation_pere,v_ersion,l1,e10,e11)
        calculer_etat_interne(e2)
        calculer_etat_sorties(t_ype,e10,e2,e3)
        assert(comportement(c_omportement,t_ype,e11,e2,e3),nil)
        fail;
```

“Règle permettant d’afficher les comportements des transformations”

```
menu_fichier_definir_comportement_transformation(n_umero)->
        bound(n_umero)
        transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l1,l2,c_omporte
ment)
        comportement(c_omportement,t_ype,e_tat_entrees,e_tat_interne,e_tat_sorties)
        out(c_omportement) outm(" : ") out(e_tat_entrees) outm(" + ") out(e_tat_interne) outm(" => ")
out(e_tat_sorties) line
        fail;

calculer_etat_entrees(_,_,_,nil,nil,nil)->!;
calculer_etat_entrees(n_umero,t_ransformation_pere,v_ersion,n_o_interne.f1,<n_absolu,d_escription,e_tat>.f2,e
_tat.f3)->
        flux(n_absolu1,t_ransformation_pere,v_ersion,n_o_interne,_,_,_,_)
        rechercher_flux_original(n_absolu1,n_absolu2)
        flux(n_absolu2,_,_,_,_,t_ype,d_escription,e_tats)
        element(e_tat,e_tats,vrai)
        calculer_etat_entrees(n_umero,t_ransformation_pere,v_ersion,f1,f2,f3)
        ;

rechercher_flux_original(n_flux_absolu,n_flux_absolu)->
        flux(n_flux_absolu,_,_,_,-1,_,_,_)
        !;
rechercher_flux_original(n_flux_absolu1,n_flux_recherche)->
        flux(n_flux_absolu1,_,_,_,n_flux_absolu2,_,_,_)
        dif(n_flux_absolu2,-1)
        rechercher_flux_original(n_flux_absolu2,n_flux_recherche)
        !;

calculer_etat_interne(nominal)->;
calculer_etat_interne(absent)->;
calculer_etat_interne(intempestif)->;
```

```
calculer_etat_sorties(standard,x.y.nil,absent,absent.nil)->!;
calculer_etat_sorties(standard,<n,d,absent>.y.nil,e2,absent.nil)->!;
calculer_etat_sorties(standard,x.<n,d,absent>.nil,e2,absent.nil)->!;
calculer_etat_sorties(standard,<n,d,intempestif>.y.nil,e2,intempestif.nil)->!;
calculer_etat_sorties(standard,x.y.nil,intempestif,intempestif.nil)-> !;
calculer_etat_sorties(standard,x.y.nil,_,nominal.nil)->;

calculer_etat_sorties(standard,x.nil,absent,absent.nil)->!;
calculer_etat_sorties(standard,<n,d,absent>.nil,e2,absent.nil)->!;
calculer_etat_sorties(standard,<n,d,intempestif>.nil,e2,intempestif.nil)->!;
calculer_etat_sorties(standard,x.nil,intempestif,intempestif.nil)->!;
calculer_etat_sorties(standard,x.y.nil,_,nominal.nil)->;

calculer_etat_sorties(safety,x.y.nil,absent,absent.nil)->!;
calculer_etat_sorties(safety,<n,d,absent>.y.nil,e2,absent.nil)->!;
calculer_etat_sorties(safety,x.<n,d,absent>.nil,e2,absent.nil)->!;
calculer_etat_sorties(safety,x.y.nil,_,nominal.nil)->;

calculer_etat_sorties(safety,x.nil,absent,absent.nil)->!;
calculer_etat_sorties(safety,<n,d,absent>.nil,e2,absent.nil)->!;
calculer_etat_sorties(safety,x.y.nil,_,nominal.nil)->;

decrire_etat_sorties(e1,e2,e3)->
        out(e1) outm(" + ") out(e2) outm(" => ") out(e3) line;

"-----------------------------------------------"
```
“Règle permettant d’insérer une fonction (donc en particulier une safety function) entre deux fonctions”
```
"-----------------------------------------------"
"menu_fichier_InsererTransformation('generer l energie mecanique','carburant'.'air'.nil,'energie mecanique'.nil)"
"on coupe un flux en deux pour y inserer une transformation"
"inserer_transformation(numero_flux,description_transformation,description_flux_entree,description_flux_sorti
e)"
"-----------------------------------------------"

inserer_transformation(n_umero_flux,d_escription_transformation,d_escription_flux_entree,d_escription_flux_s
ortie)->
        incrementer_variable(nb_max_flux,f1)
        creer_flux(d_escription_flux_entree,f1)
        modifier_transformation_flux_sortie(n_umero_flux,f1)

        incrementer_variable(nb_max_flux,f2)
        creer_flux(d_escription_flux_sortie,f2)
        modifier_transformation_flux_entree(n_umero_flux,f2)

        creer_transformation(n_umero,d_escription,l1,l2)
        ;

modifier_transformation_flux_sortie(n_umero_flux,f1)->
        transformation(t,d_escription,l1,l2,c)
        element(n_umero_flux,l2,vrai)
        supprime_element2(n_umero_flux,l2,l3)
        creer_transformation(t1,d_escription,l1,f1.l2)
        ;
modifier_transformation_flux_entree(n_umero_flux,f1)->
        transformation(t,d_escription,l1,l2,c)
        element(n_umero_flux,l1,vrai)
        supprime_element2(n_umero_flux,l1,l3)
        creer_transformation(t1,d_escription,f1.l1,l2)
        ;
"flux(n_umero,t_ransformation_pere,no_interne,no externe,_,d_escription,_,_,_,_)"
```

```
"-----------------------------------------------"
modifier_transformation->;
```

“Prédicats permettant de définir les comportements par défaut”

```
"==============================================="
"-----------------------------------------------"
"predicats basiques"
"-----------------------------------------------"
"creer_transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l1,l2,c_omportement)"
creer_transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l1,l2,c_omportement)->
        free(n_umero)
        transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l10,l20,c_omportement)
        listes_identiques(l1,l10)
        listes_identiques(l2,l20)
        !;
creer_transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l1,l2,c_omportement)->
        free(n_umero)
        incrementer_variable(nb_max_transformation,n_umero)
        idem(n_umero,c_omportement)
        assert(transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l1,l2,c_omportement),nil)
        creer_comportements(c_omportement,t_ype,l1,l2)
        !;
```

“Création du comportement d’une transformation à deux entrées”

```
creer_comportements(c_omportement,standard,e1.e2.nil,s.nil)->
        assert(comportement(c_omportement,standard,_,_,nominal.nil),!.nil)
        assert(comportement(c_omportement,standard,_._.nil,<_,1>,intempestif.nil),!.nil)
        assert(comportement(c_omportement,standard,intempestif._.nil,<_,_>,intempestif.nil),!.nil)
        assert(comportement(c_omportement,standard,_.absent.nil,<_,_>,absent.nil),!.nil)
        assert(comportement(c_omportement,standard,absent._.nil,<_,_>,absent.nil),!.nil)
        assert(comportement(c_omportement,standard,_,<1,_>,absent.nil),!.nil)
        fail;
creer_comportements(c_omportement,safety,e1.e2.nil,s.nil)->
        assert(comportement(c_omportement,safety,_,_,nominal.nil),!.nil)
        assert(comportement(c_omportement,safety,_.absent.nil,<_,_>,absent.nil),!.nil)
        assert(comportement(c_omportement,safety,absent._.nil,<_,_>,absent.nil),!.nil)
        assert(comportement(c_omportement,safety,_,<1,_>,absent.nil),!.nil)
        fail;
```

“Création du comportement à une entrée”

```
creer_comportements(c_omportement,standard,e.nil,s.nil)->
        assert(comportement(c_omportement,standard,_,_,nominal.nil),!.nil)
        assert(comportement(c_omportement,standard,_,<_,1>,intempestif.nil),!.nil)
        assert(comportement(c_omportement,standard,intempestif.nil,<_,_>,intempestif.nil),!.nil)
        assert(comportement(c_omportement,standard,absent.nil,<_,_>,absent.nil),!.nil)
        assert(comportement(c_omportement,standard,_,<1,_>,absent.nil),!.nil)
        fail;
creer_comportements(c_omportement,safety,e.nil,s.nil)->
        assert(comportement(c_omportement,safety,_,_,nominal.nil),!.nil)
        assert(comportement(c_omportement,safety,absent.nil,<_,_>,absent.nil),!.nil)
        assert(comportement(c_omportement,safety,_,<1,_>,absent.nil),!.nil)
        fail;
creer_comportements(_,_,_,_)->;


"creer_flux(n_umero,t_ransformation_pere,v_ersion,n_o_interne,n_o_externe,n_ature,d_escription,l_iste_etats)"
creer_flux(n_umero,t_ransformation_pere,v_ersion,n_o_interne,n_o_externe,n_ature,d_escription,l_iste_etats)->
        free(n_umero)
```

```
        flux(n_umero,t_ransformation_pere,v_ersion,n_o_interne,n_o_externe,n_ature,d_escription,l_iste_etats)
        !;
creer_flux(n_umero,t_ransformation_pere,v_ersion,n_o_interne,n_o_externe,n_ature,d_escription,l_iste_etats)->
        free(n_umero)
        incrementer_variable(nb_max_flux,n_umero)
        assert(flux(n_umero,t_ransformation_pere,v_ersion,n_o_interne,n_o_externe,n_ature,d_escription,l_iste
_etats),nil)
        !;

"============================================="
"---------------------------------------------"
lister_elements_version(v_ersion)->
        Modele_version(v_ersion,d_ecomposition)
        line
        affiche_decomposition(d_ecomposition,"|->")
        ;

affiche_decomposition(nil,s2)->
        conc_string(s1,"->",s2)
        outml(s1)
        !
        fail;
affiche_decomposition(<t,l1>.l2,s1)->
        conc_string("| ",s1,s2)
        transformation(t,d_esignation,_,_,_)
        outm(s2) out(t) outm(" : ") outm(d_esignation) line
        affiche_decomposition(l1,s2)
        fail;
affiche_decomposition(t.l2,s1)->
        not(tuple(t))
        conc_string("| ",s1,s2)
        transformation(t,d_esignation,f_lux_entree,f_lux_sortie,_)
        outm(s2) out(t) outm(" : ") outm(d_esignation) outm(" ; ") out(f_lux_entree) outm(" ; ")
out(f_lux_sortie) line
        fail;
affiche_decomposition(t.l2,s1)->
        affiche_decomposition(l2,s1)
        fail;

"---------------------------------------------"
"quelques fonctions de base"
"---------------------------------------------"

lister_elements_version(v_ersion)->
        version(v_ersion,v)
        lister_elements_version2(v)
        ;
lister_elements_version2(nil)->!;
lister_elements_version2(<t,l2>.l1)->
        !
        outm("--------- decomposition : ") outl(t)
        lister_elements_version2(l2)
        lister_elements_version2(l1);
lister_elements_version2(t.l)->
        outm("--------- transformation : ") outl(t)
        transformation(i_d,d_escription, l_iste_flux_entree, l_iste_flux_sortie,c_omportement)
        outml("--------- flux entree : ")
        lister_flux(l_iste_flux_entree)
        outml("--------- flux sortie : ")
        lister_flux(l_iste_flux_sortie)
        ;
```

```
lister_flux(nil)->!;
lister_flux(f_lux.l_iste_flux)->
        flux(i_d,t_ype,n_om,v_aleurs,v_aleur,m_ini,m_axi)
        lister_flux(l_iste_flux)
        ;


"============================================="
"menu transformation"
"============================================="
"---------------------------------------------"
"tester un comportement"
gotc1->tester_comportement_collectif(<1,2.3.4.nil>,nominal.nil);
gotc2->tester_comportement_collectif(<1,<2,5.6.nil>.3.4.nil>,absent.nil);
gotc3->tester_comportement_collectif(<2,5.6.7.nil>,nominal.absent.intempestif.nil);

gotu1->tester_comportement_unitaire(1,intempestif.nil);
gotu2->tester_comportement_unitaire(5,intempestif.absent.nil);
gotu3->tester_comportement_unitaire(7,nominal.intempestif.absent.nil);

"---------------------------------------------"
```
“Règle permettant la modélisation d’une décomposition fonctionnelle sur plusieurs niveaux”
```
tester_comportement_collectif(a_rbre_vertical,e_tats_sortie_analyses)->
        outml("---------------------------------------------")
        outm("Etude des fonctions ") out(a_rbre_vertical) line
        outm("Pour les valeurs de sortie : ") out(e_tats_sortie_analyses) line line
        calculer_arbre_horizontal(a_rbre_vertical,a_rbre_horizontal)

        etat_possible_transformation(_,e_tat_transformation)
        calculer_etat_sortie(a_rbre_horizontal,e_s,e_tat_entrees,e_tat_transformation,e_tats_internes)
        premier_element(e_s,e_tat_sortie)
        element(e_tat_sortie,e_tats_sortie_analyses,vrai)
        out(e_tat_entrees) outm(" + ")
        out(e_tat_transformation) outm(" --> ")
        out(e_tat_sortie) line
        fail;

premier_element(e.l,e)->!;
premier_element(e,e)->;

calculer_arbre_horizontal(a_rbre_vertical,a_rbre_horizontal)->
        initialiser_arbre_h
        idem(a_rbre_vertical,<t,_>)
        transformation(t,t0,v_ersion,_,_,_,_,n_o_flux_interne.nil,_)
        flux(n_o_flux_absolu,t0,v_ersion,n_o_flux_interne,n_o_flux_externe,_,_,_)

        assert(place_terminale(n_o_flux_absolu),nil)
        creer_noeuds_elementaires(a_rbre_vertical)
        generer_arbre_horizontal(a_rbre_horizontal)
        ;

initialiser_arbre_h->
        efface(transition,3)
        efface(place_terminale,1);

creer_noeuds_elementaires(<t0,nil>)->!;
creer_noeuds_elementaires(<t0,t.l>)->
        creer_noeuds_elementaires2(t0,t)
        creer_noeuds_elementaires(<t0,l>);

creer_noeuds_elementaires2(t0,<t1,l>)->
        creer_interfaces(t0,t1)
```

```
		creer_noeuds_elementaires(<t1,l>)
		!;
creer_noeuds_elementaires2(t0,t1)->
		transformation(t1,t0,v_ersion,n_o_interne,_,_,l_iste_flux_entree,l_iste_flux_sortie,_)
		recherche_flux_absolus_liste(t0,v_ersion,l_iste_flux_entree,l1)
		recherche_flux_absolus_liste(t0,v_ersion,l_iste_flux_sortie,l2.nil)
		assert(transition(l1,t1,l2),nil)
		creer_interfaces(t0,t1)
		;

creer_interfaces(t0,t1)->
		transformation(t1,t0,v_ersion,n_o_interne,_,_,l_iste_flux_entree,l_iste_flux_sortie,_)
		creer_interfaces_entrantes(t0,v_ersion,t1,l_iste_flux_entree)
		creer_interfaces_sortantes(t0,v_ersion,t1,l_iste_flux_sortie)
		;

creer_interfaces_entrantes(t0,v_ersion,t1,nil)->!;
creer_interfaces_entrantes(t0,v_ersion,t1,n_o_flux_interne.l)->
		flux(n_o_flux_absolu,t0,v_ersion,n_o_flux_interne,n_o_flux_externe,_,_,_)
		dif(n_o_flux_externe,-1)
		!
		assert(transition(n_o_flux_externe.nil,nul,n_o_flux_absolu),nil)
		creer_interfaces_entrantes(t0,v_ersion,t1,l);
creer_interfaces_entrantes(t0,v_ersion,t1,n_o_flux_interne.l)->
		creer_interfaces_entrantes(t0,v_ersion,t1,l);

creer_interfaces_sortantes(t0,v_ersion,t1,nil)->!;
creer_interfaces_sortantes(t0,v_ersion,t1,n_o_flux_interne.l)->
		flux(n_o_flux_absolu,t0,v_ersion,n_o_flux_interne,n_o_flux_externe,_,_,_)
		dif(n_o_flux_externe,-1)
		!
		assert(transition(n_o_flux_absolu.nil,nul,n_o_flux_externe),nil)
		creer_interfaces_sortantes(t0,v_ersion,t1,l);
creer_interfaces_sortantes(t0,v_ersion,t1,n_o_flux_interne.l)->
		creer_interfaces_sortantes(t0,v_ersion,t1,l);

generer_arbre_horizontal(a_rbre_h)->
		place_terminale(p_t)
		generer_arbre_horizontal(p_t,a_rbre_h)
		;

generer_arbre_horizontal(p1,<p1,t,a_rbre>)->
		transition(p2.nil,t,p1)
		generer_arbre_horizontal(p2,a_rbre)
		!;
generer_arbre_horizontal(p1,<p1,t,a_rbre2,a_rbre3>)->
		transition(p2.p3.nil,t,p1)
		generer_arbre_horizontal(p2,a_rbre2)
		generer_arbre_horizontal(p3,a_rbre3)
		!;
generer_arbre_horizontal(p1,<p1,nil,nil>)->
		!;

" transition(liste_entrees,transition,sortie)"
" place_terminale(no_place) "

calculer_etat_sortie(<f_lux,nil,nil>,e_tat,<f_lux,e_tat>.nil,e_tat_transformation0,nil)->
		flux(f_lux,_,_,_,-1,_,_,l_iste_etats)
		bound(l_iste_etats)
		element(e_tat,l_iste_etats,vrai)
		;
calculer_etat_sortie(<f_lux,nil,nil>,e_tat,<f_lux,e_tat>.nil,e_tat_transformation0,nil)->
```

```
        flux(f_lux,_,_,_,_,_,_,l_iste_etats)
        free(l_iste_etats)
        element(e_tat,absent.intempestif.nominal.nil,vrai)
        ;
calculer_etat_sortie(<f_lux1,nul,a_rbre>,e_tat_sortie,e_tat_entrees,e_tat_transformation0,e_tats_internes)->
        calculer_etat_sortie(a_rbre,e_tat_sortie,e_tat_entrees,e_tat_transformation0,e_tats_internes)
        ;
calculer_etat_sortie(<f_lux1,t_ransformation,a_rbre>,e_tat_sortie,e_tat_entrees1,e_tat_transformation,<<t_ransf
ormation,e_tat_transformation>,e_tats_internes1>)->
        calculer_etat_sortie(a_rbre,e_tat_sortie1,e_tat_entrees1,e_tat_transformation,e_tats_internes1)
        etat_possible_transformation(t_ransformation,e_tat_transformation)
        transformation(t_ransformation,_,_,_,_,t_ype,_,_,c_omportement)
        comportement(c_omportement,t_ype,e_tat_sortie1.nil,e_tat_transformation,e_tat_sortie)
        ;
calculer_etat_sortie(<f_lux1,t_ransformation,a_rbre1,a_rbre2>,e_tat_sortie,<e_tat_entrees1,e_tat_entrees2>,e_ta
t_transformation,<<t_ransformation,e_tat_transformation>,e_tats_internes1,e_tats_internes2>)->
        calculer_etat_sortie(a_rbre1,e_tat_sortie1.nil,e_tat_entrees1,e_tat_transformation,e_tats_internes1)
        calculer_etat_sortie(a_rbre2,e_tat_sortie2.nil,e_tat_entrees2,e_tat_transformation,e_tats_internes2)
        etat_possible_transformation(t_ransformation,e_tat_transformation)
        transformation(t_ransformation,_,_,_,_,t_ype,_,_,c_omportement)
        comportement(c_omportement,t_ype,e_tat_sortie1.e_tat_sortie2.nil,e_tat_transformation,e_tat_sortie)
        ;
"---------------------------------------------"

tester_comportement_unitaire(n_umero,e_tats_sortie_analyses)->
        outml("---------------------------------------------")
        outm("Etude du comportement de la transformation ") outl(n_umero)
        transformation(n_umero,t_ransformation_pere,v_ersion,n_o_interne,d_escription,t_ype,l_iste_flux_entr
ee,l_iste_flux_sortie, c_omportement)
        outml(d_escription) line
        outml("etat_transformation + etat_possible_entrees -> etat_sortie") line

        recherche_flux_absolus_liste(t_ransformation_pere,v_ersion,l_iste_flux_entree,l_iste_flux_entree1)
        recherche_flux_absolus_liste(t_ransformation_pere,v_ersion,l_iste_flux_sortie,l_iste_flux_sortie1)

        recherche_flux_origine_liste(l_iste_flux_entree1,l_iste_flux_entree2)
        recherche_flux_origine_liste(l_iste_flux_sortie1,l_iste_flux_sortie2)
        outm("Liste des flux entree : ") out(l_iste_flux_entree) outm(" => ") out(l_iste_flux_entree1) outm(" =>
") outl(l_iste_flux_entree2)
        outm("Liste des flux sortie : ") out(l_iste_flux_sortie) outm(" => ") out(l_iste_flux_sortie1) outm(" =>
") outl(l_iste_flux_sortie2) line

        outm("Contraintes de sortie : ") outl(e_tats_sortie_analyses) line

        etat_possible_transformation(_,e_tat_transformation)
        etat_possible_entrees(l_iste_flux_entree2,e_tat_possible_entrees)
        comportement(c_omportement,t_ype,e_tat_possible_entrees,e_tat_transformation,e_tat_sortie.nil)
        element(e_tat_sortie,e_tats_sortie_analyses,vrai)
        out(e_tat_transformation) outm(" ") out(e_tat_possible_entrees) outm(" -> ") out(e_tat_sortie) line
        fail;

etat_possible_transformation(t_ransformation,<0,0>)->;
etat_possible_transformation(t_ransformation,<0,1>)->;
etat_possible_transformation(t_ransformation,<1,0>)->;
etat_possible_transformation(t_ransformation,<1,1>)->;

etat_possible_entrees(nil,nil)->!;
etat_possible_entrees(n_o_flux_entree.l1,e_tat.l2)->
        flux(n_o_flux_entree, t_ransformation_pere, v_ersion, n_o_interne, n_o_externe, n_ature, d_escription,
l_iste_etats)
        element(e_tat,l_iste_etats,vrai)
        etat_possible_entrees(l1,l2);
```

```
"-------------------------------------------"
recherche_flux_absolus_liste(_,_,nil,nil)->!;
recherche_flux_absolus_liste(t_ransformation_pere, v_ersion,f1.l1,f2.l2)->
        recherche_flux_absolu(t_ransformation_pere, v_ersion,f1,f2)
        recherche_flux_absolus_liste(t_ransformation_pere, v_ersion,l1,l2)
        ;

recherche_flux_absolu(t_ransformation_pere, v_ersion,n_o_interne,n_o_flux_absolu)->
        flux(n_o_flux_absolu, t_ransformation_pere, v_ersion, n_o_interne, n_o_externe, n_ature, d_escription,
l_iste_etats)
        !;


recherche_flux_origine_liste(nil,nil)->!;
recherche_flux_origine_liste(f1.l1,f2.l2)->
        recherche_flux_origine(f1,f2)
        recherche_flux_origine_liste(l1,l2)
        ;

recherche_flux_origine(n_o_flux_absolu,n_o_flux_absolu)->
        flux(n_o_flux_absolu, t_ransformation_pere, v_ersion, n_o_interne, n_o_externe, n_ature, d_escription,
l_iste_etats)
        idem(n_o_externe,-1)
        !;
recherche_flux_origine(n_o_flux_absolu,n_o_flux_origine)->
        flux(n_o_flux_absolu, t_ransformation_pere, v_ersion, n_o_interne, n_o_externe, n_ature, d_escription,
l_iste_etats)
        dif(n_o_externe,-1)
        recherche_flux_origine(n_o_externe,n_o_flux_origine)
        !;

"-------------------------------------------"
```

Nous avons donc mis en application les programmes réalisés sur notre cas d'étude.

Tout d'abord, nous avons réalisé des essais pour vérifier le comportement logique d'une fonction. Nous avons donc défini par un ensemble de règles le comportement de la transformation 1 puis nous avons vérifié quels étaient les cas pour lesquels la déviation de la sortie de cette transformation était intempestive.

" Tests de comportements unitaires "

"gotu1->tester_comportement_unitaire(1,intempestif.nil);"

Le résultat associé à ce programme est ci-dessous. Une ligne de résultat indique quels sont les deux états de la transformation (*<0,0>* avec mode de défaillant absent pour le premier chiffre et mode de défaillant intempestif pour le second chiffre) et les états des entrées (*nominal.nominal.nil* indique l'état de la première entrée par le premier *nominal* et l'état de la seconde entrée par le second *nominal*, *nil* étant un indicateur de fin de liste associé à Prolog) permettant d'aboutir à la sortie recherchée (*intempestif* pour ce cas).

> gotu1;

Etude du comportement de la transformation 1 - generer l energie mecanique

etat_transformation + etat_possible_entrees -> etat_sortie

Liste des flux entree : 1.2.nil => 1.2.nil => 1.2.nil

Liste des flux sortie : 3.nil => 3.nil => 3.nil

Contraintes de sortie : intempestif.nil

<0,0> intempestif.intempestif.nil -> intempestif

<0,0> intempestif.nominal.nil -> intempestif

<0,1> intempestif.intempestif.nil -> intempestif

<0,1> intempestif.nominal.nil -> intempestif

<0,1> nominal.intempestif.nil -> intempestif

<0,1> nominal.nominal.nil -> intempestif

Le résultat est cohérent. Il indique que la transformation est intempestive si l'entrée est intempestive ou le mode de défaillance intempestif. Il serait possible d'améliorer cette modélisation en factorisant ce résultat.

Nous avons fait également des essais sur le comportement logique d'un ensemble de transformations pour vérifier si le comportement logique global était bien identique. Dans cet exemple, nous avons fait l'essai avec la version 3 en déterminant quel était le comportement logique global des transformations 5, 6, 3 et 4. Nous avons déterminé dans le cas où le flux de sortie était absent.

" Tests de comportements collectifs "

"gotc2->tester_comportement_collectif(<1,<2,5.6.nil>.3.4.nil>,absent.nil);"

Le résultat associé à ce programme est ci-dessous. Une ligne de résultat indique quels sont les états des entrées (*<<1,absent>.nil,<2,absent>.nil>* indique l'état de la première entrée par le premier *<1,absent>.nil* et l'état de la seconde entrée par le second *<2,absent>.nil*) et les deux états de la transformation (*<0,0>* comme pour le cas précédent) et permettant d'aboutir à la sortie recherchée (*absent* pour ce cas).

> gotc2;

Etude des fonctions <1,<2,5.6.nil>.3.4.nil>

Pour les valeurs de sortie : absent.nil

<<1,absent>.nil,<2,absent>.nil> + <0,0> --> absent

<<1,intempestif>.nil,<2,absent>.nil> + <0,0> --> absent

<<1,nominal>.nil,<2,absent>.nil> + <0,0> --> absent

<<1,absent>.nil,<2,absent>.nil> + <0,1> --> absent

<<1,intempestif>.nil,<2,absent>.nil> + <0,1> --> absent

<<1,nominal>.nil,<2,absent>.nil> + <0,1> --> absent

<<1,absent>.nil,<2,absent>.nil> + <1,0> --> absent

<<1,absent>.nil,<2,intempestif>.nil> + <1,0> --> absent

<<1,absent>.nil,<2,nominal>.nil> + <1,0> --> absent

<<1,intempestif>.nil,<2,absent>.nil> + <1,0> --> absent

<<1,intempestif>.nil,<2,intempestif>.nil> + <1,0> --> absent

<<1,intempestif>.nil,<2,nominal>.nil> + <1,0> --> absent

<<1,nominal>.nil,<2,absent>.nil> + <1,0> --> absent

<<1,nominal>.nil,<2,intempestif>.nil> + <1,0> --> absent

<<1,nominal>.nil,<2,nominal>.nil> + <1,0> --> absent

<<1,absent>.nil,<2,absent>.nil> + <1,1> --> absent

<<1,absent>.nil,<2,intempestif>.nil> + <1,1> --> absent

<<1,absent>.nil,<2,nominal>.nil> + <1,1> --> absent

<<1,intempestif>.nil,<2,absent>.nil> + <1,1> --> absent

<<1,intempestif>.nil,<2,intempestif>.nil> + <1,1> --> absent

<<1,intempestif>.nil,<2,nominal>.nil> + <1,1> --> absent

<<1,nominal>.nil,<2,absent>.nil> + <1,1> --> absent

<<1,nominal>.nil,<2,intempestif>.nil> + <1,1> --> absent

<<1,nominal>.nil,<2,nominal>.nil> + <1,1> --> absent

Le résultat est également cohérent car la sortie est absente si une des deux entrées est absente ou si le mode de défaillance absent est activé.

RÉSUMÉ

Contexte — La Sûreté de Fonctionnement (SdF) est un domaine étudié de façon de plus en plus rigoureuse par les concepteurs. Si la SdF relative aux organes bénéficie de retours d'expériences sur lesquels le développement de solutions physiques peut s'appuyer, tel n'est pas le cas lorsqu'il s'agit de concevoir les architectures fonctionnelles d'un produit. L'architecte Système (AS) conceptualise et conçoit un système qui doit fournir un service selon certaines performances ; son cadre de référence, à savoir l'Ingénierie Système (IS) étant lacunaire en matière de SdF. De la sorte, l'ingénieur expert en SdF intervient après l'architecte, réalise ses propres analyses fonctionnelles, perturbant le travail réalisé et obligeant à des boucles qui pourraient être évitées. Cependant, des normes comme l'ISO 26262:2011 obligent les bureaux d'études des constructeurs de passer du SdF constatée à une SdF véritablement conçue. Il convient de ce fait, pour les industriels, de s'intéresser de près à l'alignement entre la SdF et l'IS.

Résultats — Nous avons obtenu quatre résultats. Le premier est un modèle conceptuel de l'Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF). Ce modèle conceptuel, découpé en trois vues (vues opérationnelle, fonctionnelle, organique), permet de définir les concepts et les liens entre le domaine de l'IS et celui de la SdF. L'ensemble présente une forte cohérence sémantique. Ce modèle a permis de définir ensuite un macroprocessus d'ISSdF cohérent avec les processus prescrits par l'ISO 15288:2008 et conforme aux livrables de l'ISO 26262:2011. Le macroprocessus présenté permet de définir l'architecture d'un système sûr. Pour cela, nous l'avons scindé en trois phases de conception (définition de concept, conception préliminaire et conception détaillée) ainsi que quatre points de vue (opérationnel, fonctionnel externe, fonctionnel interne, organique). Plusieurs activités spécifiques sont présentes dans le macroprocessus. Il a fallu ensuite proposer des méthodes pour les réaliser au mieux. Ainsi, nous avons développé une méthode qui concerne l'analyse des risques en vue externe. Vu les limites des méthodes comme l'Analyse Préliminaire des Risques (APR) vis-à-vis des livrables demandés par l'ISO 26262:2011 (Hazard Analysis and Risk Assessment), nous avons proposé une nouvelle méthode qui permet de définir les Safety Goals (exigence de sûreté de haut niveau) avec un ASIL (niveau d'intégrité de sûreté) associé pour les scénarios critiques. De plus, nous avons développé une méthode permettant l'analyse des architectures fonctionnelles d'un point de vue dysfonctionnel pour définir les Functional Safety Requirements. Cette méthode repose sur des mécanismes de propagation de défaillance afin de déterminer et de visualiser le comportement dysfonctionnel pour l'architecte système. Nous avons aussi proposé les références de syntaxe SysML pour élaborer les modèles associés à ces deux méthodes. Elles ont été appliquées dans le cas de l'accélération intempestive du véhicule.

ABSTRACT

Context – Safety is an area which is increasingly stringent by designers. If Safety on components benefits from experience feedback on which the development of physical solutions can support, this is not the case when it comes to designing functional architecture of a product. The System Architect (SA) conceptualizes and designs a system that must provide service according to a defined level of performance; its framework, namely the System Engineering (SE) is incomplete for safety. In this way, the safety expert comes after architects, performs his/her own functional analyzes, disrupting the work and forcing loops that could be avoided. However, standards such as ISO 26262:2011 require manufacturers to transfer from observed safety to designed safety. It should therefore, for industry to pay close attention to the alignment between the SE and safety.

Results - We got four results. The first one is a conceptual model of the Safe Systems Engineering (SSE). This conceptual model, divided into three views (operational, functional, physical), defines the concepts and relationships between the field of SE and that of safety. The assembly has a strong semantic consistency. This model has then set a SSE macroprocess compliant with the process prescribed by ISO 15288:2008 and complies with the deliverables of ISO 26262:2011. The proposed macroprocess permits to define the architecture of a safe system. For this, we have divided into three design phases (concept definition, preliminary design and detailed design) and four views (operational, functional external, internal functional, physical). Several specific activities are present in the macroprocess. Thus, we developed a methodology on risk analysis in external view. Given the limitations of methods like Preliminary Hazard Analysis (PHA) regarding the deliverables required by the ISO 26262:2011 (Hazard Analysis and Risk Assessment), we proposed a new method for defining the Safety Goals (high level safety requirement) with ASIL (safety integrity level) associated for critical scenarios. In addition, we have developed a method for analyzing the functional architecture with a dysfunctional point of view to define the Functional Safety Requirements. This method relies on failure propagation mechanisms to determine and visualize the dysfunctional behavior for the system architect. We have proposed SysML syntax references too to elaborate the models associated to these two methods. They have been applied in the case of unintended acceleration of the vehicle.