Démarche de conception sûre de la Supervision de la fonction de Conduite Autonome

Romain Cuer

N°d'ordre NNT : 2018LYSEI091

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
INSA Lyon

Ecole Doctorale N° 160
Electronique Electrotechnique Automatique (EEA)

Spécialité Automatique

Soutenue publiquement le 23/11/2018, par :
Romain Cuer

Démarche de conception sûre de la Supervision de la fonction de Conduite Autonome

---

Devant le jury composé de :

Kratz, Frédéric	Professeur des Universités	INSA CVL, Bourges-Blois	Rapporteur
Levrat, Eric	Professeur des Universités	Université de Lorraine, Nancy	Rapporteur
Ghazel, Mohamed	Directeur de Recherche	IFSTTAR, Lille	Examinateur
Lanusse, Agnès	Docteure	CEA-List, Saclay	Examinatrice
Niel, Eric	Professeur des Universités	INSA de Lyon	Directeur de thèse
Piétrac, Laurent	Maître de Conférences, HDR	INSA de Lyon	Co-directeur de thèse
Dang-Van-Nhan, Christophe	Chef de Projet Innovation	Groupe Renault, Guyancourt	Encadrant industriel
Dumitrescu, Emil	Maître de Conférences	INSA de Lyon	Invité

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	M. Stéphane DANIELE Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES **secretariat.e2m2@univ-lyon1.fr**	M. Philippe NORMAND UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 Avenue Jean CAPELLE INSA de Lyon 69 621 Villeurbanne Tél : 04.72.68.49.09 Fax : 04.72.68.49.16 **emmanuelle.canet@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 Fax : 04.72.43.16.87 **infomaths@univ-lyon1.fr**	M. Luca ZAMBONI Bât. Braconnier 43 Boulevard du 11 novembre 1918 69 622 Villeurbanne CEDEX Tél : 04.26.23.45.52 **zamboni@maths.univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Marion COMBE Tél : 04.72.43.71.70 Fax : 04.72.43.87.12 Bât. Direction **ed.materiaux@insa-lyon.fr**	M. Jean-Yves BUFFIÈRE INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Marion COMBE Tél : 04.72.43.71.70 Fax : 04.72.43.87.12 Bât. Direction **mega@insa-lyon.fr**	M. Jocelyn BONJOUR INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Viviane POLSINELLI Brigitte DUBOIS INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **viviane.polsinelli@univ-lyon2.fr**	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Table des matières

Table des figures

Liste des tableaux

1
Introduction générale

1.1 Avant-propos

Cette thèse, débutée en septembre 2015, s'est déroulée dans le cadre d'une Convention Industrielle de Formation par la REcherche (CIFRE), signée entre le groupe Renault et le Laboratoire Ampère. Plus précisément, les travaux ont pris place entre le site Renault du *Technocentre* à Guyancourt et celui de l'INSA de Lyon, au sein du département Méthodes pour l'Ingénierie des Systèmes (MIS) du Laboratoire Ampère.

1.2 Le projet du Véhicule Autonome et ses enjeux

Depuis les années 1980, les véhicules intègrent un nombre de plus en plus élevé de systèmes d'aide à la conduite (Maurer et Winner 2013) [1], qui sont des systèmes embarqués électroniques, critiques du point de vue de la sûreté (Judalet 2016) (*i.e.* leur dysfonctionnement peut causer des dommages graves pour l'Homme et/ou l'Environnement). Ces systèmes embarqués et les fonctions qu'ils assurent (aide au maintien dans sa ligne, freinage d'urgence, parking automatique...) constituent un ensemble de briques technologiques à partir desquelles une innovation majeure pour l'industrie automobile est en cours de développement : le Véhicule Autonome. Ce véhicule doit se conduire à terme sans aucune intervention du conducteur, dans n'importe quelle situation de conduite. Avant toutefois d'arriver à un tel stade, des organismes internationaux majeurs tels que la National Highway Traffic Safety Administration (NHTSA) et l'Organisation Internationale des Constructeurs Automobiles (OICA) ont défini des niveaux progressifs d'autonomisation (NHTSA 2017 ; OICA 2014) selon : le désengagement graduel du conducteur, qui sera toujours moins impliqué dans les tâches de conduite, et l'extension, également progressive, des situations pour lesquelles la conduite pourra être déléguée au véhicule. L'argument majeur en faveur de l'introduction de ce nouveau véhicule est la réduction drastique du nombre d'accidents, en supprimant les erreurs humaines qui sont à l'origine de l'immense majorité des accidents (les erreurs humaines sont impliquées dans 93% des accidents aux Etats-Unis (Chong 2016)). Cependant, pour recueillir de tels bénéfices, des défis autant ardus que multiples doivent être préalablement relevés par les métiers de l'ingénierie (Fagnant et Kockelman 2015), à commencer par celui de la Sûreté de Fonctionnement.

1. Les termes en caractères bleus correspondent aux références bibliographiques

L'absence totale de conducteur dans la boucle de contrôle constitue une véritable rupture du point de vue de la Sûreté de Fonctionnement. En effet, dans la norme ISO 26262 (ISO 2011), un critère de *Contrôlabilité* quantifie la capacité d'un conducteur à éviter certains événements redoutés ou, du moins, en amoindrir leurs conséquences. Ce critère de contrôlabilité a une incidence directe sur les niveaux de contraintes (appelés Automotive Safety Integrity Level (ASIL)) associés aux exigences visant à éviter les événements redoutés. Ainsi, plus un événement redouté est considéré comme contrôlable par le conducteur, moins le niveau de contrainte sera élevé. Ceci a naturellement des impacts notables sur les architectures fonctionnelles et physiques du véhicule. Pour le véhicule autonome, il apparaît qu'aucun des événements redoutés ne peut désormais être considéré comme contrôlable par le conducteur (puisqu'il est hors de la boucle de contrôle). Ceci explique l'aspect central des enjeux relatifs à la Sûreté de Fonctionnement pour ce type de véhicules. Dans ce contexte, les démonstrations de sûreté des Véhicules Autonomes revêtiront une importance cruciale afin, d'une part, d'éviter le maximum d'accidents ou d'incidents grâce à la réduction des erreurs de conception et, d'autre part, de servir de base à d'éventuelles futures certifications, menées par des organismes indépendants. Ce travail de thèse s'inscrit dans ce contexte particulier et contribue à la réalisation de ces démonstrations de sûreté.

1.3 Problématique et objectifs

1.3.1 Problématique scientifique générale

La problématique principale à laquelle nous avons été confrontés consiste à prendre en compte au plus tôt dans le processus de conception (*i.e.* en phase amont de conception) les exigences émises par le métier de Sûreté de Fonctionnement (SdF), exercé, chez Renault, par les pilotes Sûreté de Fonctionnement (pilotes SdF). Au sein du groupe Renault, comme dans d'autres entreprises (Mauborgne 2016 ; Legendre 2017), un autre métier revêt une importance particulière : celui d'Architecte Système, appelé, chez Renault, Architecte Métier Système (AMS). Son rôle principal est de recueillir, synthétiser, analyser et restituer sous forme d'exigences techniques (nommées aussi exigences système) les besoins des différentes parties prenantes. L'AMS définit également les principales composantes du système, leur agencement, leur organisation, leurs interfaces qui constituent les architectures fonctionnelles et physiques du système conçu (SEBoK 2017).

La prise en compte des exigences émanant des pilotes SdF dans les activités des AMS est source de difficultés et d'incompréhensions car ces deux disciplines d'ingénierie sont, dans l'entreprise, organisées « en silos », c'est-à-dire que chaque métier a ses propres objectifs (livrables, jalons), contraintes (planning, coûts) et dispose pour atteindre ses objectifs de moyens spécifiques (outils, modèles, équipes de travail...). La logique de cette organisation repose sur le principe de *séparation des préoccupations* (Hürsch et Lopes 1995 ; Ernst 2003 ; Legendre 2017) qu'il convient de respecter afin que les pilotes SdF puissent, indépendamment des activités des AMS, critiquer la conception que ces derniers proposent pour élaborer des architectures systèmes sûres. Notons à ce propos que l'on retrouve cette séparation dans l'organisation des bureaux d'étude : les AMS et les pilotes SdF ne partagent pas le même espace de travail. Toutefois, bien que cette organisation en silos soit donc justifiée, elle soulève des questions sur la collaboration entre les AMS et les pilotes SdF :

— Comment s'assurer que, malgré le décalage dû à la conception en silos, les deux métiers travaillent sur le même système ?

— Comment détecter et traiter les potentielles incohérences et contradictions entre les deux points de vue métier ?

— Quels sont les moments judicieux (jalons) auxquels une mise en cohérence doit être organisée ?

— Quels moyens mettre en place pour réaliser cette convergence ?

Étant donné les différences de compétences, de connaissances, mais aussi de cultures industrielles (qui s'ajoutent à la séparation des activités), des incohérences de fort impact peuvent exister entre le point de vue de l'AMS et celui du pilote SdF. Cette problématique large a été abordée dans le secteur automobile (Taofifenua 2012 ; Mauborgne 2016), dans le domaine spatial (P. David 2009 ; Cressent 2012) ou de manière plus générique (*i.e.* ces travaux ne sont pas spécifiques à un domaine d'application particulier) (Guillerm 2011 ; Güdemann 2011 ; Legendre 2017). Elle constitue en conséquence un problème scientifique qui reste ouvert, dont les enjeux sont importants notamment dans un contexte industriel. Il ressort de ces travaux que la principale voie opérationnelle pour traiter cette question réside dans l'adoption, premièrement, du cadre méthodologique fourni par l'Ingénierie Système, puis, dans un second temps, des modèles, utilisés pour mettre en pratique ces méthodes (*Ingénierie Système Basée sur les Modèles* ou Model Based Systems Engineering, MBSE) largement adoptées à ce jour.

1.3.2 Cas d'étude

Afin de préciser le problème que nous avons traité, il convient au préalable de présenter succinctement le cas d'étude. Basiquement, la conception du véhicule autonome revient à intégrer une nouvelle fonction, nommée fonction de Conduite Autonome ou fonction Autonomous Driving (AD), dans l'architecture fonctionnelle d'un véhicule équipé des systèmes d'aide à la conduite appropriés (c'est-à-dire qui fournissent une perception suffisante de l'environnement et de son évolution). Le rôle premier de la fonction AD est d'élaborer continuellement une commande de trajectoire, transmise aux actionneurs, et construite à partir des événements extérieurs.

Globalement, la fonction AD peut être active lorsque la conduite est entièrement déléguée au véhicule, ou inactive, lorsque ce n'est pas le cas. Or, cette fonction peut également se trouver dans des états intermédiaires (*Disponible*, *Non Disponible*, *Activable*...), dus à l'évolution de l'environnement et la vérification de certaines conditions particulières. Ces états sont gérés par une fonction de supervision, nommée **Supervision AD**, qui est donc chargée de déterminer à tout instant dans quel état la fonction AD doit se trouver.

Le principal objectif de ces travaux consiste alors à garantir que la fonction AD se trouve constamment dans un état sûr. Ceci revient à s'assurer que la Supervision AD respecte l'ensemble des exigences fonctionnelles et de sûreté qui spécifient son comportement. Ces deux types d'exigences sont émis respectivement par les AMS et par les pilotes SdF, dont les points de vue ne sont pas *a priori* convergents. Ce phénomène s'est concrétisé par les différences entre les exigences considérées : les exigences émanant des AMS sont allouées à la fonction AD globale et traitent des phases nominales de fonctionnement, alors que les

exigences produites par les pilotes SdF déterminent le comportement de trois sous-fonctions redondantes, assurant une continuité de service en cas de défaillances. En effet, la fonction AD est vue par les AMS comme une boîte noire, qui interagit avec son environnement, tandis que les pilotes SdF requièrent, pour des raisons de fiabilité, une structure redondante composée de trois sous-fonctions pouvant chacune prendre le contrôle du véhicule. Le problème majeur posé est donc de garantir que la fonction AD se trouve toujours dans un état sûr, ce qui revient, dans notre cas, à poser la question en ces termes :

Interrogation 1.1
Comment garantir que la Supervision AD respecte l'ensemble des exigences de comportement, fonctionnelles et de sûreté, qui lui sont allouées ? ♠

1.4 Verrous scientifiques

Pour présenter les différents verrous scientifiques que nous avons eus à lever, nous restituons brièvement le cheminement intellectuel suivi. Chaque solution contribuant à lever un verrou a généralement pour contrepartie d'en créer un ou plusieurs autres. La réflexion a donc débuté par la question principale suivante, qui correspond à la traduction scientifique, et plus générique, de l'Interrogation 1.1 [2] :

Interrogation 1.2
Comment mettre en cohérence les points de vue de l'AMS et du pilote SdF en phase amont de conception et dans un contexte opérationnel ? ♠

Le fait que le problème soit posé en phase amont de conception explique que nous avons traité des exigences. Par ailleurs, le contexte opérationnel évoqué correspond au cadre MBSE, en cours de déploiement dans l'entreprise hôte. Ceci est cohérent avec les travaux de recherche abordant cette question, qui préconisent, dans leur immense majorité, de se fonder sur l'approche MBSE. En effet, la convergence des points de vue est rendue possible, ou, du moins, facilitée par l'utilisation de modèles formels ou semi-formels manipulés dans ce cadre. Toutefois, pour que la mise en place de cette approche soit efficace, il faut que ces modèles reflètent les vues des deux métiers, *i.e.* que les exigences émises par les deux métiers soient vérifiées sur ces modèles. Il en découle une première question scientifique :

1. *Comment vérifier un ensemble d'exigences en phase amont de conception ?*

   Plusieurs techniques de vérification existent (simulation, tests, méthodes formelles). Nous avons opté pour une *vérification formelle* des exigences traitées car les modèles comportementaux manipulés dans un contexte MBSE sont sujets à ce type de vérification. De plus, les techniques de simulation et de tests ne suffisent pas à prouver exhaustivement la sûreté du véhicule autonome (Kalra et Paddock 2016). Enfin,

2. Les termes en caractères rouges sont soit des notions définies dans le Glossaire, placées en fin de ce mémoire, soit des renvois internes

la technique de vérification retenue établit un lien formel entre les exigences vérifiées (plus précisément les propriétés formelles déduites de ces exigences) et les modèles, ce qui contribue aux démonstrations de sûreté du véhicule. Or, nous ne disposions que d'*exigences informelles* exprimées en langage naturel sans règles précises (structure, syntaxe, sémantique), et dépourvues de contenus formalisés. Un second verrou scientifique est ainsi apparu :

2. *Comment formaliser un ensemble d'exigences et construire des modèles sur lesquels elles soient vérifiées ?*

   Outre le fait que les exigences initiales soient informelles, aucun modèle de comportement formel pour la Supervision AD n'était disponible. C'est pourquoi nous avons dû mettre au point une démarche permettant à la fois de formaliser les exigences, c'est-à-dire d'exprimer ces dernières sous la forme d'une ou plusieurs propriétés formelles, et de construire des modèles de comportement formels. Dans cette optique, nous avons pu nous appuyer sur une littérature riche mais avons dû également adapter ces travaux aux spécificités de notre cadre de travail. Après avoir obtenu, selon les deux points de vue métiers, des modèles de comportement et des exigences vérifiées formellement sur ces modèles, le tout validé par chacune des expertises, une dernière interrogation scientifique a été soulevée :

3. *Comment mettre en cohérence deux modèles partageant un formalisme commun mais bâtis selon deux perspectives différentes ?*

   Concernant ce dernier point, un vocabulaire partagé par les deux métiers a été, dans un premier temps, établi. L'utilisation d'un même formalisme pour les deux perspectives lors de la construction des modèles, a facilité cette tâche. De cette manière, les modèles produits « en silos » ont été enrichis. Les modèles reflétant le point de vue des AMS (respectivement des pilotes SdF) prennent désormais compte des aspects dysfonctionnels (respectivement fonctionnels). Enfin, la cohérence entre les vues locales (perspective SdF) et globales (point de vue des AMS) est vérifiée formellement (en utilisant pour ce faire l'opération de composition parallèle d'automates finis).

En conclusion, les premier et second verrous sont liés à l'aspect *vérification* de l'Interrogation 1.2, alors que le dernier verrou concerne la notion de *mise en cohérence*.

1.5 Principaux attendus

Les attendus sont d'ordre méthodologique et scientifique mais revêtent aussi un aspect opérationnel et pragmatique. Concernant l'aspect méthodologique, les objectifs sont les suivants :

— Élaborer une démarche de conception sûre de la Supervision AD, outillée et collaborative ;

— Intégrer cette démarche dans les processus existants de l'entreprise hôte mais également dans le cadre normatif en vigueur (normes ISO 26262 et ISO 15288) ;

— Définir précisément le rôle des experts métier dans ces processus.

Quant aux attendus scientifiques, ils concernent :

— La mise au point d'un processus de formalisation d'exigences de comportement faisant intervenir l'expertise et permettant de traduire un ensemble d'exigences informelles en un ensemble de propriétés formelles (exprimées en logique temporelle) ;

— La mise en œuvre de la vérification formelle sur les modèles de comportement construits et l'exploitation des résultats (contre-exemples) ;

— L'introduction progressive du contexte MBSE par la création d'un lien formel entre les exigences informelles et les modèles formels, assurant également une traçabilité des exigences ;

— L'élaboration d'une méthode de convergence des modèles reflétant les vues des AMS et des pilotes SdF.

Enfin, étant donné le contexte CIFRE de cette thèse, des objectifs plus opérationnels et pragmatiques ont aussi été déterminés :

— La démarche proposée doit être construite et directement applicable avec des données réelles d'un projet industriel ;

— Les travaux doivent être réutilisables ;

— La démarche doit être comprise, acceptée et adoptée par les acteurs industriels concernés.

1.6 Plan de thèse

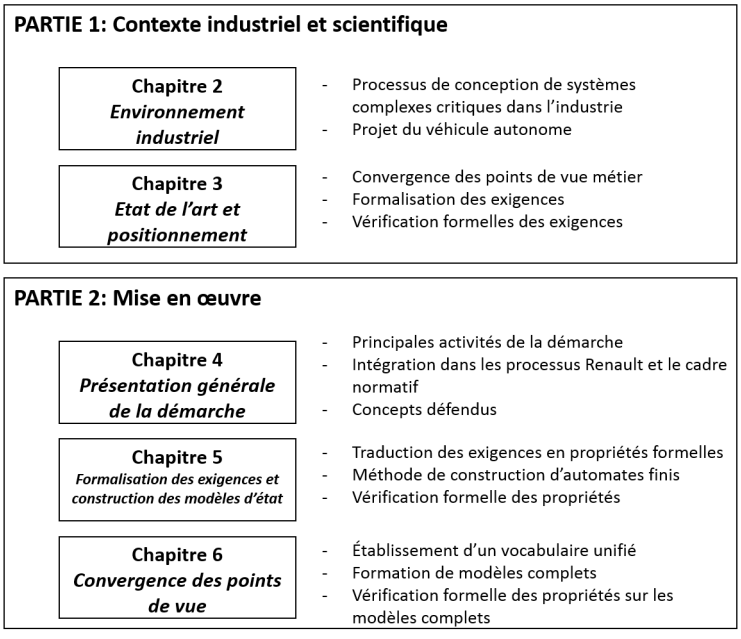


FIGURE 1.1 – Plan de la thèse

Le contenu de ce mémoire est décomposé en deux parties principales (figure 1.1) : une première partie présentant le **contexte industriel et scientifique** et une seconde dédiée à la **mise en œuvre** de la démarche proposée. Plus précisément :

1. **Partie I : Contexte industriel et scientifique** : cette première partie est constituée de deux chapitres :
   - *Chapitre 2 : Environnement industriel* : le cadre dans lequel les travaux de thèse se sont déroulés est présenté. Tout d'abord les évolutions de l'automobile conduisant au projet du véhicule autonome sont rappelées. Cet historique est centré sur l'introduction progressive de systèmes embarqués électroniques critiques du point de vue de la sûreté, dans les véhicules. L'augmentation du nombre de ces systèmes nécessite qu'ils soient agencés et organisés selon une architecture, appelée architecture Électronique Embarquée. L'automobile est ainsi devenu un système complexe, au sens de l'Ingénierie Système. Le cadre normatif ainsi que les processus déployés pour concevoir et réaliser ce type de systèmes (complexes et critiques) sont ensuite présentés pour les différents secteurs industriels des transports concernés (aéronautique, ferroviaire, automobile) ainsi qu'au sein du groupe Renault. Enfin, ce chapitre est conclu par la description du projet du véhicule autonome en général, puis, plus particulièrement chez Renault. Ces considérations permettent, en dernier lieu, d'exposer la problématique ainsi que les questionnements de recherche ;
   - *Chapitre 3 : État de l'art et positionnement* : les principales voies explorées dans les travaux de recherche pour résoudre les questionnements sont décrites. Tout d'abord, nous nous intéressons aux études qui abordent le problème de convergence des points de vue des AMS et des pilotes SdF dans un contexte industriel. La grande majorité de ces travaux appelle à l'utilisation du cadre méthodologique de l'Ingénierie Système et, plus spécifiquement, de la mise en œuvre de l'Ingénierie Système Basée sur les Modèles. En effet, la convergence passe par les modèles qui sont formels, ou semi-formels. Néanmoins, une grande part de ces travaux présuppose d'un contexte MBSE préétabli, ce qui ne correspond pas à notre contexte de travail. Afin de construire les modèles formels, une activité de formalisation des exigences, initialement informelles, est requise. Après avoir présenté les travaux qui contribuent à répondre à cette question, nous avons pu en conclure que peu d'approches de formalisation des exigences étaient également centrées sur la problématique de mise en cohérence des points de vue métier de l'AMS et du pilote SdF. En définitive, l'originalité de nos travaux, au vue de l'état de l'art, réside dans la proposition d'un processus détaillé de formalisation et de construction de modèles d'état, faisant intervenir l'expertise et répercutant les conséquences de l'activité de formalisation sur la formulation des exigences. En outre, les impacts de la confrontation des deux points de vue métier sur les exigences et les modèles sont analysés et exploités.
2. **Partie II : Mise en œuvre** : cette seconde partie, qui contient les contributions de ce travail, s'articule selon les trois chapitres suivants :
   - *Chapitre 4 : Présentation générale de la démarche* : après avoir rappelé le cas d'étude, nous donnons un premier aperçu global de la démarche proposée. Globalement, cette démarche se déroule en deux temps. Une première phase est dédiée à la formalisation des exigences et à la construction de deux ensembles de modèles d'état : l'un conforme au point de vue des AMS, et, l'autre, à celui des pilotes SdF. Puis une seconde phase, de convergence, vise à construire un modèle complet validé conjointement par les deux expertises. L'intégration de cette démarche,

dans le cadre normatif global, et au sein de l'appareil documentaire Renault, est ensuite présentée. Enfin, les principaux concepts défendus dans ce mémoire sont exposés et comparés aux propositions de la littérature ;

— *Chapitre 5 : Formalisation des exigences et construction des modèles d'état* : le processus de formalisation des exigences et de construction des modèles d'état est présenté. Il est appliqué aux exigences spécifiant le comportement de la Supervision AD émises par les AMS et par les pilotes SdF. Ce processus débute par une activité de sélection des exigences qui a pour objectif de retenir seules les exigences effectivement vérifiables sur les modèles. Des groupes d'exigences sont alors créés dans le but de réduire le nombre de spécifications formelles possibles à partir des exigences informelles retenues. Les énoncés des exigences contenues dans ces groupes sont ensuite précisés et complétés. Les hypothèses formulées pour effectuer ces deux activités sont soumises à la validation par les AMS, d'une part, et par les pilotes SdF, d'autre part. Ensuite, les modèles d'état sont construits à partir de ces groupements d'exigences et les propriétés formelles sont parallèlement élaborées. Dans un dernier temps, les propriétés sont vérifiées formellement sur les modèles construits et les résultats obtenus validés, en silos, par les AMS et les pilotes SdF ;

— *Chapitre 6 : Convergence des points de vue* : durant ce chapitre final, les deux perspectives sont mises en cohérence. L'utilisation d'un formalisme commun pour exprimer les deux vues facilitent sensiblement le travail de confrontation et de convergence. Les objets manipulés sont désormais de même type. Ainsi les écarts sémantiques, par exemple, sont détectables plus aisément que par la comparaison de deux listes d'exigences textuelles. La première étape pour la mise en cohérence consiste à établir un vocabulaire unifié pour les deux disciplines d'ingénierie. Les modèles résultants de la perspective des AMS sont enrichis par des considérations relatives à la SdF. Les modèles obtenus spécifient donc la Supervision AD globale, en prenant en compte les deux points de vue métier. Par la suite, les modèles de sûreté sont consolidés par la perspective des AMS. La cohérence entre les vues locales, ainsi obtenues, et la vue globale est ensuite vérifiée formellement à l'aide de l'opération de composition parallèle d'automates finis. En dernier lieu, l'ensemble des exigences est vérifié sur les modèles complets.

Première partie

Contexte industriel et scientifique

2
Environnement industriel

Sommaire

2.1 Architecture Electronique Embarquée (E/E)

2.1.1 Équipements E/E dans les automobiles : historique

Les années 1980 ont vu l'introduction des premiers systèmes électroniques embarqués dans les automobiles (Maurer et Winner 2013). Le système d'allumage électronique ainsi que l'AntiBlockierSystem (ABS) figurent parmi les premiers équipements électroniques, développés à la fin des années 1970. Le premier système permet d'accroître l'efficacité énergétique en enflammant le carburant au moment optimal, tandis que le second contribue à la réduction, voire la suppression, du patinage en cas de freinage soudain. Ces deux exemples illustrent deux intérêts de ces systèmes : l'amélioration des performances dynamiques des véhicules et la sécurisation accrue des utilisateurs. Progressivement, un nombre de plus en plus élevé d'équipements de ce type sont apparus dans les véhicules de série. Les principaux domaines améliorés sont la **sécurité**, avec les systèmes de régulation de vitesse (régulateur et limiteur de vitesse), d'aide au freinage et maintien de la trajectoire (ABS, Electronic Stability Program (ESP), aide au freinage d'urgence), d'alerte (détection de piétons, de fatigue du conducteur, d'obstacles en angles morts) ; le **confort** avec la direction assistée, la climatisation automatique, le Global Positioning System (GPS) ; l'**automatisation des tâches** telles que l'allumage des feux de croisement, l'activation des essuie-glaces ; l'**aide au stationnement** avec les radars, caméras de recul, stationnement automatique (Mécatronique 2014).

La plupart de ces systèmes embarqués sont dédiés à l'aide à la conduite. Parmi ces systèmes, certains interviennent en tant que simple assistance, comme l'aide au maintien dans la voie par exemple. Dans ce cas, le conducteur est capable de s'opposer aux forces exercées par les actionneurs du véhicule. En revanche, d'autres systèmes, réalisant des fonctionnalités comme le stationnement automatique ou le freinage d'urgence en cas de détection d'obstacles, nécessitent des braquages puissants, difficilement maîtrisables par le conducteur (Judalet 2016) (surtout en cas de déclenchement intempestif). Ces systèmes sont dits **critiques** du point de vue de la sûreté car leur dysfonctionnement peut avoir des conséquences graves pour les personnes et/ou l'environnement.

De quelques systèmes Électroniques Embarqués (E/E) isolés à l'intégration d'un nombre important de calculateurs électroniques équipant les véhicules récents (une quarantaine de calculateurs pour la dernière version de Renault Clio ou encore 100 et 105 pour, respectivement, la BMW i3 et BMW i8 de 2015 (Leibinger 2015)), l'automobile est ainsi passée en 30 ans d'un ensemble purement mécanique à un système mécatronique. Ce constat est renforcé par la valeur croissante des équipements E/E dans le coût global d'un véhicule. Celui-ci n'a cessé de croître depuis 60 ans, comme l'illustre le diagramme représenté sur la figure 2.1 (Mckinley 2016). Par ailleurs, les fonctionnalités offertes par ces systèmes deviennent imbriquées et interdépendantes, il n'est plus possible de les concevoir de manière indépendante. Il est en conséquence nécessaire de s'interroger sur l'interconnexion de l'ensemble des équipements E/E, c'est-à-dire l'architecture E/E.

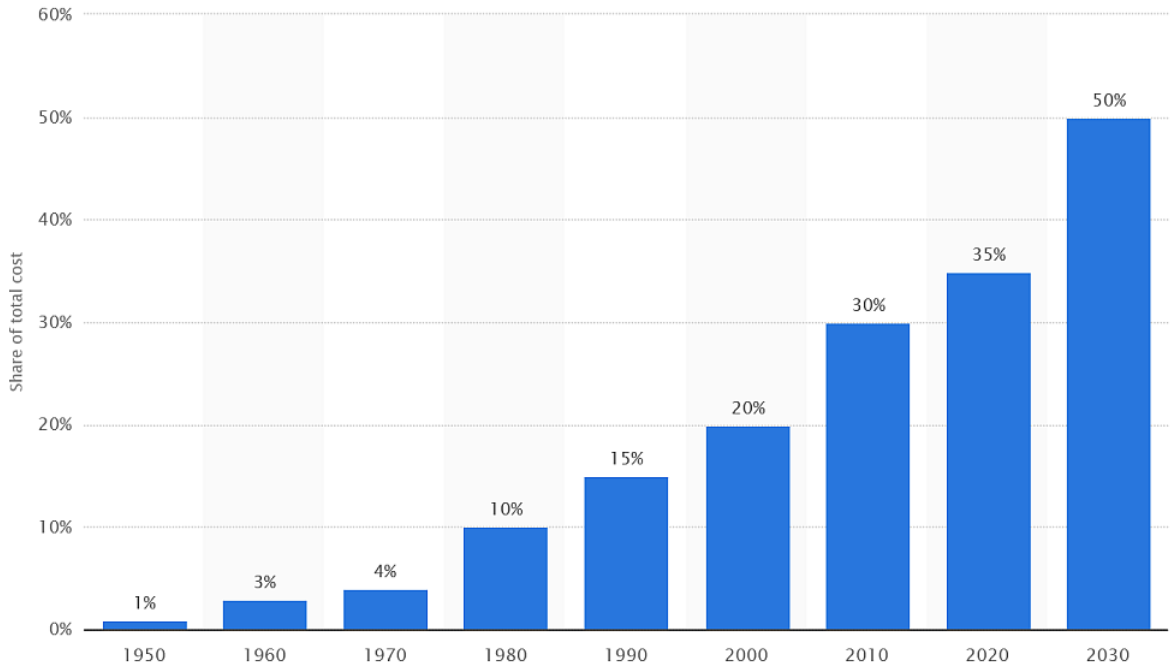


FIGURE 2.1 – Coût de l'électronique embarquée par rapport au coût total des automobiles (MCKINLEY 2016)

2.1.2 Évolution de l'architecture E/E

2.1.2.1 D'une architecture « fédérée » à une architecture « intégrée »

Une architecture E/E est définie comme l'agencement des composants électroniques telles que des Unités de Commande Électronique (UCE), capteurs ou actionneurs pour réaliser un ensemble de prestations, processus et services (STINEAU 2015). Les ingénieurs en charge de la conception de l'architecture E/E doivent trouver le meilleur compromis possible afin de satisfaire au mieux des exigences, parfois contradictoires (fiabilité/coût, volume physique/performances...). L'augmentation considérable du nombre de systèmes embarqués électroniques évoquée dans la partie précédente a singulièrement modifié les architectures E/E au cours du temps. Les travaux de Gilles Moncelet (MONCELET 1998) fournissent une rétrospective intéressante des architectures E/E implantées dans les automobiles. Afin de caractériser ces architectures, les cinq attributs suivants ont été proposés :

(a) *Allocation des UCE*
- UCE dédiées : chaque UCE réalise une fonction spécifique ;
- UCE banalisées : les UCE peuvent réaliser plusieurs fonctions.

(b) *Accès aux entrées/sorties*
- Accès spécifique : chaque UCE possède sa propre liaison avec les capteurs et actionneurs ;

— Accès partagé : la communication entre les UCE et les capteurs/actionneurs est assurée par un médium commun.

(c) *Type d'allocation de redondance*

— Redondance hétérogène : chaque fonction est caractérisée par son degré de redondance propre ;

— Redondance homogène : degré de redondance identique pour toutes les fonctions.

(d) *Organisation fonctionnelle*

— Fédération : les fonctions sont indépendantes ;

— Intégration : les fonctions interagissent entre elles pour atteindre des objectifs partagés.

(e) *Placement des UCE*

— Centralisation : UCE localisées à un emplacement unique du véhicule ;

— Distribution : UCE réparties sur l'ensemble du véhicule.

Ces différents attributs définissent un panel d'architectures : d'une architecture totalement *fédérée* à une architecture complètement *intégrée*. Les architectures E/E implémentées dans les automobiles suivent cette évolution. Les premières architectures E/E ont été des architectures fédérées. Ceci signifie que les UCE sont vues comme des boîtes noires implémentant des fonctions indépendantes. La détection et le traitement d'une défaillance interne d'une UCE se font localement. La localisation des UCE est généralement distribuée, ce qui rend la communication avec les capteurs et actionneurs plus aisée. Chaque fonctionnalité peut ainsi être pensée, conçue et réalisée de manière indépendante. Ce type d'architecture est bien adapté lorsque peu de fonctions sont implémentées.

En revanche, lorsque les fonctionnalités se multiplient et se complexifient, il est nécessaire d'introduire peu à peu l'intégration fonctionnelle : une UCE réalise plusieurs fonctions (de la même manière, plusieurs UCE peuvent être utilisées pour remplir une même fonction). Ce type d'architecture permet de diminuer le nombre d'UCE utilisées, donc les coûts par la même occasion. Par ailleurs, ceci facilite une démarche d'analyse de Sûreté de Fonctionnement (SdF) globale en harmonisant le comportement en cas de défaillance. En revanche, la cohabitation de fonctionnalités de criticités différentes ainsi que la forte interdépendance et imbrication fonctionnelle compliquent considérablement ces mêmes analyses de SdF.

2.1.2.2 Bilan : une interconnexion de systèmes complexes et critiques

Les systèmes embarqués électroniques dans les véhicules offrent des fonctionnalités nouvelles et de plus en plus élaborées. Les systèmes qui mettent en œuvre ces fonctionnalités sont caractérisés par leur :

(a) *Complexité* : sur le plan *technique*, avec l'implémentation de fonctions interdépendantes collaborant pour atteindre des objectifs communs ; et sur le plan *organisationnel* étant donné le nombre important d'acteurs impliqués (différents acteurs métiers : ingénierie, achat, marketing... appartenant à différentes entités : constructeur, fournisseurs, sous-traitants) dans leur réalisation ;

(b) *Criticité* : certains systèmes requièrent des actions puissantes de la part des actionneurs, non contrôlables par le conducteur.

La maîtrise de ces deux aspects relève de deux disciplines d'ingénierie distinctes : l'Ingénierie Système (IS) et la SdF. La section suivante est consacrée à la présentation des processus cadrant les activités de ces deux ingénieries.

2.2 Processus de conception mis en œuvre

2.2.1 Conception de systèmes critiques

2.2.1.1 Gestion de la complexité

Cadre normatif de l'Ingénierie Système

La pratique historique de l'ingénierie dans l'industrie est basée sur les documents (Micouin 2011). Le principal moyen de communication et de partage d'informations entre les différentes *parties prenantes* (notion définie ci-après) repose sur les documents, qu'ils soient sous forme de papier, fichiers ou bases de données électroniques, et, également, sur les échanges verbaux. Ce type d'ingénierie est adapté au développement de systèmes d'une complexité raisonnable mais s'avère inapproprié pour la conception de systèmes plus complexes. Leur réalisation implique de nombreux acteurs dont les activités sont difficiles à coordonner. De plus, des accidents industriels tragiques, tels que l'explosion du lanceur spatial Ariane 5 en 1996 (Lions et al. 1996) ou encore, très récemment, la perte du satellite russe Meteor M2-1 (Le Monde 2017) sont dus à des défauts systémiques. Il s'agit de lacunes dans l'expression des besoins, d'imprécision des spécifications, du manque de justification et validation des solutions envisagées. Dans le cadre d'une ingénierie basée sur les documents, ces défauts sont fréquents et peuvent être lourds de conséquences. C'est pour ces raisons qu'une science nommée Ingénierie Système (IS) a été développée (à l'origine par la National Aeronautics and Space Administration (NASA) et l'United States Air Force (USAF)) à la fin des années 1950. Une organisation internationale, fondée en 1991 et appelée International Council On Systems Engineering (INCOSE), est en charge de capitaliser et de diffuser les activités intellectuelles ainsi que l'échange des bonnes pratiques de l'IS. L'Association Française d'Ingénierie Système (AFIS), son chapitre français créé en 1999, donne la définition suivante de l'IS (Fiorèse et Meinadier 2012) :

Définition 2.1
Ingénierie Sytème *: l'Ingénierie Système (ou ingénierie de systèmes) est une démarche méthodologique générale qui englobe l'ensemble des activités adéquates pour concevoir, faire évoluer et vérifier un système apportant une solution économique et performante aux besoins d'un client tout en satisfaisant l'ensemble des parties prenantes.* ♣

Cette définition en appelle une autre (toujours selon l'AFIS (Fiorèse et Meinadier 2012)) :

Définition 2.2
Partie prenante *: partie intéressée par l'utilisation et l'exploitation du système (voire par ses impacts sur son environnement), mais aussi participant à sa conception, sa production, son déploiement, sa commercialisation, son maintien en condition opérationnelle et son retrait de service.* ♣

Le cadre méthodologique de l'IS est défini par les trois normes suivantes :

- **ISO/IEC 15288** (ISO/CEI 2008) : cette norme établit un cadre général de description des processus mis en œuvre au long des cycles de vie d'un système conçu par l'Homme. Elle définit un ensemble de processus ainsi qu'une terminologie associée. Ces processus sont de types différents : organisationnel, managérial et technique. La dernière version de cette norme a été publiée en 2015. Néanmoins, les processus effectivement en vigueur chez Renault lors des travaux de thèse (débutés en 2015) reposent sur la version précédente de cette norme qui date de 2008 ;
- **EIA 632** (Electronic Industries Alliance 2003) : ce standard décrit une approche systématique de conception d'un système qui intègre et formalise les bonnes pratiques. Cette norme se focalise sur les processus de conception et les processus support. La dernière version disponible a été publiée en 2003 ;
- **IEEE 1220** (IEEE 2005) : issue du standard militaire MIL STD 499B, cette norme de l'Institute of Electrical and Electronics Engineers (IEEE) est centrée sur les processus techniques d'IS qui vont de l'analyse des exigences jusqu'à la définition physique du système. Les processus d'analyse des exigences, d'analyse fonctionnelle et d'allocation des exigences sont finement détaillés. La dernière version applicable de cette norme date de 2005.

Deux notions récurrentes apparaissent dans ces 3 normes et méritent à ce titre d'être définies :

Définition 2.3
Processus *: ensemble d'activités corrélées ou en interaction qui utilise des éléments d'entrée pour produire un résultat escompté (d'après la norme ISO 9000 :2015 (*ISO 2015*)).* ♣

Définition 2.4
Activité *: ensemble d'actions qui consomment du temps et des ressources et dont l'exécution est nécessaire pour obtenir ou contribuer à la réalisation d'un ou de plusieurs résultats (d'après la norme ISO 15288 :2008).* ♣

Les trois normes de l'IS ont des périmètres d'application différents. L'ISO 15288 a le spectre le plus large car elle couvre l'ensemble du cycle de vie du système. En revanche,

les normes EIA 632 et IEEE 1220 détaillent certains processus. Ces différentes normes ne sont pas au cœur de ces travaux de thèse. Nous n'entrerons donc pas dans le détail de leur description, que l'on peut en revanche trouver dans les travaux de Jean Verries (Verries 2010). Outre ces trois normes, nous pouvons également citer le *« Guide to the Systems Engineering Body of Knowledge »* (SEBoK 2017) qui n'est pas un standard mais un recueil de bonnes pratiques.

Le cycle de vie préconisé par l'ISO 15288 est le cycle en V, conventionnel dans l'industrie (Forsberg, Mooz et Cotterman 2005). Cette norme définit huit processus principaux, déployés au cours du déroulement de ce cycle comme l'illustre la figure 2.2. Les deux processus, identifiés en caractères gras : le **Processus d'analyse des exigences** et le **Processus de conception d'architectures**, délimitent le périmètre des travaux présentés dans ce mémoire. Nous en donnons ici une première description générale (Standards Council of Canada 2002). Le premier processus cité consiste à traduire les exigences des parties prenantes en exigences techniques (ou exigences système). En effet les exigences des parties prenantes ne sont pas toujours exprimées dans des termes techniques. Durant ce processus, les différents types d'exigences doivent également être déterminés. Le second processus vise à définir l'architecture fonctionnelle et l'architecture physique du système. Il s'agit de déterminer une ou plusieurs architectures solutions qui respectent les exigences issues du processus d'analyse des exigences. Les exigences sont alors allouées à des éléments architecturaux. Ceci signifie que ces éléments satisfont à ces exigences.

Nous reviendrons plus en détail sur ces deux processus en présentant leur mise en œuvre au sein de l'entreprise Renault, ainsi que sur la définition des deux types d'architecture évoqués (voir paragraphe 2.2.2.3). Par ailleurs, il est important de noter que la linéarité du cycle en V n'est qu'apparente. Il existe en effet de nombreuses rétroactions des phases aval vers les phases amont, suite à des découvertes tardives d'erreurs notamment. C'est pourquoi d'autres cycles de conception ont été proposés tels que le modèle incrémental, le cycle en Y (Mihal et al. 2002) ou encore le modèle à spirale (Boehm 1988).

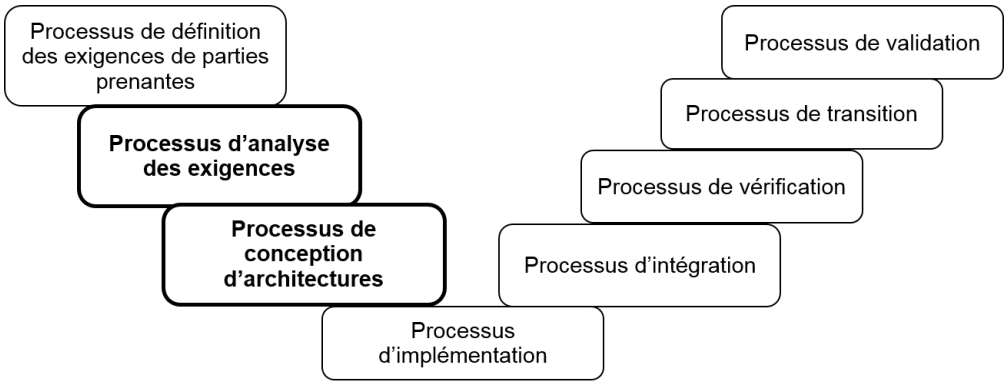


Figure 2.2 – Cycle en V de l'Ingénierie Système selon la norme ISO 15288

Dans le cadre de ce mémoire, nous ne présenterons pas l'ensemble des activités préconisées et détaillées par l'IS. En effet, le champ de la démarche méthodologique proposée par l'IS est très large, car il couvre l'ensemble du cycle de vie d'un système complexe,

de ses premières conceptualisations jusqu'à son retrait définitif de service. Le lecteur intéressé pourra se référer aux travaux (Verries 2010 ; Guillerm 2011 ; Mauborgne 2016 ; Legendre 2017) qui fournissent des panoramas complets de l'IS. Par ailleurs, la démarche proposée par l'IS est théorique. Sa mise en pratique repose notamment sur des supports opérationnels, directement utilisables par les acteurs impliqués. C'est justement pour répondre à cet impératif que l'*Ingénierie Système basée sur les Modèles* a été développée.

Ingénierie Système Basée sur les Modèles

La démarche méthodologique de l'IS doit, pour convaincre et être adoptée par les industriels, être appliquée et fournir des résultats. Le moyen le plus approprié est de rendre cette démarche opérationnelle. Pour ce faire, des modèles, dont le rôle est de concrétiser et de tracer les différentes activités de l'IS (Idasiak et Kajdan 2014) ont été établis. Ces modèles vont ainsi représenter les besoins auxquels répond le système, ses fonctions, les exigences qui lui sont allouées, les contraintes qu'il subit, ainsi que les constituants du système et leurs comportements. Un langage de modélisation, doté d'une syntaxe et d'une sémantique définies, est requis pour élaborer ces modèles, qui sont par conséquent formels ou semi-formels. Le langage de modélisation des systèmes le plus répandu dans le cadre de l'Ingénierie Système Basée sur les Modèles (ou Model Based Systems Engineering (MBSE) en anglais) est le Systems Modeling Language (SysML), promu et standardisé par l'Object Management Group (OMG). Ainsi, comme l'illustre la figure 2.3, les industriels concevant des systèmes complexes passent progressivement d'une ingénierie basée sur les documents (pratique traditionnelle) à une ingénierie basée sur les modèles (Piques et Andrianarison 2011).

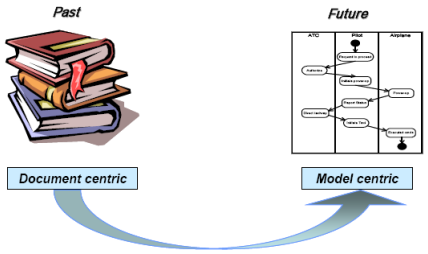


Figure 2.3 – Evolution du type d'ingénierie (Micouin 2011)

L'IS constitue donc un cadre méthodologique bien structuré contribuant à résoudre les problématiques de coopérations multidisciplinaires émergeant lors de la conception et réalisation de systèmes complexes. Néanmoins, comme nous l'avons vu dans la section 2.1.1, les questions liées à la sûreté de ces systèmes sont devenues particulièrement critiques dans le secteur automobile. De façon générale, la sûreté des systèmes électroniques embarqués complexes est une question à laquelle les domaines dans lesquels ces systèmes sont utilisés sont confrontés depuis de nombreuses années. Les aspects relatifs à la sûreté font en conséquence l'objet d'un cadre normatif.

2.2.1.2 Gestion des risques

Le domaine de l'aéronautique a été pionnier en matière de normes relatives à la sûreté des systèmes embarqués. Leur rédaction a débuté au début des années 1980 : en 1982 a été publiée la première version de la norme DO 178 (RTCA-Inc 2012) (la dernière version date de 2012). Cette norme fixe les conditions de sécurité applicables aux logiciels critiques de l'avionique. Celles-ci déterminent notamment cinq niveaux de criticité, nommés Development Assurance Level (DAL), associés à des contraintes de développement plus ou moins strictes selon le niveau. Deux autres normes (ARP 4754 :2011 (SAE International 2010) et ARP 4761 :1996 (SAE International 1996)) émanent de la Society of Automotive Engineers (SAE) international et fournissent un ensemble de pratiques recommandées, organisées en processus et usant de techniques de modélisation partagées afin d'évaluer la sûreté des systèmes conçus.

Suite aux efforts initiés par le secteur aéronautique, l'Organisation internationale de normalisation (ISO) édite en 1999 le standard ISO/IEC Guide 51 (sa dernière version date de 2014 (ISO/CEI 2014)) qui, selon ses rédacteurs, « 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. » Ce guide vise à réduire le risque engendré par l'utilisation de produits, procédés ou services tout au long de leurs cycles de vie. La norme IEC 61508 (ISO/CEI 2000) traite plus particulièrement de la sécurité fonctionnelle des systèmes électriques, électroniques et électroniques programmables relatifs à la sûreté. Cette norme, intersectorielle, a donné lieu à plusieurs déclinaisons couvrant différents domaines, comme le montre la figure 2.4.

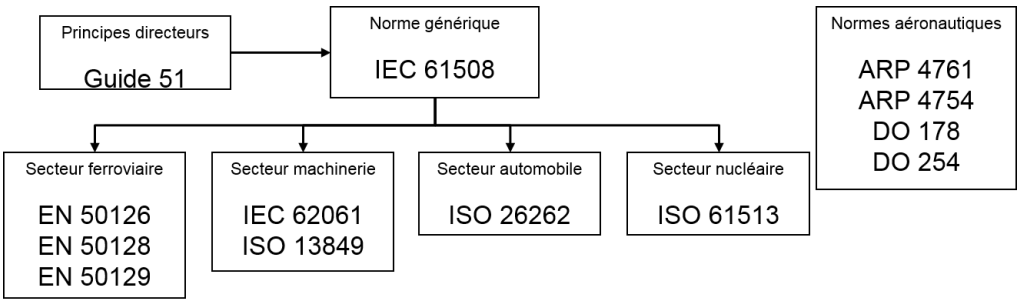


Figure 2.4 – Normes dérivées de l'IEC 61508, adapté de (Mauborgne 2016)

Pour le secteur automobile, qui nous intéresse, la norme de référence est l'ISO 26262 (ISO 2011), dont la dernière version est datée de novembre 2011. Cette norme est dédiée à la conception de systèmes automobiles embarqués électriques, électroniques et électroniques programmables relatifs à la sûreté. Les processus de déploiement de la SdF chez Renault se fondent sur cette version mais la prochaine parution est imminente (prévue pour 2018). Au même titre que la norme ISO 15288, cette norme préconise un cycle de conception, dont une version simplifiée est représentée sur la figure 2.5. Nous ne décrirons pas le détail de chacune des activités qui sortent du périmètre des travaux de thèse. Seul le contenu des activités pertinentes dans le cadre des travaux menés (identifiées en caractères gras sur la figure 2.5) sera fourni dans les parties suivantes (voir paragraphe 2.2.2). Ce qu'il importe de relever ici est la coexistence de deux

cycles de conception - l'un rattaché à l'IS et l'autre au métier SdF - qui entretiennent des liens entre eux, sans pour autant que ces derniers ne soient explicités dans les normes présentées. Nous illustrons cette idée sur la figure 2.6 : les deux cycles de conception sont accolés et les liens entre les activités des deux métiers sont ébauchés. Le détail de ces interactions, et surtout les problèmes qu'elles peuvent engendrer, sera donné uniquement pour les activités qui nous intéressent (indiquées par les textes en caractères gras sur les figures 2.2 et 2.5). De façon générale, une coordination des activités des deux métiers est nécessaire. Cette coordination se traduit notamment par la nécessité de partager des données (documents, analyses, modèles) entre les acteurs de l'IS et ceux de la SdF.

A l'instar du cadre MBSE pour l'IS, une approche, nommée Model Based Safety Analysis/Assessment (MBSA), a été mise au point pour supporter certaines activités de SdF. Le principe du MBSA (Joshi, Whalen et Heimdahl 2006 ; Joshi et Heimdahl 2005) consiste à enrichir les modèles élaborés dans le cadre du MBSE avec les aspects dysfonctionnels. Par exemple, les modèles de fautes, pour chaque composant, sont ajoutés. Il est alors possible d'étudier la propagation de fautes dans une architecture donnée ou encore de générer automatiquement des analyses de SdF, telles que les AMDEC ou les AdD.

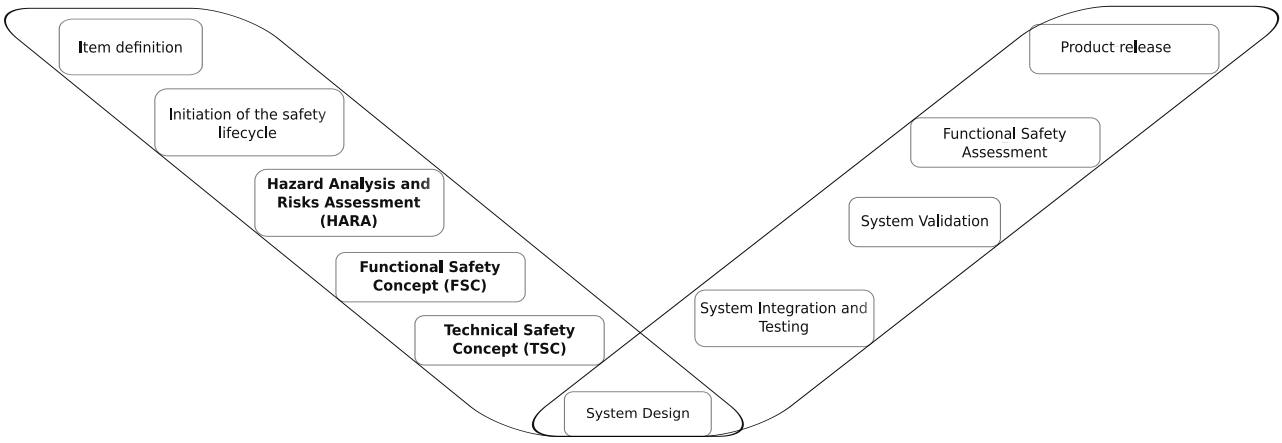


Figure 2.5 – Cycle de conception selon la norme ISO 26262, adapté de (ISO 2011)

Enfin, un standard, nommé AUTomotive Open System ARchitecture (AUTOSAR), émanant d'un consortium de plusieurs grands constructeurs et équipementiers a vu le jour en 2003. AUTOSAR propose une méthodologie de développement des systèmes électroniques embarqués dans les automobiles. L'objectif principal est de maîtriser la complexité croissante de ces systèmes (Fennel et al. 2006 ; Fürst et al. 2009). Pour ce faire, une architecture logicielle standardisée et ouverte a été définie à l'aide d'un ensemble de spécifications décrivant les modules logiciels de base. Un format d'échange standardisé entre ces modules a également été déterminé. Parmi les intérêts de cette initiative, on peut noter l'amélioration de la réutilisabilité des composants logiciels ainsi que leur intégration, et la réduction des coûts de conception. Néanmoins, le domaine d'application de ce standard concerne essentiellement le processus d'implémentation (voir figure 2.2). Or les travaux de thèse se situent en amont de ces processus. C'est la raison pour laquelle il ne sera plus fait référence à ce standard par la suite.

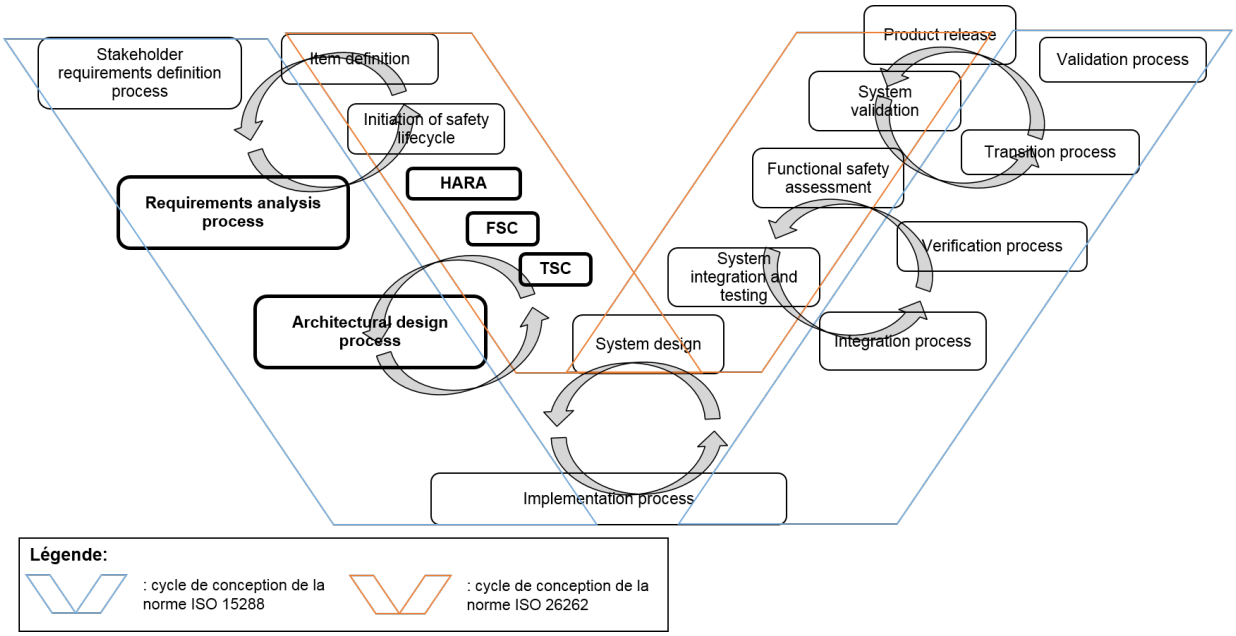


FIGURE 2.6 – Liens entre le cycle en V de la norme ISO 15288 et le cycle de la norme ISO 26262, adapté de (TAOFIFENUA 2012)

2.2.1.3 Conclusions relatives au domaine automobile

L'automobile étant un système complexe intégrant désormais un nombre élevé de systèmes embarqués électroniques, la démarche méthodologique de l'IS est progressivement adoptée par les différents constructeurs (LASALLE, PEUREUX et FONDEMENT 2011 ; FLORIAN et al. 2014), mais également leurs principaux fournisseurs (PIQUES et ANDRIANARISON 2011). Par ailleurs, certains de ces systèmes embarqués sont critiques du point de vue de la sûreté. Afin de guider la conception de tels systèmes, la norme ISO 26262 a été éditée en 2011. Cependant, les liens entre les normes encadrant les processus de l'IS et les processus décrits dans la norme ISO 26262 n'ont pas été clairement établis. Cette problématique, touchant l'ensemble des constructeurs automobiles, a été identifiée et traitée dans plusieurs travaux de recherche (TAOFIFENUA 2012 ; MAUBORGNE 2016). Notons qu'elle n'est pas propre au secteur automobile : ainsi les problèmes liés à l'intégration des activités de SdF dans les processus de l'IS ont été abordés dans le domaine aéronautique (GUILLERM 2011) et spatial (P. DAVID 2009 ; CRESSENT 2012), et, plus généralement sous un prisme méthodologique (GUILLERM 2011 ; GÜDEMANN 2011 ; LEGENDRE 2017). Si les travaux réalisés dans des domaines différents de l'automobile sont sources d'inspiration pour ce dernier, les contraintes propres à ce secteur compliquent leur adoption ou leur adaptation.

(MAUBORGNE 2016) souligne par exemple les différences entre les contextes automobile et aéronautique, que ce soit en termes de série (production de masse *versus* petites séries), de taille, ce qui facilite la redondance pour les aéronefs, ou encore de conducteurs (pilotes professionnels formés pour les aéronefs). Les spécificités des contraintes de l'automobile, telles que le temps de mise sur le marché (*« time to market »*), la pression forte des coûts, le volume disponible limité, l'extrême variabilité des conditions

(pays, lois, capacités des conducteurs, conditions climatiques) (Maurer et Winner 2013) compliquent également l'adoption de pratiques propres à des domaines connexes au secteur automobile. Par ailleurs, (Simonot-Lion et Navet 2006) insiste sur la lourdeur causée par l'application des normes aéronautiques qui, d'une part, accroissent considérablement le cycle de développement et, d'autre part, augmentent le coût, le poids et l'encombrement en raison du recours massif à la redondance matérielle. De ce fait, ces contraintes sont jugées totalement incompatibles avec celles du secteur automobile. Enfin, le rôle clef que jouent les équipementiers, ou fournisseurs de rang 1, doit être relevé. Ces collaborateurs ne se contentent pas de suivre un cahier des charges mais constituent des parties prenantes à part entière, forces de proposition dans la conception des systèmes (Chanaron 1995). Ainsi, l'ensemble des grands constructeurs automobiles s'est plutôt positionné comme assembleur (Moati 2001).

Cette question du désalignement entre des processus de l'IS et ceux relatifs à la sûreté affecte tous les domaines industriels produisant des systèmes complexes critiques. Toutefois, il convient de présenter le contexte et les processus particuliers mis en œuvre dans l'entreprise au sein de laquelle s'est déroulée cette thèse.

2.2.2 Processus de conception chez Renault

La première étape du déploiement de l'IS consiste en un effort de formalisation de l'organisation globale de l'entreprise visant à clarifier les finalités suivies, les processus et méthodes mis en œuvre, les acteurs impliqués, les jalons à respecter ou encore les livrables à produire. Ce travail s'est concrétisé par l'élaboration du Système de Conception Renault (SCR). Il s'agit d'un ensemble documentaire structuré qui définit la logique de développement des véhicules, les processus opérationnels constituant les référentiels des activités de l'ingénierie et les méthodes/outils d'ingénierie utilisés tout au long du cycle de développement. Nous ne présenterons naturellement pas l'ensemble de ce système. Cependant, nous en donnerons une vue globale dans le but de comprendre comment les travaux de thèse s'y inscrivent, et de délimiter leur périmètre afin d'en saisir leur portée.

2.2.2.1 Vue hiérarchique

La figure 2.7 représente la structuration du SCR de manière hiérarchique (Groupe Renault - Direction de la Qualité et de la Satisfaction Client 2018). Dans un premier temps, la stratégie globale de l'entreprise est déterminée puis déclinée en différentes finalités. Pour atteindre ces dernières, un ensemble de processus opérationnels est déployé. Cette notion est définie chez Renault de la façon suivante :

Définition 2.5
Processus opérationnel : processus constituant l'ossature de la logique de développement et décrivant l'enchaînement et la synchronisation des activités métiers ainsi que leur contribution aux livrables de l'ingénierie. ♣

Notons que deux autres types de processus sont identifiés chez Renault : les processus de management et les processus support. Néanmoins, dans le cadre de ce mémoire, nous nous focaliserons sur les processus opérationnels, simplement dénommés processus par la suite.

À chaque processus est associée une *Règle* qui en décrit les activités globales (voir figure 2.7). Les *Procédures* détaillent ensuite une ou plusieurs de ces activités, consistant en une série d'actions. Chaque action donne lieu à une *Instruction*. Enfin, les instructions renvoient à un ou plusieurs *Formulaires* qui sont des documents modèles génériques (*templates*) des livrables à fournir. Le niveau de détail de description des activités à réaliser dépend en fait de la maîtrise que l'entreprise a de ces activités. Puisque certaines des activités de Renault sont réalisées par des fournisseurs (voir paragraphe 2.2.1.3), ces dernières n'ont ainsi pas à faire l'objet de formulaires spécifiques. Renault s'assure en revanche de la conformité des produits (livrables, systèmes, composants) de ces équipementiers aux cahiers des charges qu'ils doivent respecter. Pour ce faire, des *design reviews* (réunions périodiques avec le fournisseur au cours desquelles la satisfaction des exigences est examinée) ou encore des audits réguliers sont organisés.

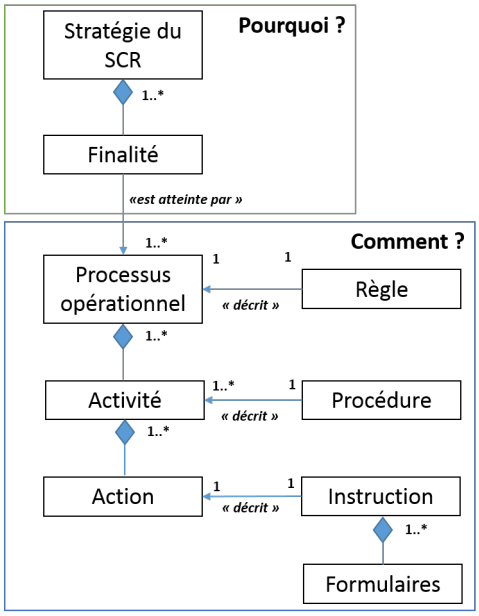


FIGURE 2.7 – Vue hiérarchique du Système de Conception Renault

2.2.2.2 Vue chronologique

A la vue hiérarchique, qui détaille les processus opérationnels à mettre en œuvre par les différents métiers, s'ajoute une vue chronologique qui précise l'ordonnancement des différentes phases de conception d'un véhicule : des premières idées, en phase amont de conception, jusqu'à la production de masse. Trois phases majeures sont distinguées (GROUPE RENAULT - DIRECTION DE LA QUALITÉ ET DE LA SATISFACTION CLIENT 2016) :

1 *Phase amont* : durant cette phase, le *business model*, les concepts de véhicule et les solutions possibles sont cadrés. Les concepts de véhicule incluent les prestations à valeur ajoutée ou encore les thèmes design qui font l'objet de démonstrateurs ;

2 *Phase de développement* : l'objectif de cette phase est de garantir une conception « bonne du premier coup ». Un véhicule entièrement numérique est construit et analysé. Toutes les pièces sont étudiées et documentées. Les performances usine et de l'après-vente doivent être confirmées ;

3 *Phase d'industrialisation* : la validation du véhicule est entreprise au cours de cette phase par niveau d'intégration successif : composant/ système/ synthèse véhicule. Ces validations se font au maximum sur des bancs systèmes afin de limiter le nombre de véhicules de validation. Les moyens de fabrication sont transférés sur le site définitif de production. La montée en cadence est préparée.

Chaque phase est close par un jalon et ponctuée de jalons intermédiaires. Les processus opérationnels sont déployés selon ces phases et des versions de livrables différentes sont attendues selon les jalons. Le cycle de conception en vigueur chez Renault repose donc bien sur le cycle en V classique présenté dans le paragraphe 2.2.1.1. Le lien entre ces deux cycles est représenté sur la figure 2.8.

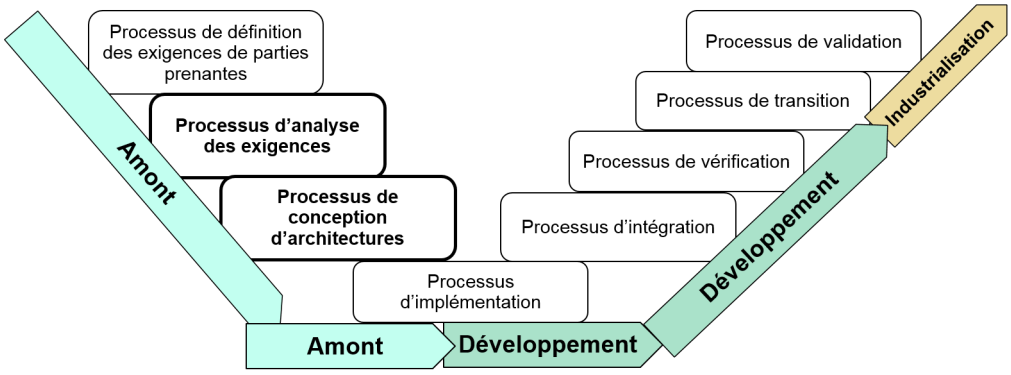


FIGURE 2.8 – Lien entre le cycle en V Renault et le cycle de l'ISO 15288

La description des trois phases constitue naturellement une vue simplifiée du cycle de conception d'un véhicule qui comprend, en réalité, un nombre de phases et de jalons beaucoup plus élevé. Toutefois, cette présentation a pour vocation de circonscrire le périmètre des travaux de thèse. C'est pourquoi l'ensemble des sous-phases et jalons intermédiaires ne sont pas décrits dans ce mémoire.

Parmi les finalités qui déclinent la stratégie globale de l'entreprise (figure 2.7), les travaux entrepris relèvent de la finalité qui consiste à fournir aux usines les éléments pour fabriquer une voiture conforme, vendable et réparable. Plus particulièrement, ils s'inscrivent dans le cadre du processus opérationnel nommé « *Concevoir et valider les systèmes* ». Du point de vue chronologique, les travaux ont été menés en phase amont de conception. L'intitulé du processus opérationnel de travail fait apparaître la notion centrale de système, qui prend, chez Renault, le sens suivant :

Définition 2.6
Système : un système est un arrangement d'éléments en interaction, organisé pour atteindre un ou plusieurs objectifs définis. ♣

Remarque 2.1
Cette définition est extraite de la norme ISO 15288 :2002. A titre d'illustration, la Direction, le Freinage ou encore la Surveillance de pression des pneus sont des systèmes d'un véhicule Renault. ♢

2.2.2.3 Intégration des travaux de thèse

Chez Renault, les deux acteurs majeurs prenant part au déploiement des processus d'analyse des exigences et de conception des architectures (ciblés sur la figure 2.2) sont : les Architecte Métier Système (AMS) d'une part, et les pilotes SdF, d'autre part. Les premiers assument les activités décrites par la démarche de l'IS, tandis que les seconds s'attellent aux analyses de risques. Les activités qu'ils mettent en œuvre sont détaillées dans des procédures internes chez Renault (décrivant les processus, voir figure 2.7), basés sur les normes en vigueur. Avant de présenter les activités de ces acteurs en rapport avec les travaux de thèse, deux notions centrales sont à définir : celle d'**exigence** et d'**architecture**. Il s'agit effectivement de données communes aux AMS et aux pilotes SdF qu'ils manipulent tout au long du cycle de conception. Ils sont donc naturellement amenés à partager ces données.

La définition de la notion d'exigence adoptée chez Renault est celle fournie par la norme EIA 632 (selon les sources, la définition de la notion d'exigence varie) :

Définition 2.7
***Exigence** : énoncé qui prescrit une fonction, une aptitude, une caractéristique ou une limitation à laquelle doit satisfaire un produit ou un processus dans des conditions d'environnement données.* ♣

Remarque 2.2
Des attributs obligatoires, tels qu'un identifiant, une version, un intitulé, une description, une source, etc. sont associés à une exigence. En revanche, il n'existe pas d'impératifs ou de recommandations sur l'énoncé de l'exigence en lui-même, qui est exprimé généralement en langage naturel, accompagné parfois de schémas explicatifs. ♢

La notion d'architecture est très large mais les deux types d'architecture qui nous intéressent sont les suivants, définis de cette manière chez Renault :

Définition 2.8
***Architecture fonctionnelle** : arrangement de fonctions et de flux présenté sous un point de vue statique.* ♣

Définition 2.9
Architecture physique *: arrangement statique de composants, de flux physiques et d'interfaces.* ♣

La table 2.1 fournit les principales caractéristiques du déploiement des deux perspectives métier (AMS et pilote SdF). Afin de comparer ces deux points de vue, nous avons opté pour la déclinaison suivante :

— *Objectifs* : buts globaux et principaux de chaque métier, ils donnent une idée de la philosophie de chaque discipline ;

— *Contraintes* : description des principaux obstacles s'opposant à l'atteinte des objectifs des deux acteurs ;

— *Procédure* : document interne Renault qui décrit les activités d'ingénierie. Nous présentons uniquement les procédures relatives aux activités dites pertinentes ;

— *Activités pertinentes* : activités menées par les AMS et pilotes SdF en relation avec les activités de thèse, c'est-à-dire fournissant des données d'entrée ou réceptionnant des résultats de ces travaux ;

— *Livrables* : documents ou données que les AMS et pilotes SdF doivent produire dans le cadre des activités pertinentes ;

— *Outils* : moyens, généralement logiciels, à disposition des acteurs pour appliquer les méthodes relevant de leur expertise.

Les deux paragraphes suivants détaillent les données de cette table, selon les deux points de vue.

	AMS	Pilote SdF
Objectifs	Proposer une architecture système optimisée	Acquérir un niveau de confiance acceptable dans le comportement du système
Contraintes	Coûts, délai	Coûts, fonctionnalités
Procédure	Concevoir le système	Maîtriser la sûreté de fonctionnement des systèmes mécatroniques
Activités pertinentes	A_AMS1. Capturer les exigences A_AMS2. Concevoir les architectures A_AMS3. Allouer les exigences	A_SdF1. Identifier et hiérarchiser les risques A_SdF2. Définir la stratégie de sécurisation A_SdF3. S'assurer de la conformité vis-à-vis des exigences de SdF
Livrables	System Technical Requirements (STR) System Design Document (SDD)	Functional Safety Concept (FSC) Technical Safety Concept (TSC) Matrice de conformité
Outils	*IBM Rational Doors* *Arkitekt*	*Aralia Fault Tree Analyzer* *Simfia*

TABLE 2.1 – Comparaison des points de vue de l'AMS et de la SdF

Point de vue de l'Architecte Métier Système

Objectif principal : Proposer une architecture système optimisée, *i.e.* satisfaisant à l'ensemble des exigences qui lui sont allouées.

Contraintes majeures : Les contraintes de coûts sont les contraintes majeures. Elles sont liées au fait que le marché automobile est extrêmement compétitif et que le critère du coût final pour le client reste décisif : il est même le premier critère de décision pour l'achat d'un véhicule (Promoneuve Opinion Way 2016 ; TNS Sofres 2014). Par ailleurs, le délai de commercialisation (*« time-to-market »*) est également une contrainte très forte. En effet, la conception, le développement et la réalisation d'un nouveau véhicule doivent se faire dans un délai maximum de 3 ans.

Activités : L'AMS suit une procédure intitulée *« Concevoir le système »* (Groupe Renault - Direction Méthodes et Standards de Conception 2018) fondée sur la norme ISO 15288. L'objectif de cette procédure est de maîtriser la conception des véhicules. Pour ce faire, les trois activités principales, identifiées dans la table 2.1 sont accomplies :

1 **A_AMS1** : cette activité consiste à collecter les besoins de toutes les parties prenantes. Ces besoins doivent être ensuite transformés en exigences dites techniques, c'est-à-dire réalisables par le système (appelées *exigences système* par l'AFIS) ;

2 Durant l'activité **A_AMS2**, une ou des solutions techniques sont définies en deux parties :

   — *Partie fonctionnelle* qui contient l'architecture fonctionnelle du système et la description détaillée de chaque flux et fonction élémentaire ;

   — *Partie physique* qui présente l'architecture physique du système, la description détaillée des composants et des interfaces entre les composants ainsi que l'allocation des fonctions sur les composants et des flux sur les interfaces.

3 **A_AMS3** : il s'agit d'allouer les exigences sur chaque fonction, flux, composant et interface des architectures fonctionnelle et physique.

Livrables : Les AMS prennent part (en tant qu'auteur, relecteur ou responsable) à l'élaboration de nombreux livrables. Nous nous focalisons, dans ce mémoire, sur deux d'entre eux :

— Le System Technical Requirements (STR), qui est le résultat de la capture de l'ensemble des exigences émises par les parties prenantes ainsi que leur traduction en exigences techniques allouées au système ;

— Le System Design Document (SDD), dans lequel les architectures fonctionnelle et physique sont décrites et les exigences techniques issues du STR sont déclinées et allouées sur les fonctions et composants du système conçu.

Outils : Les AMS utilisent deux logiciels spécifiques pour réaliser les activités décrites ci-dessus :

(a) Un outil de gestion des exigences, **IBM Rational Doors**[1], qui permet de capturer un ensemble d'exigences, de les décrire de façon hiérarchisée, d'établir une traçabilité entre les exigences, et de travailler de manière collaborative grâce à des accès partagés aux ressources ;

1. https ://www.ibm.com/fr-fr/marketplace/requirements-management

(b) Un modeleur contribuant au déploiement des activités de l'IS. Le formalisme du modeleur utilisé actuellement par les AMS chez Renault (**Arkitekt**[2], développé par la société *Knowledge Inside*) repose sur le langage de modélisation SysML, adapté aux besoins et caractéristiques spécifiques de l'entreprise. Ainsi, un ensemble de diagrammes supporte la mise en œuvre des différentes activités de l'AMS, telles que la conception d'architectures fonctionnelle et physique ou l'allocation des exigences aux fonctions et composants.

Point de vue de la SdF

Objectif principal : Il s'agit de concevoir et réaliser des systèmes fiables, disponibles, maintenables et de garantir un niveau sécuritaire acceptable pour l'entreprise lié à l'utilisation de ces systèmes.

Contraintes majeures : Les coûts viennent également contraindre les propositions du métier SdF. Par exemple, l'adjonction de mécanismes de sécurité, préconisée par les analyses de risques, entraîne un surcoût, accepté ou non, selon le niveau du risque à limiter. En outre, les fonctionnalités, de plus en plus nombreuses, offertes par les automobiles modernes, rendent les démonstrations de sûreté toujours plus difficiles. L'introduction de systèmes de sécurité active, c'est-à-dire pouvant commander les actionneurs sans requérir l'intervention du conducteur et sans que ce dernier ne puisse s'y opposer (freinage d'urgence en cas de détection d'obstacle, aide au maintien dans sa ligne, voir section 2.1.1), a rendu ces démonstration particulièrement ardues. Il est désormais primordial de se prémunir de toute activation intempestive de ce type de système. Ceux-ci peuvent en effet mener à des situations à haut risque (Evénements Indésirés Client (EIC)).

Activités : A l'instar des AMS, les pilotes SdF suivent une procédure, intitulée *« Maîtriser la SdF des systèmes mécatroniques »* (Groupe Renault - Direction de la Qualité/ Sûreté de Fonctionnement - Sécurité générale du Produit 2017). Cette procédure repose essentiellement sur deux normes : la norme ISO 15288 et le standard ISO 26262. Les activités à suivre, en rapport avec les travaux de thèse, sont détaillées ci-dessous :

(a) **A_SdF1** : l'Analyse Préliminaire des Risques (APR) débute durant cette phase. Les EIC et Evénements Redoutés Système (ERS) sont identifiés et cotés. Il s'agit d'analyser les causes au niveau système ;

(b) **A_SdF2** : les principes et les moyens de sécurisation sont définis. Ainsi, les exigences fonctionnelles de sûreté de fonctionnement (*« functional safety requirements »* en anglais dans la norme ISO 26262) sont déterminées et répertoriées dans le Functional Safety Concept (FSC). Ces exigences sont essentiellement issues de l'Analyse des Modes de Défaillance de leurs Effets et de leurs Criticités (AMDEC) fonctionnelle. A partir de ces exigences et de l'architecture physique, la rédaction du Technical Safety Concept est également initiée durant cette activité ;

(c) **A_SdF3** : pendant cette activité, les actions de levée de risques SdF sont suivies et pilotées. Les réponses des fournisseurs sont analysées et la vérification des ana-

2. http ://www.k-inside.com/web/fr

lyses de risques (Arbres de Défaillances (AdD), AMDEC) est organisée jusqu'au composant. L'ensemble de ces actions est tracé dans la Matrice de conformité.

Livrables : Les pilotes SdF, tout comme les AMS, sont impliqués dans l'élaboration de différents livrables. Dans le cadre de ces travaux, nous ne présenterons que les trois documents suivants :

— Le Functional Safety Concept (FSC) : ce document contient, d'après la norme ISO 26262, la spécification des exigences fonctionnelles de sureté de fonctionnement (*« functional safety requirement »* dans la norme ISO 26262) accompagnées des informations associées : leur allocation aux éléments de l'architecture fonctionnelle construite par les AMS, et leurs interactions requises pour atteindre les objectifs de sûreté (*safety goal*) ;

— Le Technical Safety Concept (TSC) : il est constitué, toujours selon la norme ISO 26262, des exigences techniques de sûreté de fonctionnement (*« technical safety requirement »* dans la norme ISO 26262) , ainsi que de leurs allocations aux éléments de l'architecture physique (fournie, également, par les AMS) du système conçu ;

— La Matrice de conformité : ce document permet de tracer l'application et la justification de toutes les exigences issues des concepts de sécurité (FSC, TSC).

Remarque 2.3
Dans les travaux de thèse, les données traitées proviennent du Functional Safety Concept. En effet, seule une version préliminaire du Technical Safety Concept (pour la fonction de Conduite Autonome étudiée) existait au moment d'initier la thèse et la Matrice de conformité n'était pas encore débutée, en raison de l'état d'avancement du projet. ♢

Remarque 2.4
*Nous avons choisi de traduire le terme anglais « safety » apparaissant dans les énoncés des exigences par le terme français « sûreté de fonctionnement ». En effet, les « functional safety requirements » peuvent être relatifs aux quatre principales aptitudes de la SdF (*Villemeur *1988) : la fiabilité, la maintenabilité, la disponibilité et la sécurité.*♢

Outils : Les deux outils listés dans le tableau 2.1 servent à réaliser des AdD et procéder aux calculs de probabilité (**Aralia Fault Tree Analyzer**) et à mettre en œuvre les analyses MBSA (voir paragraphe 2.2.1.2) au moyen de l'outil **Simfia**[3]. Ainsi, les modèles fonctionnels formels fournis par les AMS (construits dans **Arkitekt**) sont complétés par la prise en compte des aspects dysfonctionnels. Il apparaît que les AMS et les pilotes SdF partagent des données et des concepts communs. Par exemple, le FSC ne peut être entrepris que si l'architecture a été définie au préalable par les AMS. Pour garantir une conception sûre, les deux points de vue doivent donc être mis en cohérence. Or, l'organisation en silos et les contraintes de planning ne facilitent pas cette convergence.

3. https ://www.apsys-airbus.com/digital-software/

Cohérence des points de vue AMS et SdF

La figure 2.9 montre le déroulement des activités des AMS et des pilotes SdF présentées dans les paragraphes précédents. Les mentions *V0* et *V1* correspondent aux différentes versions des livrables (respectivement version 0 et version 1). Par ailleurs, le document nommé AFB correspond à l'Analyse Fonctionnelle du Besoin, qui est produite par les AMS lors d'une activité antérieure à A_AMS1 Les documents manipulés par les deux acteurs apparaissent en caractères gras. La nécessaire synchronisation des activités des deux métiers est mise en évidence par le partage de ces données communes. A titre d'illustration, les données de sortie de l'activité A_SdF2 sont requises pour concevoir les architectures et allouer les exigences (activités A_AMS2 et A_AMS3). Mais, pour mener à bien l'activité de SdF mentionnée (A_SdF2), les architectures fonctionnelle et physique, produites par les activités A_AMS2 et A_AMS3, sont également nécessaires. Cette circularité impose une étroite collaboration entre les deux disciplines d'ingénierie. Sans une telle collaboration, les exigences contenues dans les documents STR (côté AMS) et FSC et TSC (pour la SdF) peuvent être contradictoires, incomplètes, redondantes, porter sur des objets différents... La présentation des deux points de vue illustre

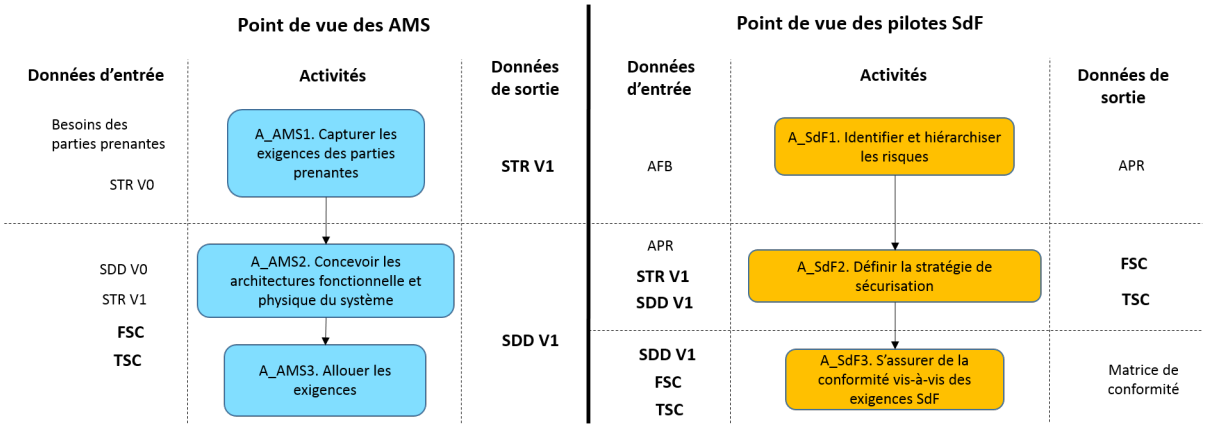


FIGURE 2.9 – Synchronisation des activités des AMS et des pilotes SdF

clairement leurs différences et renforce l'organisation traditionnelle de l'ingénierie en silos. Ainsi l'intégration des activités de SdF dans le processus de conception d'architecture système est source de difficultés récurrentes. Ce problème de mise en cohérence des points de vue de l'AMS et de la SdF n'est ni récent (LEVESON 2002) ni propre à Renault (P. DAVID 2009 ; GUILLERM 2011 ; CRESSENT 2012). En revanche, cette question reste ouverte, comme en témoigne de multiples recherches actuelles, notamment dans le domaine automobile (KAISER et al. 2010 ; CHEN et al. 2011 ; TAOFIFENUA 2012 ; KANG et al. 2013 ; WEISSNEGGER et al. 2015 ; MAUBORGNE 2016). Dans ces travaux de thèse, nous abordons cette problématique sous un angle opérationnel. En effet, dans le cadre de la conception et du développement du véhicule autonome, les enjeux liés à la sûreté sont centraux. Non seulement ce type d'automobile embarque des systèmes capables de piloter les actionneurs sans que le conducteur ne puisse s'y opposer ; mais aussi, à partir d'un certain niveau d'autonomie, le véhicule doit être capable de se mettre seul en sécurité, sans aucune intervention du conducteur et quelle

que soit la situation de conduite. Etant donné l'imminence de la commercialisation de ce type de véhicule (les premiers modèles Renault dotés de délégation de conduite sont attendus pour 2020 [4]), la mise en œuvre de démarches méthodologiques pratiques et opérationnelles contribuant au rapprochement des points de vue de l'architecte système et de la SdF est une priorité pour les constructeurs. Dans la section suivante, nous présentons le projet du véhicule autonome chez Renault. Ceci parachève la présentation du contexte industriel et débouche sur la problématique et les questions de recherche abordées au cours des travaux de thèse.

2.3 Introduction de la fonction de Conduite Autonome

2.3.1 Principales motivations et enjeux

Le véhicule autonome est une automobile qui se conduira, à terme, sans aucune intervention du conducteur, quelle que soit la situation de conduite. Parmi les principaux arguments en faveur de l'introduction du véhicule autonome, nous pouvons citer les suivants :

(a) La réduction potentielle du nombre d'accidents car ceux-ci sont majoritairement dus à l'erreur humaine (Fagnant et Kockelman 2015) : d'après (US Department of Transportation 2008), elle est la cause première de plus de 90 pourcent des accidents ;

(b) La suppression des phases de conduite « ennuyeuses » (embouteillage, longs trajets sur l'autoroute) et un gain de temps pour le conducteur, qui peut alors se consacrer à d'autres activités (Fagnant et Kockelman 2015) ;

(c) La régulation du trafic, la diminution des congestions et de la consommation de carburant, rendues possible par la communication inter-véhicule (Fagnant et Kockelman 2015) ;

(d) La fourniture d'un moyen de locomotion aux personnes étant dans l'incapacité de conduire (Hautiere, Tattegrain et Guilbot 2017).

Naturellement, cette innovation n'est pas sans risques. Ainsi, le véhicule autonome peut également être lui-même source de danger, comme l'ont illustré les trois accidents mortels survenus depuis la mise en service de prototypes sur routes ouvertes (NHTSA 2016 ; New York Times 2016 ; l'Express 2018). En conséquence, la société *Uber*, en cause dans le dernier accident, a été contrainte de suspendre l'ensemble de ses essais de véhicules autonomes. Les marques de véhicules grand public, telle que Renault, ont besoin de garanties plus fortes avant de commercialiser ce type de véhicules, dans le but d'éviter ces déconvenues, très nuisibles aux constructeurs. Bien que ce ne soit pas encore le cas dans le domaine automobile, ce type de véhicule pourrait nécessiter une certification par un ou des organisme(s) indépendant(s) (comme dans les secteurs aéronautique et ferroviaire).

4. https ://group.renault.com/innovation/vehicule-autonome/, consulté le 19/04/2018

2.3.2 Le projet du véhicule autonome

La National Highway Traffic Safety Administration (NHTSA) et l'Organisation Internationale des Constructeurs Automobiles (OICA) ont défini différents niveaux d'autonomie visant à introduire ce nouveau type de véhicule de manière progressive (NHTSA 2017 ; OICA 2014). Les deux paramètres utilisés sont le niveau de vigilance du conducteur nécessaire (mains sur le volant, pieds sur les pédales, délai maximum de reprise en main en cas de dysfonctionnement...), et les tronçons pour lesquels la fonction de Conduite Autonome (ou fonction Autonomous Driving (AD), en anglais,) pourra être activée. Les niveaux d'autonomie déterminés par l'OICA sont les suivants (OICA 2014) :

(a) *Niveau 0 Conducteur seul* : le conducteur a le contrôle total et exclusif de toutes les fonctions primaires du véhicules (freinage, direction, accélération). Aucune des fonctions principales n'est automatisée mais il peut disposer de mécanismes d'avertissement (radars de recul, voyants lumineux...) ;

(b) *Niveau 1 Assisté* : le conducteur peut momentanément confier au véhicule une tâche de conduite. La voiture ne peut prendre en charge qu'une seule dimension du guidage : longitudinale ou latérale. Une fonction de guidage latéral ou longitudinal peut donc être confiée au véhicule par le conducteur. Exemple : le régulateur adaptatif qui est un régulateur qui maintient également la distance de sécurité avec le véhicule qui précède dans la voie de circulation ;

(c) *Niveau 2 Autonomie partielle* : l'automobile peut se déplacer en autonomie selon les deux dimensions de conduite dans des conditions prédéfinies. Le conducteur doit néanmoins surveiller à tout moment son automobile et être capable de reprendre le contrôle si nécessaire. Exemple : l'assistance au stationnement ;

(d) *Niveau 3 Autonomie sous conditions* : le conducteur peut déléguer totalement la conduite dans des situations précises, mais doit toutefois rester vigilant et être capable de reprendre le contrôle de son véhicule en cas de besoin. Exemple : la fonction « embouteillage » du Volvo XC90 ;

(e) *Niveau 4 Haute autonomie* : dans un contexte limité et des situations prédéfinies, le véhicule est capable de se déplacer sans conducteur, de manière totalement autonome ;

(f) *Niveau 5 Autonomie totale* : le véhicule n'a plus besoin d'Hommes pour être conduit, quelle que soit la situation de conduite.

Il est important de noter que le conducteur reste responsable de toutes les manœuvres et de potentiels incidents ou accidents jusqu'au niveau 3 inclus. En revanche, les constructeurs des automobiles dotés d'un niveau d'autonomie 4 ou plus peuvent être tenus responsables d'accidents s'il est avéré que le système de conduite autonome embarqué est bien à l'origine du sinistre. D'un point de vue légal, la convention de Vienne sur la circulation routière, qui est un traité multilatéral paru en 1968 et entré en vigueur dans tous les états signataires en 1977, n'autorisait pas jusqu'à présent la délégation de conduite. Effectivement, l'article 8-5 stipulait que « tout conducteur doit constamment avoir le contrôle de son véhicule ». Dans un communiqué daté du 23 mars 2016, l'United Nations Economic Commission for Europe (UNECE) indique que la convention de Vienne a été modifiée (UNECE 2016), autorisant explicitement les systèmes

de conduite automatisée à condition qu'ils puissent être contrôlés voire désactivés par le conducteur. Ainsi les niveaux 4 et 5 d'autonomie restent encore en dehors du cadre légal dans les pays signataires de cette convention.

2.3.3 Le véhicule autonome chez Renault

2.3.3.1 Niveaux d'autonomie adoptés par l'alliance Renault/Nissan

Au sein de l'alliance Renault/Nissan (conclue en 1999), la stratégie de déploiement du véhicule autonome suit également une progression échelonnée par des niveaux d'autonomie, comme l'illustre la figure 2.10. Cette figure met en évidence les liens entre les niveaux adoptés par l'alliance et ceux définis par l'OICA, présentés dans la section 2.3.2.

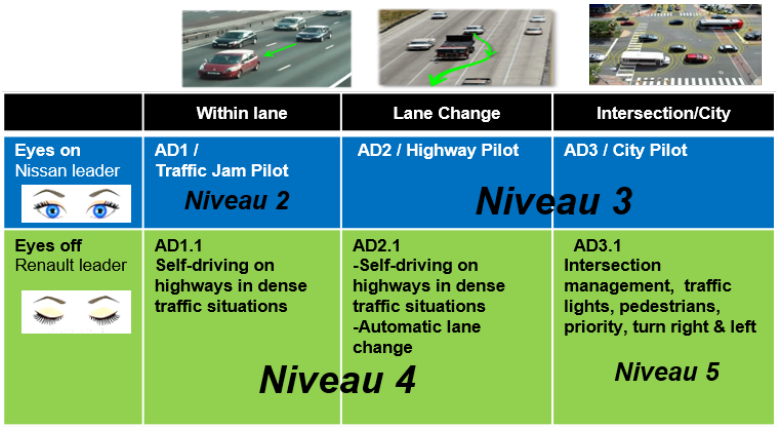


FIGURE 2.10 – Niveaux d'autonomie des véhicules de l'alliance Renault/Nissan

L'effort de développement causé par le véhicule autonome a été partagé entre Renault et Nissan de la manière suivante :

— Nissan est en charge de la conception et de la réalisation des véhicules pouvant :

(a) Se déplacer seul en restant sur leurs voies (AD1, où AD signifie Autonomous Driving, en anglais). Ceci correspond à une autonomie partielle selon l'OCIA ;

(b) Se mouvoir en autonomie en changeant de lignes (AD2) puis en gérant la conduite en ville (AD3). Ces niveaux correspondent au niveau 3 (autonomie sous conditions) de l'OICA.

Quel que soit le projet, le conducteur demeure responsable et est tenu de reprendre le contrôle lorsque le véhicule le lui demande. De même, l'automobile doit informer suffisamment à l'avance le conducteur de la nécessité de la reprise en main par le conducteur.

— Renault est affecté au développement des véhicules de niveaux d'autonomie 4 et 5. Ce déploiement sera en réalité réalisé de manière graduelle en suivant la même progression que Nissan : le véhicule devra d'abord rester dans sa voie (AD1.1), ensuite le changement de ligne sera autorisé (AD2.1). Enfin, une fois que ces

véhicules de niveau d'autonomie 4 seront éprouvés, l'objectif est de mettre au point des véhicules totalement autonomes, de niveau 5 (AD3.1).

Dans le cadre de ces travaux de thèse, nous avons travaillé sur le projet AD2.1 visant à concevoir des véhicules de niveau 4. Le conducteur peut donc totalement déléguer la conduite à la fonction de Conduite Autonome dans certaines conditions précises et prédéfinies. Ainsi, nous le verrons, la fonction AD doit assurer la mise en sécurité du véhicule si nécessaire en procédant pour ce faire à des manœuvres particulières, nommées Minimal Risk Manœuver (MRM). Les systèmes embarqués électroniques décrits dans la section 2.1.1 mettent en œuvre un ensemble de fonctionnalités qui est organisé en une architecture fonctionnelle. L'avantage d'une telle représentation est sa stabilité et son indépendance par rapport aux architectures physiques solutions. Une architecture fonctionnelle donnée peut en effet être réalisée par un ensemble d'architectures physiques (Behere et Törngren 2016). D'un point de vue fonctionnel, la conception d'un véhicule autonome revient à l'intégration de la nouvelle fonction AD dans l'architecture fonctionnelle d'un véhicule équipé des systèmes d'aide à la conduite appropriés. Les paragraphes qui suivent décrivent l'intégration de cette fonction.

2.3.3.2 Conception : point de vue des AMS

La figure 2.11 représente l'environnement dans lequel est insérée la fonction AD. Ce schéma présente les principales interactions entre la fonction AD et son environnement. Outre la fonction de fourniture d'énergie, la fonction AD interagit avec certaines fonctions du véhicule (identifiées par les blocs en fond gris, toutes les fonctions en connexion avec la fonction AD ne sont pas reproduites) dans lequel elle est implantée, ainsi que le conducteur.

Remarque 2.5
Nous faisons apparaître sur les figures 2.11, 2.12 et 2.13, la notion de ***niveau****. De façon générale, l'architecture fonctionnelle a une structure hiérarchique dans laquelle les éléments sont notamment caractérisés par leur niveau de raffinement. Dans notre cas d'étude, nous avons affaire, comme le montre les différentes figures pré-citées, à une structure hiérarchique en 3 niveaux. Le niveau 2 raffine le niveau 1 et le niveau 3 raffine le niveau 2. Nous nous intéresserons plus particulièrement aux niveaux 2 et 3.*♢

La figure 2.12 donne une représentation simplifiée de l'architecture interne de la fonction AD, issue du SDD V1 (AMS Renault 2017a). Deux sous-fonctions principales interagissent entre elles :

1 **Contrôle AD** : il s'agit de la fonctionnalité principale, à savoir la fusion et le traitement des informations issues de la perception de l'environnement dans le but d'élaborer continuellement une commande de trajectoire transmise aux actionneurs. Les informations de l'environnement sont fusionnées afin d'en extraire les données pertinentes. Parallèlement, le mouvement du véhicule est continuellement modélisé. Ceci rend possible la détermination d'une trajectoire à court terme et à plus long terme. Cette fonction comprend des sous-fonctions, conçues pour envoyer des commandes de trajectoire particulières, adaptées en temps réel à l'évolution

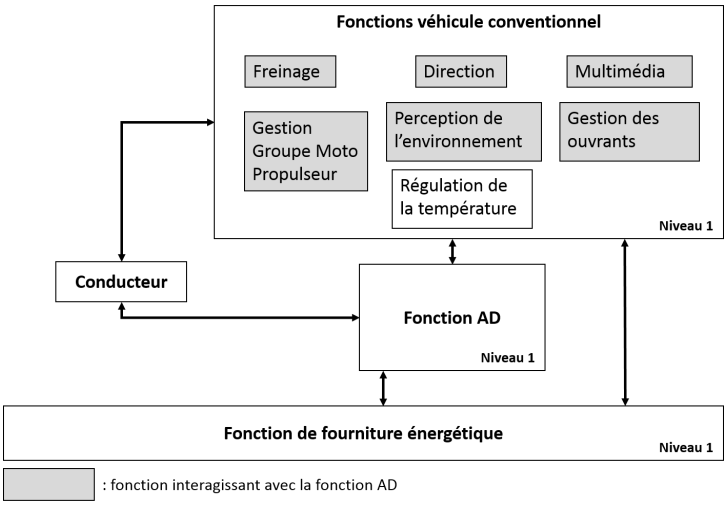


FIGURE 2.11 – Environnement de la fonction AD

de l'environnement. Par exemple, lorsque la sous-fonction *Control_MRM* (pour Minimal Risk Manoeuver (MRM)) est active, des commandes de trajectoire visant à mettre le véhicule en sécurité sont élaborées. Nous nommerons cette fonction : fonction CAD pour la suite de ce mémoire ;

2 **Supervision AD** : cette fonction a en charge de déterminer la sous-fonction de la fonction Contrôle AD qui doit être activée. La Supervision AD autorise, ou non, selon l'état dans lequel elle se trouve, l'action d'une sous-fonction de la fonction CAD. Dans notre cas, les différents états de la Supervision AD sont déterminés à partir :

- De l'état du véhicule : état des ouvrants, niveau d'huile... ;
- Des interactions avec le conducteur : appui sur les pédales, mains sur le volant, etc. ;
- De la disponibilité des fonctions contributrices à la fonction AD. Ces fonctions sont celles qui sont requises pour que la fonction AD soit opérationnelle. Il s'agit par exemple de la Direction ou du Freinage.

Cette fonction de Supervision se trouve au cœur de ces travaux de thèse. Notons que cette fonctionnalité est toujours identifiée dans les systèmes autonomes critiques, elle peut être appelée *système de management* ou de *gestion* (JO et al. 2015 ; TAŞ et al. 2016 ; BEHERE et TÖRNGREN 2016). Nous nous penchons plus précisément la spécification du comportement de cette fonction de Supervision. Dans le STR V1, document de 37 pages daté de 2017 (AMS RENAULT 2017b), se trouvent les exigences fonctionnelles, qui sont allouées aux éléments de l'architecture fonctionnelle de la fonction AD. Nous nous intéressons plus particulièrement aux *Exigences Fonctionnelles de Comportement (EFC)* (allouées, comme l'illustre la figure 2.12 aux deux sous-fonctions de la fonction AD) qui spécifient le comportement de la fonction AD en phases de fonctionnement nominal. Le document STR V1 traité en contient 175. Notons que ces exigences sont exprimées en langage naturel. En voici un exemple :

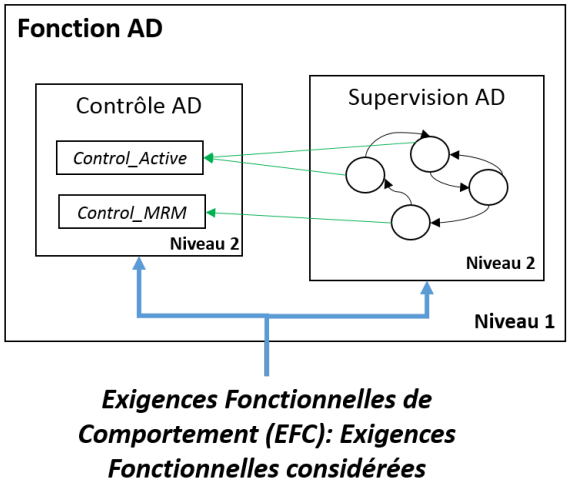


FIGURE 2.12 – Architecture fonctionnelle interne du système AD

Exemple d'Exigence Fonctionnelle de Comportement (Ex_EFC) :
« AD function shall not be available on shoulders. »

2.3.3.3 Conception : point de vue des pilotes SdF

Lorsque la fonction Contrôle AD est active, la conduite est totalement déléguée et le conducteur n'a pas l'obligation d'être disponible pour reprendre le contrôle du véhicule (le projet AD2.1, dans lequel s'inscrivent nos travaux, porte sur des véhicules de niveau d'autonomie 4). Or, le point de vue de la SdF vise à garantir que la fonction CAD se trouve toujours dans un état sûr. Ainsi, une des exigences premières des pilotes SdF, est de garantir que l'automobile puisse être mise en sécurité, à n'importe quel moment, lorsque la fonction Contrôle AD est active. Pour ce faire, une **redondance fonctionnelle chaude et partielle** a été adoptée (voir figure 2.13). La fonction de Supervision AD est, en réalité, opérée par trois sous-fonctions : Main_AD, Sub_AD et AD-3. Chacune des sous-fonctions de Supervision a également pour rôle d'activer une sous-fonction de Contrôle AD.

Cette redondance est dite **chaude** car la seconde fonction de Supervision (Sub_AD) active toujours la fonction Control_Sub_Standby lorsque la fonction principale (Main_AD) active la fonction Control_Main_Active. La fonction Control_Sub_Standby élabore des trajectoires de commande similaires à celles produites par la fonction Control_Main_Active mais non appliquées par les actionneurs si la fonction Control_Main_Active est saine. Ainsi, en cas de défaillance soudaine de la fonction Control_Main_Active, la stabilité de la dynamique du véhicule sont assurées par la fonction Control_Sub_Standby, avant que la fonction Control_Sub_MRM ne procède aux manœuvres de sécurité adéquates. La redondance est également qualifiée de **partielle** car le rôle de la seconde fonction de Supervision est de mettre le véhicule autonome en sécurité lorsque la fonction principale est perdue, et non de continuer à fournir un service nominal. Enfin, une troisième fonction (AD-3) intervient en ultime recours, en cas de double perte simultanée des deux premières fonctions de Supervi-

sion (les fonctions de Supervision elles-mêmes ou les sous-fonctions de contrôle qu'elles gèrent).

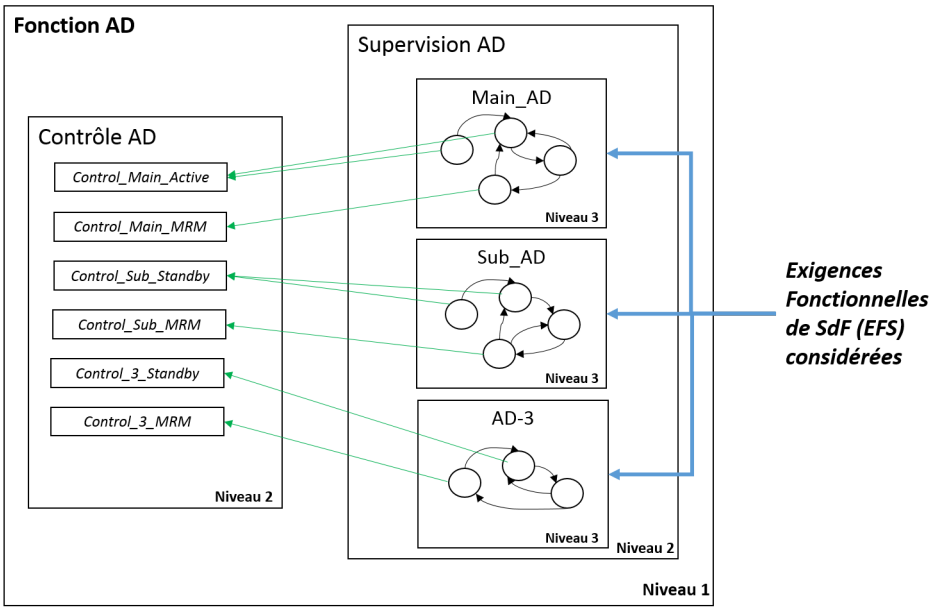


FIGURE 2.13 – Redondance de la fonction AD

Dans le FSC, document de 22 pages contenant 350 exigences, daté de septembre 2016 (IAV 2016), les *Exigences Fonctionnelles de Sûreté de fonctionnement (EFS)* sont listées. Les exigences considérées dans le cadre de ces travaux sont les EFS allouées aux trois sous-fonctions de Supervision (voir figure 2.13), qui sont au nombre de 217. Ces exigences sont également rédigées en langage naturel, comme l'illustre l'exemple qui suit :

Exemple d'exigence fonctionnelle de SdF (Ex_EFS) :

« In case of an excessive acceleration due to a dysfunction of Main_AD, Sub_AD shall enter MRM. »

2.3.3.4 Point sur les exigences considérées

Dans le cadre de ces travaux de thèse, nous avons situé les exigences considérées par rapport à la typologie en vigueur chez Renault (GAUDRÉ 2016). Cette dernière distingue les :

- *Exigences de processus* : énoncés spécifiant comment le produit doit être développé (activités, livrables...) ;
- *Exigences de produit* :
  - (a) *Fonctionnelles* : énoncés spécifiant ce que le système doit faire en précisant les informations attendues et les conditions pour atteindre un but donné ;
  - (b) *Non fonctionnelles* : énoncés spécifiant ce que le système doit être, c'est-à-dire quelles sont ses caractéristiques techniques, physiques, économiques...

Les exigences sur lesquelles est centrée l'analyse sont les *exigences fonctionnelles*, et, plus particulièrement, les exigences fonctionnelles de comportement (voir paragraphe 2.3.3). Pour éviter toute confusion, nous noterons simplement ce type d'exigences : « ***exigences de comportement*** ». Ainsi, les exigences de ce type, émises par les AMS, sont dénommées « Exigences Fonctionnelles de Comportement (EFC) », tandis que les exigences de ce type, produites par les pilotes SdF, sont notées « Exigences Fonctionnelles de SdF de Comportement (EFSC) ».

Par ailleurs, nous distinguons les exigences de comportement ***quantifiées*** et non quantifiées. Les exigences quantifiées font référence explicite à des valeurs, telles que des seuils de probabilité d'occurrence d'événement ou des durées maximales d'exécution. Voici deux exemples d'exigences de comportement quantifiées :

1 *« La probabilité d'une défaillance non détectée empêchant le système d'appliquer une pression dans les freins de plus de 20 bars dans l'état Freinage doit être inférieure à* 10^{-8} *par heure. »* (Lebeaupin 2017) (seuil de probabilité) ;

2 *« The acquisition duration window must be large enough to acquire at most 2 samples of the knock sensor. »* (Albinet, J.-L. Boulanger et al. 2007a) (durée maximale d'exécution).

Enfin, les exigences de comportement décrivent le comportement attendu d'un élément de l'architecture fonctionnelle en termes :

(a) *D'état* (exigences d'état) : ces exigences définissent les actions particulières à effectuer dans un état donné de l'élément spécifié. Ces exigences sont allouées à la fonction de Contrôle AD (voir figure 2.12) ;

(b) *De transition* : ces exigences déterminent les conditions sous lesquelles un changement d'état de l'élément spécifié doit se produire. Elles sont allouées à la fonction de Supervision.

Dans le contexte de notre étude, la méthode que nous proposons s'applique aux « ***exigences de transition*** » non quantifiées, c'est-à-dire aux exigences de comportement portant sur une transition et allouées à la Supervision AD. Les exigences d'état sont évidemment à vérifier, mais par un autre moyen que celui développé dans l'approche déployée (simulation, tests de validation) ou par la même technique (vérification formelle par *model checking*, voir paragraphe 4.1.2), en utilisant des modèles plus détaillés.

La Table 2.2 récapitule les dénominations adoptées. Nous voyons donc que, du point de vue des AMS, les exigences considérées sont déjà des exigences de comportement, mais elles sont allouées à la fois à la fonction CAD et à la fonction Supervision AD ; tandis que selon la perspective SdF, les exigences du FSC considérées sont déjà allouées aux trois sous-fonctions de Supervision locales mais ne sont pas nécessairement des exigences de comportement (ni, *a fortiori*, des exigences de transition). Notons enfin que les exigences analysées (ou traitées) sont celles qui sont retenues pour être vérifiées à l'aide de la démarche de travail que nous proposons. Le détail de cette sélection sera donné dans le chapitre 5.

2.3.3.5 Conclusions

La présentation de la conception de la fonction AD illustre en pratique les différences (présentées dans le paragraphe 2.2.1) entre les deux disciplines d'ingénierie (AMS et

Point de vue	Dénomination	Allocation	Document
AMS	Exigences fonctionnelles	Tous les éléments de l'architecture fonctionnelle	STR
	Exigences Fonctionnelles de Comportement (EFC) : exigences considérées (175)	Tous les éléments de l'architecture fonctionnelle	
	EFC de transition : exigences analysées (69)	Supervision AD	
Pilotes SdF	Exigences Fonctionnelles de SdF (EFS) (350)	Tous les éléments de de l'architecture fonctionnelle	FSC
	EFS considérées (217)	Main_AD, Sub_AD AD-3	
	EFS de transition : exigences analysées (61)	Main_AD, Sub_AD AD-3	

TABLE 2.2 – Dénomination des exigences adoptée

SdF) qui les ont produites. Les figures 2.12 et 2.13 mettent notamment en exergue les différences de niveaux d'abstraction entre les exigences émises par les AMS et celles des pilotes SdF. C'est donc à partir de ce cas d'étude que nous abordons la problématique de mise en cohérence des points de vue des AMS et des pilotes SdF, évoquée dans la section 2.2.2. Les questions afférentes, telles que l'introduction du MBSE et des méthodes formelles entrent également dans le périmètre des travaux menés.

2.4 Problématique

2.4.1 Conclusions relatives au contexte industriel

Au vu des éléments présentés dans les paragraphes précédents, le véhicule autonome apparaît comme une continuité logique de l'introduction progressive de systèmes embarqués électroniques dans les automobiles depuis une trentaine d'années. Cependant, cette innovation constitue également une rupture du point de vue de la SdF, étant donné l'absence de conducteur dans la boucle de contrôle. En effet, le véhicule autonome doit toujours être capable d'atteindre un état sûr, et ceci sans l'intervention du conducteur. Cette rupture nécessite notamment de consolider les démonstrations de sûreté pour le véhicule autonome. Dans cette optique, les points de vue des AMS et des pilotes SdF doivent être rapprochés et doivent converger au plus tôt.

C'est dans ce but que les constructeurs adoptent progressivement les méthodes de l'IS préconisées par l'INCOSE ainsi que l'AFIS, sa branche française. Afin d'appliquer et implémenter de manière efficace ces méthodes, il est nécessaire de passer d'une ingénierie dite *basée sur les documents* à une ingénierie *basée sur les modèles*. Les modèles manipulés dans le second type d'ingénierie étant formels ou semi-formels, le changement évoqué s'accompagne par conséquent de l'introduction la mise en œuvre de *méthodes formelles*. Concernant les processus, méthodes et outils, le passage d'un

type d'ingénierie à l'autre devra se faire progressivement. C'est pourquoi, une période de cohabitation entre une ingénierie basée sur les documents et une ingénierie basée sur les modèles est à prévoir. L'organisation et la mise en place de cette cohabitation doivent être anticipées.

2.4.2 Principaux verrous

Les parties précédentes ont donné un aperçu de l'ampleur du défi et des enjeux provoqués par le projet du véhicule autonome. Dans le cadre de ces travaux de thèse, deux verrous majeurs ont été identifiés :

1 La convergence nécessaire entre les points de vue des AMS et des pilotes SdF au plus tôt dans le cycle de conception ;

2 L'adoption d'une approche MBSE.

Le premier point a été présenté premièrement d'un point de vue des processus (voir paragraphe 2.2.2). Nous avons pu constater que cette question n'était propre ni à Renault ni même aux constructeurs automobiles mais touchait l'ensemble des industriels réalisant des systèmes complexes critiques. Les travaux présentés dans ce mémoire contribuent à traiter cette question en se basant sur les exigences émises par les deux disciplines d'ingénierie. Ceci présente un double avantage : opérationnel d'une part, car les documents manipulés dans le cadre de cette thèse reflètent exactement la pratique de l'ingénierie dans l'entreprise d'accueil, mais également scientifique, car le traitement des exigences est la première étape, fondamentale, pour adopter une approche MBSE.

La problématique de mise en cohérence a naturellement été corrélée avec celle de l'adoption d'une approche MBSE. En effet, le moyen le plus approprié pour partager des données entre les AMS et les pilotes SdF est l'utilisation de modèles décrits dans un formalisme commun. De plus, les modèles formels, manipulés dans le cadre d'une approche MBSE, peuvent être vérifiables au moyen de méthodes formelles, telles que le *model checking* par exemple. Cet aspect est primordial dans l'objectif de consolider les démonstrations de sûreté car il faut garantir un lien (si possible formel) entre les architectures solutions proposées et les exigences initiales, correspondant aux besoins des parties prenantes. Cependant, les liens entre les exigences contenues dans les documents de travail (STR et FSC), exprimées en langage naturel, et les modèles formels ou semi-formels utilisés dans le cadre d'une approche MBSE, ne sont pas explicites. La détermination et l'explicitation de ces liens font partie des points traités dans ces travaux. Cette question est celle de la formalisation des exigences exprimées en langage naturel, qui fait partie des questionnements de recherche, présentés dans le dernier paragraphe.

2.4.3 Questionnement de recherche

Comme nous l'avons constaté précédemment (paragraphe 2.3.3), et conformément au cadre de la Convention Industrielle de Formation par la REcherche (CIFRE) dans lequel s'est déroulée cette thèse, nous abordons une problématique scientifique classique,

la mise en cohérence des points de vue de l'IS et de la SdF, mais sous un angle opérationnel. Ainsi, la principale question à laquelle ces travaux apportent une réponse est résumée dans l'énoncé suivant :

Problématique : *Comment s'assurer que la fonction AD se trouve toujours dans un état entièrement spécifié et sûr ?*

Ceci revient à garantir que la fonction Supervision AD respecte l'ensemble des exigences de comportement (issues des deux points de vue métier : AMS et pilotes SdF) qui lui sont allouées. Cette problématique étant très large, nous l'avons, en considérant le contexte exposé ci-dessus, décomposée selon les trois questions de recherche suivantes :

1 Comment mettre en cohérence les points de vue de l'AMS et de la SdF ?

2 Comment vérifier, en phase amont de conception, un ensemble d'exigences ?

3 Comment formaliser un ensemble d'exigences ?

Ces trois points structurent l'état de l'art présenté dans le chapitre suivant. Chacune de ces questions sera, à cette occasion, détaillée.

3

Etat de l'art et positionnement

Sommaire

3.1 Mise en cohérence des points de vue de l'AMS et de la SdF

La convergence du point de vue des AMS et de celui des pilotes SdF dans un contexte industriel demeure, nous l'avons vu, un problème reconnu, largement abordé, mais encore ouvert. De plus, cette question devient cruciale dans le secteur automobile, avec l'introduction du véhicule autonome. Des expérimentations et méthodes pratiques, comme les propositions de ce mémoire, doivent être élaborées pour faire face au défi de conception que représente le véhicule autonome. La sûreté doit être démontrée et la traçabilité des exigences assurée afin de se prémunir en cas d'accidents. Les paragraphes qui suivent restituent les principales approches proposées dans la littérature pour contribuer à la mise en cohérence des deux points de vue sus-cités. Pour cette partie, nous nous appuierons sur la classification représentée sur la figure 3.1, établie par Anthony Legendre dans ses travaux de thèse (Legendre 2017). Ces travaux récents traitent également de la mise en cohérence des points de vue des deux métiers en jeu et bâtissent une classification intéressante. Trois principaux types d'approche sont identifiés :

1 Approche orientée ;

2 Approche collaborative ;

3 Approche d'analyses conjointes.

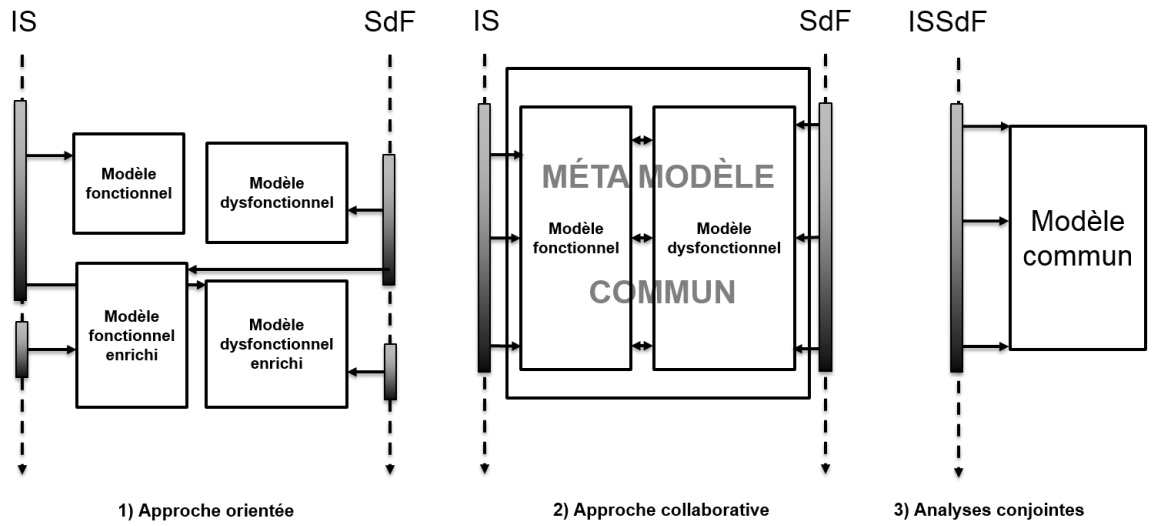


Figure 3.1 – Les 3 types d'approche pour la convergence des points de vue des AMS et des pilotes SdF

3.1.1 Approche orientée

Ce type d'approche consiste à étendre les modèles produits par un point de vue en utilisant les informations pertinentes fournies par l'autre perspective. Ainsi les deux métiers en jeu conservent leurs propres outils et modèles (fonctionnels pour les AMS, dans le cadre des processus d'IS, et dysfonctionnels pour les pilotes SdF). Le principe de séparation des préoccupations, préconisé en premier lieu par l'ingénierie logicielle (Hürsch et Lopes 1995 ; Ernst 2003), mais également en vigueur dans notre contexte (Legendre 2017), est ainsi respecté. En revanche les difficultés naissent des incohérences sémantiques entre les modèles manipulés par les deux ingénieries, des choix des informations à retenir (informations pertinentes) et des instants judicieux (jalons) auxquels les modèles doivent être confrontés et enrichis.

3.1.1.1 Enrichissement des modèles des AMS

Cette méthode consiste à construire, dans un premier temps, les modèles d'architectures fonctionnelle et physique. Ceci est fait généralement dans un modeleur de manière à obtenir des modèles formels ou semi-formels. Cette activité est menée par les AMS. Les modèles résultants sont ensuite complétés par les pilotes SdF qui ajoutent les aspects dysfonctionnels.

L'équipe de Frédéric Kratz et Vincent Idasiak a mis en place une méthode outillée nommée Méthode d'Intégration des analyses de Sûreté de Fonctionnement au processus d'Ingénierie Systèmes (MéDISIS) (P. David 2009 ; P. David, Idasiak et Kratz 2010 ; Cressent, Idasiak, Kratz et P. David 2011 ; Cressent, Idasiak et Kratz 2011 ; Cressent, P. David et al. 2013) dont le but est de faciliter l'intégration des activités de SdF à celles de l'IS dès les premières phases de conception. Cette approche se fonde sur l'Ingénierie Basée sur les Modèles et utilise le formalisme de modélisation standard : SysML. Les diagrammes SysML de structure (*Diagrammes de Définition de Blocs* et *Diagrammes de Blocs Internes*), ainsi que les diagrammes de comportement (*Diagrammes de Séquences* et *Diagrammes d'Activités*), sont analysés en détail afin d'établir une liste exhaustive des modes de défaillances pour chaque fonction et composant. Ceci est rendu possible par l'établissement et la maintenance d'une base de données contenant les comportements dysfonctionnels possibles. Des AMDEC préliminaires, analysées ensuite par les pilotes SdF, peuvent être générées, ainsi qu'une description détaillée des modèles SysML vers le langage Alta Rica Data Flow (Boiteau et al. 2006). Cette transformation est destinée aux analyses quantitatives de SdF.

Des travaux proches (Tucci-Piergiovanni et al. 2014 ; Blom et al. 2016), appliqués au secteur automobile, étendent le formalisme Embedded electronic Architecture and STudy - Architecture Description Language (EAST-ADL), dédié à la description des architectures des systèmes embarqués automobiles, par une méthodologie conforme à la norme ISO 26262 (projets Advancing Traffic Efficiency and Safety through Software Technology (ATESST) et Model-based Analysis and Engineering of Novel Architectures for Dependable Electric Vehicles (MAENAD)). Les analyses de risque préconisées par l'ISO 26262 sont intégrées dans la démarche globale, comme l'illustre la figure 3.2 (sur cette figure HW signifie Hardware et SW, Software). Les vues offertes par le formalisme EAST-ADL sont liées aux différentes activités de la norme ISO 26262.

Un modèle d'erreurs (*Error Model*) capture les informations détaillées relatives aux comportements de défaillances du système. L'intégration de l'outil Hierarchically Performed Hazard Origin and Propagation Studies (HIP-HOPS) (Papadopoulos et al. 2011) permet de procéder à des analyses de risques statiques, telles que l'APR, l'AMDEC ou l'AdD. Ces analyses sont générées automatiquement à partir des modèles du système et de l'*Error Model*. Notons que les aspects relatifs à l'implémentation sont pris en charge au travers du formalisme AUTOSAR (voir paragraphe 2.4), dont les liens avec EAST-ADL ont été définis.

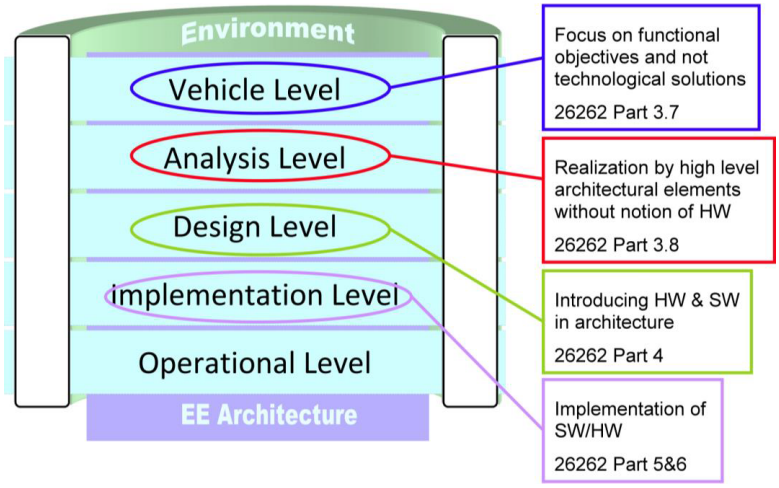


Figure 3.2 – Lien entre les vues EAST-ADL et les activités de la norme ISO 26262 (Blom et al. 2016)

Des approches complètent ces travaux en offrant la possibilité de mener des analyses de risques dynamiques. Ainsi, dans le domaine aérospatial, (Bozzano et al. 2010) mettent au point une démarche multidisciplinaire de conception de systèmes complexes dans le cadre du projet Correctness Modeling and Performance of Aerospace Systems (COMPASS). Des modèles fonctionnels sont construits en utilisant le formalisme Architecture Analysis and Design Language (AADL), émanant de la SAE International. Ensuite, la propagation des erreurs dans cette architecture est analysée au moyen d'une extension nommée *Error Model Annex*. Il est ensuite possible de procéder à des vérifications formelles sur les modèles AADL en utilisant des techniques de *model checking* de l'état de l'art.

Enfin, une dernière voie de recherche est fondée sur les passerelles logicielles. Dans (Yakymets, Dhouib et al. 2013 ; Yakymets, Perin et Lanusse 2015), un cadre de travail, appelé *Sophia*, a été établi dans le but de générer automatiquement des analyses de risques (arbres de défaillances, AMDEC) à partir de modèles SysML dans un environnement *Eclipse* (logiciel *Papyrus*). Des méthodes de vérification formelle peuvent ensuite être appliquées : le *model checking* pour vérifier la conformité du comportement du système à des propriétés ou la génération des coupes minimales des arbres de défaillances en utilisant AltaRica. Dans la thèse de Faïda Mhenni (Mhenni 2014), une méthodologie, **Safe**ty integration in **Sys**tems **E**ngineering (**SafeSysE**) est proposée. Comme le montre la figure 3.3, une interface XML a été développée entre les

modèles du système décrits dans SysML et un outil générant les trois analyses de risques suivantes : AMDE, AdD et vérification formelle par *model checking* en utilisant le *model checker* NuSMV [1]. Naturellement, des modèles de fautes (pour les AMDE et les AdD) ainsi que des propriétés à vérifier (pour le *model checking*) doivent aussi être fournis pour générer ces analyses. En poursuivant les mêmes intentions, un profil SysML, dénommé *SafeML*, a été développé dans (Biggs, Sakamoto et Kotoku 2016). Dans ce travail, les concepts propres à la SdF sont modélisés sous la forme d'éléments du langage SysML. Ainsi les *Diagrammes de Définition de Blocs*, *Diagrammes de Blocs Internes* et *Diagrammes d'Exigences* sont-ils utilisés pour mettre en œuvre et modéliser l'APR. L'intérêt majeur est de garantir que les mêmes objets soient manipulés, tant du point de vue des AMS que des pilotes SdF.

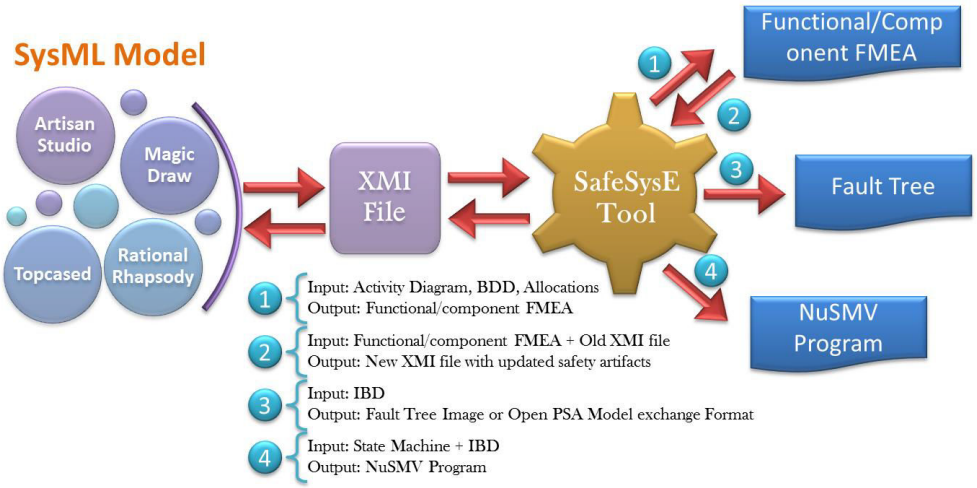


Figure 3.3 – Méthodologie SafeSysE (Mhenni 2014)

Si, dans le cadre des approches orientées, la piste consistant à étendre les modèles fonctionnels est la plus exploitée, il existe tout de même des travaux portant sur l'enrichissement des modèles utilisés en SdF.

3.1.1.2 Enrichissement des modèles dysfonctionnels

Comme nous l'avons vu dans le chapitre 2 (voir paragraphe 2.2.1), le pendant de l'approche MBSE pour la sûreté est l'approche MBSA (Model Based Safety Analysis/Assessment). Dans le cadre de cette approche, l'accent est porté sur les analyses de risques, notamment l'étude de la propagation des fautes dans un système et de ses conséquences. Il est vrai que les modèles MBSA sont généralement construits à partir des modèles fonctionnels (MBSE) (Mauborgne, Deniaud, Levrat, Bonjour, Lamothe et al. 2013). Cependant, le fait que ces deux types de modèles soient ensuite développés « en silos » par deux équipes distinctes engendre des incohérences entre les deux modélisations. Les travaux présentés ci-dessous traitent de cette problématique.

1. http ://nusmv.fbk.eu/

L'outil *Safety Architect*[2] (Dumont, Sadmi et Vallee 2011) affiche les équations dysfonctionnelles associées à une architecture fonctionnelle d'un système afin de générer les arbres de défaillances ainsi que les AMDE. Il est possible d'importer ces architectures fonctionnelles directement depuis des modèles décrits dans le formalisme SysML. L'outil se charge ensuite de générer les analyses de risques précitées. Mais les calculs de probabilité relatifs aux AdD ne peuvent, par exemple, pas être réalisés par le logiciel.

Le langage AltaRica, implémenté dans plusieurs ateliers logiciels, Cecilia OCAS (Dassault Aviation), Simfia (EADS Apsys) et Safety Designer (Dassault Système) (Yakymets, Perin et Lanusse 2014), permet également de procéder à des analyses de sûreté quantifiées (AMDE, AdD) ainsi qu'à des vérifications formelles de propriétés. Certains travaux proposent d'établir des liens entre ce langage et le formalisme SysML. Ainsi, dans le secteur ferroviaire, (Belmonte et Soubiran 2012) élabore un Domain Specific Modeling Language (DSML) dans le but de formaliser les analyses de risques statiques APR et AMDE en se basant sur SysML. A partir des modèles fonctionnels et de ces analyses de risques, le tout décrit au moyen du formalisme SysML, il est possible de générer automatiquement des arbres de défaillances et de calculer leurs coupes minimales en utilisant AltaRica. Contrairement à l'outil *Safety Architect*, cité précédemment, ces travaux sont spécialisés dans le domaine ferroviaire et, plus spécifiquement, adaptés aux pratiques de l'entreprise Alstom. En effet, ces travaux ont été initiés dans le cadre du projet Ingénierie des MOdèles de FonctIons Sécuritaires (IMOFIS), sous l'impulsion de cette entreprise.

La plupart des travaux de recherche abordant le problème de mise en cohérence des points de vue des AMS et des pilotes SdF s'appuient sur les approches orientées présentées ci-dessus. En effet, deux paradigmes structurés, reposant sur un ensemble de modèles formels ou semi-formels, supportent les activités de ces deux métiers : le MBSE pour les AMS et le MBSA pour la SdF. C'est pourquoi il est logique de vouloir étendre l'une ou l'autre de ces deux approches afin de proposer un cadre de travail complet. Toutefois, ce n'est pas la seule manière de traiter la question soulevée. Deux autres approches sont ainsi proposées dans la littérature : l'approche collaborative et l'approche dite d'analyses conjointes.

3.1.2 Approche collaborative et analyses conjointes

3.1.2.1 Approche collaborative

Contrairement aux approches orientées, qui imposent un sens d'extension (du MBSE vers le MBSA, ou l'inverse), l'approche collaborative ne se soucie pas de l'orientation des interactions. Lorsque les acteurs de l'ingénierie construisent leurs modèles, aucun point de vue ou niveau d'abstraction particulier ne leur est imposé. En revanche, une coordination est nécessaire afin de mettre en cohérence l'ensemble des modèles hétérogènes ainsi établis. Deux techniques principales sont mises en œuvre pour déployer ce type d'approche : la **fédération de modèles** (Zhang et al. 2012 ; Guychard et al. 2013) et la **synchronisation de modèles** (Legendre, Lanusse et Rauzy 2017 ; Legendre 2017).

2. https ://www.all4tec.net/safety-architect

La première approche consiste à confronter les différents modèles des métiers impliqués dans la conception d'un système. Pour ce faire, trois espaces interopérables sont définis (voir figure 3.4). L'espace d'information est une collection de bases de données, fichiers et outils dans lesquels l'information est conservée. C'est dans l'espace conceptuel que la fédération de modèles est mise en œuvre à partir des modèles de l'espace d'information. Le troisième espace (*Design Space* sur la figure 3.4) est dédié à la gestion des deux espaces précédents et de l'outillage nécessaire à leur synchronisation. Nous ne détaillerons pas plus cette approche car elle requiert un important système d'information ainsi que de riches bases de données pour être efficace.

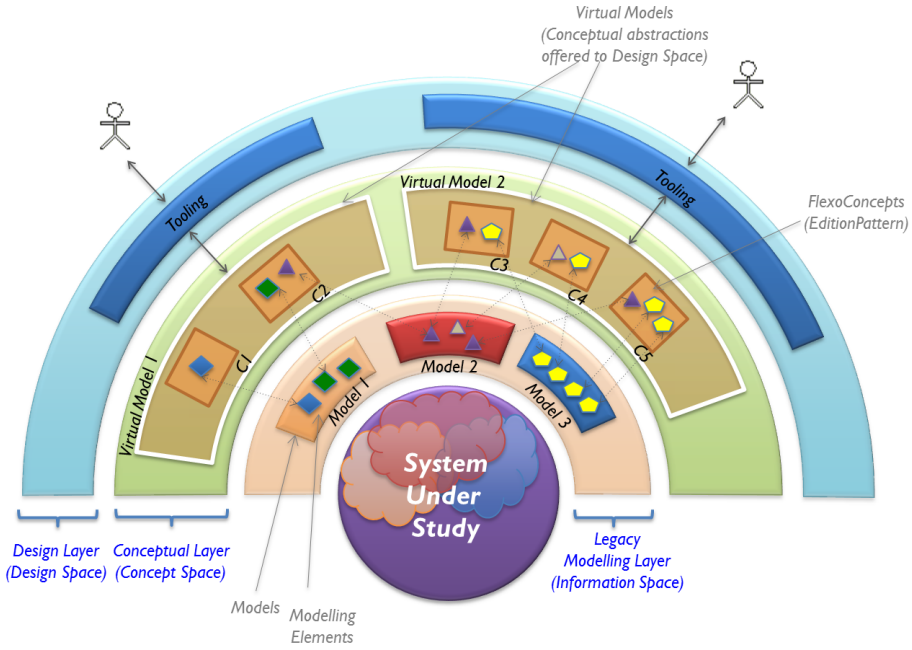


FIGURE 3.4 – Espaces interopérables du paradigme de fédération de modèles (GUYCHARD et al. 2013)

La seconde approche a été déployée par Anthony Legendre dans ses travaux de thèse (LEGENDRE 2017). Il s'agit d'une démarche collaborative et itérative basée sur le principe de synchronisation des modèles. Ces travaux sont plus particulièrement centrés sur les interactions entre les activités de construction des architectures effectuées par les AMS et les analyses de risques produites par les pilotes SdF. Les activités de synchronisation sont incrémentales dans le sens où les incohérences naissant de la confrontation des deux points de vue sont identifiées, puis font l'objet de compromis. Ces derniers font eux-mêmes évoluer progressivement les deux types de modèles. La figure 3.5 montre les objets, produits par les deux métiers, qui peuvent être comparés puis mis en cohérence. Des besoins de synchronisations sont tout d'abord définis puis ces synchronisations sont mises en œuvre au moyen d'un méta-modèle commun (modèle pivot, voir figure 3.1). Les incohérences sont déterminées suite à la confrontation des objets fournis par les deux métiers. Puis, ces incohérences sont résolues par la recherche de compromis, qui modifient l'un des deux, ou les deux, objets initiaux. L'ensemble de ces activités est chapeauté par un « responsable de synchronisation ».

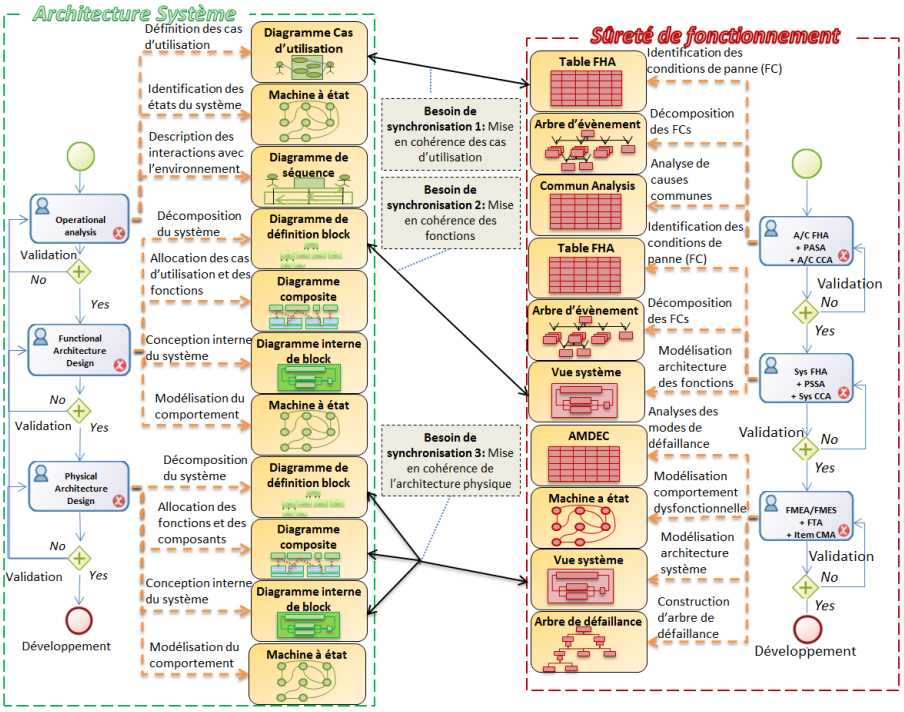


FIGURE 3.5 – Démarche de convergence des points de vue des AMS et des pilotes SdF (LEGENDRE 2017)

3.1.2.2 Analyses conjointes

Cette approche, élaborée et développée dans les travaux de thèse de Pierre Mauborgne (MAUBORGNE 2016), propose d'intégrer l'IS à la SdF par le biais d'un modèle conceptuel qui définit la sémantique d'un nouveau cadre méthodologique d'ingénierie nommé Ingénierie de Systèmes Sûrs de Fonctionnement (ISSdF). Ainsi un processus ISSdF, applicable par les architectes système, y est défini. Ce processus comprend des activités classiques de l'IS, enrichies par des aspects de SdF, ainsi que de nouvelles activités orientées SdF que devra réaliser l'AMS. Plus précisément, dans (MAUBORGNE, DENIAUD, LEVRAT, BONJOUR, MICAËLLI et al. 2016), une méthode nommée *Operational and System Hazard Analysis*, est proposée pour déterminer et classifier des scénarios dysfonctionnels dans l'objectif d'établir des *safety goals*. Cette méthode repose sur l'APR classique mais, contrairement à cette dernière, la détermination des dangers principaux est effectuée par les AMS, et non, comme c'est traditionnellement le cas, par les pilotes SdF. Les activités nécessitant des connaissances plus pointues en SdF sont en effet réservées à ces derniers. La démarche déployée fournit des scénarios dysfonctionnels plus riches et pertinents que l'APR conventionnelle. En outre, elle contribue à faire converger très tôt dans le cycle de conception les points de vue des AMS et des pilotes SdF.

3.1.3 Synthèse

La question de la mise en convergence des points de vue des AMS et des pilotes SdF a donné lieu à de multiples recherches dont les paragraphes précédents offrent un aperçu. Ces travaux se déroulant dans le contexte décrit dans le chapitre 2, il en résulte un certain nombre de spécificités et de contraintes qui sont à mettre en regard avec les propositions de la littérature :

- **Contexte industriel et processus en jeu** : la plupart des approches s'inscrivent dans un contexte industriel et sont majoritairement centrées sur les processus de conception d'architectures et d'implémentation de la phase amont du cycle de conception en V (voir figure 2.2). Plus particulièrement, il s'agit de synchroniser judicieusement ces deux processus, menés parallèlement par les AMS et les pilotes SdF (voir figure 2.6) ;
- **Approche MBSE** : l'ensemble des approches sont développées dans le cadre MBSE ou, du moins sont conformes à ce cadre ;
- **Traitement des exigences** : les travaux exposés se focalisent peu sur les liens entre les exigences et les modèles : comment prendre en compte les résultats de la vérification des exigences sur les modèles ? Quels impacts ont-ils sur la formulation des exigences ou sur les modèles ? Quelles sont les conséquences de la confrontation des points de vue des AMS et des pilotes SdF sur les exigences ?

Ainsi les approches présentées abordent la mise en cohérence de manière globale, mais portent tout de même l'accent sur les modèles d'architectures. Les travaux relevant des approches orientées et collaboratives supposent des modèles MBSE suffisamment aboutis pour être exploitables (méthode MéDISIS (P. David 2009 ; P. David, Idasiak et Kratz 2010 ; Cressent, Idasiak, Kratz et P. David 2011), approches AADL - Error Annex (Bozzano et al. 2010) et EAST-ADL - Error Annex (Tucci-Piergiovanni et al. 2014 ; Blom et al. 2016)). Or, le niveau de maturité actuel des modèles MBSE chez Renault n'est pas le même : l'introduction de la démarche d'Ingénierie Système a été initiée depuis une dizaine d'années mais l'approche MBSE est en cours de déploiement. Il en va de même pour l'approche MBSA, qui ne fait pas partie des pratiques courantes et entérinées chez Renault.

En revanche l'approche d'analyses conjointes, mise au point par Pierre Mauborgne (Mauborgne 2016), ne présuppose pas d'un contexte MBSE établi et consolidé. En effet, la méthode *Operational and System Hazard Analysis* déployée est un préalable à la construction des modèles d'architectures manipulés dans le cadre d'une approche MBSE. C'est pourquoi l'approche d'analyses conjointes est adaptée au contexte de ces travaux de thèse. Le principe de séparation des préoccupations est respecté en conservant isolées certaines activités des deux points de vue bien qu'ils partagent, pour certaines tâches, un modèle commun (doté d'une sémantique et d'une syntaxe précises). L'utilisation d'un modèle commun aux deux disciplines d'ingénierie n'est possible que si elles poursuivent le/les mêmes objectifs de modélisation et restent à un niveau d'abstraction approprié. Les analyses poussées de SdF utilisent effectivement des modèles difficilement manipulables par les AMS (Réseaux de Petri, chaînes de Markov...).

Le rapprochement des activités des AMS et celles de la SdF préconisé par l'ISSdF s'accorde avec le contexte de ces travaux de thèse. C'est pourquoi les propositions

de ce mémoire relèvent de cette approche. En revanche, l'ISSdF n'est pas focalisée sur le traitement des exigences et le lien de satisfaction à établir (par vérification) entre les exigences et les modèles d'architectures sur lesquelles elles portent. Cette question, relative au traitement des exigences, à leur formulation, analyse, allocation, formalisation, vérification et gestion au fil du cycle de conception, le tout dans une approche visant à faire converger les points de vue de l'AMS et des pilotes SdF, fait l'objet du paragraphe suivant.

3.2 Traitement et formalisation des exigences

Dans le contexte de travail qui est le nôtre, la problématique de convergence des points de vue des AMS et des pilotes SdF est abordée sous le prisme des exigences. Une des questions les plus préoccupantes soulevée par Renault et à laquelle ces travaux doivent fournir une réponse (au moins partielle), concerne la possibilité de vérifier les exigences au plus tôt dans le cycle de conception (voir figure 2.8). En fait, l'objectif est de vérifier le caractère *correct* des exigences dès que celles-ci ont été émises. Le qualificatif *correct* est ici entendu comme répondant à des critères de qualités donnés. Ces critères, fixés avec les experts métier, sont relatifs à la cohérence des exigences entre elles et leur complétude. Ceci revient à ajouter des tests de conformité à chaque phase des cycles de conception (du point de vue des AMS et des pilotes SdF, voir figures 2.8 et 2.5), comme préconisé dans (Filax, Gonschorek et Ortmeier 2016). Il convient, dans un premier temps, de rappeler les différentes méthodes de vérification disponibles lors des phases amont de conception.

3.2.1 Méthodes de vérification des exigences en phase amont de conception

En phase amont de conception, les exigences sont vérifiées non sur le système lui-même, mais sur des modèles du système. Selon le type d'exigence que l'on souhaite vérifier, des modèles appropriés doivent être utilisés. En ce qui concerne nos travaux, nous avons vu que les exigences traitées portaient sur le comportement d'une fonction (voir section 2.3.3), que ce soit du point de vue des AMS ou de celui des pilotes SdF. Trois techniques principales (Evrot 2008) sont employées pour vérifier ce type d'exigences :

1 *La simulation* : technique répandue (Cressent, Idasiak, Kratz et P. David 2011 ; Nouacer et al. 2016) qui repose sur l'exécution symbolique des modèles et la réalisation de tests de conformité aux besoins exprimés. L'exécution de la simulation nécessite une sémantique opérationnelle qui définit de manière déterministe le comportement d'un modèle en réaction à des stimuli d'entrée. La principale limite réside dans l'exhaustivité des scénarios testés. La simulation ne donne qu'une présomption de comportement correct mais demeure fortement corrélée à l'expérience et l'expertise des personnes qui la pratiquent ;

2 *Le theorem proving* : cette méthode consiste à considérer la vérification comme un théorème à prouver à partir d'un ensemble d'axiomes. Le modèle du système et les exigences à vérifier doivent être transformés en objets mathématiques. Des conclusions sont inférées directement à partir d'une description d'événements ou

d'opérations faisant évoluer le système (Roussel et Denis 2002). Toutefois l'automatisation complète est rarement possible étant donné que l'utilisateur doit souvent interagir avec l'assistant de preuves et que la préparation de la preuve nécessite la spécification d'éléments sortant du cadre direct de l'expression des besoins du cahier des charges ;

3 ***Le model checking*** : il s'agit d'une technique automatisée qui, partant d'un modèle d'état fini d'un système et d'une propriété formelle, vérifie systématiquement si cette propriété est valide pour ce modèle (Baier et Katoen 2008). Cette méthode comporte trois phases : la modélisation du système à étudier et des propriétés à vérifier, l'exécution de l'algorithme de *model checking* et l'analyse des propriétés (satisfaites, non satisfaites ou mémoire saturée). Cette technique est implémentée par différents outils, selon le type de modèles à vérifier ou l'algorithme à utiliser : UPPAAL [3], adapté aux automates temporisés, NuSMV, qui implémente le *model checking* symbolique, PRISM [4], conçu pour modéliser des comportements probabilistes ou encore SPIN [5], qui utilise les ordres partiels.

Notons que les deux dernières méthodes de vérification sont formelles, contrairement à la simulation. La sûreté du véhicule autonome ne saurait être démontrée uniquement par des tests de validation ou des simulations. (Kalra et Paddock 2016) montre que le véhicule autonome devrait rouler des centaines de millions de kilomètres pour atteindre un niveau de fiabilité satisfaisant. Qui plus est, l'introduction progressive de l'approche MBSE autorise l'utilisation des méthodes formelles. Parmi les deux principales méthodes formelles présentées, le *model checking* a été choisi car les limites mentionnées de la technique du *theorem proving* empêchent pour le moment son application dans un contexte automobile opérationnel. La technique de vérification retenue impose de traiter des exigences exprimées sous forme de propriétés formelles. Plus particulièrement, le *model checking* requiert que les propriétés à vérifier soient exprimées dans une logique temporelle adaptée (Baier et Katoen 2008). Or, comme ceci a été exposé dans le chapitre 2 (voir paragraphe 2.3.3), les données d'entrée de ce travail, fournies respectivement par les AMS et les pilotes SdF, sont des exigences exprimées en langage naturel. C'est pourquoi, la question de formalisation des exigences, *i.e.* de la transformation d'énoncés en langage naturel en propriétés formelles, se pose. Ce point fait l'objet d'une littérature riche.

3.2.2 Formalisation des exigences

La question de la formalisation est centrale dans le cadre de l'ingénierie des exigences et donc largement abordée dans la littérature relevant de cette discipline.

Un processus, intitulé *Natural Language Processing (NLP)*, de traduction automatique d'exigences exprimées en langage naturel en modèles a été développé par (Gervasi et Nuseibeh 2002 ; Cabral et Sampaio 2008 ; Almeida Ferreira et Silva 2009). Malgré l'intérêt de ces travaux, ils ne sont applicables qu'à des exigences qui ne sont pas rédigées en langage naturel libre. En effet, les exigences traitées doivent être très structurées et régulières, utiliser un langage précis ainsi qu'un vocabulaire bien défini.

3. http ://www.uppaal.org/
4. http ://www.prismmodelchecker.org/
5. http ://spinroot.com/spin/whatispin.html

Dans le domaine informatique, la question de la formalisation est également pertinente. La méthode B (Behm et al. 1999 ; J.-L. Boulanger 2014) a prouvé son efficacité dans un contexte industriel ferroviaire. Cette méthode, dédiée à la spécification de logiciels embarqués critiques permet, par raffinements successifs, de démontrer que l'implémentation concrète (*i.e.* le code exécutable, en ADA ou en C) est bien conforme aux exigences initiales. L'écart entre la spécification et le code est ainsi réduit. Néanmoins, cette méthode requiert des données d'entrée déjà exprimées sous forme de machines abstraites alors que, dans notre contexte, les exigences formulées initialement restent exprimées en langage naturel. De plus, pour appliquer la méthode B, l'allocation des exigences aux composants logiciels doit être connue. Dans (Zhioua, Roudier et Ameur-Boulifa 2017 ; Zhioua, Roudier et Ameur 2017), les auteurs proposent une méthode pour formaliser des directives de sécurité complexes dans le but de les rendre moins ambigües. Ces directives sont exprimées en langage naturel. Afin de procéder à la formalisation, des actions abstraites sont définies à partir du programme devant respecter les directives. Or, de la même manière que le NLP, les directives de sécurité analysées, bien que rédigées en langage naturel, sont aussi très structurées.

Les méthodes précédentes considèrent des énoncés qui sont certes structurés mais elles n'utilisent pas de gabarits (nommés « templates » ou « boilerplates » en anglais). D'autres méthodes de formalisation reposent sur leur utilisation. Cet usage est effectivement recommandé par les guides de bonnes pratiques de l'ingénierie des exigences (Pohl 2010 ; Badreau et J.-L. Boulanger 2014). Cependant le principal problème de ces gabarits est la rigidité qu'ils imposent aux rédacteurs d'exigences, en raison des formes fixes à respecter. Ceci réduit en conséquence l'expressivité. Pour pallier cette difficulté, il est possible d'avoir recours à un grand nombre de gabarits, mais ceci complexifie leur gestion et leur utilisation. Malgré ces limitations, plusieurs démarches de formalisation se basent sur ce principe.

(Peres, Yang et Ghazel 2012 ; Ghazel, Yang et El-Koursi 2015) construisent une méthode basée sur les *patterns* pour formaliser des exigences portant sur le comportement d'un système. Cette démarche prend en données d'entrée des exigences exprimées en langage naturel pour les transformer en propriétés formelles. Pour ce faire, le processus détaillé et documenté, représenté sur la figure 3.6 est appliqué. Plus précisément, trois opérations sont définies : *Clarify* pour supprimer les ambigüités, *Split* afin de décomposer les exigences trop longues en sous-exigences et *Modify* pour ajouter, supprimer ou corriger des informations contenues dans les exigences. Une fois ces différentes opérations réalisées, l'exigence initiale est transformée en une ou plusieurs exigence(s) formalisable(s). La notion d'exigence formalisable est définie ainsi : « une exigence qui peut être directement formalisée comme une propriété », *i.e.* une formule logique en Full Computation Tree Logic (CTL*). L'approche de formalisation proposée dans le cadre de ces travaux de thèse partagent des points communs avec cette méthode, telles que le type d'exigences traité (exigences portant sur un comportement) et l'intervention de l'expertise dans ce processus de formalisation. Toutefois, les exigences sont analysées une à une tandis que nous avons affaire à deux ensembles d'exigences (les exigences fonctionnelles et les exigences de sûreté, présentées dans le paragraphe 2.3.3). De plus, le rôle de l'expertise dans ce processus n'est pas clairement défini en termes d'activités et de planning.

(Lebeaupin 2015 ; Lebeaupin, Rauzy et Roussel 2017) partent d'une observation

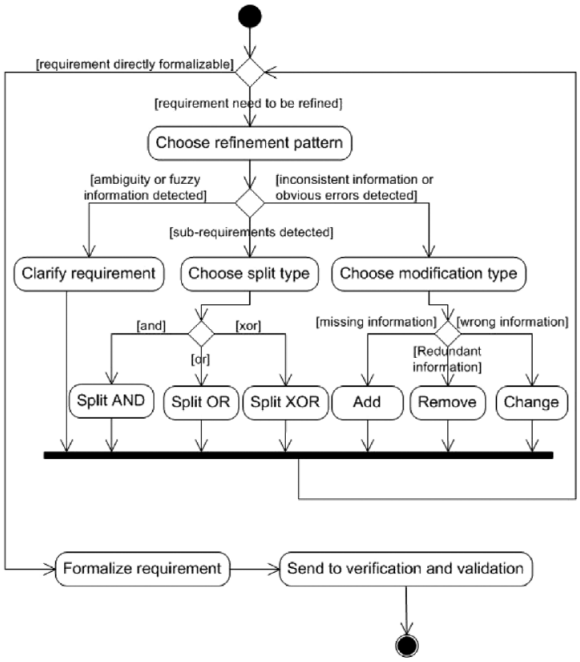


FIGURE 3.6 – Processus de formalisation des exigences (GHAZEL, YANG et EL-KOURSI 2015)

que nous avons également faite : la difficulté d'articuler correctement une approche MBSE avec les exigences exprimées en langage naturel, et, plus généralement, avec l'ingénierie basée sur les documents. Pour contribuer à résoudre ce problème, Benoît Lebeaupin propose, dans ses travaux de thèse (LEBEAUPIN 2017) (menés, pour partie, au sein de l'entreprise *Safran*), un langage « semi-formel » doté d'une syntaxe modulable. Ce langage n'est pas complètement formel car des fragments d'exigences, voire des exigences complètes, demeurent exprimés en langage naturel. Par ailleurs la modularité de la syntaxe signifie qu'un nombre important de structures d'exigences est possible. Un double objectif est poursuivi : réduire les ambigüités inhérentes au langage naturel tout en conservant son pouvoir d'expressivité. La figure 3.7 illustre le principe de la démarche mise au point pour répondre à cette question.

Les modèles utilisés dans (LEBEAUPIN 2017) sont directement empruntés au langage de modélisation SysML : *Diagrammes de Bloc Interne* pour les modèles d'architectures et *Diagrammes d'état* pour les modèles comportementaux. Les exigences doivent contenir des références explicites aux éléments des modèles construits. Enfin, une vérification de cohérence entre ces trois éléments : exigences, syntaxe formelle et éléments de modèle, est réalisée afin de permettre au lecteur de choisir le sens correct. Ceci signifie que le lecteur associe alors une exigence textuelle à un élément correct de l'« espace de sens ». L'objectif de la démarche est de créer une bijection entre texte et sens. Les ambigüités, telles qu'un texte associé à plusieurs sens possibles, ou plusieurs exigences ayant la même signification, sont ainsi éliminées. Néanmoins, les exigences traitées sont des exigences dites de « bas niveau » (voir paragraphe 3.2.3) dans le sens où elles portent sur des éléments d'une architecture physique ou temps réel pour Stimulus. En outre, les ambiguïtés sont supprimées et non exploitées pour mettre en lumière plusieurs solutions concurrentes crédibles (voir paragraphe 4.2.2). Enfin, la vérification formelle

des exigences sur les modèles de comportement du système spécifié n'est pas mise en œuvre.

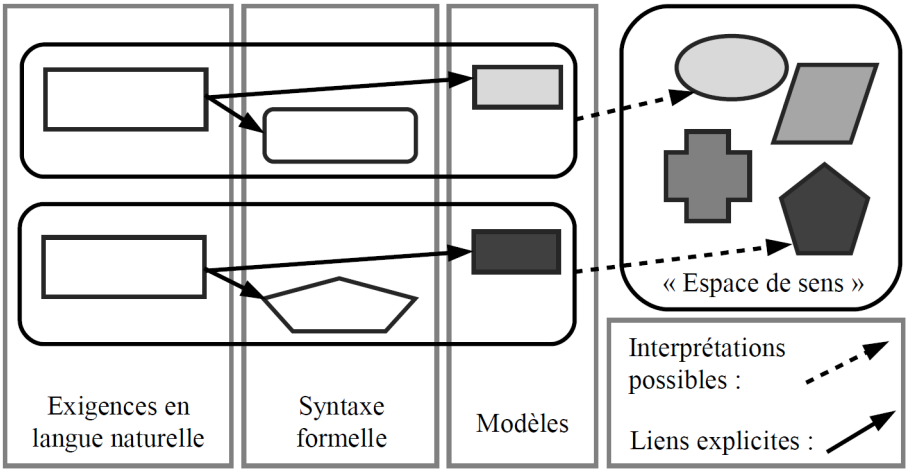


FIGURE 3.7 – Principe de formalisation proposée par (LEBEAUPIN 2017)

En conclusion, la littérature offre de nombreux éléments pour aborder le problème de formalisation des exigences. Ces approches reposent sur l'équilibre entre un traitement *a priori* qui consiste à imposer une structure, syntaxe et sémantique précises dès la rédaction de l'exigence, et une analyse *a posteriori* laissant plus libre l'expression mais compliquant sensiblement le processus de formalisation. Si ces méthodes s'avèrent performantes pour résoudre le problème de formalisation en lui-même, ces travaux ne sont toutefois pas centrés sur la mise en œuvre pratique de la vérification formelle des exigences, *i.e.* la vérification des exigences sur un modèle de comportement formel (par exemple, à l'aide d'un *model checker*) et l'exploitation des résultats de cette vérification (du ou des contre-exemples retournés).

3.2.3 Mise en œuvre de la vérification formelle des exigences

3.2.3.1 Remarques lexicales

Afin de clarifier les propos qui vont suivre, il nous a paru important de définir, dans le cadre de ce mémoire, les notions de ***point de vue*** et de ***niveau***. En effet, le principal inconvénient de ces deux termes est leur flou sémantique. A notre connaissance, il n'existe pas de définitions normées auxquelles nous puissions nous référer. En conséquence, nous nous contenterons de préciser la manière dont nous employons ces notions. En règle générale, le *point de vue* désigne, d'après le dictionnaire *Larousse*, un *« plan, aspect sous lequel on se place pour examiner quelque chose »*. Nous utilisons bien ce terme en ce sens, mais il recouvre deux réalités différentes selon le contexte de son emploi :

— *Conception d'un système* : dans le cadre de l'IS, les ouvrages de référence (SEBOK 2017 ; FIORÈSE et MEINADIER 2012) définissent, entre autres (d'autres points de vue sont donnés mais sortent de notre périmètre), trois points de vue : *opérationnel, fonctionnel* et *organique* (appelée *physique* chez Renault) ;

— *Métier* : nous utilisons le terme *point de vue* ou *perspective* pour décrire la façon dont un métier donné (en l'occurrence AMS et pilote SdF) pense et voit le système qu'il contribue à concevoir. Cette perception du système, propre à chaque métier, est conditionnée par de multiples facteurs, tels que les contraintes subies, les objectifs à tenir, ou encore les cultures métier spécifiques (voir paragraphe 2.2.2.3).

En définitive, le *point de vue* est toujours une manière particulière d'appréhender le système à réaliser. Néanmoins, selon l'acception du terme, les réalités ne se recoupent pas nécessairement. Ainsi, le point de vue de l'AMS est tout autant opérationnel, fonctionnel que physique, selon la phase du cycle de conception dans laquelle le système se trouve. Et, inversement, le point de vue fonctionnel, par exemple, peut être tout aussi bien considéré par les AMS que par les pilotes SdF.

L'emploi du terme « niveau » porte également à confusion. En effet, les exigences dites de « bas niveau », sont généralement des exigences qui portent sur un élément précis d'une architecture physique, et souvent quantifiées (voir paragraphe 2.3.3 (Albinet, J.-L. Boulanger et al. 2007a ; Evrot 2008 ; Kang et al. 2013 ; Genius et Apvrille 2016 ; Lebeaupin 2017)). Or, nous pensons que cette dénomination est source de confusions et de malentendus. Effectivement : l'obtention de telles exigences, qualifiées de « bas niveau », à partir d'une ou plusieurs exigences de haut niveau, est, en réalité, le résultat de deux opérations :

1 Un raffinement ;

2 Une modification du point de vue : du fonctionnel au physique.

Dans le contexte de ce travail, nous faisons usage du terme « niveau », à propos des exigences, uniquement pour désigner le **degré de raffinement** (plus ou moins élevé). Ainsi, les exigences que nous traitons sont bien des exigences de bas niveau (*i.e.* niveaux 2 et 3 dans notre cas d'étude, voir paragraphe 2.3.3), mais qui sont allouées à des éléments de l'architecture fonctionnelle et, non de l'architecture physique. Par exemple (voir paragraphe 2.3.3), « Ex_EFC » est allouée à la fonction *Supervision AD* (figure 2.12) et « Ex_EFS » à la fonction *Sup_sub* (figure 2.13).

3.2.3.2 Caractérisation des approches proposées

La figure 3.8 offre une représentation simplifiée (seules les exigences qui nous intéressent apparaissent) de la détermination et du traitement des exigences de comportement chez Renault. Nous reprenons la structure hiérarchique en 3 niveaux, adaptée à notre cas d'étude (paragraphe 2.3.3). Les besoins des parties prenantes sont d'abord capturés et transformés en exigences de produit (voir paragraphe 2.3.3). Puis, ces dernières sont progressivement raffinées et allouées sur l'architecture fonctionnelle. La vérification des exigences de comportement quantifiées repose, pour partie, sur l'exécution de modèles simulables. Elles sont donc vérifiées, en phase amont de conception, par la technique de simulation, présentée dans le paragraphe 3.2.1. Les modèles utilisés sont des modèles de comportement des composants (majoritairement construits grâce à *Matlab Simulink*[6]). Les autres exigences (quantifiées ou non) sont vérifiées par d'autres moyens, tels que des revues de conception ou des tests.

6. https ://fr.mathworks.com/products/simulink.html

Néanmoins, les exigences demeurent informelles, *i.e.* rédigées en langage naturel non contraint, malgré le raffinement opéré (du niveau 1 au niveau 3). Par exemple, il n'existe pas d'obligation de faire référence explicite à un élément du modèle de comportement lorsque l'on rédige une exigence de comportement. C'est pourquoi les liens qui existent (ou doivent exister) entre le contenu des exigences de comportement informelles de niveau 3 non quantifiées, celui des exigences de comportement informelles quantifiées, et les modèles de comportement des composants ne sont pas clairement définis. Comme

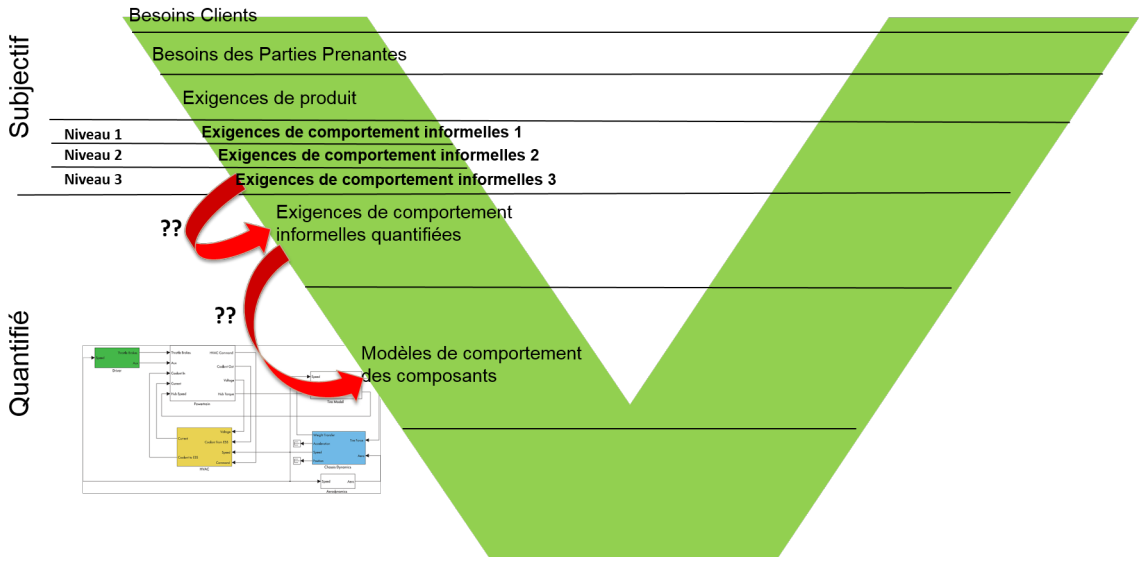


FIGURE 3.8 – Processus de vérification en vigueur chez Renault

le montre la figure 3.9, deux principales approches, complémentaires, pour vérifier les exigences de comportement de bas niveau (niveaux 2 et 3 dans notre contexte) peuvent être distinguées :

1 *Approche orientée vérification* : cette approche rassemble des méthodes de vérification formelle d'exigences de comportement de bas niveau qui peuvent être quantifiées (ALBINET, BEGOC et al. 2008 ; SHARVIA et PAPADOPOULOS 2015 ; GENIUS et APVRILLE 2016), ou non (KANG et al. 2013 ; DJOUDI, BOUANAKA et ZEGHIB 2016 ; MARINESCU et al. 2014) ;

2 *Approche orientée formalisation* : deux types de travaux relèvent de cette approche : des études focalisées uniquement sur l'opération de formalisation (GERVASI et NUSEIBEH 2002 ; PERES, YANG et GHAZEL 2012 ; ZHIOUA, ROUDIER et AMEUR-BOULIFA 2017), et des travaux dans lesquels le processus de formalisation s'intègre dans un paradigme MBSE plus global : un processus de vérification est également déployé (EVROT 2008 ; LEBEAUPIN 2017).

Ces deux approches introduisent dans le processus de vérification un nouveau domaine, qualifié d'« objectif » (figure 3.9). Il s'agit de l'espace des objets (exigences, modèles d'architectures, modèles de comportement) formels. Nous utilisons également ce concept pour la démarche proposée. En revanche, la première approche ne crée pas de liens entre le contenu des exigences de comportement formelles de bas niveau vérifiées (sur les modèles de comportement) et celui des exigences initiales informelles,

dont les exigences vérifiées (formelles) sont issues. Quant à la seconde approche, plus proche de ce que nous proposons, elle n'exploite pas pour autant les ambigüités des exigences informelles traitées, contrairement à notre démarche (voir paragraphe 4.2.2). Enfin, notons que la problématique de rapprochement des points de vue des AMS et des

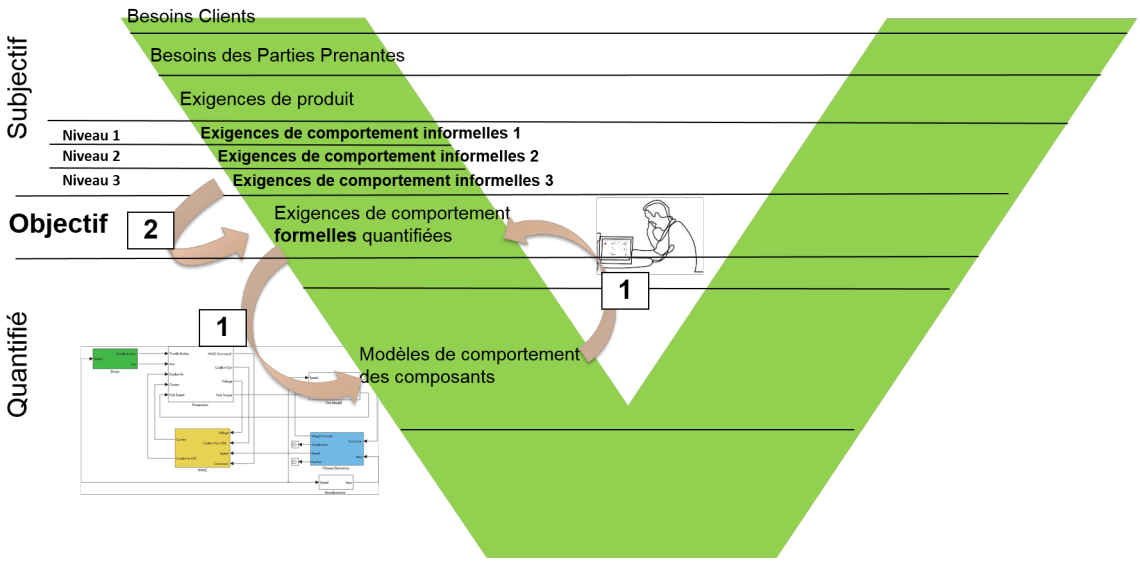


FIGURE 3.9 – Propositions de la littérature

pilotes SdF n'est pas abordée dans ces travaux. C'est pourquoi nous avons recherché les travaux qui proposaient des approches globales de rapprochement des disciplines de l'AMS et de la SdF en se focalisant sur le traitement (formalisation et vérification) des exigences.

3.2.4 Convergence des points de vue centrée sur le traitement des exigences

Dans (EVROT 2008 ; PÉTIN et al. 2010), les auteurs ne visent pas explicitement à faire converger les points de vue des AMS et des pilotes SdF. Cependant, ils élaborent une approche méthodologique permettant l'identification, la formalisation et la structuration d'exigences globales relatives à un système, puis leur projection sur une architecture physique. Le langage SysML est utilisé pour la description de la structure et du comportement du système ainsi que celle des exigences qui lui sont allouées. Les systèmes étudiés sont des systèmes manufacturiers. Un processus rigoureux de décomposition puis allocation des exigences aux éléments architecturaux est proposé. Par ailleurs deux types d'exigences, proches de ceux que nous analysons, sont traités : les exigences fonctionnelles d'une part, et les exigences de sécurité, d'une autre. En outre, les exigences analysées sont formalisées dans le but d'être vérifiables par un *model checker* (l'outil UPPAAL est utilisé). Ainsi, la formalisation et la vérification sont mises en œuvre, le tout dans une approche MBSE. Néanmoins, le processus de formalisation, et les impacts que ses résultats peuvent avoir tant sur les exigences initiales que sur les modèles d'architecture ne sont pas détaillés. De plus, les conséquences potentielles,

sur les exigences et les modèles, causées par la confrontation du point de vue de l'AMS et de celui des pilotes SdF ne sont pas analysées.

Par la suite, des travaux, menés au Laboratoire d'Analyse et d'Architecture des Systèmes - CNRS (LAAS-CNRS), se sont attaqués, cette fois explicitement, au problème de convergence des deux points de vue. Dans les travaux de thèse de Romaric Guillerm (Guillerm 2011), une démarche d'Ingénierie Système intégrant la SdF est élaborée. Cette démarche repose sur les deux éléments suivants : un modèle d'information en SysML et une méthodologie de déclinaison des exigences de sûreté de fonctionnement (assurant la traçabilité de ces exigences). A chaque étape de conception, les activités à accomplir pour obtenir un système sûr de fonctionnement sont détaillées. Le secteur d'application est l'aéronautique. Le modèle d'information SysML, représenté sur la figure 3.10, met en relation les concepts manipulés et les activités effectuées par, respectivement, les AMS et les pilotes SdF. Malgré le fait que ces travaux se concentrent bien sur les exigences de SdF et soulignent l'importance de formaliser les liens qu'elles entretiennent avec les objets de l'AMS, il n'est pas fait mention du traitement des exigences fonctionnelles. Par ailleurs les exigences ne sont pas formalisées, et, bien que les *test cases* soient spécifiés, ils ne sont pas mis en pratique. Enfin les aspects relatifs aux méthodes et outils n'entrent pas dans le périmètre de ces travaux.

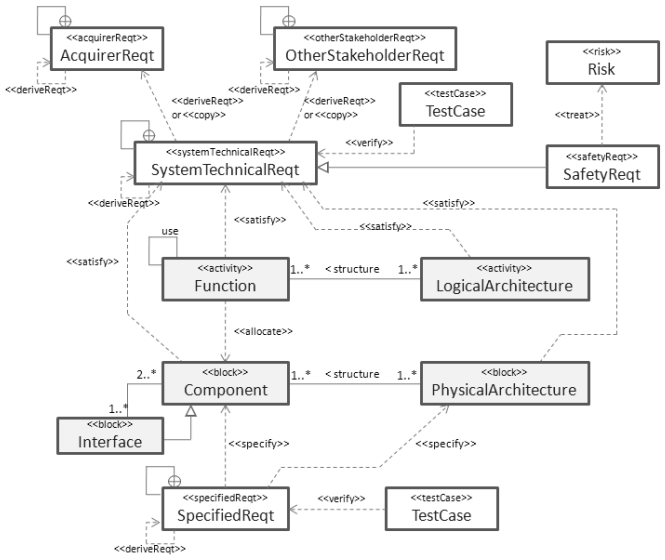


Figure 3.10 – Modèle d'information SysML (Guillerm 2011)

Pour conclure ce paragraphe, nous présentons des travaux appliqués au domaine automobile. Dans le cadre du projet MEthode de Modelisation pour la VAlidation et la Tracabilité des EXigences (MEMVATEX), porté par les entreprises Siemens et Mondi-Tech, une méthodologie de conception de systèmes embarqués automobiles temps-réel a été définie (Albinet, J.-L. Boulanger et al. 2007b ; Albinet, Begoc et al. 2008). Cette méthodologie est centrée sur l'expression, la traçabilité et la vérification d'exigences temps-réel et repose sur le formalisme EAST-ADL2, ainsi que les profils d'Unified Modeling Language (UML) Modeling and Analysis of Real Time and Embedded systems (MARTE) et SysML. Deux objectifs principaux ont été visés : enrichir

la formulation des exigences analysées, et réduire les écarts entre les spécifications et les modèles solution, le tout en respectant la séparation nette entre le domaine du problème (les exigences) et celui de la solution. Ainsi les trois éléments majeurs de cette méthode sont : les modèles d'exigences, les modèles solution et les moyens de Vérification & Validation (V&V). Les auteurs construisent un nouveau type d'exigences, nommé *« MeMVaTEx Requirement »*, par extension du stéréotype standard *Requirement* fourni par SysML, dans le but de lier les exigences directement aux processus de V&V. Cependant, les exigences traitées et ainsi vérifiées sont uniquement des exigences de temps réel. Par ailleurs, aucun processus de formalisation des exigences, comme nous l'entendons dans ces travaux, n'est proposé. Enfin, il n'est pas fait non plus mention de la convergence des points de vue des AMS et des pilotes SdF.

Cette problématique est en revanche traitée dans les travaux de thèse d'Ofaina Taofifenua (Taofifenua 2012) réalisés chez Renault. Une formalisation des deux domaines de connaissances (celui de l'IS et de la SdF), tels qu'ils sont pratiqués au sein de l'entreprise Renault, a été produite, en utilisant pour ce faire des ontologies. L'ontologie est considérée comme le modèle de référence global du système auquel tous les documents et modèles de conception doivent être conformes. Deux ontologies, relatives aux deux métiers, sont proposées, puis une vérification de la cohérence entre ces deux objets est possible. Notons que parmi les concepts formalisés dans ces ontologies, apparaît celui d'exigence. Les incohérences détectables au moyen de cette approche concernent la structure même du corpus d'exigences (toutes les exigences sont-elles bien raffinées et allouées à des éléments du système ? Est-ce que le niveau d'Automotive Safety Integrity Level (ASIL), affecté à une exigence fonctionnelle de SdF de haut niveau, est bien distribué sur les exigences de plus bas niveau qui la raffinent ?) ou encore la sémantique de chaque ensemble d'exigences (est-ce que le même terme, contenu dans les exigences des deux métiers, a la même signification ?). Cependant, la mise en œuvre efficace de cette démarche suppose la création d'une nouvelle fonction chez Renault : celle d'« ingénieur ontologie », en charge de la création, de la gestion et de l'exploitation des résultats fournis par l'utilisation d'ontologies. Or, cette fonction n'a pour le moment pas été créée au sein de l'entreprise. Plus fondamentalement, ces travaux ne sont pas centrés sur le contenu des exigences et la manière dont elles sont formulées. Il n'est également pas question de vérifier formellement les exigences traitées sur des modèles du système à concevoir.

3.3 Synthèse et positionnement des travaux

Chacune des questions de recherche soulevées dans la section 2.4.3 a donné lieu à de nombreuses études, décrites dans les paragraphes précédents. La table 3.1 résume les principaux travaux qui contribuent à répondre aux différents axes de recherche identifiés.

Nous avons commencé par présenter des approches globales visant à la convergence des points de vue des AMS et des pilotes SdF, dans un contexte industriel. Il en est ressorti que les approches orientée et collaborative reposaient sur une pratique déjà avancée et entérinée des modèles, méthodes et outils de l'approche MBSE, pour les AMS, et MBSA, pour les pilotes SdF. Or, le niveau de maturité requis n'est pas compatible

Question	Solutions	Lacunes identifiées
1. Comment mettre en cohérence les points de vue ?	Approches orientées (3.1.1) Approches collaboratives (3.1.2.1)	Présuppose une ingénierie basée sur les modèles
	Analyses conjointes (3.1.2.2)	N'intègre pas de processus de vérification des exigences
2. Comment vérifier des exigences en phase amont ?	Simulation (3.2.1)	Exhaustivité des scénarios
	Theorem proving (3.2.1)	Niveau d'expertise de l'utilisateur
	Model checking (3.2.1)	Formalisation des exigences
3. Comment formaliser un ensemble d'exigences ?	NLP (3.2.2) Formalisation directives de sécurité informatiques (3.2.2)	Exigences très structurées langage, vocabulaire très précis
	Langage « semi-formel » (3.2.2)	Type d'exigence traité Pas de vérification sur des modèles
	Méthode basée sur les *patterns* (3.2.2)	Analyse exigences une à une Rôle de l'expertise peu clair

TABLE 3.1 – Synthèse de l'état de l'art

avec le contexte industriel de la thèse (l'approche MBSE est en cours de déploiement). C'est la raison pour laquelle, la démarche proposée dans le cadre de ces travaux de thèse relève plutôt de l'approche d'analyses conjointes, dont l'ISSdF (MAUBORGNE 2016 ; MAUBORGNE, DENIAUD, LEVRAT, BONJOUR, MICAËLLI et al. 2016) constitue le principal représentant. Or, l'ISSdF n'intègre pas explicitement un processus de vérification des exigences opérationnel au plus tôt dans le cycle de conception, ce qui est un des objectifs de la thèse.

Une revue des principales techniques de vérification applicables dans la phase amont du cycle de conception nous a orienté vers le *model checking*. Ce choix est également cohérent avec l'introduction de l'approche MBSE chez Renault car les modèles manipulés dans ce cadre sont formels ou semi-formels, et vérifiables par *model checking* (du moins pour les modèles formels). Cependant, cette technique pose le problème de la formalisation des exigences : celles-ci doivent être exprimées sous forme de propriétés formelles, dans une logique temporelle appropriée.

Après avoir présenté les principales méthodes de formalisation, il nous est apparu que peu d'entre elles mettaient en œuvre un processus opérationnel de vérification des exigences, ni ne s'intégraient dans une approche MBSE globale. Parmi les approches remplissant ces deux critères (processus de formalisation des exigences et vérification effective des exigences dans un contexte MBSE), aucune ne traitait cependant de la question du rapprochement des points de vue des AMS et des pilotes SdF. Il est vrai que des approches visant à rapprocher les deux points de vue en se focalisant sur le traitement des exigences ont été proposées. Or, le processus de formalisation du contenu des exigences n'y est pas détaillé, ni les impacts qu'a ce processus sur la formulation des exigences. De plus, les conséquences de la confrontation des deux ensembles d'exigences (émises respectivement par les AMS et les pilotes SdF) sur le contenu des exigences ainsi que sur les modèles, ne sont pas clairement mises en évidence, analysées, exploitées et capitalisées.

Deuxième partie

Mise en œuvre

4

Présentation générale de la démarche

Sommaire

4.1 Principales activités de la démarche adoptée

4.1.1 Rappels sur le cas d'étude : la fonction AD

Dans la section 2.3.3, nous avons vu que, selon les deux points de vue métier (AMS et pilotes SdF), le niveau de raffinement de la même fonction AD n'était pas le même (voir figures 2.12 et 2.13). La figure 4.1 résume l'allocation des exigences considérées, selon les deux perspectives métier. Nous y distinguons donc les *exigences considérées*, qui correspondent aux données brutes que nous avons eues à traiter, des *exigences analysées*, qui sont celles que nous avons retenues pour notre démarche.

Les AMS considèrent la fonction AD dans son ensemble, c'est-à-dire constituée des deux sous-fonctions : de Contrôle (CAD), d'une part, et de Supervision (Supervision AD), d'autre part. Selon cette perspective, le comportement de la fonction AD est spécifié globalement, *i.e.* sans que les exigences ne soient allouées à une de ces deux sous-fonctions ; et en phase nominale, sans prendre en compte de dysfonctionnements potentiels. Ces exigences, nommées Exigences Fonctionnelles de Comportement (EFC), sont au nombre de 175 et contenues dans le document intitulé System Technical Requirements dont nous avons considéré la version suivante : STR V1 (AMS Renault 2017b).

Par ailleurs, du point de vue des pilotes SdF, la Supervision AD est décomposée en 3 sous-fonctions partiellement redondantes, comme le représente la figure 4.1. Les exigences fonctionnelles de SdF que nous considérons sont comprises dans le document FSC (IAV 2016). On en dénombre 217 qui sont allouées aux 3 sous-fonctions de Supervision (Main_AD, Sub_AD, AD-3). Ce cas d'étude illustre ainsi en pratique les différences entre les deux points de vue métier évoquées dans les chapitres précédents. Il met notamment en exergue deux points importants :

1 *La différence de niveaux d'abstraction* : les exigences fonctionnelles de comportement retenues relèvent du niveau 2 tandis que les exigences fonctionnelles de SdF retenues sont de niveau 3 ;

2 *La différence des phases traitées* : les exigences fonctionnelles de comportement retenues se focalisent sur les phases de comportement nominal alors que les exigences fonctionnelles de SdF retenues ont trait aux phases dégradées (après l'occurrence d'un dysfonctionnement).

Après avoir rappelé les données essentielles du problème traité, nous pouvons désormais présenter, de façon globale, l'approche que nous avons proposée. Cette dernière sera appliquée à la fonction AD dans les chapitres 5 et 6 de ce mémoire.

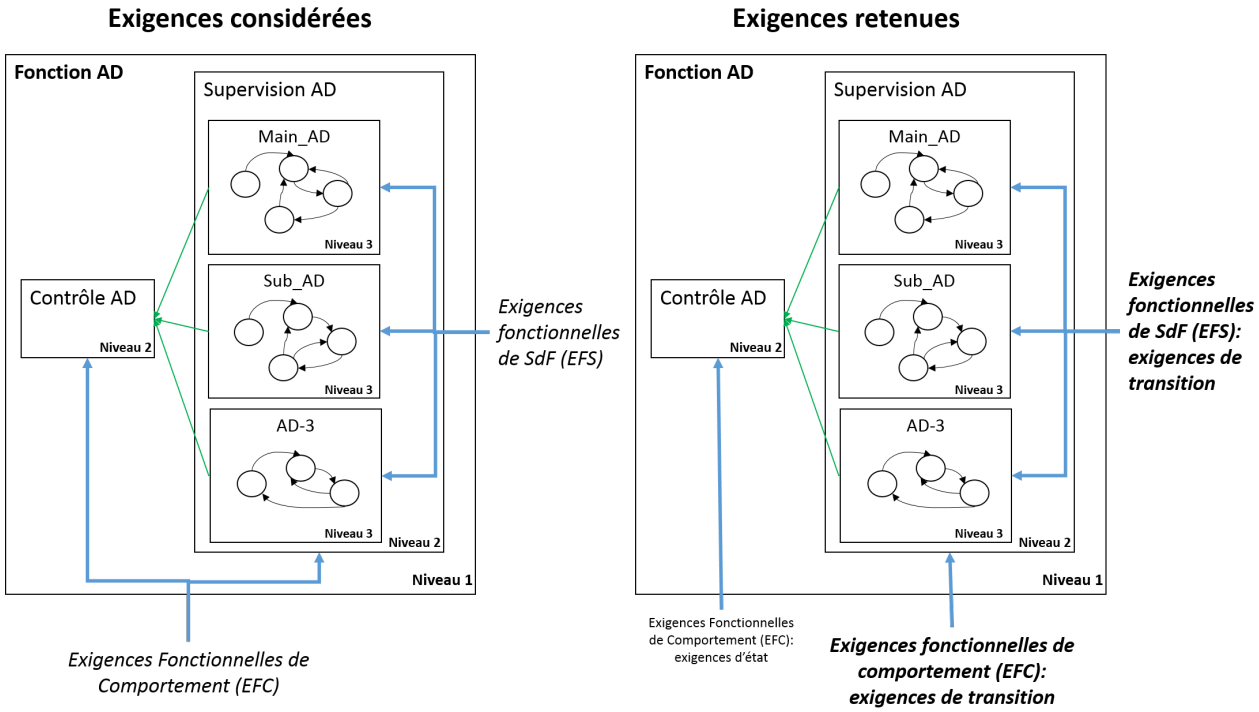


FIGURE 4.1 – Exigences considérées et analysées

4.1.2 Aperçu général de la démarche

La figure 4.2 donne une première représentation du déroulement global de la démarche proposée. Cette vue se focalise sur les données manipulées et leur évolution au cours de la démarche. Elle représente également le rôle des acteurs (AMS, pilotes SdF et « ingénieurs méthodes formelles »), le séquencement général des activités et les principales contributions revendiquées. En revanche, les activités, symbolisées par des flèches, ne sont volontairement pas détaillées ici. Concernant les activités sur fond vert, elles font intervenir des acteurs formés dans le domaine des méthodes formelles (appelés précédemment *ingénieurs méthodes formelles*). Notons que ces acteurs ne sont pas encore en place dans le contexte actuel chez Renault, mais l'entreprise vise à en définir précisément le rôle, les missions et attributions et à affecter les ressources nécessaires (voir paragraphe 4.2.4). Ces activités correspondent, en pratique, à celles qui ont été menées dans le cadre de la thèse. Cette démarche, dont l'application est pour le moment limitée à la fonction AD, a pour but d'être intégrée dans le nouvel ensemble de méthodes, outils et processus en cours de déploiement chez Renault.

4.1.2.1 Construction des MEF et MES (AF1, AS1)

Les activités **AF1** et **AS1** consistent à construire plusieurs ensembles : { *Modèle d'État (formel), Propriétés formelles, Exigences modifiées, Hypothèses* } du point de vue des AMS (activité **AF1**) et d'autres ensembles, constitués des mêmes éléments, du point de vue des pilotes SdF (**AS1**), le tout à partir des deux ensembles d'exigences (les

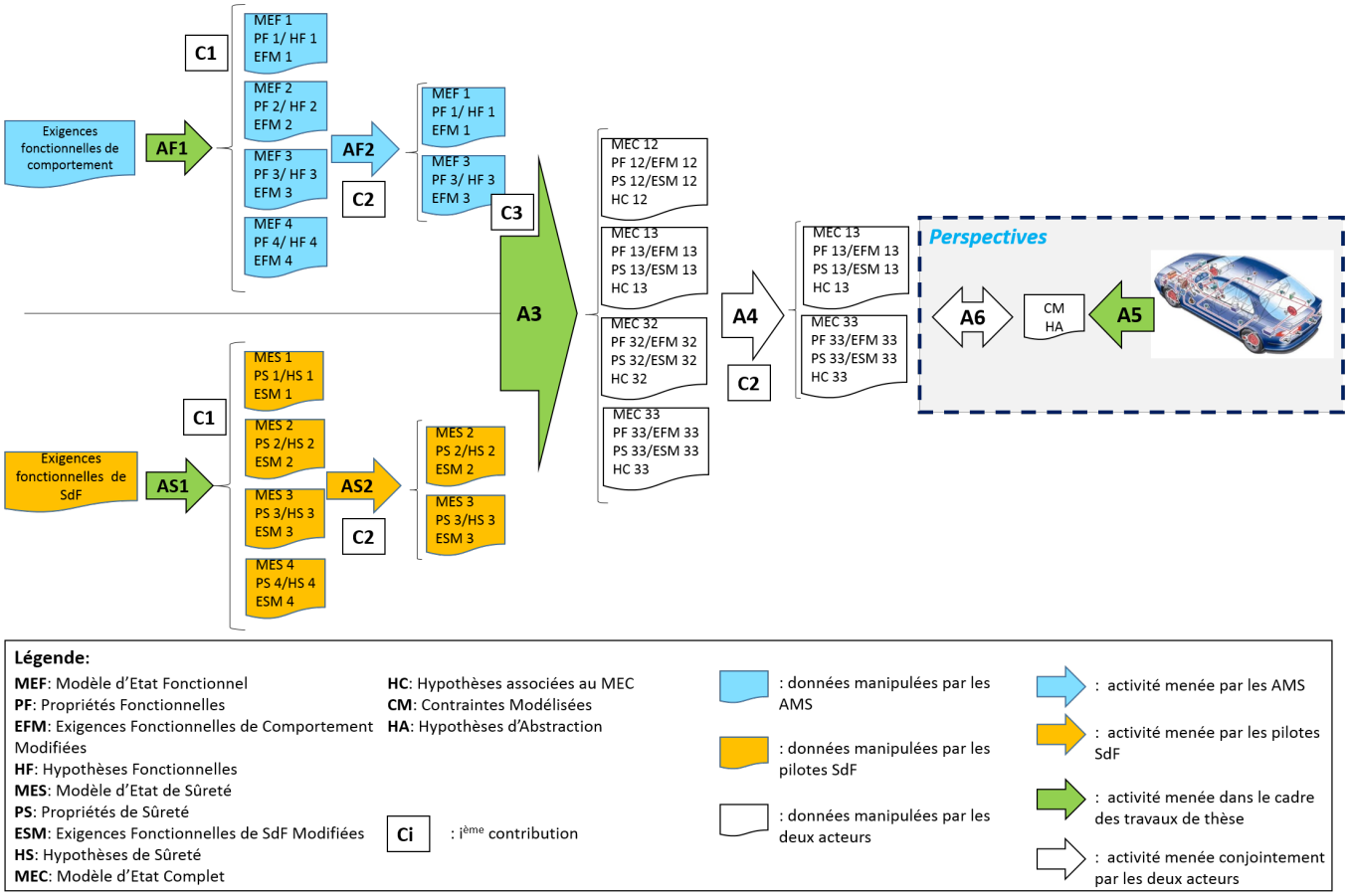


FIGURE 4.2 – Aperçu de la démarche proposée

exigences considérées sur la figure 4.1), exprimées en langage naturel. Chaque modèle obtenu, à partir d'un même ensemble d'exigences, correspond à une interprétation formelle possible de ces exigences. Ces interprétations sont tracées au moyen des hypothèses prises. Pour chaque ensemble, les propriétés formelles sont vérifiées par *model checking* sur le modèle. Par ailleurs, les hypothèses formulées pour formaliser les exigences peuvent modifier ces dernières de deux manières : soit le contenu d'une exigence est modifié (ajout, suppression ou correction d'un élément), soit le corpus d'exigences est modifié (ajout, suppression d'une ou plusieurs exigences). La mention « *exigences modifiées* » (fonctionnelles ou de sûreté) recouvre ces aspects. Ceci illustre la première contribution (**C1**) de ces travaux, qui consiste à mettre en évidence les conceptions alternatives, *i.e.* toutes les spécifications formelles possibles à partir d'un ensemble d'exigences en langage naturel. De plus, les exigences initiales, informelles, sont conservées et « coexistent » avec les modèles et propriétés formelles qui en sont déduits.

En revanche, il n'est pas possible de vérifier directement les exigences informelles initiales sur les modèles construits. Seules les propriétés formelles déduites des exigences modifiées (qui correspondent à une interprétation particulière des exigences initiales) sont effectivement vérifiées sur les modèles d'état. Ainsi, la démarche que nous pro-

posons ne vise pas à remplacer les énoncés initiaux par des propriétés formelles, mais bien à conserver ces exigences informelles, ainsi que les spécifications formelles auxquelles elles peuvent donner lieu. En outre, il n'est ni réaliste ni désirable de remplacer définitivement les exigences rédigées en langage naturel par des modèles, afin de ne pas contraindre (par l'usage de langages formels) l'expression des besoins initiaux. L'objectif est plutôt d'être en mesure d'établir un lien de traçabilité entre ces premières exigences informelles, les modèles de comportement formels des systèmes puis le comportement des systèmes eux-mêmes, dans le but final de garantir que le système respecte effectivement les exigences formulées en phase amont de conception. Les interprétations qui ne sont pas retenues sont également stockées, en prenant soin de mentionner la raison de ces choix.

4.1.2.2 Choix des modèles corrects (AF2, AS2)

Ensuite, lors de l'activité **AF2** (respectivement **AS2**), les AMS (respectivement les pilotes SdF) choisissent un ou plusieurs ensembles { *Modèle d'état (formel), Propriétés formelles, Exigences modifiées, Hypothèses* } qui correspondent effectivement à la spécification attendue. En effet, la phase de formalisation met en exergue des potentielles contradictions entre les exigences ou des lacunes dans les expressions qui n'apparaissent pas clairement dans les ensembles d'exigences textuelles. Ces manques sont, en revanche, plus aisément détectables lorsque les ensembles d'exigences prennent la forme graphique d'un modèle de comportement (le formalisme des automates à états finis a été adopté). Plus précisément la définition formelle (*i.e.* rigoureuse) de ce modèle facilitent l'identification des lacunes. La seconde contribution **C2** est la définition précise, en termes de planning et d'activités, du rôle de l'expertise métier dans le processus de formalisation des exigences.

Du point de vue applicatif, nous verrons dans le chapitre 5 que les activités de construction des modèles et d'élaboration des propriétés formelles (activités **AF1** et **AS1**) sont, dans les faits, menées plutôt de manière parallèle que successive avec les activités de choix des modèles corrects (**AF2** et **AS2**). Plus précisément, un certain nombre d'hypothèses sont prises pour construire les modèles et formaliser les propriétés et ces hypothèses sont régulièrement validées par les deux acteurs métier. En effet, un nombre très élevé de modèles d'état sont initialement possibles, à partir des 175 exigences fonctionnelles de comportement (contenues dans le STR V1 (AMS Renault 2017b)) et des 217 exigences fonctionnelles de SdF (se trouvant dans le document FSC (IAV 2016)) considérées. Nous quantifierons donc l'ensemble des formalisations possibles avec des hypothèses très générales. Nous illustrerons ensuite comment la considération d'hypothèses plus spécifiques réduit peu à peu le nombre de modèles. Afin de justifier le bien-fondé de ces hypothèses, celles-ci sont donc régulièrement soumises aux deux expertises métier pour validation. En définitive, nous obtenons plusieurs ensembles (six, pour le cas d'étude) { *Modèle d'état (formel), Propriétés formelles, Exigences modifiées, Hypothèses* } crédibles du point de vue des AMS et 1 ensemble (toujours pour l'exemple d'application) du même type à partir des exigences fonctionnelles de sûreté. Par ailleurs, toutes les propriétés formelles sont vérifiées sur les modèles d'état obtenus par *model checking*. Enfin, les résultats de la vérification formelle ont également conduit à enrichir les gardes de transition des modèles d'état, et, par suite, les exigences

analysées.

4.1.2.3 Mise en cohérence des points de vue (A3, A4)

Au terme des quatre activités **AF1, AF2** et **AS1, AS2**, menées en parallèle, chaque point de vue produit des ensembles { *Modèle d'état (formel), Propriétés formelles, Exigences modifiées, Hypothèses* } cohérents. Néanmoins, puisque les activités ont été réalisées en silos, rien ne garantit que les ensembles soient cohérents entre eux. Par exemple, les Modèles d'État de Sûreté ne respectent pas nécessairement les propriétés formelles fonctionnelles déduites des exigences fonctionnelles de comportement retenues (et inversement). Le rôle de l'activité **A3** est, en conséquence, de construire des Modèles d'État Complets, c'est-à-dire sur lesquels toutes les exigences (plus précisément les propriétés déduites de ces exigences) sont vérifiables, puis vérifiées. La confrontation entre modèles d'état fonctionnel et de sûreté se fait grâce à la comparaison du *produit synchronisé d'automates à états finis* décrivant le comportement des Supervisions AD locales (constituant le MES), du point de vue des pilotes SdF, avec le modèle d'état de la Supervision AD globale, selon la perspective des AMS. Ce produit est mis en œuvre au moyen de l'outil *Supremica*[1], qui offre effectivement une vue graphique du produit synchronisé, facilitant la comparaison des vues locale (pilotes SdF) et globale (AMS) dans notre contexte de travail. La troisième contribution **C3** est donc une méthode de mise en cohérence des points de vue des AMS et des pilotes SdF par la confrontation des modèles d'état traduisant les spécifications émises par ces deux métiers.

À la suite de cette activité **A3**, l'activité **A4**, consistant à choisir les Modèles d'État Complets satisfaisant à la fois aux attentes des AMS et à celle des pilotes SdF, est initiée. Il en résulte des Modèles d'État Complets conformes aux vues des deux disciplines d'ingénierie. Les activités **A3** et **A4** sont détaillées dans le chapitre 6. La confrontation des deux points de vue métier modifie également les ensembles { *Modèle d'État (formel), Propriétés formelles, Exigences modifiées, Hypothèses* } résultant des activités **AF2** et **AS2**. Nous verrons que la spécification d'états supplémentaires cause l'ajout de 642 à 750 exigences (selon les versions de modèle d'état complet) par rapport aux 392 (= 175 exigences fonctionnelles de comportement + 217 exigences fonctionnelles de SdF) exigences initiales, suite à la mise en cohérence des deux modèles des deux perspectives. Toutefois, ces modèles reflètent le comportement attendu, correspondant aux besoins et attentes initiaux, qui est réalisé par des fonctions abstraites. Or, il est possible, très tôt dans le cycle de conception, d'obtenir des informations sur les composants réalisant ces fonctions. La question de la faisabilité se pose donc : la spécification fonctionnelle peut-elle être effectivement implémentée par les composants ? En effet, leur utilisation amène des contraintes, dont certaines sont modélisables en phase amont de conception. Ces questions n'ont pas été traitées complètement au cours des travaux de thèse. Elles ont cependant fait l'objet d'analyses préliminaires (voir figure 4.8) qui constituent, nous le verrons en conclusion (paragraphe 7.2.2), des perspectives de recherche intéressantes, également pertinentes d'un point de vue opérationnel.

A cette démarche, s'ajoute une réflexion relative à son outillage (au moins partiel), et aux liens entre les logiciels utilisés pour les travaux de thèse et ceux déployés,

1. https ://supremica.org/

ou en cours de déploiement, à l'échelle de l'entreprise (notamment *Rational Doors*[2] et *MagicDraw*[3]). Enfin, ces travaux de thèse ont donné lieu à un stage, débuté en février 2018, portant sur la possibilité d'automatiser le processus de formalisation des exigences. À l'issue de ce stage, un algorithme générant des *squelettes de modèles d'état* (voir paragraphe 5.5.4) à partir d'exigences exprimées en langage naturel, du même type que celles considérées dans la thèse, a été développé, en se basant sur la démarche proposée (plus de détails à ce sujet sont donnés dans le paragraphe 7.2.2). Après avoir présenté globalement la démarche proposée, il convient de s'intéresser à son intégration dans les processus en vigueur chez Renault, dans une optique de réutilisabilité.

4.1.3 Intégration de la démarche

4.1.3.1 Activités Renault et cadre normatif

Les activités, menées par les AMS et les pilotes SdF, qui sont en lien avec les travaux de thèse sont représentées sur la figure 4.3. Cette figure montre comment ces activités (décrites dans le paragraphe 4.1.2) s'inscrivent dans les processus normatifs en vigueur. Il s'agit, pour les AMS, des processus de la norme ISO 15288, tandis que les pilotes SdF sont concernés par le standard ISO 26262. Dans les deux cas, les processus en question se trouvent dans la phase amont du cycle en V. La figure 4.3 conduit aux deux remarques suivantes :

1 L'activité **A_AMS1** consiste non seulement en un recueil des exigences des parties prenantes, mais aussi en une analyse de ces exigences visant à exclure des exigences non pertinentes, dès le début du projet. C'est pourquoi, cette activité tient à la fois du processus normatif de définition des exigences des parties prenantes que du processus d'analyse des exigences ;

2 L'activité **A_AMS2** est également une activité qui relève de deux processus normatifs : le processus d'analyse des exigences et celui de conception d'architectures. Effectivement, afin de concevoir les architectures, il est nécessaire de poursuivre l'analyse des exigences qui sont généralement modifiées (certaines sont raffinées, d'autres ajoutées ou supprimées par exemple) au fur et à mesure de l'élaboration des architectures.

4.1.3.2 Démarche proposée et Processus Renault existants

La question du positionnement des activités de la démarche proposée par rapport aux activités classiques chez Renault mérite d'être posée car l'ensemble des activités de la démarche sont *a priori* nouvelles (voir figures 4.3 et 4.2). L'intégration précise en termes de livrables, acteurs, jalons, ne fait pas partie du périmètre des travaux de thèse mais constitue une perspective intéressante. Cette intégration est rendue complexe du fait du changement de paradigme d'ingénierie en cours chez Renault (qui implique nécessairement des modifications dans le Système de Conception Renault, notamment au niveau des procédures). Toutefois, il est possible de situer les activités menées dans

2. https ://www.ibm.com/fr-fr/marketplace/requirements-management
3. https ://www.nomagic.com/products/magicdraw

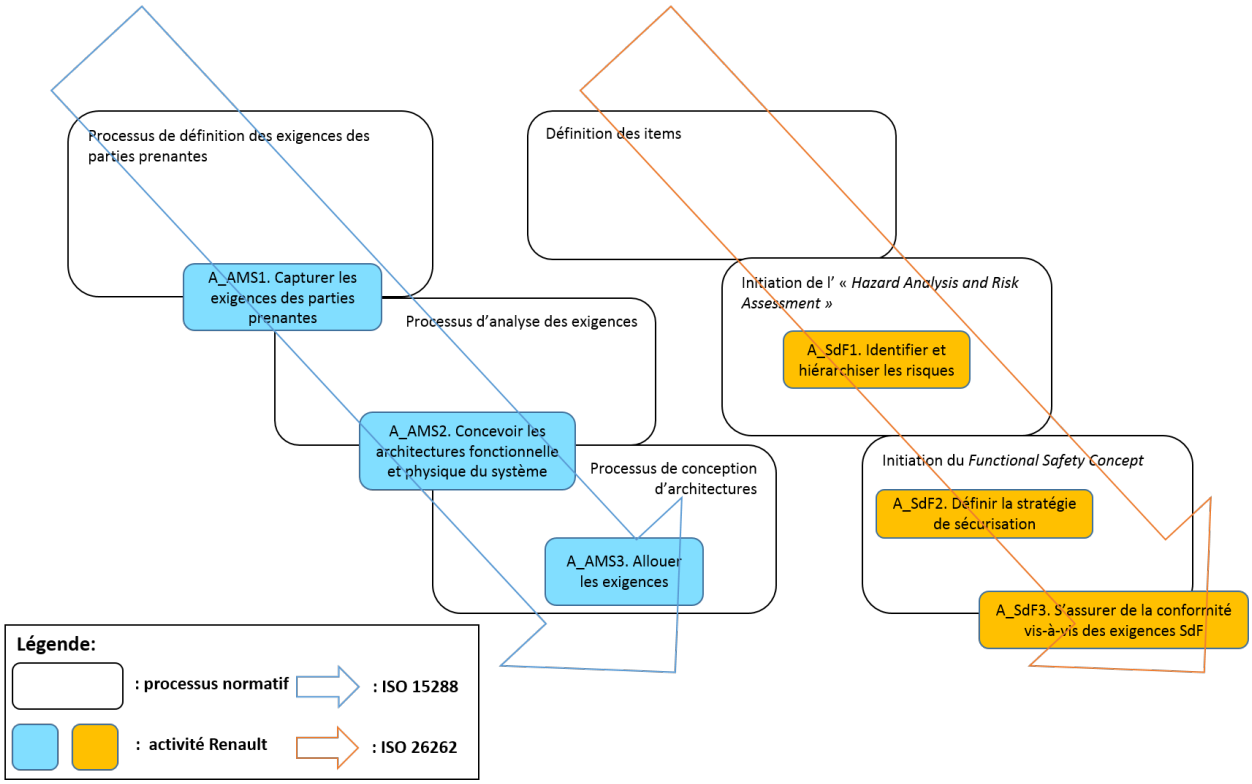


FIGURE 4.3 – Intégration des activités menées chez Renault dans les processus normatifs en vigueur

le cadre de la thèse, par rapport à celles qui sont déployées traditionnellement chez Renault, sans pour autant que les liens ne soient encore clairement et précisément établis.

Cette représentation est donnée sur la figure 4.4, sur laquelle les libellés des activités sont aussi explicités. Les nouvelles activités y sont mises en évidence par rapport aux activités classiques (chez Renault). Ces activités sont effectuées soit par l'un des deux acteurs, soit conjointement par les deux, soit encore ont été réalisées dans le cadre de la thèse. Nous reviendrons, dans le paragraphe 4.2.4, sur la répartition des rôles (actuelle et à court et moyen termes). Enfin, il convient de remarquer que la dénomination « nouvelles activités » correspond plutôt à une façon différente (méthodes, outils, jalons) de réaliser les activités traditionnelles des deux acteurs. C'est pour cette raison que ces activités sont placées en parallèle ou en chevauchement des activités classiques.

Remarque 4.1
Notons que les activités ***A5*** *et* ***A6*** *ne seront pas détaillées dans ce mémoire. Elles seront simplement présentées dans le paragraphe 4.3.2 car elles constituent des perspectives de travail, comme le montre la figure 4.8.* ♢

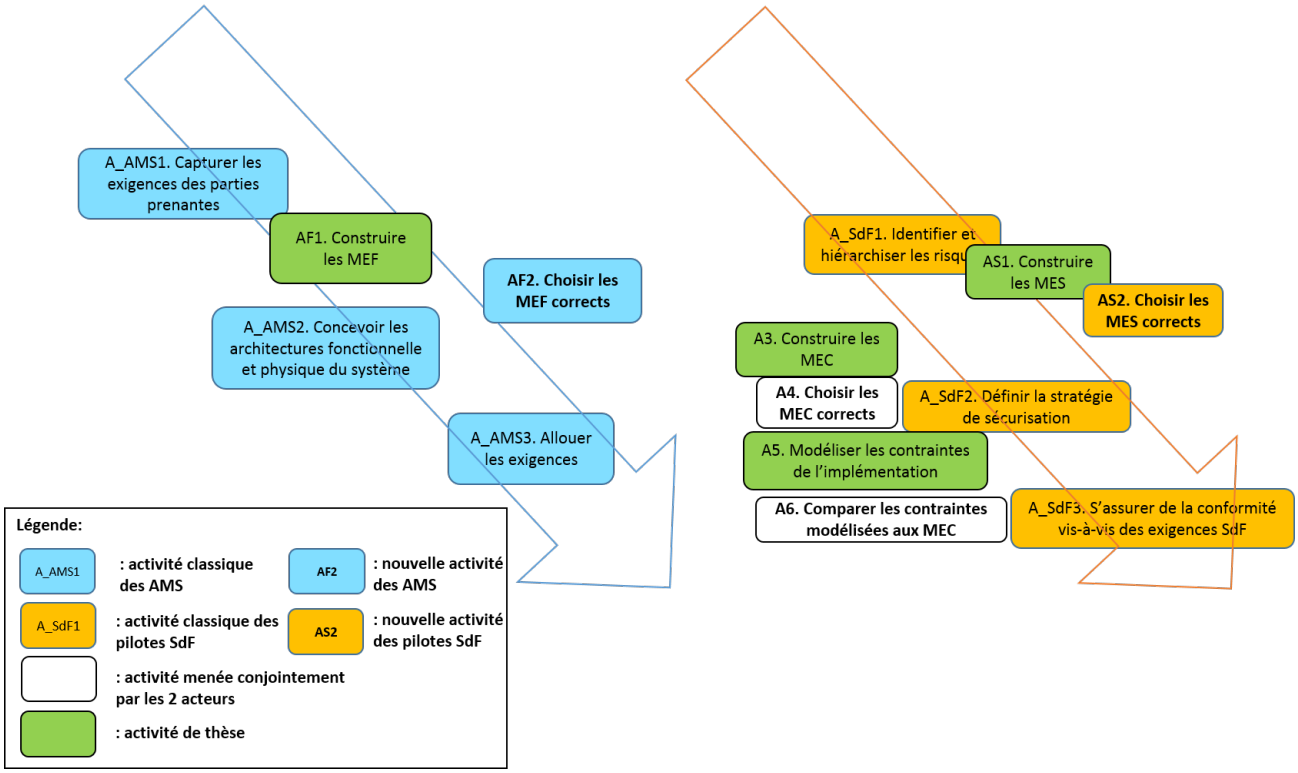


FIGURE 4.4 – Intégration des activités de la démarche dans les processus Renault actuels

4.1.3.3 Conclusions

Les considérations qui précèdent donnent une idée de l'intégration de la démarche proposée, à la fois dans le cycle de conception propre à Renault, mais aussi dans les cycles standards, préconisés par les normes en vigueur (ISO 15288 et ISO 26262). Ceci donne le cadre global et montre que les travaux menés s'inscrivent bien dans un contexte plus vaste. La démarche contribue également à la prise en compte de recommandations normatives dans les processus Renault existants. Par exemple, l'emploi des méthodes formelles, encouragé par la norme ISO 26262, est ici mis en pratique et adapté aux pratiques existantes. En effet les exigences analysées sont vérifiées formellement, par *model checking* sur les modèles d'état construits.

Par ailleurs, une des principales contributions de ces travaux est le détail de l'interaction entre les experts métier (AMS et pilotes SdF) et le processus de formalisation des exigences et de construction des modèles d'état. Il s'agit du séquencement des activités de la démarche proposée, représenté sur la figure 4.4, et détaillé dans les chapitres 5 et 6. En revanche, l'intégration des activités proposées dans les processus normatifs et dans le contexte global de l'entreprise hôte sort du cadre des travaux de thèse et n'est, en conséquence, pas étudiée en profondeur. La dernière partie de ce chapitre est consacrée à la clarification du positionnement des travaux de thèse au vu de l'état de l'art, présenté dans le chapitre 3. À la lumière de la présentation générale de l'approche proposée, il est désormais possible de dégager les principaux concepts que nous avons développés afin de les comparer avec les propositions de la littérature.

4.2 Concepts défendus

4.2.1 Positionnement dans le cycle de conception

Le premier point abordé porte sur le type d'exigences que nous avons vérifié et le positionnement de la démarche dans le cycle de conception Renault. Ce positionnement est comparé à la fois aux pratiques actuelles chez Renault, et également aux travaux de la littérature, relatifs à la formalisation et vérification d'exigences de comportement. Nous reprenons ici les éléments présentés dans le paragraphe 3.2.3. Rappelons que les exigences que nous avons retenues pour l'analyse sont des exigences spécifiant un comportement, allouées à des fonctions de bas niveau (niveaux 2 et 3 pour le cas d'étude, voir paragraphe 2.3.3) et informelles, *i.e.* rédigées en langage naturel. Le but initial, représenté sur la figure 3.8, était de lier le contenu des exigences de comportement informelles non quantifiées, celui des exigences de comportement informelles quantifiées et les modèles de comportement des composants. Il existe déjà des relations, définies notamment par le langage SyML (allocation, association, dérivation, raffinement), qui structurent l'agencement des objets manipulés dans une approche MBSE (exigences, fonctions, architectures, composants). Cependant, tant que le contenu des exigences demeure informel, il n'est pas possible de lier formellement les exigences de comportement à des modèles dont le formalisme est, lui, défini. Nous pensons donc que seule la formalisation du contenu des exigences rend pleinement efficace et utile ces relations déterminées entre les objets manipulés dans un contexte MBSE.

Comme le montre la figure 4.5, la démarche proposée vise à faire passer les exigences analysées du domaine subjectif au domaine objectif (décrit dans le paragraphe 3.2.3), en les rendant pour ce faire, précises, complètes et exemptes d'ambigüités, sans pour autant changer de niveau d'abstraction. En outre, des modèles d'état formels, sur lesquels ces exigences sont vérifiées, ont été construits. Nous avons donc défini des liens entre le contenu des exigences de comportement informelles retenues (de niveaux 2 et 3), celui des exigences de comportement formelles déduites (également de niveaux 2 et 3), et les modèles d'état construits. Pour remplir totalement l'objectif initial et garantir la traçabilité complète des exigences, des relations entre les modèles d'état que nous avons obtenus et les modèles de comportement des composants doivent être créées (flèches de couleur noire et en pointillé sur la figure 4.5). Ce travail sort du périmètre de notre étude, mais on peut tout de même signaler que des approches faisant le lien entre les diagrammes comportementaux d'un environnement SysML et les modèles de simulation *Simulink* ont été proposées (diagrammes de séquences et paramétriques (Cressent, Idasiak et Kratz 2011) et diagrammes d'état (Kawahara et al. 2009)).

4.2.2 Mise en évidence de solutions alternatives

Les activités **AF1** et **AS1** de la démarche (voir figure 4.2) consistent à générer l'espace de spécifications formelles, à partir de données informelles. L'objectif global est de contribuer à l'intégration de l'ingénierie des exigences dans une approche MBSE. Les travaux de thèse de Benoît Lebeaupin (Lebeaupin 2017), présentés dans le paragraphe 3.2.2, partagent cette préoccupation. De ce fait, le principe du processus de formalisation que nous adoptons est proche de celui de ces travaux (voir figure 3.7).

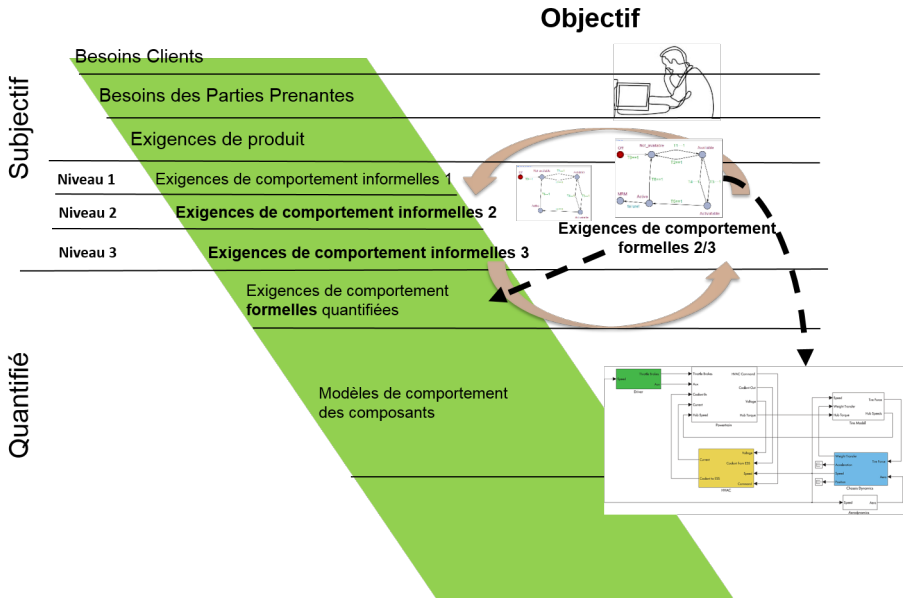


FIGURE 4.5 – Processus de vérification proposé

Néanmoins, comme l'illustre la figure 4.6, il s'en distingue par plusieurs points. En effet, l'objectif que nous poursuivons n'est pas nécessairement la *suppression* totale des ambigüités, mais plutôt la mise en évidence de leur *existence*, ainsi que les différents modèles et propriétés formels possibles (et surtout crédibles pour l'expertise) que ces ambigüités génèrent.

Ainsi, l'espace de sens, qui demeure une notion peu claire et très subjective dans (LEBEAUPIN 2017), est identifié et défini au moyen des hypothèses formulées. De cette manière, une exigence est à l'origine d'un ensemble de sens possibles. Chacun de ces sens est décrit de manière informelle dans les hypothèses et également associé à des éléments formels que sont les propriétés et les modèles. L'AMS ou le pilote SdF, suivant la provenance des exigences, est alors à même de choisir le sens « correct », c'est-à-dire correspondant au comportement attendu qu'il avait au préalable décrit dans les exigences informelles. Étant donné que plusieurs formalisations (ou sens) sont acceptables par l'expert, ceci met en évidence un ensemble de solutions alternatives possibles. Ces solutions étaient en réalité « contenues » dans les exigences initiales, sans que ceci ne soit pour autant explicite.

4.2.3 Convergence par le modèle

4.2.3.1 Co-construction des modèles et des propriétés formelles

Le cycle de conception défini chez Renault (voir figure 4.3), préconise des activités séquentielles. Ainsi les architectures doivent être théoriquement réalisées avant d'y allouer les exigences. Mais, comme ceci est souligné dans (WU et KELLY 2006 ; LEBEAUPIN 2017), le co-développement des exigences et des modèles est une réalité pratique. En effet, construire des modèles sur lesquels les exigences soient vérifiables fait apparaître

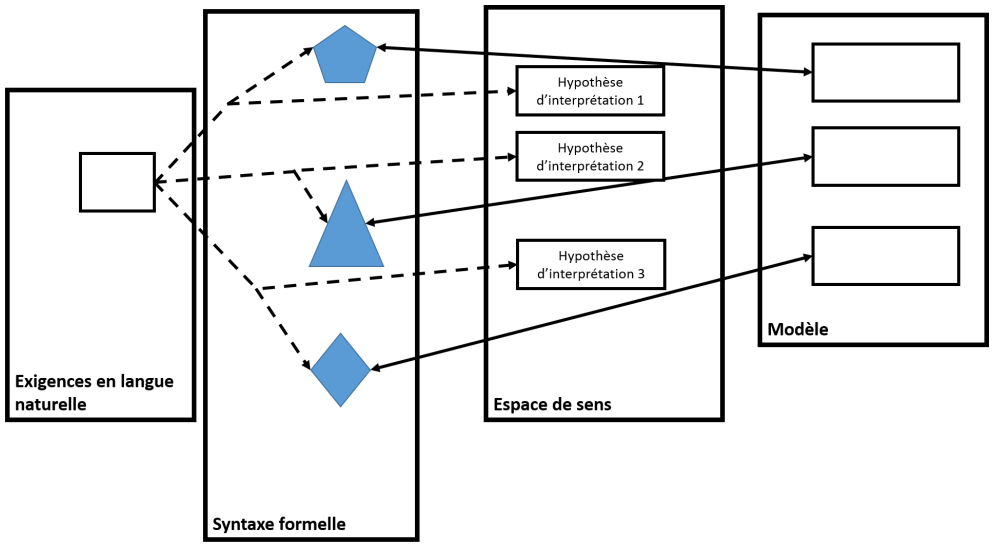


FIGURE 4.6 – Principe de la démarche de formalisation proposée

les lacunes que comportent ces exigences initialement : on ne peut vérifier une exigence sur un modèle que s'il est fait explicitement référence à des éléments du modèle dans le contenu de l'exigence. Par ailleurs, la construction d'un modèle (d'état dans notre cas) ouvre la voie à des questionnements plus larges. Par exemple, il est beaucoup plus naturel de se poser la question de la nécessité de l'existence (ou de l'absence) d'une transition entre deux états sur un modèle graphique plutôt qu'en considérant un ensemble d'exigences textuelles.

À titre illustratif, la figure 4.7 montre une première version de modèle d'état (appelée, nous le verrons, *squelette de modèle d'état*) spécifiant le comportement de la fonction *Supervision AD* que nous avons construit à partir d'exigences fonctionnelles de comportement retenues (allouées à la fonction Supervision AD, voir figure 4.1). Ce modèle, les états qu'il comporte, la signification de la plausibilité, des gardes et de l'état initial sont détaillés dans le paragraphe 5.5.4. Un des premiers résultats de la revue des modèles avec les AMS a été d'exclure cette possibilité de modèle étant donné que l'état *Not_ available* est bloquant. Ce qui est donc un résultat immédiat et évident avec un modèle ne l'est pas nécessairement avec une liste d'exigences textuelles à analyser.

4.2.3.2 Un formalisme partagé pour des objectifs de modélisation communs

Le point de départ des activités **AF1** et **AS1** (voir figure 4.2) est, pour les AMS, l'ensemble des exigences pertinentes issues du STR V1 (AMS RENAULT 2017b) et, pour les pilotes SdF, celles du FSC (IAV 2016). Comme l'illustre la figure 2.9, le STR V1 est disponible avant le FSC, car ce dernier est en partie construit à partir du STR V1. C'est pourquoi les deux activités, représentées comme parallèles sur la figure 4.2, débutent, dans les faits, de manière désynchronisée : l'activité **AF1** est initiée avant l'activité **AS1** (comme ceci apparaît sur la figure 4.4). Or, comme nous le constatons également sur la figure 2.9, l'élaboration du FSC peut causer des modifications sur les exigences

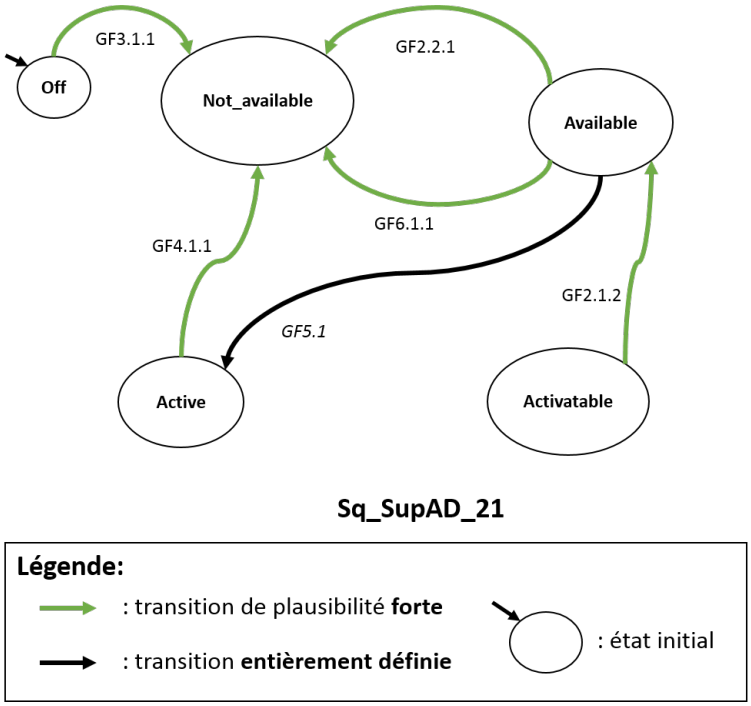


FIGURE 4.7 – Une version de modèle d'état de la Supervision AD (point de vue des AMS)

contenues dans le STR. Ces modifications concernent potentiellement les exigences retenues pour construire les Modèles d'État Fonctionnels (MEF). L'imbrication de ces deux activités, ainsi que la différence de culture et d'objectifs entre les AMS et les pilotes SdF, vont nécessairement avoir des impacts sur le vocabulaire et la sémantique utilisés. Ce phénomène explique notamment les différences de niveaux d'abstraction et de phases de comportement spécifiées relevées dans notre cas d'étude (paragraphe 4.1.1).

Cette importante question est actuellement résolue par la tenue régulière de revues communes. Cependant, il n'est pas aisé de fixer la fréquence de ces revues pour que ces dernières soient efficaces. En effet, si elles sont trop régulières, elles ne présentent pas de grande utilité car peu d'éléments auront évolués. De plus, la disponibilité des deux acteurs est réduite car ils travaillent généralement sur plusieurs projets différents. Si, au contraire, ces revues sont trop rares, les écarts pourront être devenus importants, en conséquence difficiles à identifier et à traiter dans les délais impartis. Nous pensons que l'adoption d'un même formalisme, par les AMS et les pilotes SdF, dès lors que les deux acteurs partagent un objectif de modélisation commun, facilite le traitement des incohérences de spécifications entre les deux points de vue. Dans un premier temps, la vue graphique qu'offrent les modèles d'état manipulés simplifie l'identification de ces incohérences par rapport à la comparaison de deux corpus d'exigences textuelles. Puis, dans un second temps, il est possible de procéder aux modifications requises directement sur les modèles. Enfin, les relations établies entre les modèles et les exigences textuelles permettent de répercuter ces modifications sur les exigences initiales.

Ainsi, la manipulation d'un formalisme commun contribue à faire converger les points de vue des AMS et des pilotes SdF. Toutefois, si ces modèles sont effectivement utiles

pour consolider l'expression des exigences analysées (exigences de comportement de bas niveau, non quantifiées), ce n'est plus le cas pour des exigences de comportement allouées à des composants et, généralement, quantifiées, qui nécessitent des modèles de comportement plus détaillés du système réalisé ainsi que des techniques plus adaptées et plus pointues (comme soutenu dans (Mauborgne 2016) par ailleurs). Par exemple, pour vérifier une exigence typique de SdF portant sur la fiabilité de composants, le spécialiste aura recours à des modèles plus élaborés tels que les réseaux de Petri, les chaînes de Markov ou pourra procéder à des simulations de Monte Carlo.

Enfin, le dernier point à évoquer est l'entame d'une réflexion sur l'attribution des activités menées durant la thèse. Nous avons, dans le paragraphe 4.1.2, mentionné le rôle d'un « ingénieur méthodes formelles », pour prendre en charge les activités menées durant la thèse. Cependant, la définition d'un nouveau poste n'est pas la seule voie en vue de rendre pérenne la démarche proposée dans le contexte de l'entreprise hôte.

4.2.4 Une attribution « souple » des activités

Les activités effectuées durant la thèse ne sont pas encore attribuées dans les procédures internes chez Renault. Afin de rendre nos travaux réutilisables, il est nécessaire de s'interroger sur les ressources les plus appropriées pour réaliser ces activités. Pour ce faire, deux solutions sont envisageables :

1 *Désigner un acteur supplémentaire* : c'est une voie prônée par certains travaux de recherche tels que la thèse d'Anthony Legendre (Legendre 2017) dans laquelle un *« responsable de synchronisation »* est requis pour chapeauter les activités de mise en cohérence des points de vue (l'activité **A3** dans la démarche que nous proposons, voir figure 4.2) ; ou encore, dans les travaux d'Ofaina Taofifenua (Taofifenua 2012), où un *« ingénieur ontologie »* est désigné. Ce dernier est en charge de gérer l'ontologie sur laquelle repose le processus de conception proposé (voir section 3.2.4) ;

2 *Affecter les nouvelles tâches à des acteurs existants* : parmi les travaux de l'état de l'art dressé préalablement, c'est dans le processus d'ISSdF, défini dans les travaux de Pierre Mauborgne (Mauborgne 2016), que cette solution est mise en œuvre. Ce sont essentiellement les AMS qui ont à réaliser des activités supplémentaires en initiant, par exemple, des analyses de risques (APR, AMDEC, AdD) habituellement attribuées entièrement aux spécialistes SdF. Dans cette approche, ces derniers se chargent essentiellement des analyses de SdF plus pointues.

Dans le cadre de ces travaux, la seconde voie semble préférable. Les activités de thèse, identifiées par un fond de couleur verte sur la figure 4.4, sont essentiellement focalisées sur la formalisation des exigences et la construction de modèles formels. Néanmoins, ces activités, bien qu'elles fassent évidemment appel à des connaissances relatives aux méthodes formelles, sont aussi fortement liées aux deux métiers concernés (AMS et pilote SdF). Ainsi, pour les exigences qui ont été traitées durant les travaux de thèse, les connaissances métier doivent être plus approfondies que celles des méthodes formelles.

Enfin, d'un point de vue purement opérationnel et organisationnel, il est plus rapide et aisé, à court terme, d'attribuer de nouvelles tâches à des acteurs en place que d'en recruter de nouveaux. Ceci n'est pas le cas en revanche pour une vision à plus long

terme, mais qui sort du périmètre de ces travaux. Ainsi, l'on peut envisager une phase transitoire durant laquelle les AMS et les pilotes SdF s'approprieraient les activités menées durant la thèse. Ceci suppose naturellement une communication sur les méthodes utilisées ainsi qu'un outillage approprié. Par la suite, si cette pratique donne les résultats escomptés, des nouveaux acteurs, cette fois experts en méthodes formelles, pourraient être engagés pour procéder à des vérifications plus poussées.

4.3 Synthèse

4.3.1 Bilan

Le contenu de ce chapitre a présenté les activités principales de la démarche proposée, clarifié son cadre global, et exposé les concepts défendus dans ce mémoire. En premier lieu, la démarche proposée et les principales activités menées ont été décrites. Nous nous sommes essentiellement concentrés sur les données manipulées, le rôle des différents acteurs et les contributions majeures, à savoir :

(a) **C1** : la mise en évidence de différentes solutions formelles crédibles à partir d'un même ensemble d'exigences informelles ;

(b) **C2** : la définition précise (activités, planning) du rôle de l'expertise dans le processus de formalisation des exigences et construction de modèles d'état ;

(c) **C3** : la convergence des points de vue des AMS et des pilotes SdF par l'usage d'un formalisme commun.

La portée des travaux a ensuite été délimitée. Les activités menées par les deux acteurs (AMS et pilote SdF) ainsi que celles réalisées dans le cadre des travaux de thèse se situent en phase amont des cycles de conception préconisés par le cadre normatif en vigueur et par l'entreprise hôte. En conséquence, les vérifications des propriétés formelles, issues des exigences analysées (voir paragraphe 4.1.2) qui ont été effectuées lors des travaux de thèse ont des impacts sur les exigences fonctionnelles rédigées en phase amont de conception. D'autre part, la démarche proposée contribue à la mise en pratique de méthodes formelles, ce qui est notamment une recommandation de la norme ISO 26262.

Les principaux concepts défendus dans ces travaux de thèse, liés aux contributions, ont enfin été identifiés et exposés. Tout d'abord, le *positionnement des travaux dans le cycle de conception*, représenté sur la figure 4.5, semble assez original par rapport à l'état de l'art. Effectivement, les activités de formalisation des exigences et de construction des modèles proposées visent à rendre objectives les spécifications traitées, c'est-à-dire précises, complètes et non ambigües, mais en conservant le même niveau d'abstraction que les exigences analysées. Les activités de formalisation des exigences et de construction des modèles sont menées de manière parallèle. Ce principe de *co-construction des modèles et des propriétés formelles*, déjà défendu par (Wu et Kelly 2006 ; Lebeaupin 2017), a pour double avantage de consolider la formulation des exigences tout en affinant les modèles. Au cours de ces deux activités, des *conceptions alternatives* sont mises en évidence. Le traitement des ambigüités contenues dans les exigences exprimées en langage naturel que nous proposons vise à identifier ces lacunes, puis les exploiter

en générant les spécifications formelles qu'elles suggèrent. Les représentations formelles graphiques (automates à état) ainsi construites mettent en évidence un ensemble de possibilités de spécifications formelles dont l'expert métier (AMS ou pilote SdF) n'a pas nécessairement conscience en rédigeant des exigences informelles. Ainsi, parmi ces possibilités, l'expert choisit celles qu'il juge correctes : certaines correspondent à son idée initiale, mais d'autres sont « nouvelles », dans le sens où elles n'avaient pas été considérées comme envisageables initialement.

Enfin, notons que, de manière générale, la démarche proposée œuvre à l'*introduction progressive de l'Ingénierie basée sur les modèles*. Il s'agit dans un premier temps de faire cohabiter certaines pratiques de l'ingénierie basée sur les documents avec celles de l'ingénierie basée sur les modèles. Dans cette optique, nous défendons les idées suivantes :

— *Un processus de formalisation progressif* qui fait « co-exister » les exigences exprimées en langage naturel avec les spécifications formelles (propriétés formelles et modèles) ;

— *Une traçabilité* entre les objets formels (propriétés, modèles) et informelles (exigences) assurée par un ensemble d'hypothèses. Les choix d'interprétations, ainsi que les raisons qui en sont à l'origine, doivent notamment y apparaître ;

— *Une attribution des rôles « souple »*. Ceci signifie que nous envisageons que les activités menées dans le cadre de la thèse peuvent être, dans un premier temps, effectuées par les deux acteurs concernés, à savoir les AMS et les pilotes SdF. Naturellement, ces activités ne peuvent être prises en charge par les métiers que s'ils sont convaincus de l'utilité et des apports de la démarche. Des actions de sensibilisation et de formation (aux méthodes et outils) doivent donc être entreprises. Cette perspective est réaliste car elle s'inscrit dans le cadre global de la mise en place d'une approche MBSE outillée (*MagicDraw*, *Rational Doors*). Enfin, toujours dans le contexte de ce chantier global, *i.e.* à l'échelle de l'entreprise, le recrutement de personnels spécialisés dans les méthodes formelles est possible. Les voies ouvertes par des initiatives telles que cette thèse (mais aussi, chez Renault, d'autres travaux (Taofifenua 2012 ; Dumitrescu et al. 2013 ; Góngora, Gaudré et Tucci-Piergiovanni 2013)) posent les bases sur lesquelles de tels experts pourront se reposer.

4.3.2 Suite de la thèse

Jusqu'à présent, nous nous sommes focalisés sur les aspects conceptuels des travaux réalisés. Les activités principales de la démarche ont été présentées de manière générale, afin d'exhiber les principales idées défendues, puis les comparer avec les concepts proposés dans la littérature.

Dans la suite du mémoire, nous nous proposons de mettre en pratique, dans le détail, la démarche proposée, les techniques et outils utilisés, en illustrant chacune des activités menées au moyen du cas d'étude portant sur la Supervision AD. Nous rappelons, sur la figure 4.8, les principales activités suivies. Cette figure a pour but de donner le fil conducteur de la suite de ce mémoire. Le chapitre 5 sera donc consacré aux activités de formalisation des exigences et de construction de modèles d'état, tant du point de

vue des AMS que de celui des pilotes SdF. Nous verrons notamment dans ce chapitre comment ces deux activités se déroulent précisément en étudiant chacune des étapes (*i.e.* sous-activités) mises en œuvre, les données produites et les hypothèses formulées. Nous analyserons également le séquencement de ces étapes de travail. Enfin, le rôle de chacun des acteurs dans le déploiement de ces activités sera également étudié. L'accent sera mis sur l'implication nécessaire des deux acteurs métier (AMS et pilotes SdF) dans les activités de formalisation des exigences et de construction des modèles d'état menées durant la thèse. Ces activités aboutissent à des modèles d'état sur lesquels les exigences formalisées sont vérifiables, et vérifiées grâce à deux *model checkers* : UPPAAL, NuSMV. Néanmoins, la cohérence entre les deux points de vue n'est pas assurée à ce stade.

C'est la raison pour laquelle le chapitre 6 explicite la convergence des deux perspectives, fondée sur la confrontation des modèles obtenus à l'issue des activités précédentes. Plus précisément, nous nous sommes appuyés sur l'opération formelle de *composition parallèle d'automates à état fini* (mis en œuvre dans l'outil Supremica) en vue de comparer les spécifications locales (émanant des pilotes SdF) avec le point de vue global (des AMS). Nous verrons que cette activité résulte, de nouveau, en l'enrichissement à la fois des modèles d'état fonctionnels et des modèles d'état de sûreté, donc également en la modification des exigences initiales sélectionnées.

Sur la figure 4.8, nous avons également représenté comment les perspectives relatives à la prise en compte de contraintes de l'implémentation (voir paragraphe 4.1.2) pourraient être initiées. Il s'agirait, à partir d'une seconde version du *System Design Document* (SDD V2), dans laquelle est restituée l'architecture physique cible, de modéliser les contraintes induites par les éléments de cette architecture (activité **A5**). Ces contraintes seraient ensuite introduites dans les Modèles d'État Complets résultants de l'activité **A4**. Enfin, il serait vérifié, lors de l'activité **A6**, que ces contraintes ne remettent pas en cause les exigences exprimées au point de vue fonctionnel. Ces réflexions n'ont pas pu être abouties et ne seront donc pas détaillées dans ce mémoire. Elles correspondent toutefois à des pistes de travail envisageables.

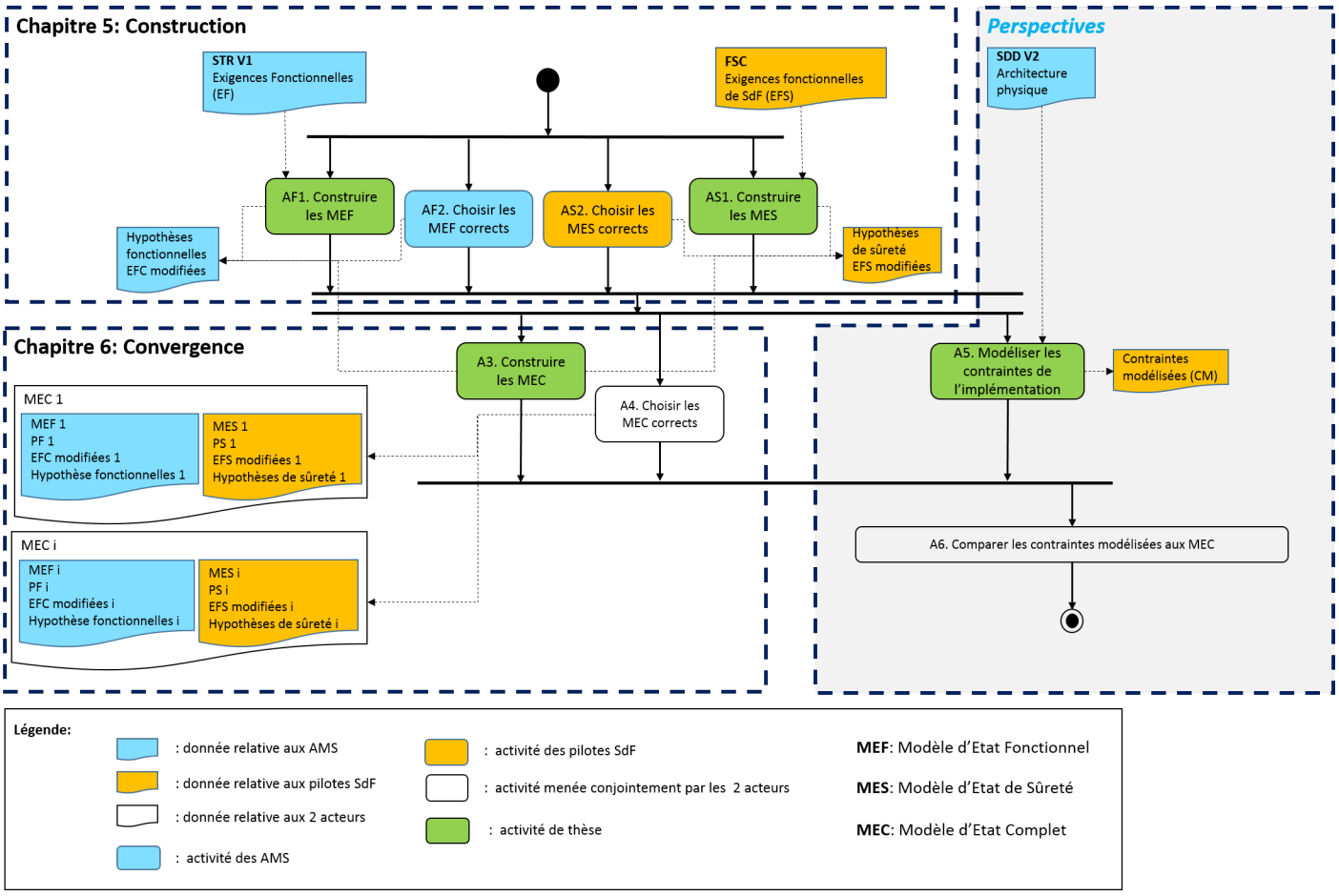


FIGURE 4.8 – Contenu des chapitres 5 et 6

5

Formalisation des exigences et construction des modèles d'état

Sommaire

5.1 Introduction : fil conducteur

Avant de présenter dans le détail les différentes activités de la démarche proposée, nous en rappelons le double objectif global : la construction de modèles d'état, plus précisément d'automates à états finis, et l'élaboration de propriétés formelles, vérifiables et vérifiées sur ces modèles. Nous présentons les principales étapes de transformation des données qui aboutissent à ces résultats sur la figure 5.1 et introduisons par la même occasion le vocabulaire employé.

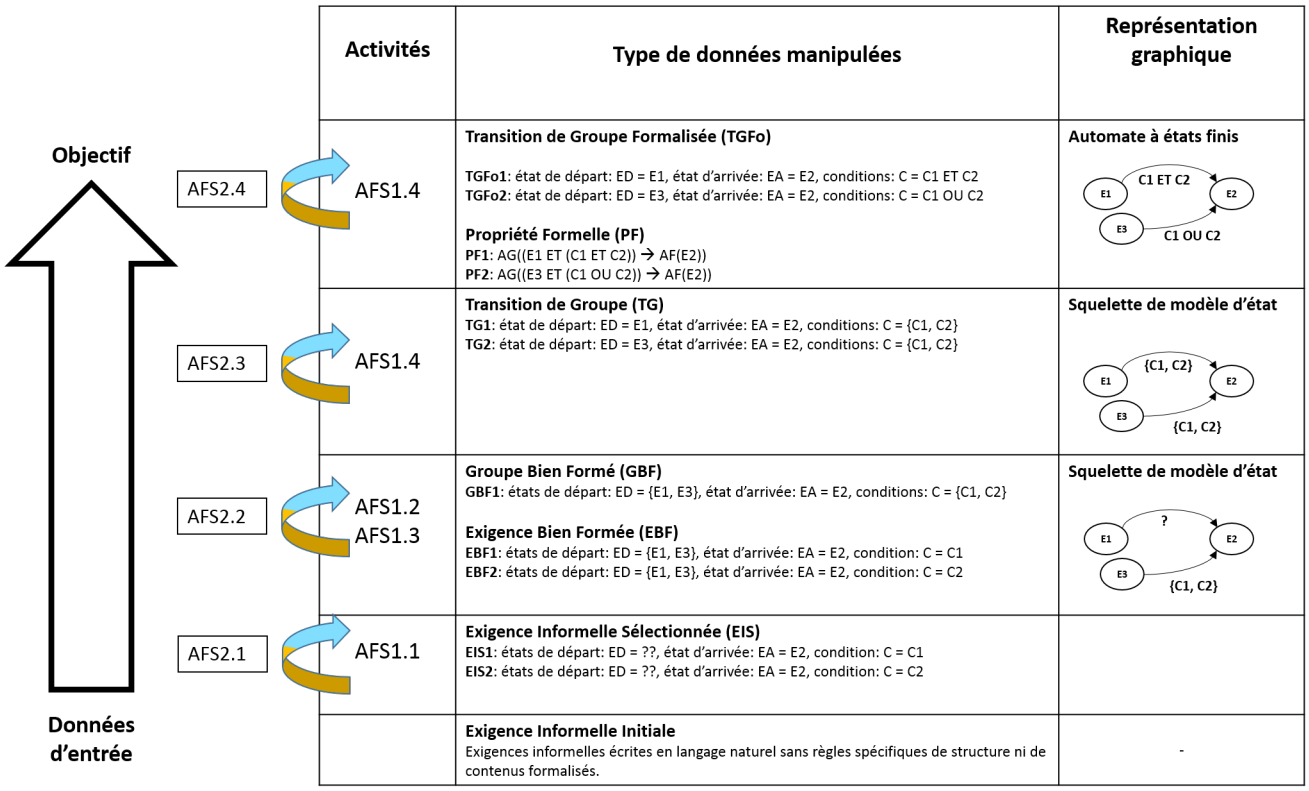


FIGURE 5.1 – Fil conducteur des activités présentées dans le chapitre 5

Le cheminement suivi est représenté sur la figure 5.1 et décrit ci-dessous de manière très succincte. Sur cette figure, une brève description des types de données manipulées est fournie, ainsi qu'une représentation graphique simplifiée, lorsque ceci est pertinent. L'objectif est de donner l'évolution de la *forme* des deux principales données : les exigences et les modèles d'état, tout au long du processus de formalisation. Par ailleurs, les activités à l'issue desquelles ces données sont produites sont également identifiées. Ces activités sont des sous-activités de l'activité de construction des modèles d'état, décrites dans le paragraphe suivant (paragraphe 5.2, dans lequel la signification du sigle **AFS1** est donnée). Nous remarquons aussi sur la figure 5.1, que des sous-activités de l'activité **AFS2** (également défini dans le paragraphe suivant) sont représentées : il s'agit de boucles de validation effectuées **en silos**, *i.e.* séparément, par les deux experts métier (AMS et pilotes SdF). Nous n'entrons volontairement pas dans le détail des définitions des notions évoquées. Celui-ci sera effectivement fourni dans les paragraphes

qui suivent.

Dans un premier temps, il s'agit donc de sélectionner, parmi les ***exigences informelles initiales***, celles qui seront traitées (***exigences informelles sélectionnées***). Les exigences retenues sont celles qui déterminent un changement d'état. Certaines exigences sélectionnées doivent être ensuite complétées en ajoutant les informations d'état manquantes, qui permettront de les vérifier formellement. Des états (de départ ou d'arrivée) sont par exemple proposés, pour obtenir, après modification, des ***exigences bien formées***, c'est-à-dire des exigences comprenant toutes les informations d'état requises. Dans le cas présenté sur la figure 5.1, les états de départ (ED) **E1** et **E3** sont ajoutés aux deux exigences sélectionnées **EIS1** et **EIS2**. Les exigences bien formées **EBF1** et **EBF2** sont obtenues de la sorte. Puis ces exigences bien formées sont regroupées selon les informations qu'elles contiennent pour constituer des ***groupes bien formés***. À partir de ces groupes bien formés, il est possible de produire une première représentation graphique, nommée ***squelette de modèle d'état***, qui facilite l'analyse. Enfin, les groupes bien formés comprennent un ensemble d'états de départ alors que nous désirons entreprendre la vérification sur une seule transition. En conséquence, il convient de construire des ***transitions de groupe*** à partir des groupes bien formés. Les transitions de groupe ont un unique état de départ et un unique état d'arrivée. Ainsi, à partir du groupe bien formé **GBF1**, deux transitions de groupe **TG1** et **TG2** sont constructibles selon que l'état **E1** ou l'état **E3** est considéré comme l'état de départ. L'état de départ est unique et clairement spécifié pour une transition de groupe, il manque seulement l'information relative aux opérateurs logiques entre les conditions pour aboutir aux ***transitions de groupe formalisées*** et aux ***propriétés formelles*** objectifs. Pour l'exemple représenté sur la figure 5.1, un opérateur ET logique a été choisi pour lier les conditions de la transition de groupe **TG1** entre elles, donnant lieu à **TGFo1** ; alors qu'un opérateur OU logique est placé entre les conditions de la transition de groupe **TG2**, ce qui donne **TGFo2**. Les propriétés formelles **PF1** et **PF2** sont directement déduites des transitions de groupe formalisées **TGFo1** et **TGFo2** et d'une structure type, qui sera détaillée dans le paragraphe 5.3.1.

5.2 Principales activités déployées

Les différentes données manipulées, présentées dans le paragraphe précédent, sont produites à l'issue de plusieurs activités, représentées sur la figure 5.2. Cette figure décompose les deux activités **AF1** et **AS1** de construction des modèles de comportement opérées respectivement du point de vue des AMS et des pilotes SdF (voir figure 4.8). Étant donné que ces deux activités sont menées de la même façon, selon les deux perspectives, nous employons le sigle **AFS1**. Par ceci nous entendons que les étapes suivies sont les mêmes, ainsi que leur séquencement et les types d'objets manipulés. En revanche, les sens que les deux métiers accordent à ces objets ne sont pas encore comparés entre eux. Ils peuvent différer en raison du cloisonnement partiel entre les deux ingénieries.

Quatre activités sont donc déployées pour réaliser l'activité globale de construction des modèles d'état. Il en résulte des modèles d'état du point de vue des AMS (notés MEF, pour Modèles d'État Fonctionnels) et des modèles selon la perspective des pilotes SdF

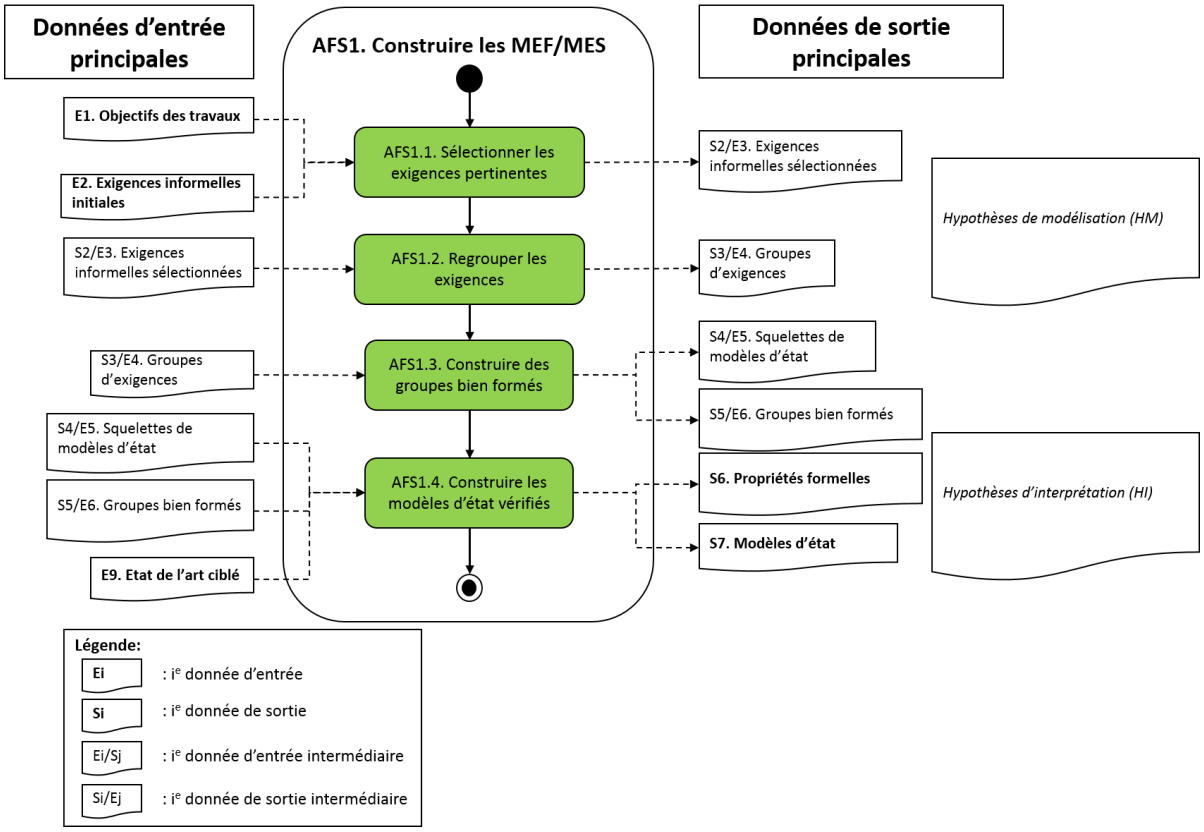


FIGURE 5.2 – Activités de construction des modèles d'état et de formalisation des exigences

(MES, pour Modèles d'État de Sûreté). Les principales données d'entrée et de sortie de chacune des activités sont reportées sur la figure 5.2. Ces données seront détaillées et contextualisées dans les paragraphes qui suivent. Par contre, il convient de définir dès à présent une notion très importante pour la traçabilité des exigences et la validité des modèles construits, il s'agit des *hypothèses*. Nous en distinguons deux sortes :

(a) ***Les hypothèses de modélisation (HM)*** : hypothèses portant sur la modélisation en elle-même. Il s'agit par exemple de règles pour regrouper les exigences ou de la définition de variables et de paramètres pour la formalisation. Il revient aux experts de choisir parmi ces hypothèses lesquelles sont correctes et d'exclure celles qui ne le sont pas ;

(b) ***Les hypothèses d'interprétation (HI)*** : il s'agit d'hypothèses émises pour compléter une exigence (ou un groupe d'exigence, voir paragraphe 5.4.2) qui n'est pas de la forme attendue. Cette hypothèse implique la modification ou la création d'exigences. La confrontation avec les experts permet de trancher entre les hypothèses et seules les interprétations les plus crédibles sont retenues.

Remarque 5.1
La traçabilité des exigences dans le processus de transformation est assurée par l'application de la méthode ainsi que l'adoption d'une nomenclature adaptée. ♢

Remarque 5.2
Afin de ne pas charger outre-mesure la figure 5.2, nous n'avons volontairement pas représenté les liens entre les activités et les hypothèses formulées. Ceux-ci seront détaillés par la suite. ♢

Remarque 5.3
*Pour la même raison, nous n'avons pas représenté comment les activités de choix des modèles corrects (**AF2** et **AS2** sur la figure 4.8), menées par les AMS et les pilotes SdF, sont en interaction avec l'activité AFS1. Ce parallélisme, représenté schématiquement sur la figure 4.8, est, dans la suite, précisé. Par ailleurs, l'état de maturité des données (proposées, vérifiées, validées...) n'est également pas indiqué sur la figure 5.2 car il sera spécifié dans les paragraphes suivants.* ♢

5.3 Sélection des exigences (AFS1.1)

5.3.1 Activité de sélection des exigences

5.3.1.1 Présentation des données d'entrée

Afin d'appréhender au mieux les activités qui vont suivre, d'en comprendre la logique et le fondement, nous commençons par donner des exemples d'exigences considérées (données d'entrée **E2** sur la figure 5.2). Rappelons que, du point de vue des AMS, les exigences considérées sont des exigences fonctionnelles de comportement contenues dans le document STR V1 (AMS Renault 2017b) et sont au nombre de 175 (voir paragraphe 2.3.3). Ces exigences sont allouées à la fonction AD globale (voir figure 2.12). Voici 10 exemples d'Exigences Fonctionnelles de Comportement (EFC) :

1 **EFC 1** : *« The lateral jerk requested by AD function shall be limited to p_lateralJerkBoundary »*
2 **EFC 2** : *« AD function shall not request brutal acceleration or braking in order to keep the Autonomous Vehicle stable, comfortable and quiet »*
3 **EFC 3** : *« AD function shall not request brutal steering wheel turns for visual comfort of the driver »*
4 **EFC 4** : *« IF AD function is **Available** THEN AD function shall give feedback via HMI »*
5 **EFC 5** : *« AD function shall be **Available** in Germany and France »*
6 **EFC 6** : *« AD function shall be **Available** at night without impact on Autonomous Vehicle performance »*
7 **EFC 7** : *« AD function shall be **Not_available** on shoulders »*
8 **EFC 8** : *« IF AD function is **Available** AND (turn indicator is on OR val_vEgo≥val_vActivation) THEN AD function shall be not **Activatable** »*
9 **EFC 9** : *« IF AD function is **Active** AND driver deletes current route definition in navigation system THEN AD function shall exit **Active** state and give a specific HMI feedback to the driver »*

10 **EFC 10** : *« IF AD function is **Active** AND driver has a foot on the brake pedal AND braking pressure>p_pMasterCylinderPressure THEN AD function shall exit **Active** state and give a specific HMI feedback to the driver »*

Remarque 5.4
« p_lateralJerkBoundary », « val_vActivation »,
*« p_pMasterCylinderPressure » sont des **paramètres** et « val_vEgo » est une **variable**, déjà définis dans les exigences textuelles initiales. Nous donnerons, dans le paragraphe 5.10, les définitions de ces deux notions. Nous verrons que nous utilisons autant que possible les paramètres et variables déjà déterminés, mais qu'il a malgré cela été nécessaire d'en proposer de nouveaux.* ♢

Par ailleurs, du point de vue des pilotes SdF, nous donnons également les 10 Exigences Fonctionnelles de SdF (EFS) suivantes, extraites des 217 exigences allouées aux Supervisions locales et contenues dans le Functional Safety Concept (IAV 2016) (nous notons EFSM, les EFS allouées à la fonction Main_AD, EFSS celles allouées à Sub_AD et EFS3, celles allouées à AD-3, voir figure 2.13) :

1 **EFSM 1** : *« The self-detection shall detect an excessive acceleration due to dysfunction of Main_AD function »*

2 **EFSM 2** : *« The self-detection shall detect an insufficient acceleration due to dysfunction of the Main_AD function »*

3 **EFSM 3** : *« No excessive acceleration due to a dysfunction of application of a fail silent strategy for Main_AD function »*

4 **EFSS 4** : *« No insufficient acceleration due to a dysfunction of application of a fail silent strategy for Sub_AD function »*

5 **EFSS 5** : *« No OR too low visibility of autonomous vehicle to other road users due to dysfunction of Sub_AD function »*

6 **EFSM 6** : *« In case of a faulty deactivation of the AD function due to dysfunction of Main_AD function, Main_AD shall switch itself **Off** »*

7 **EFSM 7** : *« In case of a faulty activation of the AD function due to dysfunction of Sub_AD function, Main_AD shall enter **MRM** »*

8 **EFSS 8** : *« In case of an excessive deceleration due to dysfunction of Sub_AD function, Sub_AD shall switch itself **Off** »*

9 **EFS3 9** : *« In case of excessive deceleration due to dysfunction of Sub_AD function, AD-3 shall enter **MRM** »*

10 **EFSS 10** : *« When the Main_AD is **Active**, the Sub_AD shall be in **Standby** »*

Rappelons que notre objectif principal est de garantir que le comportement de la Supervision AD respecte l'ensemble des exigences de comportement fonctionnels et de sûreté qui lui sont allouées. Dans ce but, nous avons choisi de construire des modèles d'état sur lesquels les exigences de comportement relatives au changement d'état de la Supervision AD puissent être vérifiées. C'est donc ce type d'exigences qui est retenu pour l'analyse et la vérification. Ainsi, par exemple, il apparaît que l'exigence EFC 3,

qui interdit un braquage trop brutal dû à la fonction AD doit être respectée par la fonction de contrôle (Contrôle AD, voir figure 2.12) et non par la fonction de Supervision AD. En effet, cette exigence ne spécifie pas un changement d'état de la Supervision AD mais une action particulière à effectuer par la fonction de contrôle. Il résulte de ce raisonnement que la première étape de notre travail doit consister en la détermination d'un *critère* qui systématise cette sélection sur l'ensemble des exigences considérées. Pour ce faire, les activités suivantes ont été déployées.

5.3.1.2 Activités mises en jeu

La figure 5.3 illustre les activités mises en œuvre pour procéder à la sélection des exigences qui seront par la suite analysées, parmi l'ensemble des exigences informelles initiales fournies. Dans un premier temps, un critère de sélection est déterminé à partir des objectifs principaux de ces travaux (**AFS1.1.1**). Ceci donne lieu à deux hypothèses (HM1, HM2), confrontées lors de l'activité **AFS2.1** aux AMS et aux pilotes SdF. Après leur validation, il est possible d'effectuer la sélection des exigences pertinentes (**AFS1.1.2**). Les deux acteurs métier sont donc impliqués dans le processus de formalisation des exigences et de construction des modèles d'état. Ces interventions rendent ce processus beaucoup plus efficace car elles évitent, nous le verrons notamment dans le paragraphe 5.5.3, l'explosion combinatoire.

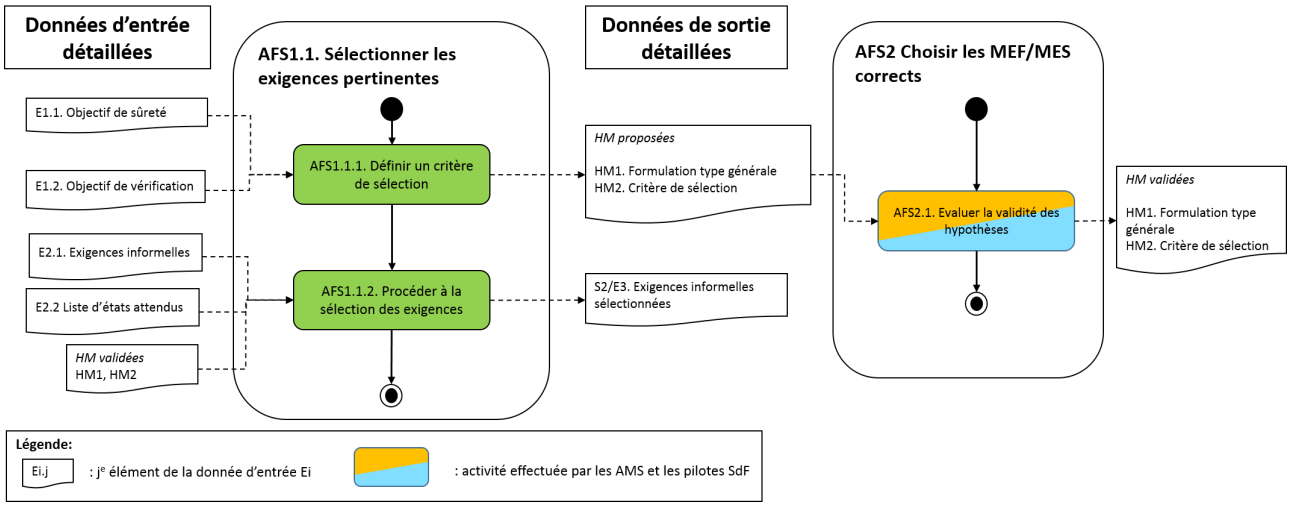


FIGURE 5.3 – Sélection des exigences pertinentes (**AFS1.1**)

5.3.1.3 Détermination d'un critère de sélection (AFS1.1.1)

Les deux objectifs de l'étude, **E1.1** et **E1.2**, qui nous ont amenés à nous focaliser sur les exigences dites *de transition* (*i.e.* exigences de comportement allouées à la Supervision AD, voir paragraphe 2.3.3.4) sont rappelés sur la figure 5.3. L'*objectif de sûreté* revient à s'assurer que la Supervision AD respecte l'ensemble des exigences de comportement qui lui sont allouées ; tandis que l'*objectif de vérification* est relatif à la possibilité de vérifier ces exigences en phase amont de conception. À partir de la définition d'exigence

de transition, nous avons été en mesure de déterminer une formulation générale que les exigences analysées doivent respecter :

Proposition 5.1
((Etat(s) de départ) ET (Condition(s) environnementale(s))) IMPLIQUENT (Etat d'arrivée)) ♠

La Proposition 5.1 fait l'objet de la première Hypothèse de Modélisation (**HM1**). Nous précisons cette proposition en donnant ci-dessous la définition des notions d'*état de départ*, de *conditions environnementales* et d'*état d'arrivée* :

Définition 5.1
Etat de départ *: dans un contexte donné par une exigence, l'ensemble d'états de départ est l'ensemble des états de l'automate pour lequel l'exigence s'applique. Cet ensemble peut comprendre (ou non) l'état initial de l'automate.* ♣

Définition 5.2
Condition environnementale *: une condition environnementale est relative à une évolution perceptible par les équipements embarqués dans le véhicule autonome de l'environnement. Ces changements sont principalement liés aux conditions extérieures (climat, type de route, signalétique...), l'état du véhicule (état des ouvrants, niveau d'huile moteur, etc), au comportement du conducteur (interactions avec l'IHM, les pédales, le volant...), ou encore à la disponibilité des fonctions de contrôle.* ♣

Définition 5.3
Etat d'arrivée *: dans un contexte donné par une exigence définissant une ou plusieurs conditions environnementales et un ensemble d'états de départ, l'état d'arrivée est l'**unique** état dans lequel la fonction spécifiée doit se trouver suite à l'occurrence des conditions environnementales. L'état d'arrivée ne peut pas être un élément de l'ensemble des états de départ.* ♣

Remarque 5.5
Pour une exigence, nous considérons que plusieurs états de départ sont possibles. C'est-à-dire que dans deux états A et B différents (états de départ) d'un modèle, si les mêmes conditions C sont remplies, alors un même état D est atteint (état d'arrivée). ♢

Remarque 5.6
Étant donné que les modèles d'état ne peuvent, au même moment, se trouver dans deux états différents, un seul état d'arrivée doit être défini. En outre, cet état d'arrivée doit être différent de l'ensemble des états de départ possibles. En effet si l'état de départ est le même que l'état d'arrivée, alors l'énoncé ne concerne pas un changement d'état et

l'exigence n'est pas une exigence de transition. ♢

Enfin, une notion découle de la Proposition 5.1, il s'agit de celle d'***exigence bien formée*** :

Définition 5.4
Exigence bien formée *: une exigence est dite bien formée si les trois éléments de la Proposition 5.1 se retrouvent dans son énoncé, à savoir, au moins un* ***état de départ*** *et au moins une* ***condition environnementale*** *ainsi qu'un (unique)* ***état d'arrivée.*** ♣

Notons que pour la suite, nous parlerons de ***type d'état*** pour signifier si, dans le contexte d'une exigence donnée, un état est un état **de départ** ou **d'arrivée**.

La Proposition 5.1 n'est pas exprimée dans une logique temporelle, donc non vérifiable par *model checking*, comme nous le souhaitons. Nous verrons par la suite (paragraphe 5.6.3), quelle formulation précise nous avons adoptée pour procéder à ces vérifications sur les modèles construits. En revanche, la Proposition 5.1 suffit à définir un *critère de sélection* des exigences considérées.

Définition 5.5
Critère de sélection des exigences *: les exigences retenues sont celles dont l'énoncé est relatif à* ***un changement d'état*** ♣

Ce critère de sélection constitue la deuxième hypothèse de modélisation : **HM2**. Plus précisément, les exigences sélectionnées sont celles qui comportent au moins, parmi les trois éléments constitutifs de la Proposition 5.1, une *Condition environnementale* ET un *Etat* (qui peut être de départ ou d'arrivée). Il résulte de ceci que cette sélection repose sur la définition des états de la Supervision AD. Avant de procéder à la sélection des exigences, menée lors de l'activité **AFS1.1.2**, les deux hypothèses de modélisation prises doivent être confrontées aux deux acteurs métier : les AMS et les pilotes SdF.

5.3.1.4 Évaluation de la validité des hypothèses (AFS2.1)

La Proposition 5.1 (**HM1**) a été jugée valide, à la fois par les AMS et les pilotes SdF, au vu du niveau de détail des exigences considérées et des objectifs de sûreté et de vérification poursuivis. Quant à l'hypothèse **HM2**, correspondant au critère de sélection, les deux acteurs métier l'ont approuvée également car elle découle de la première hypothèse.

5.3.1.5 Réalisation de la sélection (AFS1.1.2)

Point de vue des AMS

La sélection des exigences revient ici à ne retenir, parmi les exigences fonctionnelles de comportement, uniquement celles qui sont allouées à la fonction Supervision AD. L'autre information contenue dans la donnée d'entrée **E2** (voir figure 5.3) correspond à la *liste d'états attendus* de la fonction **Supervision AD** (voir paragraphe 2.3.3, figure 2.12), restituée ci-dessous :

- **Off** : état dans lequel la Supervision AD n'est pas alimentée électriquement. Il s'agit de l'*état initial* ;
- **Not_available** : premier état atteint suite à la mise sous tension du/des composants implémentant la Supervision AD ;
- **Available** : état atteint suite à la vérification de conditions environnementales définies dans certaines exigences fonctionnelles de comportement ;
- **Activatable** : état atteint après la vérification de conditions supplémentaires. Dans cet état la fonction AD est proposée au conducteur ;
- **Active** : état pour lequel la conduite est entièrement déléguée par le conducteur à la fonction AD.

Dans les 10 exigences exemples (EFC) présentées précédemment, les états sont identifiés en caractères gras. Il apparaît immédiatement que les exigences **EFC 1**, **EFC 2** et **EFC 3** ne font aucune mention à un état. C'est pourquoi elles sont directement éliminées. Ces exigences portent sur des actions qui ne donnent pas lieu à un changement d'état et sont à entreprendre dans un état donné. Il s'agit donc d'exigences que nous avons qualifiées d'*exigences d'état* (paragraphe 2.3.3.4). Concernant l'exigence suivante (**EFC 4**), on remarque qu'elle mentionne bien un état. Néanmoins, la condition de cette exigence n'est pas une condition environnementale. En effet, cette exigence spécifie une action particulière à effectuer par la fonction AD dans un état donné. C'est donc à la fonction Contrôle AD qu'elle est affectée. Elle est également une exigence d'état. Enfin, les six dernières exigences sont retenues. Effectivement, toutes ces exigences font référence à au moins un état. De plus, toutes les conditions qu'elles contiennent sont des conditions environnementales relatives :

- *à l'environnement extérieur* : **EFC 5**, la localisation du véhicule autonome (France, Allemagne), **EFC 6**, les conditions de luminosité (la nuit) et **EFC 7**, la localisation précise (accotement) ;
- *à l'état du véhicule* : **EFC 8**, l'état des clignotants et la vitesse du véhicule ;
- *au comportement du conducteur* : **EFC 9**, **EFC 10**, interactions avec l'IHM (respectivement le système de navigation et la pédale de freinage).

En définitive, sur les 175 exigences fonctionnelles de comportement contenues dans le document STR V1, **69** ont été retenues de cette manière, soit 39 % des exigences considérées. Les autres exigences de comportement sont allouées à la fonction de Contrôle AD et vérifiées par d'autres moyens, plus appropriés (tests, simulation).

Point de vue des pilotes SdF

Les exigences fonctionnelles de sûreté de fonctionnement, émises par les pilotes SdF, sont contenues dans le document FSC (IAV 2016). Ce document compte au total 350 exigences, allouées à tous les éléments de la fonction AD. Mais nous nous concentrons

sur les exigences portant sur la Supervision AD (au nombre de 217), plus précisément les trois sous-fonctions de Supervision (Main_AD, Sub_AD, AD-3, voir figure 2.13). Le problème, du point de vue de la SdF, est donc posé différemment. En effet, toutes les exigences sont déjà allouées aux sous-fonctions de Supervision. Par contre, toutes les exigences considérées ne sont pas des exigences de comportement. La liste d'états attendus de la Supervision AD est en réalité constituée de trois « sous-listes » (une par sous-fonction de supervision), que voici :

1 **Liste d'états attendus de la fonction Main_AD** :

- ***Off*** : état atteint suite à la défaillance de la fonction Control_Main_Active (voir figure 2.13) ;
- ***Active*** : état pour lequel la conduite est entièrement déléguée par le conducteur à la fonction Control_Main_Active, en phase de fonctionnement nominal. Cet état est l'*état initial* de la fonction Main_AD ;
- ***MRM*** : état dans lequel la fonction Control_Main_MRM est activée et réalise des manœuvres appropriées afin de mettre le véhicule en sécurité (MRM pour Minimal Risk Manoeuver).

2 **Liste d'états attendus de la fonction Sub_AD** :

- ***Off*** : état atteint suite à la défaillance de la fonction Control_Sub_Standby ;
- ***Standby*** : état pour lequel la conduite est entièrement déléguée par le conducteur à la fonction Control_Main_Active, en phase de fonctionnement nominal. Il s'agit de l'*état initial* de la fonction Sub_AD ;
- ***MRM*** : état dans lequel la fonction Control_Sub_MRM est activée et réalise des manœuvres appropriées afin de mettre le véhicule en sécurité.

3 **Liste d'états attendus de la fonction AD-3**

- ***Standby*** : état pour lequel la conduite est entièrement déléguée par le conducteur à la fonction Control_Main_Active, en phase de fonctionnement nominal. L'état *Standby* est l'*état initial* de la fonction AD-3 ;
- ***MRM*** : état dans lequel la fonction Control_3_MRM est activée et réalise des manœuvres appropriées afin de mettre le véhicule en sécurité.

De la même manière que le point de vue des AMS, nous avons mis en évidence les références aux états en les marquant par des caractères gras, dans les exigences exemples présentées précédemment. Nous voyons que les cinq premières exigences, ne faisant aucune mention à un état, sont d'emblée exclues. Pour **EFSM 1** et **EFSM 2**, aucun événement particulier n'est spécifié, ces exigences doivent donc être toujours vérifiées. En revanche, les trois exigences suivantes (**EFSM 3**, **EFSS 4** et **EFSS 5**) concernent l'occurrence d'événements élémentaires redoutés (accélération intempestive, insuffisante, visibilité trop faible) qui doit être réduite au minimum. Ces exigences sont en fait associées à un niveau d'intégrité (niveau ASIL, défini dans la norme ISO 26262), qui prescrit une probabilité d'occurrence maximale. Elles sont ensuite vérifiées par la fiabilité intrinsèque des composants utilisés ou la mise en place de redondance matérielle par exemple. Les cinq dernières exigences sont, quant à elles, retenues. En effet, ces exigences contiennent clairement un état, mais aussi des conditions environnementales, cette fois-ci relatives à des défaillances de fonctions. Ces événements peuvent affecter

les sous-fonctions de contrôle. Le cas de l'exigence **EFSS 10** est plus singulier car la condition environnementale concerne cette fois un état de fonctionnement normal (*Active*) de la fonction Main_AD.

Au final, sur les 217 exigences prises en compte, **61** ont été sélectionnées pour être analysées et vérifiées, ce qui représente 28 % des exigences considérées. Comme pour le point de vue des AMS, les exigences non retenues sont vérifiées selon un autre procédé que celui de la démarche proposée.

Remarque 5.7
L'activité de sélection des exigences consiste à filtrer uniquement les exigences de comportement allouées à la Supervision AD. Ainsi, du point de vue de la SdF, étant donné que les sous-fonctions de Supervision AD sont initialement nommées Main_AD, Sub_AD et AD-3, aucune modification n'est à apporter aux énoncés des exigences retenues. En revanche, les exigences non sélectionnées devront être modifiées pour éviter les ambiguïtés. En effet, l'exigence ***EFSM 1*** *par exemple, ne porte pas sur la fonction Main_AD mais sur une fonction de Contrôle AD. Concernant la perspective des AMS, nous avons dû préciser tous les énoncés des exigences retenues en substituant à la mention « **AD function** » le terme : « **AD Supervision** ». Ainsi, l'exigence sélectionnée* ***EFC 5*** *s'écrit désormais : « **AD Supervision** shall be Available in Germany and France ».* ♢

5.3.2 Bilan

Cette première activité de sélection précise l'objet de notre étude en rappelant le type d'exigences retenu et en les illustrant par des exemples pratiques. Nous avons vu qu'entre 28 % (point de vue SdF) et 39 %, pour les AMS, des exigences ont été sélectionnées. Les exigences exclues peuvent être vérifiées par d'autres techniques que celle que nous utilisons. Le constat qui s'impose pour les exigences retenues est qu'elles ne sont pas rédigées selon la formulation type générale (Proposition 5.1). Au moins un élément de la Proposition est toujours manquant. Ceci est bloquant par la suite car nous ne serons pas en mesure de vérifier les énoncés initiaux à l'aide de modèles d'état. Par conséquent, il est nécessaire de compléter les exigences sélectionnée.

Nous verrons qu'un traitement exigence par exigence se heurte à l'explosion combinatoire. C'est pourquoi les exigences doivent être, au préalable, regroupées en fonction des informations d'état qu'elles contiennent.

5.4 Création de groupes d'exigences (AFS1.2)

5.4.1 Constat sur les exigences sélectionnées

Les exigences retenues (tant du point de vue des AMS que des pilotes SdF) ne sont pas des exigences bien formées (voir Définition 5.4 et figure 5.1). Or, afin d'analyser et de vérifier ces exigences sur des modèles d'état, les exigences sélectionnées doivent être exprimées sous la forme de la Proposition 5.1. En conséquence, il est nécessaire de

compléter chacune des exigences retenues, *i.e.* d'ajouter au moins un des trois éléments constitutifs de la formulation générale souhaitée. Chacun de ces ajouts donne lieu à la formulation d'une *hypothèse d'interprétation*. Prenons l'exemple de l'exigence retenue **EFC 7**, dont nous rappelons ci-dessous l'énoncé :

EFC 7 : *« AD Supervision shall be* ***Not_ available*** *on shoulders*[1] *»*

Seuls deux éléments de la Proposition 5.1 sur les trois requis figurent dans le contenu de cette exigence. Une *condition environnementale*, (*on shoulders*) et l'**état d'arrivée**, mis en évidence par les caractères gras (***Not_ available***). Par conséquent, le/les **états de départ** sont manquants. Il est donc nécessaire d'ajouter cet élément afin que l'exigence EFC 7 devienne une exigence bien formée. On compte 15 manières possibles de procéder à cela :

- *1 seul état de départ (4 possibilités)* : Off, Available, Activatable, Active (rappelons que l'état de départ et l'état d'arrivée doivent être différents) ;
- *2 états de départ (6 possibilités)* : (Off, Available), (Off, Activatable), (Off, Active), (Available, Activatable), (Available, Active), (Activatable, Active) ;
- *3 états de départ (4 possibilités)* : (Off, Available, Activatable), (Off, Available, Active), (Off, Activatable, Active), (Available, Activatable, Active) ;
- *4 états de départ (1 possibilité)* : (Off, Available, Activatable, Active).

Chaque possibilité donne lieu à une *Hypothèse d'Interprétation (HI)*. Ainsi, nous obtenons par exemple :

- *HI 4* : l'état de départ est **Off**. Ce qui donne :
  EFC 7.4 : *« **IF ((AD Supervision is in the state Off)** AND Autonomous Vehicle is on shoudlers) THEN AD Supervision shall be **Not_ available** »*
- *HI 5* : les états de départ possibles sont **Off** OU **Available**. Ce qui donne :
  EFC 7.5 : *« **IF ((AD Supervision is in the state Off OR AD Supervision is in the state Available)** AND Autonomous Vehicle is on shoudlers) THEN AD Supervision shall be **Not_ available** »*

Dans l'ensemble des EFC sélectionnées, 14 sont de la même forme que EFC 7 , *i.e.* définissent une ou plusieurs conditions, un état d'arrivée (*Not_ available*) et aucun état de départ. Puisque pour chacune de ces exigences il existe 15 manières de les compléter, on en déduit que l'on peut trouver 15^{14} ($= 2.92.10^{16}$) combinaisons de ces exigences bien formées. Ainsi, il est assez clair que la prise en compte de l'ensemble de ces possibilités pour chacune des exigences rend le problème inanalysable (voir paragraphe 5.5.3), étant donné la quantité de possibilités ainsi produites. Nous avons partagé ce constat avec les deux métiers concernés. Après concertation, il a été acté que le nombre de possibilités devait être réduit afin de palier cette difficulté. Le principal écueil est de ne pas supprimer des possibilités intéressantes. Pour ce faire, des ***groupes d'exigences*** ont été formés en travaillant avec les AMS et les pilotes SdF.

1. accotement

5.4.2 Formation de groupes d'exigences (AFS1.2)

5.4.2.1 Aperçu des activités mises en œuvre

La figure 5.4 fait apparaître les activités déployées pour former des groupes d'exigences. Dans un premier temps (activité **AFS1.2.1**), une *règle de regroupement* a été formulée. Elle fait l'objet de l'hypothèse de modélisation **HM3**. Cette dernière a ensuite été soumise à la fois aux AMS et aux pilotes SdF pour validation (**AFS2.2**). Enfin, il est possible, à l'aide des exigences sélectionnées et de cette règle, de former les différents groupes d'exigences durant l'activité **AFS1.2.2**.

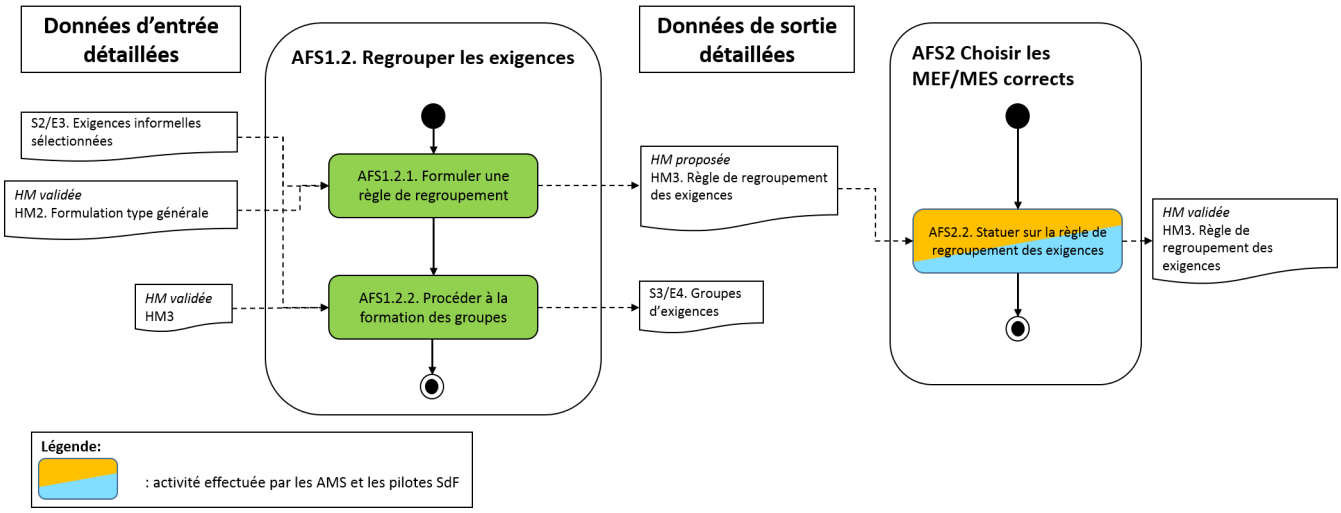


FIGURE 5.4 – Regroupement des exigences (**AFS1.2**)

5.4.2.2 Élaboration et validation d'une règle de regroupement (AFS1.2.1, AFS2.2)

L'application du critère de sélection implique que toutes les exigences retenues contiennent au moins une condition environnementale. En revanche, nous n'avons *a priori* aucune garantie sur les deux autres informations requises pour obtenir des exigences bien formées. Avant de définir la notion de groupe d'exigences, nous devons au préalable préciser les deux termes suivants :

Définition 5.6
Degré de définition d'un état *: pour une exigence retenue, il s'agit de la quantité d'information disponible à propos d'un état. C'est-à-dire qu'un état peut être :*

(a) ***Entièrement défini (Df)*** *: l'intitulé de l'état apparaît de manière claire et non ambigüe dans l'énoncé de l'exigence considérée ;*

(b) ***Non défini (NDf)*** *: l'intitulé de l'état n'apparaît pas du tout dans l'énoncé de l'exigence considérée ;*

(c) ***Partiellement défini (PDf)*** *: l'intitulé de l'état apparaît mais sous une forme négative. L'exigence* ***EFC 8****, avec l'état* ***Activatable****, en est un exemple.* ♣

Définition 5.7
Ensemble d'exigences *: agrégation d'exigences (sélectionnées) contenant le ou les* ***même(s) type(s) d'état*** *caractérisés par le* ***même degré de définition****.* ♣

Par exemple, EFC 5 et EFC 7 appartiennent au même ensemble d'exigence. En effet, pour ces deux exigences :

— L'état de départ (*type d'état*) n'est pas défini (*NDf*) ;

— L'état d'arrivée (*type d'état*) est défini (*Df*) : *Available* pour EFC 5 et *Not_ available* pour EFC 7.

Ainsi, neuf ensembles d'exigences sont constitués de la sorte. En effet nous considérons deux types d'états selon trois modalités différentes (degrés de définition). Ce premier regroupement est illustré par la Table 5.1 dans laquelle les neuf ensembles sont représentés. De plus, nous y avons, à titre illustratif, placé chacune des 11 exigences retenues dans le paragraphe 5.3.1.

		État d'arrivée		
		NDf	**Df**	**PDf**
État de départ	**NDf**	*Ensemble 1*	*Ensemble 2* EFSM 6, EFSM 7, EFSS 8 EFS3 9, EFSS 10 EFC 5, EFC 6, EFC 7	*Ensemble 3*
	Df	*Ensemble 4* EFC 9, EFC 10	*Ensemble 5*	*Ensemble 6* EFC 8
	PDf	*Ensemble 7*	*Ensemble 8*	*Ensemble 9*

TABLE 5.1 – Positionnement des EFC et EFS exemples retenues dans les ensembles d'exigences

Il est désormais possible de définir les ***groupes d'exigences*** :

Définition 5.8
Groupe d'exigences *: agrégation des exigences appartenant à un même ensemble d'exigences et partageant le ou les mêmes état(s).* ♣

Chaque groupe d'exigences et chaque exigence sélectionnée est donc caractérisé de la manière suivante : (États de départ, État d'arrivée) = ((NDf, Df, PDf), (NDf, Df,

PDf)). À partir de cette caractérisation, la règle de regroupement suivante a été proposée :

Règle de regroupement des exigences (HM3) : *une exigence retenue appartient à un groupe d'exigences si elle possède les mêmes caractéristiques que le groupe en termes de types d'état, de degrés de définition des états et de définition des états s'ils sont définis ou partiellement définis.*

Par exemple, pour **EFC 5** : (État de départ, État d'arrivée) = (**NdF**, **Df** : *Available*) et pour **EFC 7** : (État de départ, État d'arrivée) = (**NdF**, **Df** : *Not_ available*). Ces deux exigences appartiennent donc à deux groupes distincts car, bien que le type d'état et le degré de définition des états soient partagés, les deux états d'arrivée sont différents. Un ensemble d'exigences contient donc plusieurs groupes d'exigences.

Cette règle a été soumise aux deux expertises métier durant l'activité **AFS2.2**. Le postulat selon lequel les exigences ayant les mêmes caractéristiques en termes d'informations d'état peuvent être regroupées a été validé par les deux expertises. La règle a donc ensuite été appliquée à toutes les exigences sélectionnées afin de procéder à la formation des groupes (**AFS1.2.2**).

5.4.2.3 Formation des groupes d'exigences (AFS1.2.2)

La Table 5.2 illustre la formation et la répartition dans les différents ensembles des groupes d'exigences, respectivement du point de vue des AMS, et de celui des pilotes SdF. Dans la Table 5.3, la répartition des exigences exemples (EFC et EFS) dans les différents groupes est restituée. Les notations adoptées sont les suivantes :

- *GFi.j* : j^{e} groupe d'exigences contenant les exigences fonctionnelles de comportement retenues du i^{e} ensemble d'exigences ;
- *GSMi.j* : j^{e} groupe d'exigences contenant les exigences de SdF retenues portant sur la fonction Main_AD et appartenant au i^{e} ensemble d'exigences ;
- *GSSi.j* : j^{e} groupe d'exigences contenant les exigences de SdF retenues portant sur la fonction Sub_AD et appartenant au i^{e} ensemble d'exigences ;
- *GS3i.j* : j^{e} groupe d'exigences contenant les exigences de SdF retenues portant sur la fonction AD-3 et appartenant au i^{e} ensemble d'exigences ;
- *Gi.j : (État 1, État 2)* signifie que toutes les exigences du groupe *Gi.j* :
  (a) Ont l'É*tat 1* pour état de départ ET l'É*tat 2* comme état d'arrivée si les deux états sont définis ;
  (b) Ont l'É*tat 1* pour état de départ OU l'É*tat 2* comme état d'arrivée si l'un des deux états est défini et l'autre ne l'est pas ;
  (c) N'ont pas l'É*tat 1* pour état de départ OU (resp. ET) n'ont pas l'É*tat 2* comme état d'arrivée si, respectivement, l'un des deux états est partiellement défini ou les deux états sont partiellement définis.

		État d'arrivée		
		NDf	**Df**	**PDf**
État de départ	**NDf**	*Ensemble 1*	*Ensemble 2* **GF2.1** : *(-, Available)* **GF2.2** : *(-, Not_available)* **GSM2.1** : *(-, Off)* **GSM2.2** : *(-, MRM)* **GSS2.1** : *(-, Off)* **GSS2.2** : *(-, MRM)* **GSS2.3** : *(-, Standby)* **GS32.1** : *(-, MRM)*	*Ensemble 3* **GF3.1** : *(-, Active)*
	Df	*Ensemble 4* **GF4.1** : *(Active, -)*	*Ensemble 5* **GF5.1** : *(Available, Active)*	*Ensemble 6* **GF6.1** : *(Available, Activatable)*
	PDf	*Ensemble 7*	*Ensemble 8*	*Ensemble 9*

TABLE 5.2 – Groupes d'exigences du point de vue des AMS et des pilotes SdF

		Etat d'arrivée		
		NDf	**Df**	**PDf**
Etat de départ	**NDf**	*Ensemble 1*	*Ensemble 2* {EFSM 6} ∈ **GSM2.1**, {EFSM 7} ∈ **GSM2.2** {EFSS 8} ∈ **GSS2.1**, {EFSS 10} ∈ **GSS2.3** {EFS3 9} ∈ **GS32.1** {EFC 5, EFC 6} ∈ **GF2.1** {EFC 7} ∈ **GF2.2**	*Ensemble 3*
	Df	*Ensemble 4* {EFC 9, EFC 10} ∈ **GF4.1**	*Ensemble 5*	*Ensemble 6* {EFC 8} ∈ **GF6.1**
	PDf	*Ensemble 7*	*Ensemble 8*	*Ensemble 9*

TABLE 5.3 – Répartition des exigences exemples dans les groupes d'exigences

Remarque 5.8
Dans la Table 5.2, il apparaît que les ensembles 7, 8 et 9 demeurent vides. Ceci est simplement dû à l'échantillon d'exigences retenues. Effectivement, pour aucune de ces exigences, l'état de départ était Partiellement Défini. Avec un autre cas d'étude, ces ensembles pourraient être non vides. ♢

On constate (Table 5.3) que 6 groupes d'exigence ont été créés, tant pour les exigences fonctionnelles de comportement que pour les exigences de SdF. Cette activité **AFS1.2** est finalement une réorganisation des données d'entrée. Son objectif principal est de réduire le champ des possibles pour la formalisation à venir. En revanche, les

groupes ainsi formés ne contiennent pas d'exigences bien formées (excepté les groupes de l'ensemble 5). Or, pour vérifier les exigences, celles-ci doivent être des exigences bien formées. L'activité suivante (**AFS1.3**) décrit la manière d'obtenir des groupes bien formés, à partir des exigences retenues.

5.5 Élaboration de groupes bien formés (AFS1.3)

5.5.1 Activités déployées

Nous avions vu comment (voir paragraphe 5.4.1), pour l'exigence **EFC 7**, obtenir l'ensemble des exigences bien formées possibles. Pour ce faire, nous avions complété les informations d'état manquantes dans l'énoncé initial de EFC 7. Nous appliquons donc exactement le même principe aux groupes d'exigences formés lors de l'activité précédente (**AFS1.2**). L'idée est de réduire le nombre d'exigences bien formées possibles en ne traitant pas les exigences une à une, mais par groupe. La figure 5.5 illustre les activités menées pour obtenir des groupes *bien formés validés* à partir des groupes d'exigences. Tout d'abord, nous verrons comment construire des groupes bien formés (**AFS1.3.1**). Nous comparons ensuite le nombre d'interprétations que nous pourrions obtenir sans considérer les groupes d'exigences et en les prenant en compte. Nous remarquons que le regroupement des exigences, même s'il réduit le nombre de possibilités, ne le rend pas encore analysable. C'est la raison pour laquelle nous avons estimé la **plausibilité** des interprétations (**AFS1.3.2**). Ce travail constitue un dernier filtre, qui doit être validé par l'expertise appropriée, lors de l'activité **AFS2.3**.

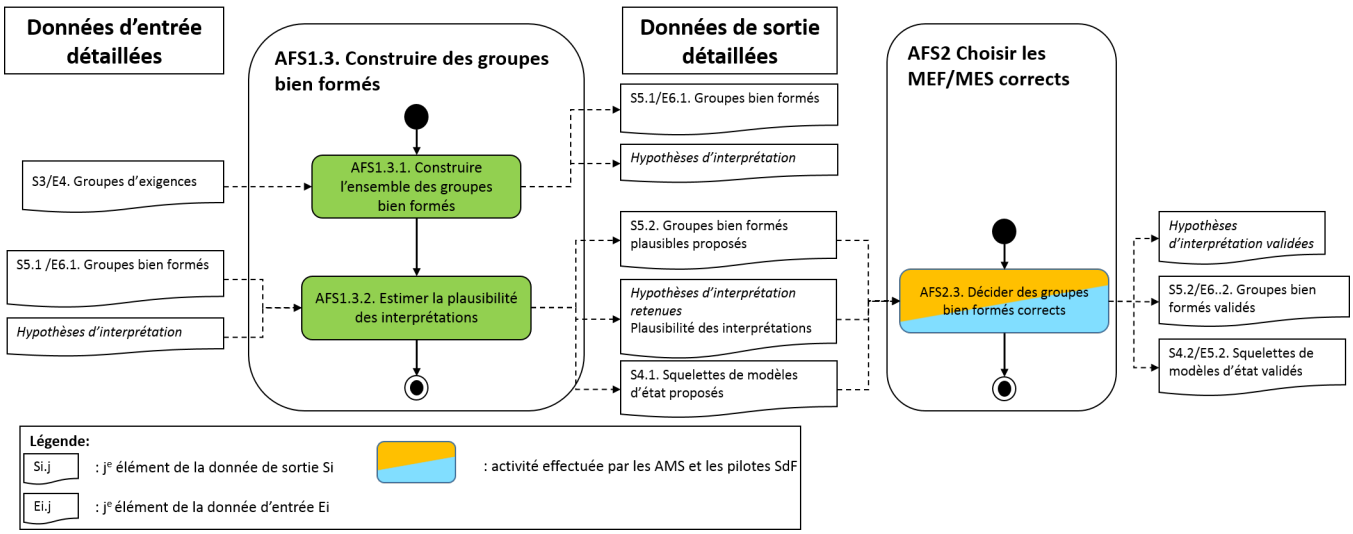


FIGURE 5.5 – Construction des groupes bien formés (**AFS1.3**)

5.5.2 Groupes bien formés (AFS1.3.1)

La notion de *groupe bien formé* est définie à partir de celle d'*exigence bien formée* (voir Définition 5.4) :

Définition 5.9
Groupe bien formé *: un groupe bien formé est un groupe qui ne contient que des exigences bien formées partageant le/les mêmes états de départ et le même état d'arrivée.* ♣

À titre illustratif, il est possible de créer 15 groupes bien formés à partir du groupe GF2.2. En effet, le raisonnement que nous avions tenu pour l'exigence EFC 7 (qui appartient au groupe GF2.2, voir Table 5.3) afin d'obtenir les exigences bien formées possibles à partir de l'exigence EFC 7, est alors appliqué à GF2.2. Par exemple, un des groupes bien formés, déduit à partir de GF2.2 et correspondant à l'hypothèse d'interprétation **HI5**, est le groupe noté **GF2.2.5**, pour lequel l'état de départ est *Off* (voir paragraphe 5.4.1). Les *hypothèses d'interprétation* prises pour une seule exigence, sont alors formulées pour toutes les exigences appartenant au groupe d'exigences analysé.

A l'issue de l'activité **AFS1.3.1**, tous les groupes bien formés possibles sont déterminés. Dans le paragraphe suivant, deux points clef de notre démarche sont justifiés :

1 *La formation de groupe d'exigences*, dont la nécessité s'est fait sentir au préalable (voir paragraphe 5.4.2) sans que nous soyons encore en mesure d'évaluer les effets de cette hypothèse sur l'ensemble des exigences retenues ;

2 *La détermination de la plausibilité des groupes bien formés* afin de réduire le nombre de ceux qui seront analysés.

Pour ce faire, nous donnons par la suite une évaluation du nombre de possibilités d'interprétations. Tous les résultats sont exprimés en fonction du nombre d'états de la liste d'états attendus.

5.5.3 Justification des hypothèses relatives aux groupes d'exigences

5.5.3.1 Application aux exigences fonctionnelles de comportement retenues

La Table 5.4 donne le nombre de combinaisons de groupes bien formés possibles par ensemble (*Calcul avec regroupement*), et le nombre de combinaisons d'exigences bien formées possibles, également par ensemble (*Calcul sans regroupement*).

Ces calculs sont opérés de deux manières :

— *Calcul avec regroupement* : le nombre de groupes bien formés qu'il est possible de déduire à partir d'un groupe d'exigences (voir paragraphe suivant) d'un ensemble est multiplié autant de fois que le nombre de groupes d'exigences par ensemble (quantité en exposant) ;

— *Calcul sans regroupement* : pour un ensemble d'exigences donné, le nombre d'exigences bien formées possibles à partir d'une exigence analysée est multiplié autant de fois que la somme de toutes les exigences contenues dans l'ensemble d'exigences considéré. Par exemple, pour l'ensemble 2, 15 exigences bien formées sont constructibles à partir de chacune des exigences se trouvant dans cet ensemble (comme EFC 7). Étant donné que l'on trouve 27 (13 dans GF2.1 et 14 dans

		État d'arrivée		
		NDf	**Df**	**PDf**
État de départ	**NDf**	*Ensemble 1*	*Ensemble 2* Avec regroupement $[15]^2 = 225$ Sans regroupement $[15]^{(13+14)} = 5,68.10^{31}$	*Ensemble 3* Avec regroupement $[60]^1 = 60$ Sans regroupement $[60]^1 = 60$
	Df	*Ensemble 4* Avec regroupement $[4]^1 = 4$ Sans regroupement $[4]^{36} = 4,72.10^{21}$	*Ensemble 5* Avec regroupement $[1]^1 = 1$ Sans regroupement $[1]^4 = 1$	*Ensemble 6* Avec regroupement $[4]^1 = 4$ Sans regroupement $[4]^1 = 4$
	PDf	*Ensemble 7*	*Ensemble 8*	*Ensemble 9*

TABLE 5.4 – Nombre de combinaisons de groupes bien formés et d'exigences bien formées possibles pour les EFC retenues

GF2.2) exigences de cette forme dans les exigences fonctionnelles retenues, on en déduit la quantité donnée dans la Table 5.4.

5.5.3.2 Comparaison du nombre de combinaisons possibles

Pour évaluer le nombre total de combinaisons possibles, de groupes bien formés ou d'exigences bien formées, à partir des exigences fonctionnelles de comportement retenues, il suffit de multiplier les valeurs obtenues précédemment de la façon suivante :

— *Calcul avec regroupement* : **216 000** ($= 225.60.4.1.4$) combinaisons sont possibles, c'est-à-dire 216 000 combinaisons de groupes bien formés ;

— *Calcul sans regroupement* : **6,43.10^{55}** ($= 5,68.10^{31}.60.4,72.10^{21}.1.4$) combinaisons sont possibles. Dans ce cas, $6,43.10^{55}$ combinaisons d'exigences bien formées sont obtenues.

Ce raisonnement vient donc appuyer le constat fait dans le paragraphe 5.4.2 qui a conduit à regrouper les exigences et émettre des hypothèses d'interprétation sur des groupes d'exigences et non des exigences seules. En effet, le nombre de possibilités obtenu sans regrouper les exigences est clairement non analysable dans un délai raisonnable.

Etant donné que les exigences sont en constante évolution au cours de la vie d'un projet, la manière spécifique (*i.e.* avec les exigences fonctionnelles de comportement retenues dans le cadre de cette étude) dont est remplie la Table 5.4 peut changer avec ces évolutions. En outre, la démarche proposée doit être, autant que possible, réutilisable. Pour ces raisons, nous avons évalué, dans le cas général, le nombre de possibilités d'interprétations en fonction du nombre d'états.

5.5.3.3 Nombre d'interprétations possibles en fonction du nombre d'états

Dans un premier temps, nous avons, pour chaque ensemble, établi le nombre maximum de groupes d'exigences possibles en fonction du nombre d'états **N** de la liste d'états attendus. Cette quantité est dénommée *Card[Ei]* (i variant de 1 à 9) dans la Table 5.5. Le cardinal peut être nul si un ensemble ne contient aucune exigence. C'est le cas, par exemple, des ensembles 1, 7, 8 et 9 pour les EFC retenues (voir Table 5.2). Ensuite, nous avons évalué, pour chaque groupe d'exigences, le nombre de groupes bien formés qu'il était possible de déduire, toujours en fonction de **N**. Les résultats que nous avons obtenus apparaissent dans la Table 5.5. Dans la Table 5.5, les expressions

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="3">État d'arrivée</th></tr>
<tr><th>NDf</th><th>Df</th><th>PDf</th></tr>
<tr><td rowspan="5">État de départ</td><td>NDf</td><td>Ensemble 1
Card[E1] $\in [0,1]$
$[\sum_{i=1}^{N} \binom{N}{i}(N-i)]^{Card[E1]}$</td><td>Ensemble 2
Card[E2] $\in [0,N]$
$[\sum_{i=1}^{N} \binom{N}{i}(1-\frac{i}{N})]^{Card[E2]}$</td><td>Ensemble 3
Card[E3] $\in [0,N]$
$[\sum_{i=1}^{N} \binom{N}{i}(1-\frac{i}{N})$
$(N-i-1)+\frac{i}{N}\binom{N}{i}(N-i)]^{Card[E3]}$</td></tr>
<tr><td rowspan="2">Df</td><td rowspan="2">Ensemble 4
Card[E4] $\in [0,N]$
$[(N-1)]^{Card[E4]}$</td><td rowspan="2">Ensemble 5
Card[E5] $\in [0,N.(N-1)]$
$[1]^{Card[E5]}$</td><td>Ensemble 6
Cas 1 :
Card[E6.1] $\in [0,N.(N-1)]$
$[(N-1)]^{Card[E6.1]}$</td></tr>
<tr><td>Cas 2 :
Card[E6.2] $\in [0,N.(N-2)]$
$[(N-2)]^{Card[E6.2]}$</td></tr>
<tr><td rowspan="2">PDf</td><td rowspan="2">Ensemble 7
Card[E7] $\in [0,N]$
$[\sum_{i=1}^{N} \binom{N-1}{i}(N-i)]^{Card[E7]}$</td><td>Ensemble 8
Cas 1 :
Card[E8.1] $\in [0,N.(N-1)]$
$[\sum_{i=1}^{N} \binom{N-1}{i}]^{Card[E8.1]}$</td><td>Ensemble 9
Cas 1 :
Card[E9.1] $\in [0,(N-2)^2+(N-1)]$
$[\sum_{i=1}^{N} \binom{N-1}{i}(1-\frac{i}{N-1}).(N-i-1)$
$+\frac{i}{N-1}\binom{N-1}{i}(N-i)]^{Card[E9.1]}$</td></tr>
<tr><td>Cas 2 :
Card[E8.2] $\in [0,N.(N-2)]$
$[\sum_{i=1}^{N} \binom{N-1}{i}.$
$(1-\frac{i}{N-1})]^{Card[E8.2]}$</td><td>Cas 2 :
Card[E9.2] $\in [0,(N-2).(N-1)]$
$[\sum_{i=1}^{N} \binom{N-1}{i}(N-i-1)]^{Card[E9.2]}$</td></tr>
</table>

TABLE 5.5 – Nombre de combinaisons groupes bien formés possibles par ensemble en fonction du nombre d'états **N**

de chaque cellule donnent le nombre de combinaisons de groupes bien formés possibles par ensemble d'exigences. C'est-à-dire :

— Pour un ensemble d'exigences donné, le terme entre crochets (sans l'exposant) correspond au nombre de groupes bien formés qu'il est possible de construire à partir d'un groupe de l'ensemble considéré. C'est justement l'application de ces formules (pour N=5, le nombre d'états du point de vue des AMS) qui explique les quantités (également entre crochets) apparaissant dans la Table 5.4 ;

— L'expression complète (avec l'exposant) donne le nombre de combinaisons possibles de groupes bien formés, pour un ensemble donné. En effet, étant donné que chacun des groupes d'un ensemble est indépendant (puisque chacun spécifie un changement d'état différent), il faut multiplier tous les groupes bien formés possibles, à partir d'un groupe de l'ensemble considéré, autant de fois que de groupes constituant l'ensemble (Card[Ei]).

Par ailleurs, deux cas sont identifiés dans la Table 5.5. Dans le *cas 2*, les ensembles d'états de départ et d'arrivée ont une intersection non nulle. Or, les états de départ et d'arrivée ne doivent pas être identiques. Il y a donc moins de combinaisons possibles que dans le *cas 1*. En effet, pour ce cas, soit cette intersection est nulle (pour les Ensembles 6 et 8), soit elle comprend un nombre d'éléments inférieur à l'intersection du cas 2, pour l'Ensemble 9. L'exigence **EFC 8** (appartenant à l'*Ensemble 6*) donne une illustration du cas 2 :

— L'état de départ est **Défini** : il s'agit de l'état *Available*, donc l'ensemble d'états de départ possible est : *{Available}* ;

— L'état d'arrivée est **Partiellement Défini** : il s'agit de l'état *Activatable*, donc l'ensemble d'états d'arrivée possible est : *{Off, Not_ available, Available, Active}* ;

— Il y a ainsi **3** (soit effectivement $(5 - 2)^1$, voir Table 5.5) possibilités de complétion : *(État de départ, État d'arrivée) = (Available, Off)* OU *(Available, Not_ available)* OU *(Available, Active)*.

Si l'on suppose maintenant que l'état d'arrivée, toujours partiellement défini, de l'exigence **EFC 8**, est *Available*, alors on se trouve dans le *cas 1* et :

— L'ensemble d'états de départ possible est : *{Available}* ;

— L'ensemble d'états d'arrivée possible est : *{Off, Not_ available, Activatable, Active}* ;

— Il y a donc **4** (soit $(5 - 1)^1$, voir Table 5.5) possibilités de complétion : *(État de départ, État d'arrivée) = (Available, Off)* OU *(Available, Not_ available)* OU *(Available, Activatable)* OU *(Available, Active)*.

En conclusion, on remarque que, même en formant des groupes d'exigences, le nombre de combinaisons possibles reste encore considérable (voir Table 5.4). C'est pourquoi nous sommes contraints de prendre de nouvelles hypothèses pour réduire encore le nombre de possibilités. Ceci fait justement l'objet de l'activité **AFS1.3.2** (voir figure 5.5).

5.5.4 Estimation de la plausibilité des groupes (AFS1.3.2)

5.5.4.1 Critères considérés

Au cours de cette activité, chaque groupe bien formé, construit à partir des groupes initiaux (comprenant des exigences qui ne sont pas des exigences bien formées) est analysé. Plus précisément, nous nous intéressons cette fois au contenu des exigences de chaque groupe et prenons en compte la définition des états. Ainsi, cette activité fait appel à la fois à des connaissances sur la fonction étudiée (la fonction AD), mais aussi à des connaissances relatives aux deux métiers en jeu. Cette activité est très importante

car elle réduit considérablement le nombre de possibilités à étudier. Pour procéder à cette estimation, nous avons analysé d'abord chaque groupe bien formé obtenu selon les deux critères suivants :

1 ***Critère 1 (Cr1)*** *: Plausibilité du changement d'état* : le changement d'état respecte-t-il la logique de l'enchaînement des états, au vu de la définition de ceux-ci ?

2 ***Critère 2 (Cr2)*** *: Analyse rapide du contenu* : le contenu des exigences (*i.e.* les conditions qu'elles définissent) est-il cohérent avec le changement d'état ?

Les groupes bien formés sont alors classés par plausibilité : *forte, moyenne* ou *faible*.

Nous étudions ensuite la plausibilité « inter-groupes » en construisant des ***squelettes de modèles d'état***. Nous appelons squelette de modèle d'état un graphe d'état sur lequel n'apparait que le libellé des états et les transitions orientées possibles entre ces états, dont les gardes ne sont pas définies. Il ne s'agit pas encore de modèles formels mais leur utilité est grande. En effet, ils fournissent aux experts métier (AMS et pilotes SdF) une première représentation graphique des ensembles d'exigences textuelles, ce qui autorise la détection d'erreurs de spécification, telles que la présence d'états bloquants ou non atteignables, qui peuvent poser problème.

Méthode de construction d'un squelette de modèle d'état

Un squelette de modèle d'état se construit en suivant les étapes suivantes :

1 Représenter la liste d'états attendus, selon la perspective métier considérée ;

2 Pour chaque groupe d'exigences construit à l'issue de l'activité **AFS1.2.1**, choisir un groupe bien formé (résultant de l'activité **AFS1.3.1**) ;

3 Pour chaque groupe bien formé :

   i. Si le groupe bien formé choisi n'a qu'un seul état de départ, alors tracer une transition de l'état de départ à l'état d'arrivée ayant comme garde le nom du groupe bien formé considéré ;

   ii. Sinon, tracer autant de transitions que d'états de départ possibles. Toutes ces transitions contiennent la même garde : le nom du groupe bien formé considéré.

Exemple d'application aux exigences fonctionnelles de comportement retenues :

1 Le schéma **(1)** de la figure 5.6 donne une première représentation graphique de la liste d'états ;

2 La Table 5.6 restitue la combinaison de groupes bien formés choisie pour cet exemple d'application ;

3 Le schéma **(2)** de la figure 5.6 représente le squelette de modèle d'état obtenu avec les groupes bien formés choisis.

		État d'arrivée		
		NDf	**Df**	**PDf**
État de départ	**NDf**	*Ensemble 1*	*Ensemble 2* **GF2.1.3** : *((Not_ available, Activatable), Available)* **GF2.2.4** : *(Off, Not_ available)*	*Ensemble 3* **GF3.1.2** : *(Activatable, Active)*
	Df	*Ensemble 4* **GF4.1.1** : *(Active, Not_ available)*	*Ensemble 5* **GF5.1** : *(Available, Active)*	*Ensemble 6* **GF6.1.1** : *(Available, Not_ available)*
	PDf	*Ensemble 7*	*Ensemble 8*	*Ensemble 9*

TABLE 5.6 – Combinaison possible de groupes bien formés

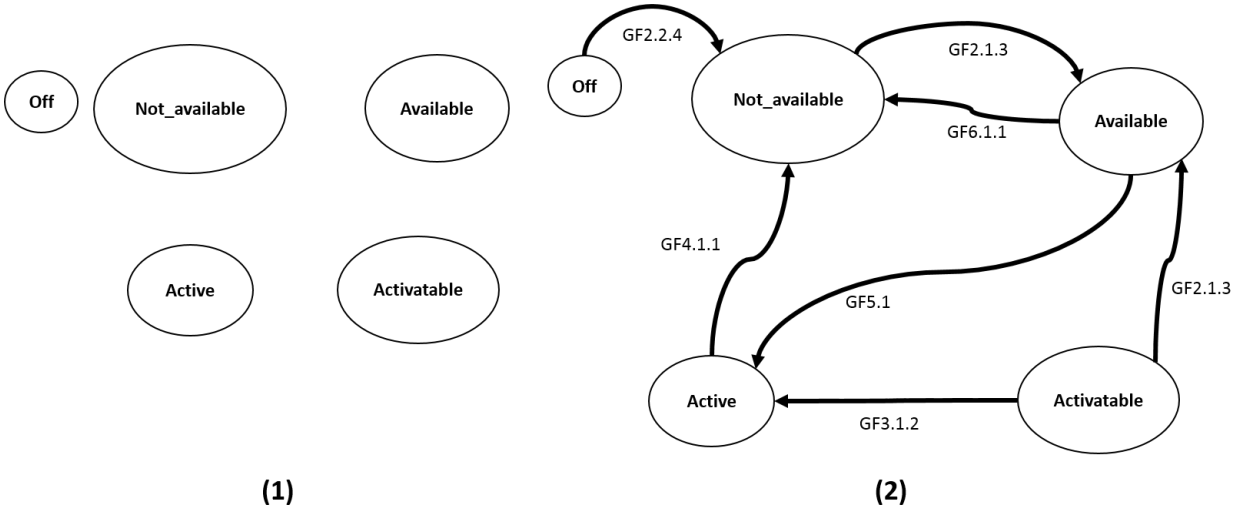


FIGURE 5.6 – Exemple de squelette de modèle d'état fonctionnel

Ainsi, un squelette de modèle d'état donne une représentation d'une combinaison de groupes bien formés. Il y a donc autant de squelettes de modèles d'état que de combinaisons de groupes bien formés possibles.

5.5.4.2 Plausibilité des groupes d'exigences fonctionnelles retenues

Chacun des six groupes d'exigences déterminés est examiné selon les deux critères définis précédemment. Nous donnons ci-dessous un exemple de cette analyse pour les deux groupes de l'ensemble 2 (voir la Table 5.2) :

— *GF2.1 : état d'arrivée* ***Available*** : D'après la Table 5.4, 15 groupes bien formés sont déductibles à partir de GF2.1. Appliquons à ces groupes les deux critères :

1 *Cr1* : l'état *Available* est défini comme un état atteignable lorsqu'un certain nombre de conditions environnementales sont remplies. *A priori*, les trois états les plus probables pour l'état de départ de ce groupe sont *Not_ available*,

Activatable et *Active*. En effet, étant donné que l'état *Not_ available* est déterminé justement pour être le premier état dans lequel se trouve la fonction Supervision AD lorsqu'elle est alimentée, il apparaît donc très peu probable que l'état de départ soit l'état *Off* ;

2 *Cr2* : en étudiant le contenu des exigences du groupe GF2.1, aucune ne fait référence à une alimentation électrique, l'état *Off* comme état de départ semble d'autant moins plausible. Par ailleurs, le contenu d'autres exigences (appartenant au groupe GF4.1) détermine des conditions plus précises pour sortir de l'état *Active*. Ainsi, il est moins probable que l'état *Active* soit un état de départ.

Résultats : deux états de départ de **plausibilité forte** : *Not_ available* et *Activatable*, un état de départ de **plausibilité moyenne** : *Active* et un état de départ de **plausibilité faible** : *Off*. Ceci nous donne donc : 3 groupes bien formés de plausibilité forte, 7 de plausibilité moyenne et 15 de plausibilité faible. Cet ensemble d'hypothèses d'interprétation est noté **HIF2.1** (**F** pour **F**onctionnel). Par exemple, pour la première hypothèse de cet ensemble, HIF2.1.1, l'état de départ du groupe GF2.1 est *Not_ available*. Ce groupe bien formé est alors noté **GF2.1.1** ;

— *GF2.2 : état d'arrivée* ***Not_ available*** : de même que pour le groupe précédent, 15 groupes bien formés sont possibles. On leur applique également les deux critères :

1 *Cr1* : l'état *Not_ available* est défini comme le premier état à atteindre après l'alimentation de la fonction AD. L'état le plus probable pour l'état de départ de ce groupe semblerait être *a priori* l'état *Off*. Il peut néanmoins être un autre état, dans lequel la fonction Supervision AD se retrouve afin, par exemple, d'être réinitialisée ;

2 *Cr2* : en étudiant le contenu des exigences du groupe GF2.2, il apparaît en effet qu'aucune ne fait référence à une alimentation électrique, ainsi l'état *Off* est finalement peu plausible. Les trois autres états restants ont, à première vue, la même plausibilité. Toutefois, nous attribuons une plausibilité plus faible à l'état de départ *Active* étant donné qu'il existe des conditions plus précises pour sortir de cet état. De ce fait, toutes les combinaisons d'états de départ comprenant l'état *Active* sont moins probables.

Résultats : deux états de départ de **plausibilité forte** : *Available* et *Activatable*, un état de départ de **plausibilité moyenne** : *Active* et un état de **plausibilité faible** : *Off*. Il en résulte 3 groupes bien formés de plausibilité forte, 7 de plausibilité moyenne et 15 de plausibilité faible. Cet ensemble d'hypothèses est noté **HIF2.2**.

Le même type de raisonnement a été mené pour les autres groupes d'exigences et il en a résulté (seules les possibilités de plausibilité forte et moyenne sont citées) :

— *GF3.1* : un couple (état de départ, état d'arrivée) de plausibilité forte ((*Off*, *Not_ available*)) (**HIF3.1.1**) ;

— *GF4.1* : un état d'arrivée de plausibilité forte : *Not_ available* (**HIF4.1.1**) ;

— *GF6.1* : un état d'arrivée de plausibilité moyenne : *Not_ available* (**HIF6.1.1**).

À partir de la détermination de ces plausibilités, groupe par groupe, nous passons à une analyse inter-groupes, appuyée par les squelettes de modèles d'état. Pour ce faire, nous

appliquons la méthode de construction d'un squelette de modèles d'état en considérant les groupes bien formés de plausibilité moyenne et forte. Les neuf squelettes de modèles d'état de plausibilité maximale par groupe sont représentés en annexe (figure 10.1). Sur la figure 5.7, sont illustrés les squelettes pertinents sur lesquels s'appuient les réflexions qui vont suivre.

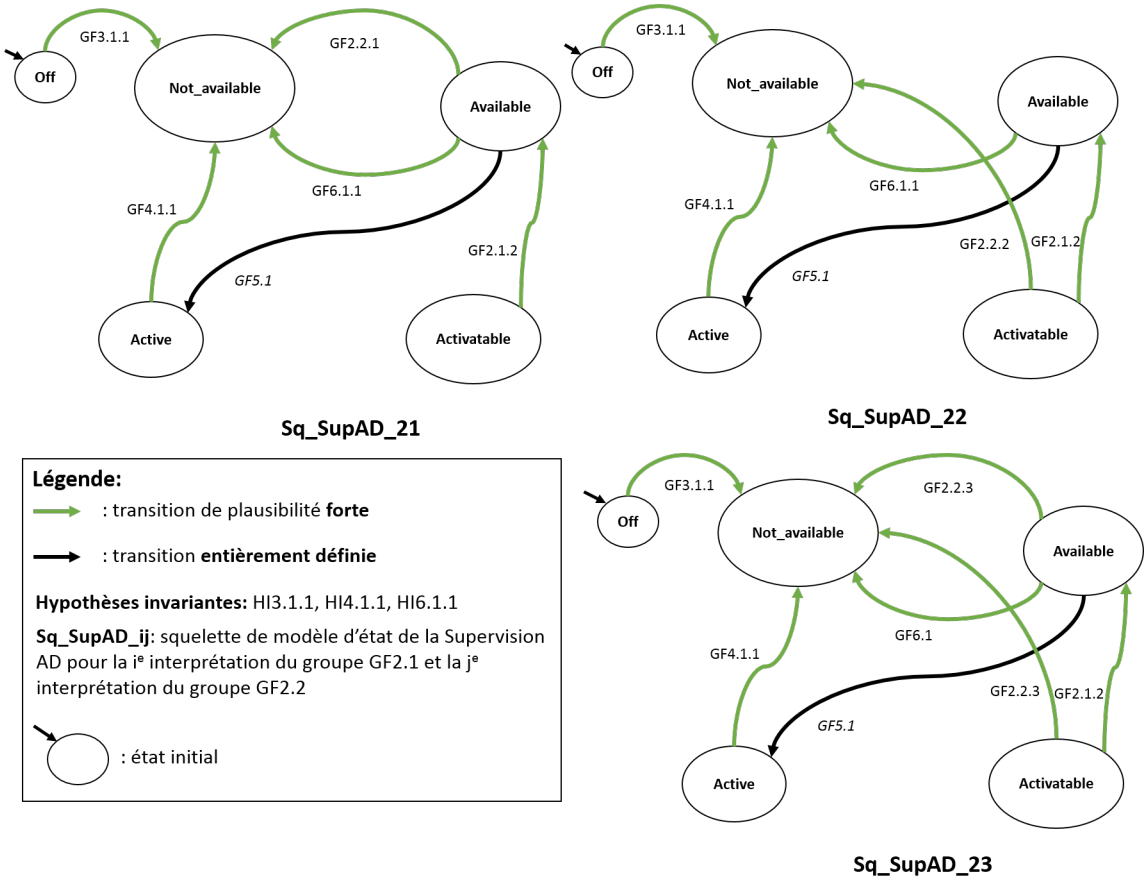


FIGURE 5.7 – Squelettes de modèles d'état fonctionnels bloquants

Étant donné que l'état initial de l'automate est l'état *Off*, il est clair, sur cette figure, que l'état *Not_ available* est un état bloquant pour chacun des squelettes de modèle. Ceci correspond à la situation pour laquelle le groupe **GF2.1** n'inclut pas dans ses états de départ l'état *Not_ available*. Ainsi cet état doit apparaître dans les états de départ possibles de ce groupe. Par exemple, on compte, pour le groupe GF2.1, initialement 3 groupes bien formés de plausibilité forte :

1 État de départ : *Not_ available* (**GF2.1.1**) ;

2 État de départ : *Activatable* (**GF2.1.2**) ;

3 États de départ : *Not_ available* ou *Activatable* (**GF2.1.3**).

Or, comme **GF2.1.2** ne comprend pas l'état *Not_ available* dans ses états de départ, ce groupe est conséquemment éliminé. Il reste donc **2** groupes bien formés de plausibilité forte. En appliquant le même raisonnement aux groupes de plausibilités moyenne et

faible, le nombre de possibilité est réduit à **4** pour la plausibilité moyenne et **7** pour la plausibilité faible.

Par ailleurs, l'état *Activatable* n'est atteignable sur aucun squelette de modèle d'état de plausibilité maximale (voir figure 10.1 en annexe). L'entrée dans cet état n'est en effet pas défini explicitement, mais seulement de façon « négative » au travers du groupe GF6.1 (ceci signifie que l'état d'arrivée de ce groupe ne doit pas être *Activatable*, c'est un état *Partiellement Défini*, voir paragraphe 5.4.2). Nous avons, en considérant le reste des exigences, supposé que l'opposé des conditions contenues dans les exigences du groupe GF6.1 devaient être remplies pour passer de l'état *Available* à l'état *Activatable*. Nous noterons cette hypothèse d'interprétation **HIF6.1.2**. Enfin, au vu des conditions définies dans l'exigence constituant le groupe GF5.1, et en raison de la définition de l'état *Activatable* (état dans lequel la fonction AD est proposée au conducteur, voir paragraphe 5.3.1), nous avons estimé qu'il était fortement probable que l'état de départ du groupe GF5.1 ne soit pas l'état *Available*, comme ceci était initialement spécifié dans les exigences de ce groupe, mais l'état *Activatable* (**HIF5.1.1**).

La Table 5.7 restitue le nombre d'interprétations restantes de plausibilité forte et moyenne. Le nombre de combinaisons possibles est ainsi réduit drastiquement par l'activité **AFS1.3.2**. En effet, nous avions vu dans le paragraphe 5.5.3, que le nombre total de combinaisons possibles (en considérant les groupes d'exigences) s'élevaient à **216 000**. Après analyse cette quantité chute donc à un maximum de **28** combinaisons de groupes bien formés. Les 6 squelettes de modèles d'état de plausibilité forte sont représentés en annexe, sur la figure 10.2.

	Groupe d'exigences initiaux	GF2.1	GF2.2	GF3.1	GF4.1	GF5.1	GF6.1	**Nombre de combinaisons possibles**
Groupes bien formés	**Plausibilité forte**	2	3	1	1	1	1	**6**
	Plausibilité moyenne et forte	4	7	1	1	1	1	**28**

TABLE 5.7 – Combinaisons de groupes bien formés de plausibilité moyenne et forte

5.5.4.3 Plausibilité des groupes d'exigences de SdF retenues

L'analyse des groupes d'exigences de SdF retenues est plus simple car moins d'états sont considérés. L'application du critère **Cr1** suffit à déterminer les plausibilités et il n'est pas nécessaire d'étudier le contenu des exigences (**Cr2**). Rappelons que pour ces exigences, nous avons en fait 2 groupes pour la sous-fonction Main_AD, 3 groupes pour Sub_AD et 1 pour AD-3 (voir Table 5.3). En outre, tous ces groupes appartiennent au même ensemble, l'ensemble 2. Ceci signifie que l'état d'arrivée est défini mais que l'état de départ ne l'est pas. Pour la fonction Main_AD :

— *GSM2.1 : état d'arrivée* ***Off*** : cet état étant atteint suite à une défaillance de la fonction de Contrôle AD (figure 2.13), nous avons estimé qu'*a priori*, cet événement pouvait survenir à n'importe quel moment. C'est pourquoi les deux états de départ possibles, que sont l'état *Active* et *MRM* ont une **plausibilité forte**.

Ceci nous donne donc 3 groupes bien formés de plausibilité forte (ensemble d'hypothèses **HISM2.1**) ;

— *GSM2.2 : état d'arrivée* ***MRM*** : cet état vise à mettre le véhicule en sécurité, la fonction Control_Main_MRM est alors activée et doit envoyer les commandes de trajectoire adéquates. La fonction de contrôle doit donc être au préalable opérationnelle, pour activer la fonction Control_Main_MRM. En conséquence, l'état *Off* a une plausibilité faible d'être un état de départ et seul l'état *Active* est un candidat crédible. L'état *Active* est donc l'état de départ le plus probable. Il en résulte un groupe bien formé de plausibilité forte (**HISM2.2.1**).

Pour la fonction Sub_AD :

— *GSS2.1 : état d'arrivée* ***Off*** : cet état est atteint suite à une défaillance de la fonction de Contrôle AD. Pour les mêmes raisons que la fonction Main_AD, les états *Standby* et *MRM* ont une **plausibilité forte**. Ceci nous donne donc 3 groupes bien formés de plausibilité forte (ensemble d'hypothèses **HISS2.1**) ;

— *GSS2.2 : état d'arrivée* ***MRM*** : de même que pour la fonction Main_AD, seul l'état *Standby* est un candidat crédible. L'état *Standby* est donc l'état de départ le plus probable et un groupe bien formé de *plausibilité forte* est mis en évidence (**HISS2.2.1**) ;

— *GSS2.3 : état d'arrivée* ***Standby*** : la définition de l'état *Standby* suppose que la fonction de contrôle n'ait pas subi de défaillance au préalable. Ainsi, le seul état de départ probable est donc l'état *MRM*, ce qui donne un groupe bien formé de *plausibilité forte* (**HISS2.3.1**).

Étant donné que fonction AD-3 n'a que deux états, le seul état de départ de plausibilité forte est l'état *Standby*. Il en résulte un seul groupe bien formé de plausibilité forte (**HIS32.1.1**). Au même titre que la fonction AD globale, l'ensemble des squelettes de modèles d'état de plausibilité forte a été construit. Ils sont restitués sur la figure 5.8.

En définitive, nous constatons que le gain, en termes de combinaisons à analyser en priorité, est moins important pour ce qui concerne les exigences de SdF retenues. En effet, on dénombre 3 combinaisons de groupes bien formés de plausibilité forte pour les fonctions Main_AD et Sub_AD alors que leurs quantités s'élèvent à 9 et 27 combinaisons pour, respectivement, la fonction Main_AD et Sub_AD, sans considérer les plausibilités. Par contre, le gain considérable en termes de nombre de combinaisons à analyser est obtenu lors de la formation des groupes. En effet :

— *Pour la fonction Main_AD* : GSM2.1 et GSM2.2 contiennent respectivement 14 et 13 exigences. Ainsi, en reprenant le mode de calcul détaillé dans le paragraphe 5.5.3, $\mathbf{7{,}63.10^{12}}$ ($=3^{27}$) combinaisons d'exigences bien formées sont possibles sans opérer à un regroupement des exigences, contre seulement **9** ($= 3^2$) combinaisons de groupes en procédant au regroupement ;

— *Pour la fonction Sub_AD* : GSS2.1, GSS2.2 et GS2.3 contiennent respectivement 13, 14 et 1 exigences. Ceci donne un total de $\mathbf{2{,}29.10^{13}}$ ($= 3^{28}$) combinaisons possibles sans opérer à un regroupement des exigences, contre **27** ($= 3^3$) en procédant au regroupement.

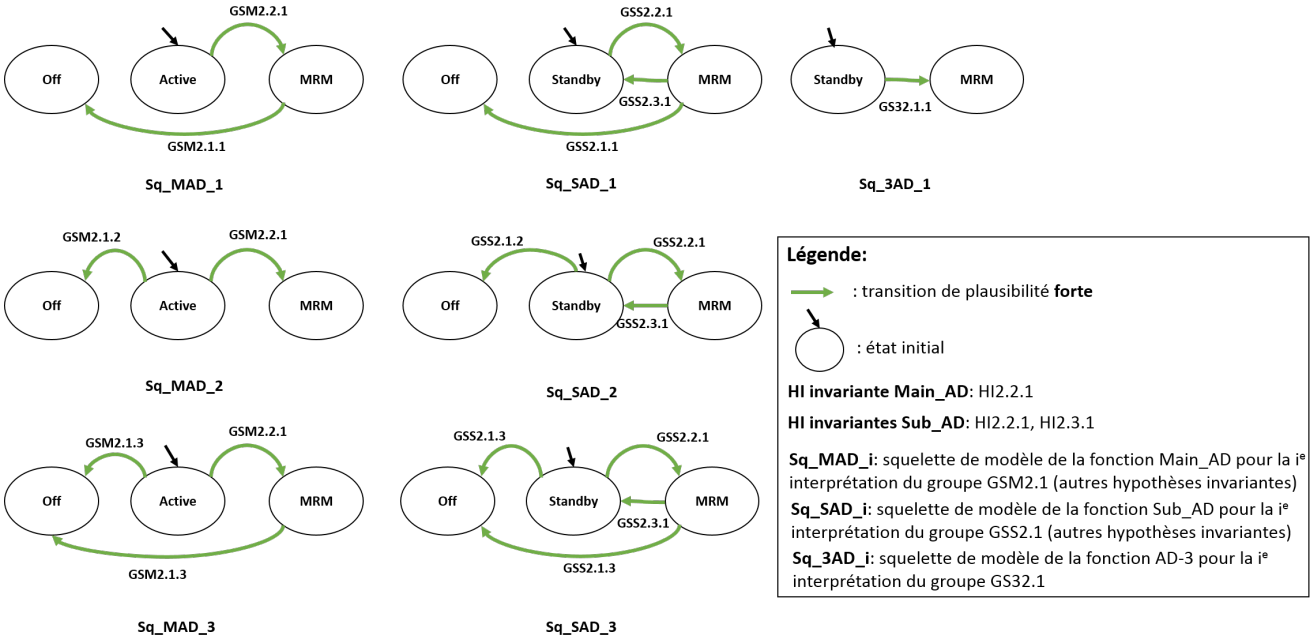


FIGURE 5.8 – Squelettes de modèles d'état de sûreté de plausibilité forte

5.5.4.4 Conclusions

L'analyse menée durant l'activité **AFS1.3.2** met en évidence les possibilités de formalisation les plus crédibles au regard de la logique de la définition des états, d'une part, et du contenu des exigences, d'autre part. Si ce travail est bien lié à la fonction AD analysée et ses spécificités, il n'en reste pas moins que les trois principes suivants semblent généralisables à d'autres études :

1 *Séquencement des états* : des règles simples, relatives à des séquences particulières d'états peuvent être édictées. Par exemple, du point de vue des AMS, une règle pourrait être que l'état *Active* ne peut pas être atteint depuis un autre état que l'état *Activatable* ;

2 *Atteignabilité des états* : tous les états considérés doivent être atteignables depuis l'état initial de l'automate ;

3 *Présence d'états bloquant* : pour certains points de vue métier (dans notre cas celui des AMS), les automates ne doivent pas être bloquant.

À l'instar de l'activité **AFS1.2**, consacrée à la formation de groupes d'exigences, l'activité **AFS1.3.2** a également pour effet de réduire le nombre d'interprétations. La Table 5.8 illustre ce propos en rappelant l'effet qu'ont ces deux activités sur le nombre potentiel de combinaisons (donc de squelettes de modèles d'état) à analyser.

Nombre de combinaisons possibles	Sans regroupement	Avec regroupement	Plausibilité moyenne et forte	Plausibilité forte
EFC	$6{,}43.10^{55}$	216 000	28	6
EFSM (Main_AD)	$7{,}63.10^{12}$	9	3	3
EFSS (Sub_AD)	$2{,}29.10^{13}$	27	3	3

TABLE 5.8 – Récapitulatif des réductions du nombre de combinaisons possibles

5.5.5 Avis de l'expertise relatif à la plausibilité (AFS2.3)

5.5.5.1 Conclusions des AMS et des pilotes SdF

Au terme de l'activité **AFS1.3.2**, les différentes hypothèses d'interprétation retenues associées aux plausibilités des interprétations des groupes sont soumises et confrontées aux deux expertises (voir figure 5.5). Au cours de cette activité, les deux acteurs métier donnent, ou non, leur assentiment concernant ces hypothèses. Voici l'issue de cette concertation pour notre cas d'étude :

— *Point de vue des AMS* : les AMS ont retenu comme possibles tous les groupes bien formés de plausibilité forte. De ce fait, 6 combinaisons de groupes bien formés, soit 6 squelettes de modèles d'état, sont encore possibles ;

— *Point de vue des pilotes SdF* : trois points ont été soulevés :

1 Pour les groupes GSM2.1 (Main_AD) et GSS2.1 (Sub_AD) ayant tous deux pour état d'arrivée *Off*, les pilotes SdF ont éliminé la possibilité que l'état *MRM* soit un état de départ. Effectivement, lorsque la sous-fonction Main_AD (resp. Sub_AD) est dans l'état *MRM*, la fonction Control_Main_MRM (resp. Control_Sub_MRM) envoie des commandes de trajectoire visant à mettre le véhicule autonome en sécurité. Il est exigé que, durant ce laps de temps, la fonction Control_Main_MRM (resp. Control_Sub_MRM) ne puisse pas subir un dysfonctionnement. Par conséquent, il n'est pas possible de passer de l'état *MRM* à l'état *Off*. Ainsi, que ce soit pour la sous-fonction Main_AD, Sub_AD et AD-3, finalement 1 seul groupe bien formé a été retenu par les pilotes SdF ;

2 Concernant le groupe GSS2.3 (Sub_AD), les pilotes SdF ont décidé que l'exigence contenue dans ce groupe (**EFSS 10**, paragraphe 5.3.1) était en fait une exigence d'état, devant être vérifiée dans l'état *Standby* de la fonction Sub_AD. En conséquence, le groupe d'exigences GSS2.3 sort du périmètre de l'analyse ;

3 Pour le groupe GS32.1 (AD-3), les pilotes SdF ont relevé, au vu des squelettes de modèles de la figure 5.8, l'absence de l'équivalent d'un état *Off* pour la fonction AD-3. Ceci signifierait que la fonction Control_3_Standby ne peut jamais dysfonctionner. Il est donc clair que cette modélisation n'est pas acceptable : un état *Off*, atteint suite à la défaillance de la fonction Control_3_Standby doit être ajouté au modèle et spécifié. Ce point est important car il implique la modification de la liste d'états attendus initiale de la fonction AD-3 (voir paragraphe 5.3.1). En outre, une exigence

a été ajoutée pour rendre ce nouvel état atteignable (uniquement, comme pour les deux autres sous-fonctions et pour les mêmes raisons, depuis l'état *Standby*). Par ailleurs il a également été conclu que la défaillance de la fonction Control_3_Standby devait causer l'entrée dans l'état *MRM* de la fonction Main_AD. Ainsi, une nouvelle exigence, ajoutée au groupe bien formé GSM2.2.1, spécifie également l'entrée de la fonction Main_AD dans l'état *MRM* en cas de défaillance de la fonction Control_3_Standby.

Les 6 squelettes de modèles d'état fonctionnels retenus ainsi que le modèle d'état de sûreté sélectionné figurent en annexe (respectivement figures 10.2 et 10.3). La Table 5.9 récapitule, pour les exigences des deux types, les groupes bien formés retenus, associés aux hypothèses d'interprétation formulées.

Le deuxième point relevé par les pilotes SdF montre en pratique le caractère itératif de notre démarche. Effectivement, nous voyons que le processus de formalisation proposé, non seulement modifie le contenu des exigences (en complétant ces dernières) mais aussi le corpus même des exigences (ajout d'exigences). De plus, la contribution **C1** (voir figure 4.2) relative à la mise en évidence de conceptions alternatives, est également illustrée. Nous avons, dans un premier temps, montré qu'un nombre très élevé de possibilités d'interprétations (puis de squelettes de modèles d'état en découlant) aurait été à analyser en essayant de traiter de manière brute toutes les exigences une à une. Ensuite, à l'aide de **regroupements judicieux d'exigences**, le nombre de possibilités a pu être diminué. Enfin, en **caractérisant la plausibilité** de ces différentes possibilités, leur nombre a pu être encore réduit. Néanmoins, malgré cette réduction importante du nombre de possibilités, plusieurs spécifications concurrentes subsistent (6 squelettes de modèles d'état du point de vue des AMS). Ceci met donc en lumière les **différentes formalisations possibles et crédibles**, initialement masquées par les ambigüités et les implicites des exigences exprimées en langage naturel.

5.5.5.2 Mise à jour des énoncés des exigences

À l'issue du regroupement des exigences, de la complétion des groupes et de la validation de ces activités par les deux expertises métier, il convient de mettre à jour les énoncés des exigences retenues. Effectivement, les exigences sélectionnées subissent, selon les deux points de vue métier, des modifications car elles sont complétées par des informations d'état, qui doivent donc être ajoutées aux énoncés de ces exigences. Après ces différentes transformations, les exigences sélectionnées prennent la forme d'**exigences bien formées** (voir Table 5.1).

Nous restituons ci-dessous les énoncés mis à jour deux exemples d'exigences fournis dans le paragraphe 5.3.1. Les notations suivantes sont adoptées pour ces exigences :

- *EFBFi_j* pour l'i^e Exigence Fonctionnelle de comportement Bien Formée découlant de la j^e hypothèse d'interprétation ;
- *ESMBFi_j* pour l'i^e Exigence fonctionnelle de SdF Bien Formée allouée à la fonction **M**ain_AD et découlant de la j^e hypothèse d'interprétation ;
- *ESSBFi_j* pour l'i^e Exigence fonctionnelle de SdF Bien Formée allouée à la fonction **S**ub_AD et découlant de la j^e hypothèse d'interprétation ;
- *ES3BFi_j* pour l'i^e Exigence fonctionnelle de SdF Bien Formée allouée à la fonction AD-**3** et découlant de la j^e hypothèse d'interprétation.

Groupe d'exigences initial	Éléments initialement définis	Hypothèse d'interprétation (HI)	Groupe(s) bien formés retenu(s)
GF2.1	État d'arrivée *Available*	*HIF2.1.1* : état de départ : *Not_ available* *HIF2.1.3* : états de départ : *(Not_ available, Activatable)*	GF2.1.1 GF2.1.3
GF2.2	État d'arrivée *Not_ available*	*HIF2.2.1* : état de départ : *Available* *HIF2.2.2* : état de départ : *Activatable* *HIF2.2.3* : états de départ : *(Available, Activatable)*	GF2.2.1 GF2.2.2 GF2.2.3
GF3.1	État d'arrivée n'est pas *Active*	*HIF3.1.1* : (état de départ, état d'arrivée) = *(Off, Not_ available)*	GF3.1.1
GF4.1	État de départ *Active*	*HIF4.1.1* : état d'arrivée : *Not_ available*	GF4.1.1
GF5.1	Complètement défini *(Available, Active)*	*HIF5.1.1* : état de départ : *Activatable*	GF5.1.1
GF6.1	État de départ : *Available* État d'arrivée n'est pas Activatable	*HIF6.1.2* : état d'arrivée : *Activatable*	GF6.1.2
GSM2.1	État d'arrivée *Off*	*HISM2.1.1* : état de départ : *Active*	GSM2.1.1
GSM2.2	État d'arrivée *MRM*	*HISM2.2.1* : état de départ : *Active*	GSM2.2.1
GSS2.1	État d'arrivée *Off*	*HISS2.1.1* : état de départ : *Standby*	GSS2.1.1
GSS2.2	État d'arrivée *MRM*	*HISS2.2.1* : état de départ : *Standby*	GSS2.2.1
GSS2.3	État d'arrivée *Standby*	*HISS2.3.1* : état de départ : *Standby*	-
GS32.1	État d'arrivée *MRM*	*HIS32.3.1* : état de départ : *Standby*	GS32.1.1
-	-	*HIS35.1.1* : (état de départ, état d'arrivée) = (*Standby, Off*)	GS35.1.1

TABLE 5.9 – Hypothèses d'interprétation retenues pour construire les groupes bien formés

Nous obtenons donc :

— **EFBF6_1** : *« If AD Supervision is in the state **Not_ available** at night without impact on Autonomous Vehicle performance, then AD Supervision shall be in the state **Available** »*

— **ESMBF6_1** : *« If Main_AD is in the state **Active** and in case of a faulty deactivation of the AD function due to dysfunction of Main_AD function, then Main_AD shall switch to the state **Off** »*

L'exigence **EFBF6_1** est donc issue de l'exigence fonctionnelle de comportement re-

tenue **EFC 6**. Puisque cette exigence appartient au groupe d'exigences fonctionnelles **GF2.1** (voir la Table 5.3) et que, la première hypothèse d'interprétation pour former ce groupe est le couple (*état de départ*, *état d'arrivée*) = (*Not_ available*, *Available*) (voir la Table 5.9), l'énoncé restitué ci-dessus pour **EFBF6_1** est obtenu. Un raisonnement similaire est tenu pour l'exigence fonctionnelle de SdF exemple.

5.6 Construction des modèles d'état et vérification (AFS1.4)

5.6.1 Récapitulatif et détails de l'activité AFS1.4

À ce stade, nous avons obtenu, selon les deux points de vue métier, un ensemble de groupes bien formés validés par l'expertise. Or, nous souhaitons construire des modèles d'état et un groupe bien formé peut contenir un ensemble d'états de départ et concerner donc plusieurs transitions. Dans un premier temps, il s'agit de traduire les conditions contenues dans les groupes bien formés en équations logiques (activité **AFS1.4.1** sur la figure 5.10). Puis, il faut établir des relations entre les groupes bien formés et les transitions des modèles d'état comme l'illustre la figure 5.9. Ceci correspond à l'activité **AFS1.4.2**. Il convient de déterminer les opérateurs logiques entre les équations des différentes transitions de groupe. Des *transitions de groupe formalisées* (notion définie ci-après) sont alors obtenues, comme le montre la figure 5.9.

À partir de celles-ci, des modèles d'état dits préliminaires (car non vérifiés) peuvent être construits (**AFS1.4.3**) et des propriétés formelles déterminées (**AFS1.4.4**). Avec l'ensemble de ces éléments, il est possible, lors de l'activité **AFS1.4.5**, de procéder à la vérification des propriétés sur les automates. Ces derniers sont, dans un dernier temps, soumis à la validation de l'expertise appropriée (**AFS2.4**).

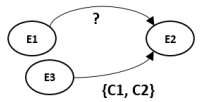
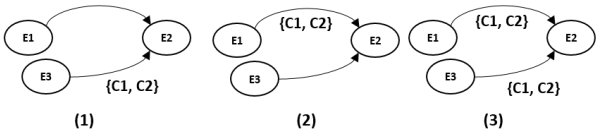
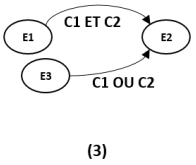
Type de données manipulées	Représentation graphique
Groupe Bien Formé (GBF) **GBF1**: états de départ: ED = {E1, E3}, état d'arrivée: EA = E2, conditions: C = {C1, C2}	**Squelette de modèle d'état** 
Transitions de Groupe (TG) **TG1:** état de départ: ED = E1, état d'arrivée: ED = E3 **TG2**: état de départ: ED = E3, état d'arrivée: ED = E3 **Version 1:** **TG1**: conditions: C = {Ø} **TG2**: conditions: C= {C1, C2} **Version 2:** **TG1**: conditions: C= {C1, C2} **TG2**: conditions: C = {Ø} **Version 3:** **TG1**: conditions: C= {C1, C2} **TG2**: conditions: C= {C1, C2}	**Squelette de modèle d'état** 
Transitions de Groupe Formalisées (TGFo) **TG1** → ET logique, **TG2** → OU logique **Version 3 retenue:** **TGFo1**: état de départ: ED = E1, état d'arrivée: EA = E2, conditions: C = C1 ET C2 **TGFo2**: état de départ: ED = E3, état d'arrivée: EA = E2, conditions: C = C1 OU C2 **Propriétés Formelles (PF)** **PF1**: AG((E1 ET (C1 ET C2)) → AF(E2)) **PF2**: AG((E3 ET (C1 OU C2)) → AF(E2))	**Automate à états finis** 

FIGURE 5.9 – Construction des Transitions de Groupe Formalisées

5.6.2 Traduction du contenu des exigences (AFS1.4.1)

Il s'agit, au cours de cette activité, de traduire les conditions contenues dans les exigences en *équations logiques*. Pour ce faire, nous faisons appel aux deux notions suivantes :

1 *Variable* : terme représentant l'évolution des données mesurables, telles que la vitesse, l'accélération ou la température, et l'état des fonctions de contrôle ;

2 *Paramètre* : valeur particulière (seuil) que peut prendre une variable donnée.

La façon la plus directe de déterminer les variables à partir des exigences est d'associer la ou les conditions contenues dans une exigence retenue à une variable booléenne. Celle-ci sera donc vraie ou fausse selon que la condition est remplie ou non. Cette méthode présente l'avantage de rester très fidèle aux exigences et de ne pas prendre d'hypothèses de modélisation relatives à la définition de variables. Elle autorise une indépendance totale entre toutes les conditions, en dépit néanmoins du sens que ces combinaisons de variables peuvent avoir. Le principal défaut de cette méthode est justement le trop grand nombre de situations modélisables : non seulement l'espace d'état ainsi généré sera certainement difficile à parcourir par un *model checker* et nous risquons d'être confrontés à des problèmes d'explosion combinatoire ; de plus, il est certain que des situations incohérentes seront simulées. Par exemple, lorsqu'une exigence spécifie une action à effectuer lorsque le véhicule est en mouvement (condition

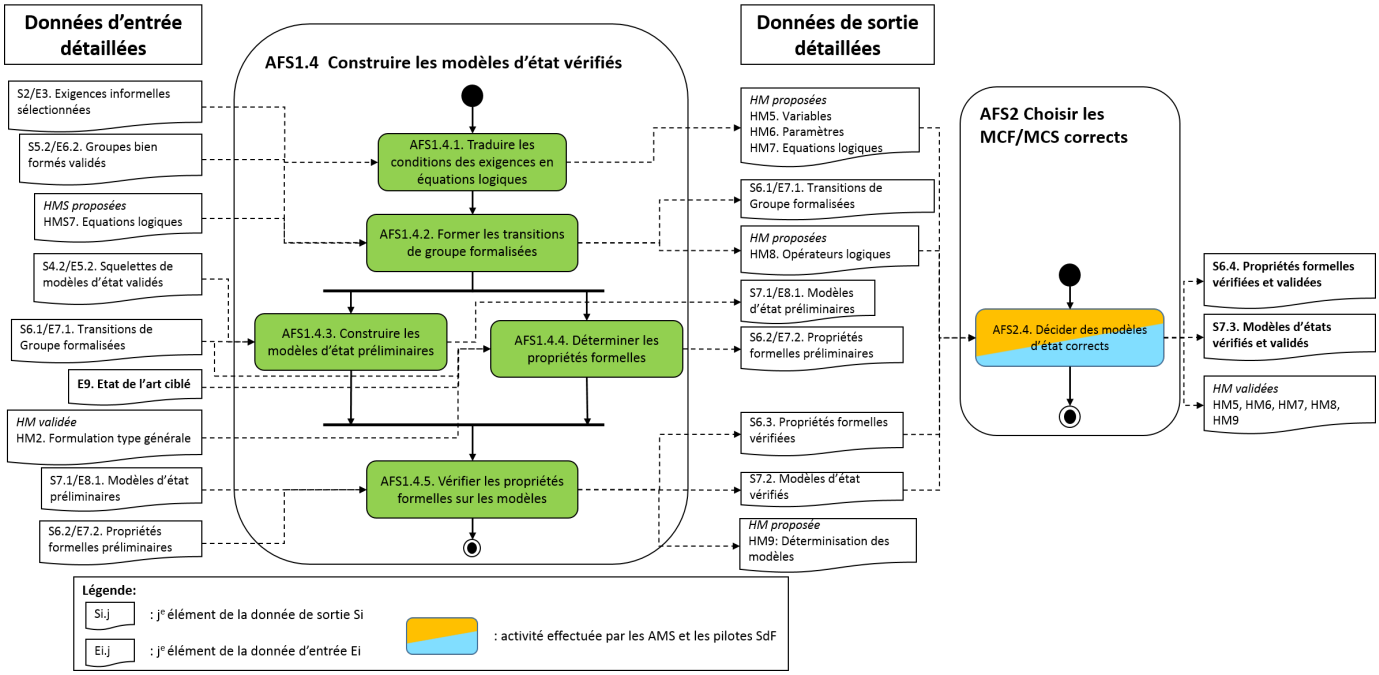


FIGURE 5.10 – Activités de construction des modèles et de vérification (**AFS1.4**)

1) et une autre exigence détermine une action à faire quand le véhicule est immobile (condition 2), la dépendance entre les conditions 1 et 2 est claire.

Il faut par conséquent avoir recours à une analyse plus fine des exigences pour déterminer les variables et les paramètres à partir desquelles les équations logiques seront construites. Nous restituons dans la Table 5.10 les équations logiques associées à deux des exigences sélectionnées (selon les deux points de vue métier), dans le paragraphe 5.3.1. Pour **EFC 8**, deux variables booléennes et un paramètre ont été définis :

- *turn_ind* : cette variable indique l'état des clignotants : activés ou non ;
- *spd_thr* : cette variable indique si la vitesse du véhicule (« val_vEgo ») est en-dessous ou au-dessus d'un seuil (« val_vActivation ») ;
- *val_vActivation* : ce paramètre correspond à la valeur maximale de la vitesse du véhicule pour laquelle la fonction de Conduite Autonome peut être activée.

Pour **EFSS 8**, une variable et deux paramètres ont été spécifiés :

- *s_decel* : cette variable correspond à la commande de décélération du véhicule élaborée par la fonction Control_Sub_Standby ;
- *excess_decel* : ce paramètre correspond à une décélération excessive du véhicule, due à la fonction Control_Sub_Standby ;
- *insuf_decel* : ce paramètre correspond à une décélération insuffisante du véhicule, due à la fonction Control_Sub_Standby.

À chaque exigence est donc associée une équation logique booléenne. La définition de ces variables et de ces paramètres donnent lieu à des hypothèses de modélisation (HM5, pour les variables, et HM6, concernant les paramètres). Les variables définies sont toutes des variables numériques. Elles peuvent être booléennes (par exemple pour

l'absence ou la présence d'une défaillance) ou entières bornées (par exemple 3 valeurs possibles pour la comparaison à deux seuils ordonnés). Par ailleurs, les équations logiques constituent également des hypothèses de modélisation (HM7). En annexe, dans les Tables 10.1 et 10.2, se trouve l'ensemble des variables à la fois pour les exigences fonctionnelles de comportement retenues (52 variables) et pour les exigences de SdF sélectionnées (19 variables).

Id	Enoncé	Variables	Paramètre	Equation logique
EFC 8	« IF AD Supervision is Available AND **(turn indicator is on OR val_vEgo⩾val_vActivation)** THEN AD Supervision shall be not Activatable »	*turn_ind* : Booléen turn_ind==0 ⟺ turn indicator : OFF turn_ind==1 ⟺ turn indicator : ON *spd_thr* : Booléen spd_thr== 0 ⟺ val_vEgo<val_vActivation spd_veh== 1 ⟺ val_vEgo⩾ *val_vActivation*	val_vActivation	(turn_indic==1 OR spd_thr==1)
EFSS 8	« In case of an excessive deceleration due to dysfunction of Sub_AD function, Sub_AD shall switch itself **Off** »	*s_decel* : Entier ∈ [0, 2] s_decel==0 ⟺ decel<insuf_decel s_decel==1 ⟺ decel ∈ [insuf_decel, excess_decel] s_decel== 2 ⟺ decel ⩾ *excess_decel*	excess_decel insuf_decel	(s_decel==2)

TABLE 5.10 – Détermination des variables et des paramètres pour EFC 8 et EFSS 8

La traduction a été faite exigence par exigence. Or, nous analysons les groupes bien formés, construits et choisis précédemment. Un lien doit donc être créé entre les équations logiques et les groupes bien formés. Ce lien se concrétise par la formation des *transitions de groupe formalisées*, qui est l'objet de l'activité **AFS1.4.2**.

5.6.3 Élaboration des modèles préliminaires et des propriétés formelles (AFS1.4.2 à AFS1.4.4)

5.6.3.1 Transitions de groupe formalisées (AFS1.4.2)

Avant d'introduire la notion de transitions de groupe formalisées, il faut définir celle de transition de groupe :

Définition 5.10
Transition de groupe *: une transition de groupe est une agrégation d'exigences qui ne contient que des exigences ayant le* ***même et unique état de départ*** *et le même*

état d'arrivée. Une transition de groupe est donc associée à une transition d'un modèle d'état. ♣

Lien entre les groupes bien formés et les transitions de groupe

Du point de vue des AMS, 20 transitions de groupe sont déterminées (notées TGF pour Transition de Groupe Fonctionnelle). Ces transitions sont listées dans la Table 10.3 en annexe et représentées sur la figure 5.12. La figure 5.11 illustre, en pratique, le lien entre les notions de groupes bien formés et celle de transitions de groupe. Nous considérons dans un premier temps les deux groupes bien formés retenus à partir du groupe initial GF2.1 (GF2.1.1 et GF2.1.3). Puisque ces deux groupes contiennent les mêmes exigences, ils définissent également les mêmes conditions. Ainsi, selon l'Hypothèse d'Interprétation retenue (HI2.1.1 ou HI2.1.3), les conditions contenues dans les transitions de groupe sont différentes. Si HI2.1.1 est choisie, TGF6 comportera (au moins) les conditions de GF2.1 et on ne pourra rien dire de TGF15. Si HI2.1.3 est sélectionnée, les conditions de TGF6 et de TGF15 comprendront, au moins, celles de GF2.1.

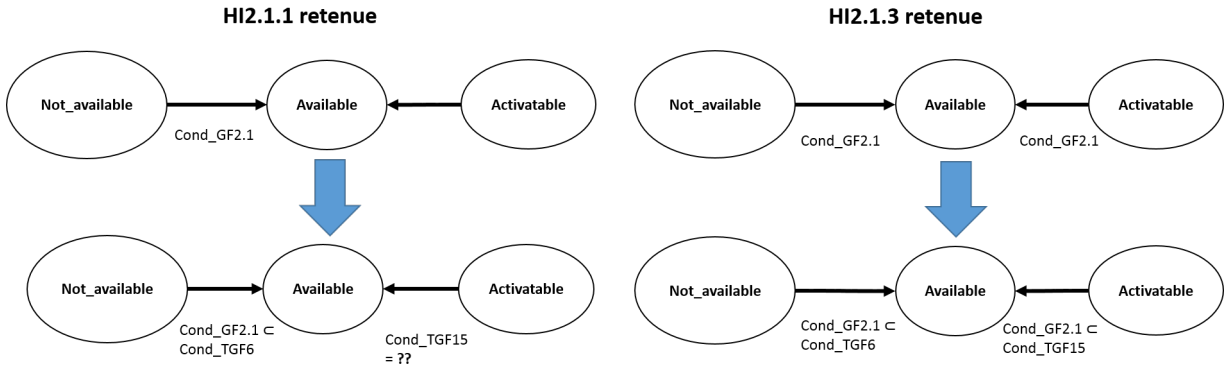


FIGURE 5.11 – Lien entre groupes bien formés et transition de groupe

L'intérêt de la notion de *transition de groupe* est donc de rapprocher les exigences retenues des modèles construits en associant ces transitions de groupe à une unique transition. Enfin, on définit :

Définition 5.11
Transition de groupe formalisée *: une transition de groupe est dite formalisée s'il existe des opérateurs logiques entre toutes les équations logiques associées aux exigences qui la constitue.* ♣

Pour former les transitions de groupe formalisées, trois règles, énoncées ci-après, ont été proposées. Avant de les restituer, nous exposons le principe sous-jacent sur lequel elles reposent, et qui les justifie.

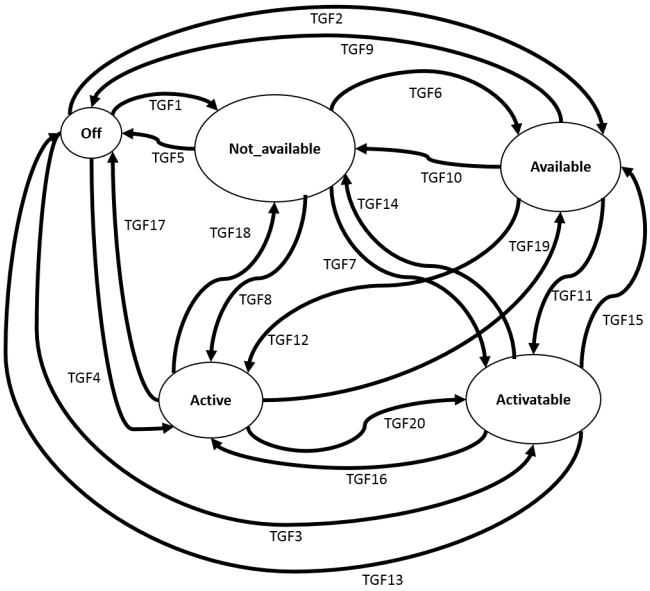


FIGURE 5.12 – Représentation graphique des Transitions de Groupe Fonctionnelles (TGF)

Tant du point de vue des AMS que de celui des pilotes SdF, les phases de fonctionnement les plus risquées sont celles durant lesquelles la conduite est entièrement déléguée à la fonction AD. En effet, une défaillance d'une fonction de contrôle peut alors avoir des conséquences graves. Concernant la modélisation, il s'agit de l'état ***actif***, *i.e.* l'état *Active* pour la Supervision AD globale et la fonction Main_AD et *Standby* pour les fonctions Sub_AD et AD-3. Il convient donc de sécuriser autant que possible l'entrée dans cet état. Pour ce faire, deux cas sont distingués :

1 **Cas 1** : *passage d'un état « plus éloigné » de l'état actif à un état « plus proche » de ce même état* : toutes les conditions de la garde de la transition entre l'état considéré et l'état *actif* doivent être remplies pour que la transition soit franchie ;

2 **Cas 2** : *passage d'un état « plus proche » de l'état actif à un état « plus éloigné » de ce même état* : une seule des conditions de la garde de la transition entre l'état considéré et l'état *actif* suffit à être remplie pour que la transition soit franchie.

Un état A est dit *« plus proche »* de l'état *actif* qu'un état B si un nombre moins élevé de conditions doit être vérifié pour passer de l'état A à l'état *actif* que pour passer de l'état B à l'état *actif*. Nous parlerons par la suite de ***niveau de contrainte d'état*** :

Définition 5.12
Niveau de contrainte d'état *: un état A est dit plus contraint qu'un état B si un nombre moins élevé de conditions doit être rempli pour passer de l'état A à l'état actif que pour passer de l'état B à l'état actif.* ♣

À partir de cette notion, nous pouvons définir celle de *transition de groupe contrainte* :

Définition 5.13
Transition de groupe contrainte *: une transition de groupe est dite contrainte si son état de départ est un état moins contraint que son état d'arrivée.* ♣

Ainsi, une transition de groupe contrainte correspond au **cas 1** identifié précédemment : toutes les conditions doivent être vérifiées au même instant pour franchir cette transition. À l'inverse, lorsque la transition de groupe est non contrainte, ceci correspond au **cas 2** pour lequel il suffit qu'une condition soit remplie pour franchir la transition. Enfin, les règles relatives à la construction des *transitions de groupe formalisées* font appel à une dernière définition :

Définition 5.14
Equations logiques indépendantes *: deux équations logiques sont dites indépendantes si elles n'ont aucune variable commune.* ♣

Les considérations précédentes permettent d'établir les règles suivantes, servant à construire les transitions de groupes formalisées :

(a) L'opérateur **OU** logique doit être placé entre toutes les équations logiques associées aux exigences constituant une transition de groupe non contrainte ;

(b) L'opérateur **ET** logique doit être placé entre toutes les équations logiques indépendantes associées aux exigences d'une transition de groupe contrainte ;

(c) L'opérateur **OU** logique doit être placé entre toutes les équations logiques non indépendantes associées aux exigences d'une transition de groupe contrainte.

Du point de vue des AMS, il est possible, à partir des définitions données dans la liste d'états attendus (voir paragraphe 5.3.1), de classer les états de l'état le moins contraint au plus contraint : *Off*, *Not_available*, *Available*, *Activatable*, *Active*. De cette manière les transitions de groupe contraintes et non contraintes sont déterminées, ce qui est restitué sur la Table 10.3 en annexe. Par exemple, la transition de groupe TGF6 est contrainte tandis que TGF15 est non contrainte (voir figure 5.11). Les conditions des deux transitions de groupe sont données par le même groupe bien formé : GF2.1. Lorsque l'hypothèse d'interprétation HI2.1.3 est choisie pour GF2.1 (voir la Table 5.9), les transitions de groupe formalisées issues des transitions de groupe TGF6 et TGF15, notées TGFFi (i^{e} Transition de Groupe Fonctionnelle Formalisée) :

— *TGFF6* : ((road_sec==0) **AND** ((veh_pos==0 AND veh_dir==1) OR (veh_pos==6)) **AND** (forec_AD_dur==1) **AND** (veh_loca==0 OR veh_loca==1) **AND** (lan_width==1) **AND** ((weat_cond==0) OR (weat_cond==1) OR (weat_cond==2) OR (weat_cond==3) OR (weat_cond==4) OR (weat_cond==5) OR (weat_cond==6))) ;

— *TGFF15* : ((road_sec==0) **OR** ((veh_pos==0 AND veh_dir==1) OR (veh_pos==6)) **OR** (forec_AD_dur==1) **OR** (veh_loca==0 OR veh_loca==1) **OR** (lan_width==1) **OR** ((weat_cond==0) OR (weat_cond==1) OR (weat_cond==2) OR (weat_cond==3) OR (weat_cond==4) OR (weat_cond==5) OR (weat_cond==6))).

Chaque équation logique (terme entre parenthèses) correspond à une des 13 exigences du groupe GF2.1. Il s'agit donc bien des mêmes équations logiques pour TGFF6 et TGFF15, mais l'opérateur logique **ET** est utilisé entre les équations indépendantes de TGFF6 tandis que l'opérateur **OU** logique est employé pour le groupe TGFF15.

Pour chacun des 6 squelettes de modèles d'état retenus, les transitions de groupe formalisées auront une expression particulière. Dans le cas où nous choisissons les hypothèses d'interprétation HI2.1.3 pour GF2.1 et HI2.2.3 pour GF2.2, nous obtenons que les transitions de groupe formalisées suivantes :

— TGFF6 (notée TGFF6_33, pour la 3^{e} interprétation de GF2.1 et de GF2.2) contient les équations logiques associées à GF2.1 et un ET logique doit être placé entre ces équations indépendantes car TGF6 est contrainte ;

— TGFF15 (notée TGFF15_33) contient les équations logiques associées à GF2.1 et un OU logique doit être placé entre les équations car TGF15 est non contrainte ;

— TGFF10 (notée TGFF10_33) contient les équations logiques associées à GF2.2 et un OU logique doit être placé entre les équations car TGF10 est non contrainte ;

— TGFF14 (notée TGFF14_33) contient les équations logiques associées à GF2.2 et un OU logique doit être placé entre les équations car TGF14 est non contrainte ;

— TGFF1 (notée TGFF1_33) contient les équations logiques associées à GF3.1 et un ET logique doit être placé entre les équations indépendantes car TGF1 est contrainte ;

— TGFF16 (notée TGFF16_33) contient les équations logiques associées à GF5.1 et un ET logique doit être placé entre les équations indépendantes car TGF16 est contrainte ;

— TGFF18 (notée TGFF18_33) contient les équations logiques associées à GF4.2 et un OU logique doit être placé entre les équations car TGF18 est non contrainte ;

— TGFF11 (notée TGFF11_33) contient les équations logiques associées à GF6.1 et un ET logique doit être placé entre les équations indépendantes car TGF11 est contrainte ;

Selon la perspective des pilotes SdF, les états « actifs » (*Active* pour Main_AD, *Standby* pour Sub_AD et AD-3) sont moins contraints que les états *Off*. La détermination des transitions de groupe contraintes et non contraintes est également fournie en annexe, dans la Table 10.4. Dans ce cas, toutes les transitions de groupe sont non contraintes, il faut donc placer l'opérateur OU logique entre les équations logiques associées aux transitions de groupe, comme l'illustre l'exemple ci-dessous (TGSFM pour Tansition de Groupe de Sûreté Formalisée relative à la fonction Main_AD) :

— *TGSFM3* : ((m_accel==2) **OR** (m_accel==0) **OR** (m_decel==2) **OR** (m_decel==0) **OR** (m_lat_accel==2) **OR** (m_lat_accel==0) **OR** (m_acti==1) **OR** (m_deac==1) **OR** (m_stay==1) **OR** (m_notif==1) **OR** (m_visib==1) **OR** (m_visib==2) **OR** (m_info==2) **OR** (m_info==1))

5.6.3.2 Élaboration des modèles d'état préliminaires (AFS1.4.3)

Les modèles d'état préliminaires doivent être vérifiés formellement, par *model checking*. En outre, une représentation graphique doit être produite pour être soumise

aux deux expertises métier. Dans cette optique, nous avons choisi l'outil de *model checking* **UPPAAL**[2] pour différentes raisons. En premier lieu, UPPAAL fournit une preuve formelle qu'un modèle comportemental est conforme à des propriétés données. Qui plus est, cet outil offre une vue graphique, facilitant la communication auprès des ingénieurs concernés. Une distinction claire est également opérée entre les modèles d'état, représentés dans l'onglet *« Éditeur »* et les propriétés, reportées dans l'onglet *« Vérifieur »*. Ceci était en accord avec notre démarche. Au-delà de la vérification formelle, il est également possible de procéder à des simulations, qui donnent une première opinion sur le comportement des modèles. Enfin, la capacité d'UPPAAL à modéliser des automates communicants a été la dernière raison de ce choix. Effectivement, les trois sous-fonctions Main_AD, Sub_AD et AD-3 communiquent entre elles, ce qui provoque des changements d'état. Par ailleurs, UPPAAL est également un outil reconnu (Pétin et al. 2010 ; Liu et Zhu 2011 ; Apvrille et Becoulet 2012 ; Marinescu et al. 2014).

Nous donnons ci-dessous une définition formelle des objets manipulés dans l'outil UPPAAL dans le but de préciser quels objets nous construirons et analyserons par la suite. Dans le cas général, les automates modélisés dans UPPAAL sont des tuplets (L, l_0, V, A, E, I) tels que (Behrmann, A. David et Larsen 2004) :

- L est l'ensemble des places ;
- $l_0 \in L$: la place initiale ;
- V : l'ensemble des variables ;
- A : l'ensemble des actions ;
- E : l'ensemble des arcs tel que $E \subseteq L \times A \times B(V) \times 2^V \times L$;
- I assigne des invariants aux places, tel que $I : \rightarrow B(V)$.

Dans notre cas d'étude, nous utilisons les variables de type *entier* et *booléen* pour déterminer les variables et paramètres, tels que définis précédemment.

Les *Actions* correspondent aux *canaux de communication (channel)* employés pour modéliser les communications entre les sous-fonctions de Supervision : Main_AD, Sub_AD et AD-3. Les gardes des transitions sont constituées d'équations logiques, associées aux transitions de groupe formalisées, et appartenant à $B(V)$. Enfin, pour mettre à jour, si nécessaire, la valeur de certaines variables, nous usons de la fonctionnalité d'UPPAAL dédiée (la valeur des variables peut être mise à jour aussi bien lors du franchissement d'une transition que lors de l'entrée dans un état). Dans nos modèles, les arcs de E sont donc dotés d'actions (appartenant à A), de *gardes* (appartenant à $B(V)$) et de mises à jour de variables. Une *place* correspond à un état. Par ailleurs, nous n'avons ni considéré les invariants, ni l'aspect temporisé. En conséquence, les objets que nous manipulons sont de la forme suivante : (L, l_0, V, A, E).

La figure 5.13 illustre deux des 6 modèles d'état préliminaires de plausibilité forte obtenus du point de vue des AMS. L'automate situé à droite sur cette figure, nommé *SupAD_33*, est issu du squelette de modèle d'état *Sq_SupAD_33* (voir figure 10.2 en annexe). Pour ce modèle, les gardes des transitions correspondent aux transitions de groupe formalisées déterminées précédemment. Les autres modèles préliminaires de plausibilité forte sont exposés sur la figure 10.4 en annexe.

2. http ://www.uppaal.org/

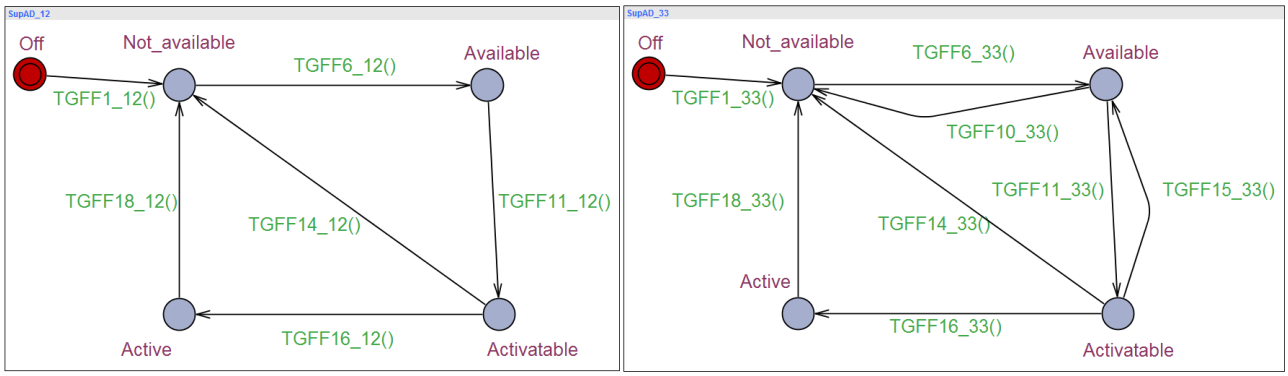


FIGURE 5.13 – Deux modèles d'état fonctionnels préliminaires de plausibilité forte

Selon la perspective de la SdF, rappelons que nous n'avons retenu qu'une alternative de squelette de modèle d'état (un squelette de modèle par sous-fonction). La façon de modéliser se distingue en revanche de celle que nous avons utilisée du point de vue des AMS car nous devons traiter trois automates qui communiquent entre eux. Nous exploitons donc les capacités d'UPPAAL en la matière et représentons ces échanges par les signaux suivants :

- *ctl_main_fail !* : défaillance de la fonction Control_Main_Active, modélisée sous la forme d'un signal **émis (!)** par la fonction Main_AD et **reçu (?)** par la fonction Sub_AD (voir figure 5.14) ;
- *ctl_sub_fail1 !* : défaillance de la fonction Control_Sub_Standby ayant pour conséquence l'entrée dans l'état MRM de la fonction Main_AD ;
- *ctl_sub_fail2 !* : défaillance de la fonction Control_Sub_Standby ayant pour conséquence l'entrée dans l'état MRM de la fonction AD-3 ;
- *ctl_3_fail !* : défaillance de la fonction Control_3_Standby.

À partir des squelettes de modèles d'état retenus (voir figure 10.3 en annexe) et des relations entre groupes bien formés et transitions de groupe formalisées, nous avons pu construire les automates représentés sur la figure 5.14. Ces automates, nous le verrons dans le paragraphe 5.6.4, constituent une version agrégée des modèles que nous avons effectivement utilisés pour la vérification. Ils sont toutefois plus appropriés en première analyse. Cette fois-ci, contrairement aux modèles fonctionnels, les équations logiques sont reportées dans le modèle sous la forme de variables dont la valeur est mise à jour suite à l'occurrence d'un des quatre événements de perte de fonction. De cette manière le détail lié à la faute (accélération excessive, insuffisante, ...) est donné par la valeur des variables. Par exemple, si une décélération excessive due à une défaillance de la fonction Control_Sub_Standby, se produit lorsque la fonction Sub_AD est dans l'état *Standby*, alors la fonction Sub_AD passera dans l'état *Off* et la valeur de la variable *s_decel* (voir la Table 5.10) sera mise à 0. De plus, la fonction Main_AD passera dans l'état *MRM*.

On remarque par ailleurs que cette manière de modéliser (par signaux) nous a contraint à scinder la transition TGSFS3 en deux : TGSFS3_1 contient les exigences déterminant une entrée de la fonction Main_AD dans l'état *MRM* tandis que TGSFS3_2 regroupe

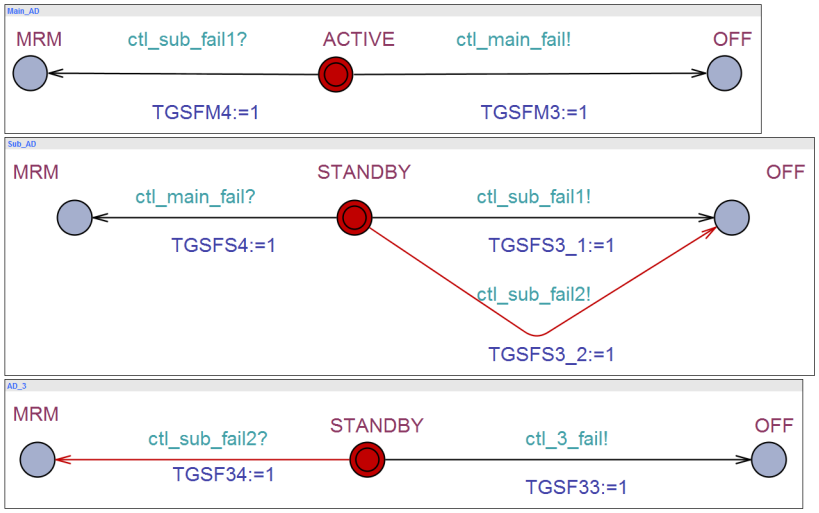


FIGURE 5.14 – Modèle d'état de sûreté préliminaire

les exigences qui spécifient un passage dans l'état *MRM* de la fonction AD-3. Ceci a pour cause la façon dont nous avons modélisé le comportement mais apporte en pratique un bénéfice important. Cette modélisation fait apparaître immédiatement aux relecteurs (les pilotes SdF) quels échanges ont les automates entre eux, donc, en creux, quelle stratégie de reconfiguration est effectivement déterminée par les exigences de sûreté retenues. Nous avons en effet constaté, lors de l'activité de validation (**AFS2.4**, voir paragraphe 5.6.4), l'efficacité de cette approche.

5.6.3.3 Détermination des propriétés formelles (AFS1.4.4)

La technique de vérification des exigences que nous avons employée est le *model checking*. Cette technique requiert d'exprimer les exigences dans une logique temporelle adéquate. Selon le type de propriété que l'on veut vérifier, l'on peut avoir recours à une logique plus ou moins expressive. Deux classes de logique temporelle existent :

— *Logique Temporelle Linéaire (LTL)* : il s'agit d'une extension de la logique classique (propositionnelle) visant à prendre en compte les aspects temporels. L'évolution temporelle du système est représentée par une séquence infinie d'états. Il n'y a qu'un seul futur possible. Cette logique a été introduite dans le domaine informatique en 1977 par Amir Pnueli (PNUELI 1977) ;

— *Logique Temporelle Branchante (CTL, pour Computation Tree Logic, en anglais)* : cette logique, également temporelle, est dite branchante car elle permet de décrire tous les futurs possibles. Elle a été développée par Edmund M.Clarke et E. Allen Emerson en 1981(CLARKE et EMERSON 1981).

Puisque nous voulons vérifier les propriétés sur tous les chemins possibles, nous avons choisi la logique CTL. Ensuite, nous avons recherché une expression dans cette logique qui puisse être fidèle à la fois à la formulation générale (Proposition 5.1) et au niveau de détail du contenu des exigences. Les travaux (DWYER, AVRUNIN et CORBETT 1998) fournissent justement des « patterns » de propriétés formelles à utiliser pour vérifier

des automates à états finis par *model checking*. Dans cet article, les patterns sont classés selon différents aspects du comportement du système étudié (absence/présence d'événements particuliers, réponse à des stimuli,...).
Pour notre cas d'étude, la formulation générale (Proposition 5.1) décrit une relation de cause à effet : entre l'occurrence d'un événement (conditions remplies) et l'entrée dans un état. Le « response pattern » correspond exactement à cette situation : les propriétés de ce pattern décrivent une relation de cause à effet entre l'occurrence d'un événement et l'entrée dans un état particulier. Nous employons donc une propriété formelle type, extraite de ces travaux et que nous adaptons à notre cas d'étude en reprenant les éléments de la Proposition 5.1. Chaque proposition formelle se vérifie par rapport à une transition particulière de l'automate considéré et est donc associée à une transition de groupe formalisée. Voici l'expression de la propriété formelle type :

Proposition 5.2
$AG(a \implies AF(b))$ ♠

où

- a : état de départ (unique) du groupe ET la garde formalisée associée à la transition de groupe ;
- b : état d'arrivée du groupe ;
- $\implies$: opérateur « imply » ;
- A : opérateur de chemin *All* ;
- G : opérateur temporel *Globally* signifiant que la propriété doit être toujours vraie dans le futur ;
- F : opérateur temporel *Futur* signifiant que la propriété sera *à un instant quelconque du futur* vraie.

La Proposition 5.2 signifie donc que, dans un état donné *a1* de l'automate sur lequel est vérifiée la propriété, si la garde formalisée *a2*, associée à la transition de groupe vérifiée, est vraie, alors, à un instant quelconque du futur, l'automate entrera dans l'état b (avec $a = a1\ ET\ a2$).
Par exemple, les deux propriétés formelles fonctionnelles préliminaires associées aux transitions de groupe formalisées déterminées précédemment (TGFF6 et TGFF15) sont les suivantes (PFF pour Propriété Formelle Fonctionnelle) :

- *PFF6_33* : AG((SupAD_33.Activatable) AND ((road_sec==0) AND ((veh_pos==0 AND veh_dir==1) OR (veh_pos==6)) AND (forec_AD_dur==1) AND (veh_loca==0 OR veh_loca==1) AND (lan_width==1) AND ((weat_cond==0) OR (weat_cond==1) OR (weat_cond==2) OR (weat_cond==3) OR (weat_cond==4) OR (weat_cond==5) OR (weat_cond==6))) $\implies$ AF(SupAD_v6.Available)) (pour TGFF6) ;
- *PFF15_33* : AG((SupAD_33.Not_available) AND ((road_sec==0) OR ((veh_pos==0 AND veh_dir==1) OR (veh_pos==6)) OR (forec_AD_dur==1) OR (veh_loca==0 OR veh_loca==1) OR (lan_width==1)

OR ((weat_cond==0)OR (weat_cond==1) OR (weat_cond==2) OR (weat_cond==3) OR (weat_cond==4) OR (weat_cond==5) OR (weat_cond==6))) $\Longrightarrow$ AF(SupAD_v6.Available)) (pour TGFF15)

L'indice *33* signifie que ces propriétés sont associées à la version 33 du modèle, nommée *SupAD_33* (voir figure 5.13). Du point de vue des pilotes SdF, la propriété formelle préliminaire associée à la transition de groupe formalisée TGSFM3 est (PFSM pour Propriété Formelle de Sûreté relative à la fonction Main_AD) :

— *PFSM3* : AG((Main_AD.ACTIVE) AND ((m_accel==2) OR (m_accel==0) OR (m_decel==2) OR (m_decel==0) OR (m_lat_accel==2) OR (m_lat_accel==0) OR (m_acti==1) OR (m_deac==1) OR (m_stay==1) OR (m_notif==1) OR (m_visib==1) OR (m_visib==2) OR (m_info==2) OR (m_info==1)) $\Longrightarrow$ AF(Main_AD.OFF))

Une fois l'ensemble de ces éléments obtenus, *i.e.* les modèles d'état préliminaires et propriétés formelles associées, il est possible de procéder à la vérification des propriétés sur les modèles (**AFS1.4.5**). Enfin, les modèles et les résultats de ces vérifications sont soumis et exploités par les deux acteurs métier.

5.6.4 Vérification et décision des experts métier (AFS1.4.5, AFS2.4)

5.6.4.1 Mise en œuvre de la vérification (AFS1.4.5)

L'ensemble des éléments est maintenant réuni pour procéder à la vérification des exigences proprement dite. Nous vérifions, en réalité, les transitions de groupe formalisées sur les modèles. Puis les liens effectués avec les groupes bien formés assurent la vérification des groupes bien formés et des exigences qu'ils contiennent également. Enfin, le fait que les méthodes de formation des groupes reposent sur des hypothèses validées par les deux acteurs métier garantit la cohérence de l'ensemble du processus et donnent du crédit à la vérification effectuée. Néanmoins, pour procéder à la vérification, deux approches différentes ont dû être adoptées selon les deux points de vue.

5.6.4.2 Perspective des AMS

L'automate construit du point de vue fonctionnel est unique (pour chaque version) et les valeurs des variables n'ont, dans l'outil UPPAAL, aucune raison d'évoluer si aucun stimulus n'est modélisé. En conséquence, tant que le changement possible de valeur des variables n'est pas modélisé, celles-ci restent à leur valeur initiale et l'automate est en *Deadlock*. Pour palier ce problème, nous avons ajouté deux automates qui simulent une évolution aléatoire de l'ensemble des variables (modèles présentés sur les figures 10.5 et 10.6 en annexe). Si la modélisation répondait bien à notre besoin, il n'a pas été possible, en revanche, de vérifier les propriétés formelles dans l'outil. En effet, l'espace d'état créé par la modélisation de l'ensemble des valeurs de variables possibles est trop grand et épuise ainsi la mémoire. À ce stade, deux pistes étaient envisageables :

1 Définir de nouvelles variables, en nombre plus restreint, pour réduire l'espace d'état ;

2 Utiliser un autre outil de vérification.

Nous avons choisi la seconde option pour les raisons suivantes :

— Ne pas formuler d'hypothèses supplémentaires (nécessaires pour définir de nouvelles variables) ;

— Rester au plus près du niveau des exigences traitées.

Nous avons, de ce fait, utilisé le *model checker* **NuSMV** (une version sans interface graphique, nommée simplement *NuSMV*[3] et une version dotée d'une interface graphique : *gNuSMV*[4]) pour vérifier les propriétés formelles. En effet, ce logiciel implémente un algorithme de *model checking* symbolique afin de réduire l'espace d'état. À titre d'illustration, le modèle *SupAD_12*, déjà présenté dans UPPAAL (figure 5.13) est retranscrit dans le paragraphe 10.5.2 en annexe. Nous avons, pour le modèle SupAD_12, restitué les résultats de la vérification obtenus sur la figure 5.15. Les propriétés 0 à 5 correspondent à la vérification de chacune des transitions de groupe formalisées, en utilisant le schéma de propriété formelle de la Proposition 5.2.

Il apparaît que la propriété 4 n'est pas vérifiée. La première conclusion est que les gardes définies par les transitions de groupe TGFF16_12 et TGFF14_12 peuvent être vraies au même moment. Nous nous intéressons ensuite au contre-exemple retourné par l'outil, représenté sur la figure 5.16. Nous constatons, au vu de la figure 5.13, que cette trace correspond au scénario pour lequel une boucle infinie est créée entre les états *Not_available*, *Available* et *Activatable*. En revanche, cette situation ne se retrouve pas lorsque les gardes TGFF14_12 et TGFF16_12 sont vraies dans l'état *Activatable* et que l'on cherche à connaître l'atteignabilité de l'état *Not_available*. En effet, comme le prouve la propriété 3, l'état *Not_available* sera nécessairement atteint, dans ces conditions. La solution que nous avons proposée est d'ajouter, dans le modèle d'état SupAD_12, pour chacune des gardes des transitions sortantes de l'état *Activatable*, l'opposée de l'autre garde. Rappelons que, pour le modèle SupAD_12 : les conditions des gardes des transitions de groupe TGFF14_12 et TGFF16_12 sont, respectivement, celles des groupes GF2.1 et GF5.1. En prenant en compte les résultats de la vérification, nous enrichissons donc ces gardes de la manière suivante :

— Garde de TGFF14_12 = Conditions de GF2.1 ET $\overline{Conditions\ de\ GF5.1}$;

— Garde de TGFF16_12 = Conditions de GF5.1 ET $\overline{Conditions\ de\ GF2.1}$.

Nous obtenons donc, pour *SupAD_12*, un nouvel automate, dont un extrait pertinent est montré dans le paragraphe 10.5.2 en annexe. Ces modifications correspondent à l'hypothèse de modélisation **HM9** (voir figure 5.10).

5.6.4.3 Point de vue des pilotes SdF

En pratique, les propriétés formelles déduites des exigences de SdF retenues ne sont pas vérifiées sur les modèles représentés sur la figure 5.14, qui sont des versions agrégées. Les modèles sur lesquels la vérification a été entreprise figurent en annexe, au paragraphe 10.7. Ceci s'explique par le fait qu'un événement de défaillance correspond au changement de valeur d'une seule variable. Ainsi, avec la version représentée sur la

3. http ://nusmv.fbk.eu/
4. http ://nusmv.fbk.eu/gnusmv/

0	☑	True		CTL	AG ((state = off & TGFF1_12) -> AF state = not_available)
1	☑	True		CTL	AG ((state = not_available & TGFF6_12) -> AF state = available)
2	☑	True		CTL	AG ((state = available & TGFF11_12) -> AF state = activatable)
3	☑	True		CTL	AG ((state = activatable & TGFF14_12) -> AF state = not_available)
4	☑	False	1	CTL	AG ((state = activatable & TGFF16_12) -> AF state = active)
5	☑	True		CTL	AG ((state = active & TGFF18_12) -> AF state = not_available)

FIGURE 5.15 – Résultats de la vérification des propriétés formelles fonctionnelles sur le modèle SupAD_12

Loop	Step	ADactiv	ADbutton	AEinter	Astate	Lgapspd	Rgapspd	TGFF1112	TGFF1412	TGFF1612	TGFF1812	TGFF112	TGFF612
	0	1	1	1	off	1	1	1	1	1	1	1	0
	1	1	1	1	not_available	1	1	1	1	1	1	1	1
	2	1	1	1	available	1	1	1	1	1	1	1	0
↱	3	1	1	1	activatable	1	1	1	1	1	1	1	0
|	4	1	1	1	not_available	1	1	1	1	1	1	1	1
|	5	1	1	1	available	1	1	1	1	1	1	1	0
└	6	1	1	1	activatable	1	1	1	1	1	1	1	0

FIGURE 5.16 – Contre-exemple suite à la vérification de propriété formelle fonctionnelle 4

figure 5.14, il n'est, par exemple, pas possible de savoir quelle variable a évolué lorsque la variable globale TGSFM3 (associée au groupe GSM2.1.2) passe à 1. Ainsi, grâce aux automates détaillés, nous vérifions, par exemple, le respect du groupe bien formés GSM2.1.2 par ces automates en utilisant l'expression suivante :

PFSM3 : ((Main_AD.ACTIVE) && ((m_accel==2) || (m_accel==0) || (m_decel==2) || (m_decel==0) || (m_lat_accel==2) || (m_lat_accel==0) || (m_acti==1) || (m_stay==1) || (m_deac==1) || (m_notif==1) || (m_visib==1) || (m_visib==2) || (m_info==2) || (m_info==1)) → (Main_AD.OFF))

Cette expression correspond à l'énoncé de la propriété formelle PFSM3, déjà exprimée précédemment, tel qu'il est entré dans l'outil UPPAAL. Effectivement, dans ce logiciel, les opérateurs ET et OU sont respectivement symbolisés par « && » et « || », et l'opérateur →, appelé « leads to », est défini tel que (a → b) ⇔ AG(a ⟹ AF(b)).

Les résultats de la vérification ont mis en évidence que les propriétés formelles de SdF associées aux transitions de groupe TGSFM4 et TGSF33 (voir figure 5.14) n'étaient pas respectées. Ceci est dû au fait que sur des mêmes événements de défaillance de la fonction Control_Sub_Standby, il était spécifié à la fois une entrée dans l'état *MRM* de la fonction Main_AD et de la fonction AD-3 (donc l'activation de la fonction Control_Main_MRM et Control_3_MRM). Cette vérification révèle donc également une indétermination dans la spécification des sous-fonctions de Supervision AD, que nous avons soumise aux pilotes SdF lors de l'activité **AFS2.4**.

5.6.4.4 Décisions des experts métier (AFS2.4)

La toute dernière activité du processus de formalisation et de construction des modèles d'état revient aux AMS et aux pilotes SdF. Ces deux acteurs doivent déterminer le caractère correct des modèles, par rapport à leur savoir-faire et leur compétence. Cette activité relève de la connaissance qu'ont les ingénieurs dans le comportement attendu de la fonction qu'ils spécifient.

Du point de vue des AMS, les hypothèses relatives à la déterminisation des modèles (HM9) ont été jugées pertinentes. De ce fait, les 6 modèles d'état fonctionnels retenus ont été modifiés de la même manière que l'exemple du modèle *SupAD_12*.

Pour notre cas d'étude, les points intéressants ont particulièrement concerné la SdF. En effet, la spécification de la fonction AD-3 ne répondait pas à l'objectif initial des pilotes SdF. D'après le modèle représenté sur la figure 5.14, il apparaît que la fonction AD-3 entre dans l'état *MRM* lorsque des dysfonctionnements spécifiques dus à la fonction Control_Sub_Standby surviennent. Ceci s'observe aisément en étudiant la forme de l'automate de la fonction Sub_AD, pour lequel deux transitions mènent de l'état *Standby* à l'état *Off*. Or, le comportement attendu par les experts, est une entrée dans l'état *MRM* de la fonction AD-3 uniquement si les fonctions Main_AD et Sub_AD subissent un dysfonctionnement dû à une cause commune, donc simultané. Par ailleurs, l'ensemble des dysfonctionnements de la fonction Control_Sub_Standby doivent uniquement causer l'entrée dans l'état *MRM* de la fonction Main_AD. Pour prendre en compte ces remarques, des automates modifiés (remplaçant la version préliminaire fournie à la figure 5.14), représentés sur la figure 5.17 ont été construits, puis validés par les pilotes SdF. On voit sur ces automates que la défaillance simultanée des fonctions Control_Main_Active et Control_Sub_Standby est modélisée par un nouveau signal, *ctl_ms_fail!*. Par ailleurs, les signaux *ctl_sub_fail1!* et *ctl_sub_fail2!* n'apparaissent logiquement plus sur le modèle. Ces modifications ont nécessairement eu un impact sur les exigences de SdF :

- 3 exigences, initialement allouées à la fonction AD-3 ont été supprimées, car elles spécifiaient une réaction non conforme aux attentes des pilotes SdF ;
- 14 nouvelles exigences ont été allouées à la fonction AD-3 pour spécifier l'entrée dans l'état *MRM*. Elles font donc partie du groupe bien formé GS32.1.1 ;
- 14 nouvelles exigences ont été allouées aux fonctions Main_AD et Sub_AD pour spécifier leur entrée simultanée dans l'état *Off*. Ces exigences constituent les gardes des transitions de groupe TGSFM3_2 (pour Main_AD, voir figure 5.17) et TGSFS3_2 pour la fonction Sub_AD.

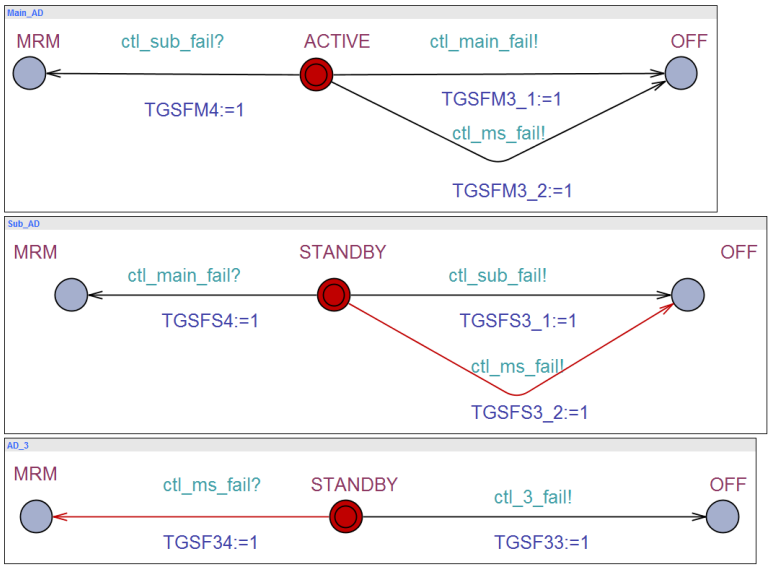


FIGURE 5.17 – Modèle d'état validé du point de vue de la sûreté

5.7 Conclusions générales

La démarche de travail suivie pour formaliser les exigences et construire les modèles d'état associés, présentée de façon détaillée dans ce chapitre, met en lumière certaines contributions revendiquées, notamment sur la figure 4.2. Plus précisément :

1 **Contribution 1** : *Mise en évidence de conceptions alternatives* : nous avons montré que la complétion des informations manquantes, nécessaires à la formalisation des exigences, fait émerger plusieurs solutions crédibles. Cependant, la difficulté principale réside dans l'exhibition de modèles plausibles, initialement « mélangés » avec des possibilités plus ou moins sensées. C'est justement pourquoi nous favorisons l'intervention de l'expertise métier dans le processus de formalisation, au détriment d'une automatisation totale. Ceci évite, nous l'avons vu, une explosion combinatoire des possibilités. Toutefois, dans le processus actuel, ces interventions sont encore trop fréquentes. En revanche, nous avons montré que des règles (par exemple, séquencement particulier d'états, critères d'atteignabilité, états bloquants, formation des groupes) ont pu être définies à l'issue de certaines de ces interventions. Ces règles sont, pour certaines, intégrables dans un processus de formalisation semi-automatisé. Ainsi, bien qu'une automatisation totale du processus de formalisation ne soit certainement ni faisable, ni désirable, il est par contre intéressant de réduire le nombre d'interventions des experts en formalisant certaines règles ;

2 **Contribution 2** : *Définition précise du rôle des experts* : le détail de chacune des sous-activités mises en œuvre pour réaliser l'activité globale de construction des modèles d'état selon les deux points de vue (AFS1) détermine à quelles étapes les deux acteurs doivent intervenir, ce qu'ils doivent faire, les données qu'ils reçoivent et celles qu'ils produisent. Le séquencement est lui corrélé au processus de formalisation, que nous avons réalisé dans le cadre de ces travaux de thèse.

La description précise des activités à effectuer ainsi que l'implication des deux expertises contribuent à rendre pérennes les travaux de thèse.

En conclusion de ce chapitre, deux ensembles cohérents *{Modèle d'état, Exigences, Propriétés Formelles, Hypothèses}* sont obtenus. En revanche, ces deux ensembles n'ont pas été confrontés afin de faire converger les deux perspectives métier. Ceci fait justement l'objet du chapitre 6.

6

Convergence des points de vue

Sommaire

6.1 Objectifs et enjeux de la convergence

6.1.1 Récapitulatif des données résultantes de la construction des modèles d'état

Au terme des activités de construction des modèles d'état et de formalisation des exigences (activités **AF1**, **AF2**, **AS1** et **AS2**, voir figure 4.8), nous avons obtenu :

— **Selon le point de vue des AMS** :

(a) 6 versions de Modèles d'État Fonctionnels vérifiés et validés par les AMS, constitués chacun d'un automate fini spécifiant le comportement de la Supervision AD globale ;

(b) 69 exigences retenues et modifiées par l'ajout d'informations d'état : entre 69 et 96 exigences vérifiées sur les modèles d'état construits, selon les versions ;

(c) Entre 6 et 8 Propriétés Formelles Fonctionnelles vérifiées, selon les versions ;

(d) 9 hypothèses d'interprétation validées par les AMS (voir Table 5.9).

— **Selon le point de vue des pilotes SdF** :

(a) 1 version de Modèle d'État de Sûreté vérifié et validé par les pilotes SdF comprenant 3 automates finis pour les 3 sous-fonctions de Supervision : Main_AD, Sub_AD et AD-3 (voir figure 2.13) ;

(b) 61 exigences sélectionnées : toutes ont été modifiées par l'ajout d'informations d'état. De plus, 3 exigences ont été supprimées et 42 nouvelles exigences ont été créées. Au final 100 exigences vérifiées sur les modèles construits ;

(c) 7 Propriétés Formelles de Sûreté vérifiées ;

(d) 7 hypothèses d'interprétation validées par les pilotes SdF (voir Table 5.9).

— **Selon les deux points de vue** :

(a) 9 hypothèses de modélisation ont été validées par les AMS et les pilotes SdF.

Ces activités de consolidation aboutissent à deux ensembles : { *Modèle d'état, Exigences, Propriétés Formelles, Hypothèses* } cohérents mais construits « en silos ». En pratique, ceci signifie que les Propriétés Formelles Fonctionnelles (respectivement les Propriétés Formelles de Sûreté) ne peuvent *a priori* pas être vérifiées sur les Modèles d'État de Sûreté (respectivement les Modèles d'État Fonctionnels). Par exemple, la Propriété Formelle Fonctionnelle **PFF6_33**, restituée dans le paragraphe 5.6.3, ne pourra pas être vérifiée sur le modèle d'état validé par les pilotes SdF (voir figure 5.17). En effet, les états *Available* et *Activatable*, figurant dans l'énoncé de PFF6_33, n'existent pas dans le Modèle d'État de Sûreté. De plus, les conditions de la propriété PFF6_33 ne sont également pas présentes dans ce modèle. En conséquence, il convient de modifier les modèles résultants des activités de construction en silos pour obtenir un modèle (constitué de l'automate de la Supervision AD globale et des 3 automates des Supervisions locales), nommé **Modèle d'État Complet**, sur lequel l'ensemble des propriétés puissent être vérifiées.

6.1.2 Fil conducteur

À l'instar du chapitre 5, nous présentons le fil rouge qui sera suivi tout au long de ce chapitre, en s'appuyant pour ce faire sur les figures 6.1 et 6.2. Ces deux figures représentent les sous-activités détaillées dans le chapitre 6 : la première est centrée sur les données manipulées tandis que la seconde précise le contenu des sous-activités de l'activité **A3**. Cependant, contrairement à la figure 5.1, ce sont les données manipulées qui sont représentées sur la figure 6.1, et non le type de données. Ces données sont donc qualifiées en fonction de leur état de maturité (préliminaires, vérifiées, validées). Par rapport à la démarche globale, illustrée sur la figure 4.8 à la fin du chapitre 4, les deux activités sur lesquelles est focalisé le chapitre 6 sont celles de construction des Modèles d'État Complet (**A3**), et le choix, à la fois par les AMS et les pilotes SdF, d'un ou plusieurs modèles d'état corrects, lors de l'activité **A4**. Contrairement aux activités **AF1** et **AS1** qui étaient réalisées selon, respectivement, le point de vue des AMS et celui des pilotes SdF, l'activité **A3** prend en considération les deux perspectives métier à la fois. En outre, l'activité **A4** est réalisée conjointement par les deux acteurs métier. De la même manière que pour les activités précédentes, la construction de modèles complets nécessite la formulation d'hypothèses qui sont régulièrement confrontées aux deux expertises. Ainsi, les activités **A3** et **A4** ne sont pas successives. Ceci est symbolisé sur la figure 4.8 par le positionnement relatif des deux activités, et par les boucles de validation apparaissant sur la figure 6.1. De ce fait, l'état de maturité des données produites par les sous-activités de **A3** est toujours préliminaire, ou à valider, par des sous-activités de l'activité **A4**. Ces interactions seront détaillées dans les paragraphes qui suivent.

À partir des données d'entrée, récapitulées dans le paragraphe 6.1.1, il s'agit en premier lieu d'établir un vocabulaire commun pour les objets de même type (états, conditions des exigences), lors des activités **A3.1** et **A4.1**. Le but est d'aboutir à une liste d'états unifiée, partagée et validée par les deux expertises, ainsi qu'une comparaison et éventuelle mise en cohérence (si besoin) des variables et équations logiques, traduisant les conditions environnementales, provenant des deux points de vue (variables fonctionnelles et de sûreté, voir Tables 10.1 et 10.2 en annexe). Cette activité donne lieu à des ***listes d'états unifiées*** (une pour chaque fonction : Supervision AD globale et les trois Supervisions locales) ainsi qu'un ensemble d'***équations logiques unifiées***.

La détermination de ces éléments permet alors d'enrichir les Modèles d'État Fonctionnels (de la Supervision AD globale), résultants de l'activité **AF2**, par la considération des aspects dysfonctionnels. Ces derniers étaient jusqu'alors analysés par les pilotes SdF, mais au niveau local (des sous-fonctions de Supervision). Cet enrichissement, opéré lors de l'activité **A3.2**, aboutit à des ***Modèles Globaux cibles***, soumis aux expertises conjointes des AMS et pilotes SdF lors de l'activité **A4.2**. L'objectif de cette activité **A3.2** est de déterminer les réactions appropriées que doit présenter la Supervision AD globale en cas d'occurrence d'un événement de défaillance spécifié par les pilotes SdF (c'est parce que les modèles résultants intègrent ces aspects de sûreté et qu'ils concernent la Supervision AD globale qu'ils sont qualifiés de « globaux » et non plus de « fonctionnels »).

Les modifications apportées aux Modèles d'État Fonctionnels sont à l'origine d'***exigences supplémentaires***, allouées à la Supervision AD globale durant l'activité. Ces exi-

gences supplémentaires sont le fruit d'une ré-allocation soit en termes de fonction (par exemple, une exigence, initialement allouée à une Supervision locale, est allouée à la Supervision AD globale), soit en termes d'espace d'états (une exigence, initialement valable pour un espace d'état donné, devient valable pour un espace d'états différent).

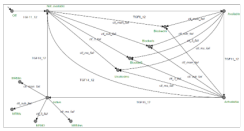
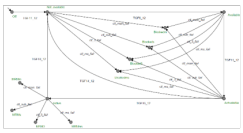
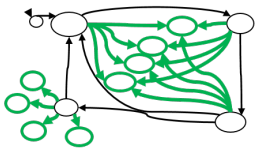
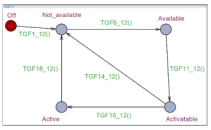
Les Modèles Globaux cibles validés à l'issue de l'activité spécifient le comportement attendu, à la fois par les AMS et par les pilotes SdF, de la Supervision AD globale. Il convient dès lors de modifier la spécification des comportements des Supervisions locales de telle sorte que leurs comportements concurrents soient cohérents avec celui de la Supervision AD globale (activité **A3.3**). La garantie de la cohérence entre les vues locale et globale repose sur l'opération de composition parallèle d'automates à états finis. Pour un Modèle Global cible donné, un Modèle d'État Local est jugé correct (et la vue globale cohérente avec les vues locales), si le résultat de la composition parallèle des 3 automates qui le constituent est identique à l'automate du Modèle Global cible considéré. Les modèles des Supervisions locales ainsi obtenus sont appelés ***Modèles Locaux***. Ils sont validés par les AMS et les pilotes SdF lors de l'activité **A4.3**.

Les modifications opérées sur les Modèles d'État de Sûreté initiaux donnent lieu également à la formulation d'exigences supplémentaires allouées cette fois aux fonctions Main_AD, Sub_AD et AD-3. Chaque ***Modèle d'État Complet préliminaire*** est donc constitué d'un Modèle Global cible et d'un Modèle Local.

Dans un dernier temps (activité **A3.4**), les Modèles d'État Complets préliminaires sont vérifiés par rapport à l'ensemble des propriétés formelles : celles déduites des exigences fonctionnelles de comportement et des exigences fonctionnelles de SdF retenues qui sont encore pertinentes pour les modèles complets ; ainsi que les propriétés déduites des exigences supplémentaires. Cette vérification est menée durant l'activité **A3.4** et les modèles d'état sur lesquels toutes les propriétés formelles ont été vérifiées (appelées ***propriétés formelles vérifiées et validées***) sont nommés ***Modèles d'État Complets vérifiés et validés***.

Remarque 6.1
Sur la figure 6.1, seul le type de données manipulées est indiqué et non leur état de maturité (comme pour la figure 5.2). ♢

Objectif

Activité d'expertise	Activité formalisée	Données manipulées
A4.4	A3.4	**Modèles d'Etat Complets vérifiés et validées**  **Propriétés formelles vérifiées et validées** - Supervision AD globale: Ex: *« AG((state = activatable & TGF6_33) → AF(state = available))* - Main_AD: Ex: *« AG((state_m = active & ctl_main_fail) → AF(state_m = silent))* - Sub_AD - AD-3
A4.3	A3.3	**Modèles Locaux validés** **Modèles d'Etat Complets Préliminaires validés**  **Exigences supplémentaires** - Main_AD: Ex: *« If Main_AD is in the state Not_available and in case of faulty deactivation of the AD function due to a dysfunction of Main_AD function, then Main_AD shall switch to the state Silent »* *« If Main_AD is in the state Not_available at night without impact on Autonomous Vehicle performance, then AD function shall be in the state Available »* - Sub_AD - AD-3
A4.2	A3.2	**Modèles Globaux cibles validés**  **Exigences supplémentaires** - Supervision AD globale: Ex: *« If AD Supervision is in the state Not_available and in case of faulty deactivation of the AD function due to a dysfunction of Main_AD, then Main_AD shall switch to the state Silent »* *« If AD Supervision is in the state Available and in case of faulty deactivation of the AD function due to a dysfunction of Main_AD, then Main_AD shall switch to the state Silent »*
A4.1	A3.1	**Listes d'états attendus unifiée validées** **Equations logiques unifiées validées**
		Modèles d'Etat - 6 Modèles d'Etat Fonctionnels - 1 Modèle d'Etat de Sûreté  **Exigences** - Entre 69 et 96 exigences fonctionnelles: Ex: *EFBF6_1: « If AD Supervision is in the state Not_available at night without impact on Autonomous Vehicle performance, then AD function shall be in the state Available »* - 100 exigences de sûreté: Ex: *ESMBF6_1: « If Main_AD is in the state Active and in case of faulty deactivation of the AD function due to a dysfunction of Main_AD, then Main_AD shall switch to the state Off »* **Propriétés formelles** - Entre 6 et 8 Propriétés Formelles Fonctionnelles - 7 Propriétés Formelles de Sûreté

Données d'entrée

FIGURE 6.1 – Fil conducteur régissant la manipulation des données

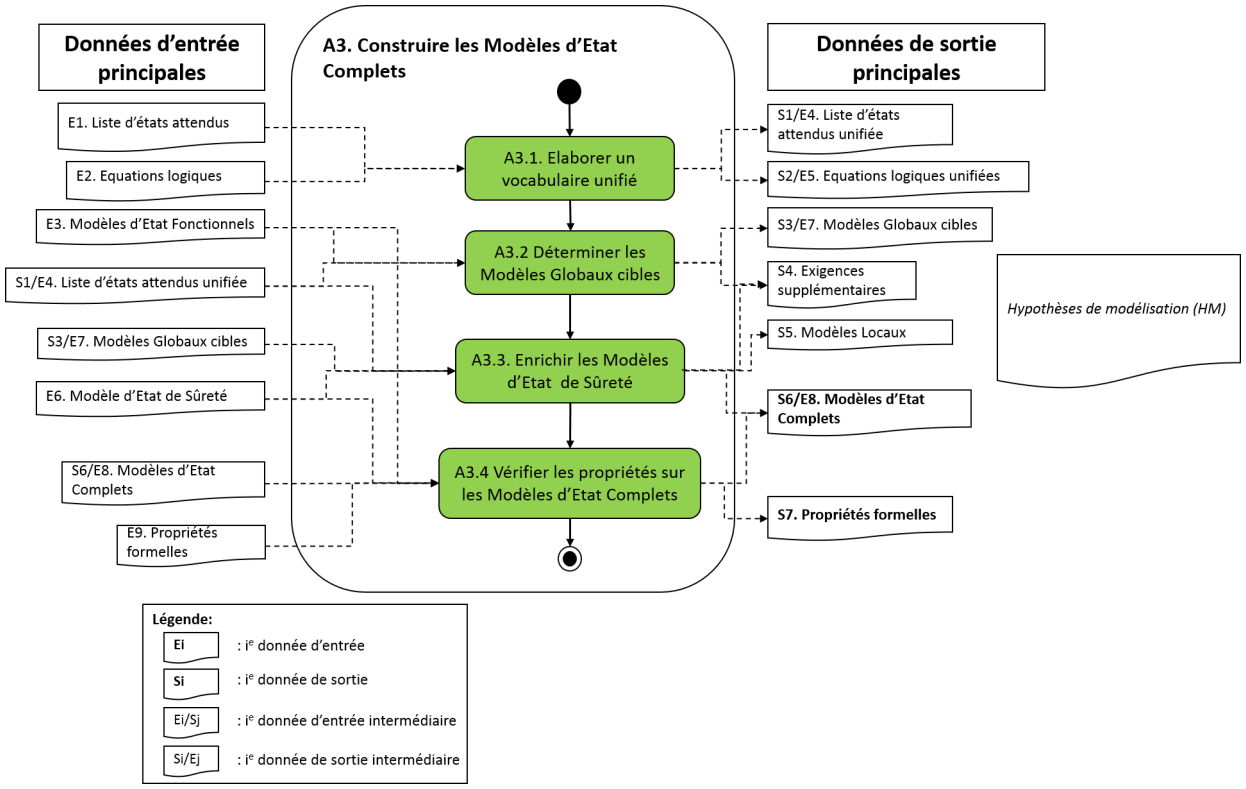


FIGURE 6.2 – Activité de convergence des points de vue (**A3**)

6.2 Établissement d'un vocabulaire unifié (A3.1)

6.2.1 Constat relatif aux vocabulaires métier

Nous avons, lors des activités précédentes, volontairement fait usage d'un formalisme commun, l'automate à états finis, afin que les objets construits soient de même type. Toutefois, manipuler des objets de même type ne garantit pas que la signification que leur accordent les deux acteurs de l'ingénierie soit convergente. Or, pour que les modèles de spécification de comportement élaborés aient du sens, il est nécessaire de s'assurer de cette convergence. Ceci passe par l'établissement d'un lexique partagé en termes :

(a) D'**événements** ou, dans notre cas d'étude, de **conditions** sous lesquelles un changement d'état est opéré ;

(b) D'**états**.

Pour le premier point, il s'agit de comparer les conditions définies par les exigences fonctionnelles de comportement retenues avec celles décrites par les exigences fonctionnelles de SdF sélectionnées. Pour ce faire, il faut étudier les variables et déterminer si elles sont communes ou décrivent une même réalité physique. Concernant les états, deux aspects sont à analyser :

1 La signification de la dénomination des états ;

2 Le niveau d'abstraction des états.

Le premier item vise à éviter des confusions ou conflits sémantiques. Par exemple, un état ayant la même dénomination mais deux sens différents suivant le point de vue métier considéré. La deuxième réflexion est relative au niveau de raffinement des fonctions considérées. En effet, comme l'illustrent les figures 2.12 et 2.13, la spécification de la Supervision AD par les AMS se fait à un niveau d'abstraction plus haut que les pilotes SdF. En conséquence, les états de la Supervision AD globale correspondent à une (ou plusieurs) combinaisons d'états des Supervisions locales. Ces combinaisons doivent être explicitement déterminées.

6.2.2 Activités déployées

6.2.2.1 Aperçu des activités visant à unifier le vocabulaire

La figure 6.3 illustre les activités à réaliser pour unifier les vocabulaires. Tout d'abord, les deux listes d'états, produites par les AMS et les pilotes SdF, sont comparées durant l'activité **A3.1.1**. Puis, les conditions contenues dans les exigences sont analysées au travers des équations logiques et variables que nous avons déterminées lors de l'activité **AF3.1.2** (voir paragraphe 5.5.4 du chapitre 5). Le but de ces deux activités est double : repérer, d'une part, d'éventuels conflits sémantiques, et, d'autre part, inclure les aspects relatifs à la SdF dans la perspective globale des AMS. Les hypothèses de modélisation auxquelles ces activités donnent lieu sont ensuite soumises aux deux expertises lors de l'activité **A4.1**. La différence notable avec les activités qui faisaient intervenir l'expertise, présentées dans le chapitre 5, est la nécessité d'une participation conjointe des AMS et pilotes SdF, qui doivent donc procéder à ces activités en collaboration.

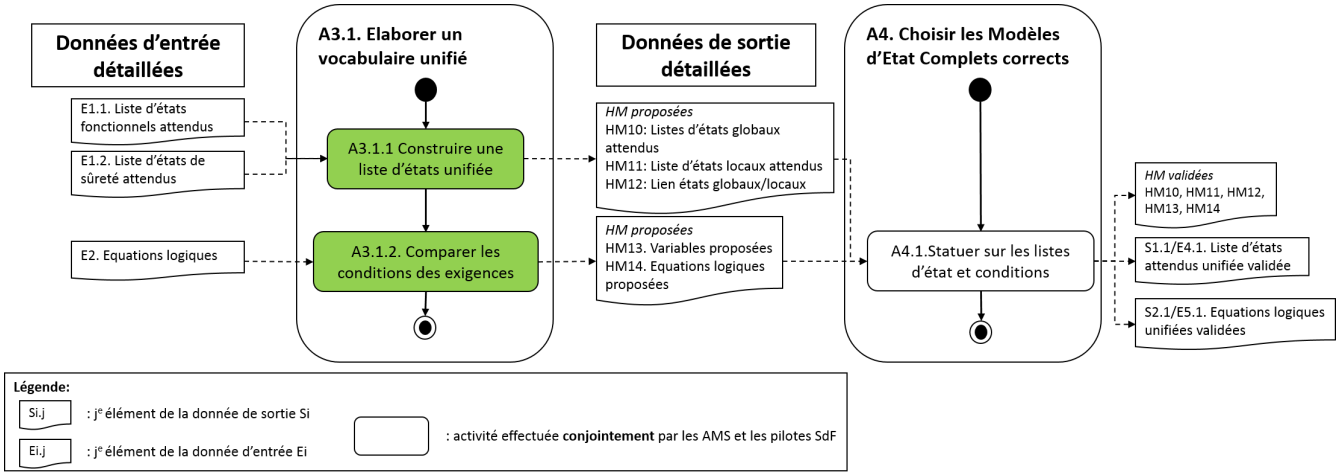


FIGURE 6.3 – Modèles Globaux cibles (**A3.1**)

6.2.2.2 Listes d'états unifiée (A3.1.1)

Signification des états

En comparant la liste d'états attendus produite par les AMS avec celle fournie par les pilotes SdF (voir paragraphe 5.3.1), il apparaît que deux états portent le même nom. Il s'agit des états ***Off*** et ***Active***.

Rappelons que, du point de vue des AMS, l'état *Off* est décrit comme étant l'état dans lequel le support physique de la Supervision AD globale n'est pas alimenté électriquement. Pour les pilotes SdF, l'état *Off*, atteignable par les trois sous-fonctions de Supervision (Main_AD, Sub_AD et AD-3), correspond à une défaillance des fonctions de Contrôle AD gérées par les Supervisions locales (Control_Main_Active, Control_Sub_Standby et Control_3_Standby). Les deux sens sont donc complètement différents alors que le nom de l'état est commun. Pour résoudre ce conflit de vocabulaire, les deux significations ont été clairement distinguées, donnant lieu aux deux définitions d'état suivantes :

- *Off* : état dans lequel les supports physiques de la Supervision AD globale ou des Supervisions locales ne sont pas alimentés électriquement ;
- ***Silent*** : état atteint par une Supervision AD locale lorsque qu'une fonction de Contrôle AD est défaillante. Plus précisément :
  - La fonction Main_AD atteint l'état *Silent* suite à une défaillance affectant la fonction Control_Main_Active ;
  - La fonction Sub_AD atteint l'état *Silent* suite à une défaillance affectant la fonction Control_Sub_Standby ;
  - La fonction AD-3 atteint l'état *Silent* suite à une défaillance affectant la fonction Control_3_Standby ;
  - Les fonctions Main_AD et Sub_AD atteignent simultanément l'état *Silent* suite à une défaillance de cause commune affectant au même instant les fonctions Control_Main_Active et Control_Sub_Standby.

Pour les pilotes SdF, l'état précédemment appelé *Off* est désormais dénommé *Silent*. En revanche, dans l'objectif d'obtention des Modèles Globaux intégrant les aspects relatifs à la sûreté, la (ou les) réaction(s) appropriée(s) de la Supervision AD globale en cas de défaillance d'une fonction de Contrôle AD doivent être déterminées. Puisque la situation pour laquelle toutes les fonctions de contrôle sont simultanément défaillantes n'est pas considérée comme envisageable, il en résulte que, du point de vue global, la fonction Contrôle AD est toujours opérationnelle. Plus précisément, à tout instant une fonction de contrôle pouvant assurer une MRM (manœuvre de sécurité) est disponible. Par conséquent, et sous cette hypothèse, la Supervision AD globale ne peut jamais se trouver dans un état « *Silent global* » (en adoptant le même sens que pour les Supervisons locales).

Ceci ne dispense pas toutefois de spécifier le comportement de la Supervision AD globale lorsqu'un événement de défaillance affectant une fonction de contrôle a lieu, à n'importe quel moment. Pour ce faire, trois situations ont été distinguées :

1 *La fonction AD n'est pas alimentée électriquement* : dans ce cas il n'est pas utile de spécifier une réaction en cas de défaillance d'une fonction de contrôle. En effet, une défaillance d'une fonction survenant alors que son support physique n'est pas alimenté n'est, de toute façon, détectée qu'à l'instant auquel le support reçoit de l'énergie. Si une fonction est déjà défaillante (avant alimentation), alors la Supervision locale qui gère la fonction de contrôle défaillante passera directement dans l'état *Silent*, après avoir atteint l'état *Not_available* suite à l'événement d'alimentation ;

2 *La fonction AD est alimentée électriquement mais n'est pas active*, c'est-à-dire que la conduite n'est pas déléguée à la fonction AD (la Supervision AD globale ne se trouve pas dans l'état *Active*). Si une défaillance d'une fonction de contrôle survient durant cette phase, alors la Supervision AD doit entrer dans un de ces quatre nouveaux états : ***Blockedm***, ***Blockeds***, ***Blocked3*** ou ***Blockedms*** pour signifier que la fonction AD ne sera plus proposée au conducteur. En effet, la fonction AD ne peut être activée par le conducteur uniquement si toutes les fonctions de contrôle sont saines. La seule sortie possible de cet état est la réparation (au garage par exemple) de la fonction de contrôle défaillante. Plus précisément ces états sont définis de la sorte :

   i. ***Blockedm*** : état atteint par la Supervision AD globale suite à une défaillance de la fonction Control_Main_Active alors que la Supervision AD globale n'était pas dans l'état *Active* ;

   ii. ***Blockeds*** : état atteint par la Supervision AD globale suite à une défaillance de la fonction Control_Sub_Standby alors que la Supervision AD globale n'était pas dans l'état *Active* ;

   iii. ***Blocked3*** : état atteint par la Supervision AD globale suite à une défaillance de la fonction Control_3_Standby alors que la Supervision AD globale n'était pas dans l'état *Active* ;

   iv. ***Blockedms*** : état atteint par la Supervision AD globale suite à une défaillance simultanée des fonctions Control_Main_Active et Control_Sub_Standby alors que la Supervision AD globale n'était pas dans l'état *Active*.

3 *La fonction AD est alimentée électriquement et active*, *i.e.* la conduite lui est complètement déléguée (la Supervision AD se trouve alors dans l'état *Active*). Dans cette situation, si une défaillance d'une fonction de contrôle survient, le véhicule doit être mis en sécurité. C'est pourquoi les quatre états supplémentaires suivants ont été proposés :

   i. ***MRMm*** : état atteint par la Supervision AD globale suite à une défaillance de la fonction Control_Main_Active alors que la Supervision AD globale était dans l'état *Active* ;

   ii. ***MRMs*** : état atteint par la Supervision AD globale suite à une défaillance de la fonction Control_Sub_Standby alors que la Supervision AD globale était dans l'état *Active* ;

   iii. ***MRM3*** : état atteint par la Supervision AD globale suite à une défaillance de la fonction Control_3_Standby alors que la Supervision AD globale était dans l'état *Active* ;

iv. ***MRMms*** : état atteint par la Supervision AD globale suite à une défaillance simultanée des fonctions Control_Main_Active et Control_Sub_Standby alors que la Supervision AD globale était dans l'état *Active*.

La liste d'états globaux attendus, qui correspond à l'hypothèse de modélisation **HM10** (voir figure 6.3), est donc constituée de la liste d'états fonctionnels attendus initiale, donnée dans le paragraphe 5.3.1, augmentée des états *Blockedm*, *Blockeds*, *Blocked3*, *Blockedms*, *MRMm*, *MRMs*, *MRM3* et *MRMms*, tels que définis précédemment.

Comme évoqué plus haut, le second état déterminé à la fois par les AMS et les pilotes SdF est l'état ***Active***. Cet état est spécifié pour la Supervision AD globale et pour la fonction Main_AD. Dans les deux cas, il s'agit de la situation pour laquelle la conduite est entièrement déléguée à la fonction AD. Ainsi, il n'existe pas de conflits particuliers de vocabulaire relatifs à la signification de cet état. Cependant, deux questions se posent :

1 Dans quel état doivent être les fonctions Sub_AD et AD-3 lorsque la Supervision AD globale est dans l'état *Active* ?

2 Quelles sont les relations entre les états définis uniquement d'un point de vue et non de l'autre, *i.e. Not_available*, *Available* et *Activatable*, du point de vue des AMS et *Standby* et *MRM* pour les pilotes SdF ?

Niveau d'abstraction des états

Les deux acteurs métier spécifient le comportement de la Supervision AD à des niveaux d'abstraction différents. La conséquence en termes d'états est la nécessité d'établir une correspondance entre les états de la Supervision AD globale ceux des sous-fonctions, locales, de Supervision. La question s'est naturellement posée dans le paragraphe précédent au sujet de l'état *Active*. En première analyse, et au vu de la définition des états des Supervisions locales (voir paragraphe 5.3.1), lorsque la Supervision AD globale est dans l'état *Active*, on peut supposer que les fonctions Sub_AD et AD-3 doivent se trouver dans l'état *Standby*.

Le problème est que les Supervisions locales ne doivent pas avoir la même réaction si une défaillance d'une fonction de contrôle survient lorsque la Supervision AD globale est dans l'état *Active* (alors une des Supervisions locales doit entrer dans l'état *MRM* pour que le véhicule soit mis en sécurité), que lorsqu'elle n'est pas dans cet état (alors les Supervisions locales gérant les fonctions de contrôle encore saines doivent entrer dans un état pour lequel ces fonctions de contrôle ne puissent plus être activées). Or, l'état *Standby*, tel qu'il a été défini dans la liste d'états de sûreté initiale (voir paragraphe 5.3.1), est atteint quand la Supervision AD globale est dans l'état *Active*. Cette définition de l'état *Standby* n'est donc pas satisfaisante. Il s'agit de le « décomposer » de deux façons différentes :

1 *Pour la fonction Sub_AD* : *Standby* est décomposé selon tous les états fonctionnels (excepté *Off*), *i.e. Not_available*, *Available*, *Activatable* et *Active*. Lorsque la Supervision AD globale est dans un de ces états, la fonction Sub_AD doit y être également ;

2 *Pour la fonction AD-3* : *Standby* est décomposé seulement en les états *Not_available* et *Available*. En effet, l'unique rôle de la fonction AD-3 est d'être

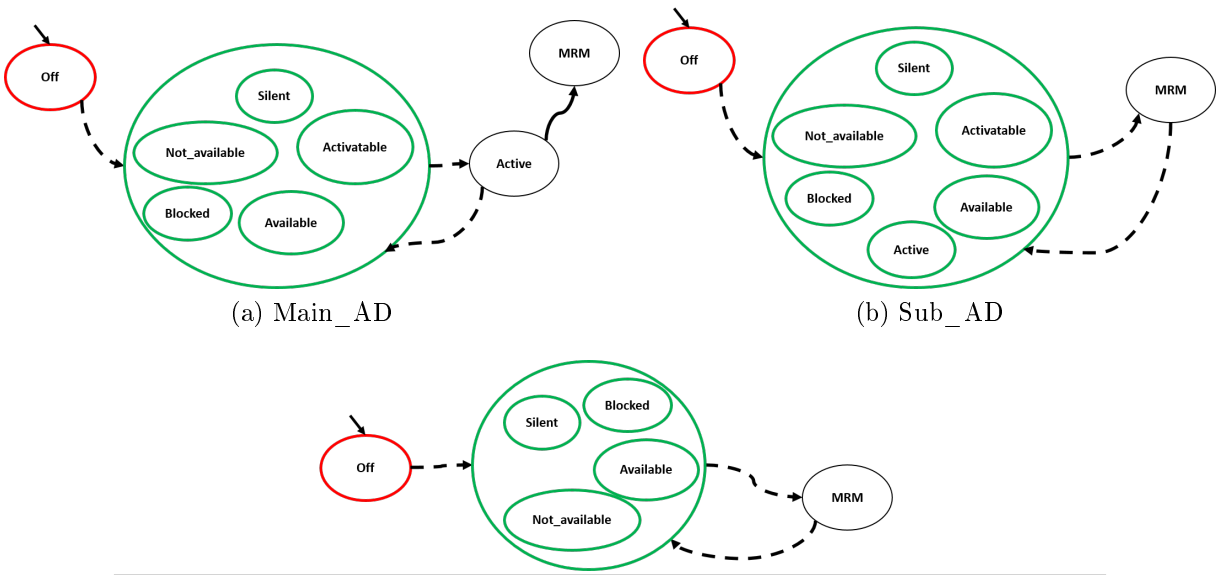
en mesure d'activer la fonction Control_3_MRM, qui procède alors aux manœuvres de sécurité adéquates, en cas de défaillance simultanée des fonctions Control_Main_Active et Control_Sub_Standby. Par conséquent, il s'agit simplement de savoir, pour la fonction AD-3, si la Supervision AD globale est dans l'état *Active* (état *Available* de la fonction AD-3) ou non (état *Not_available* de la fonction AD-3).

Par ailleurs, la fonction Main_AD ne comporte pas, dans le Modèle d'État de Sûreté déterminé à l'issue de l'activité **AS2** (voir figure 5.17), d'état *Standby*. Cependant, cette fonction ne doit pas se trouver dans l'état *Active* lorsque la Supervision AD globale et les autres Supervisions locales sont dans les états *Not_available*, *Available* ou *Activatable*. Ces états sont donc également attribués à la fonction Main_AD afin de faire converger les vues locale et globale. Ainsi, la liste d'états locaux attendus (**HM11**), dont une représentation graphique est donnée sur la figure 6.4, est la suivante :

— **Main_AD** : *Off*, *Not_available*, *Available*, *Activatable*, *Active*, *Silent*, *Blocked* et *MRM* ;

— **Sub_AD** : *Off*, *Not_available*, *Available*, *Activatable*, *Active*, *Silent*, *Blocked* et *MRM* ;

— **AD-3** : *Off*, *Not_available*, *Available*, *Silent*, *Blocked* et *MRM*.



(a) Main_AD (b) Sub_AD

(c) AD-3

FIGURE 6.4 – Représentation graphique de la liste d'états locaux

La définition des états *Not_available*, *Available*, *Activatable* est, pour chaque fonction, la même que celle donnée par la liste d'états attendus de la Supervision AD globale, dans le paragraphe 5.3.1. L'état *Silent* a aussi été défini précédemment. Enfin, l'état

Blocked est l'état dans lequel se met une Supervision locale lorsqu'une autre Supervision arrive dans l'état *Silent*, si la Supervision AD globale n'était pas dans l'état *Active* au préalable.

Remarque 6.2
Les états ***Silent*** *et* ***Blocked*** *sont tous deux des états puits, au même titre que l'état MRM.* ♢

Finalement, il convient d'établir une correspondance entre les états globaux et les états locaux. *A priori*, il existe 384 = 8.8.6 combinaisons d'états locaux possibles. Néanmoins, au lieu de considérer une à une les combinaisons locales, le raisonnement inverse a été mené, *i.e.* à quelle(s) combinaison(s) d'états locaux correspondent chacun des 13 états globaux ? L'hypothèse majeure que nous avons formulée est que, *idéalement*, à un état global doit correspondre une **unique combinaison d'états locaux**. L'objectif est effectivement de maîtriser l'état de la fonction de Supervision AD et d'être capable de déterminer, à tout moment, dans quels états les Supervisions locales doivent se trouver en fonction de l'état de la Supervision globale. De plus, les changements d'état des Supervisions locales s'effectuant suite à l'occurrence d'événements communs, ces changements doivent donc être synchronisés. Dans la réalité, il se peut qu'un même événement soit perçu à des instants différents par les Supervisions locales, ce qui engendre une désynchronisation dans les changements d'état. Mais, étant donné que ces décalages de perception de l'environnement ont des causes physiques (utilisation de capteurs différents, canaux de transmission distincts...), ils n'ont pas été pris en considération dans le cadre de notre étude.
En résumé, le but global est de **déterminer la fonction de transition** de chacun des automates locaux (fonctions Main_AD, Sub_AD et AD-3) de telle sorte que les uniques combinaisons d'états locaux possibles correspondent aux 13 combinaisons déterminées dans la Table 6.1, et ce quelle que soit l'évolution de l'environnement.
Pour chaque état global, nous avons donc proposé une unique combinaison d'états locaux. Ces relations sont restituées dans la Table 6.1 et constituent l'hypothèse de modélisation **HM12**. Pour les états *Off*, *Not_available*, *Available*, *Activatable* et *Active*, la Supervision globale et les Supervisions locales Main_AD et Sub_AD sont synchronisées. Seule la fonction AD-3 ne possède pas les mêmes états car la seule information pertinente pour cette fonction est de savoir si la Supervision AD globale est, ou n'est pas, dans l'état *Active*. Pour les états atteints suite à une défaillance d'une fonction de contrôle, la Supervision AD globale entre dans un état bloqué si la fonction de Conduite Autonome n'était pas active, ou *MRM* si la fonction était active. Chaque Supervision locale entre dans l'état *Silent*, si la fonction de contrôle qu'elle gère est défaillante, *Blocked*, si la fonction de contrôle qu'elle gère n'a pas à commander de manœuvres de sécurité ou *MRM* si la fonction de contrôle gérée doit piloter ces manœuvres.
Deux exemples, identifiés en caractères gras dans la Table 6.1, illustrent ces principes :

— *État global :* ***Activatable*** : dans cet état de la Supervision AD globale, les deux Supervisions locales Main_AD et Sub_AD doivent avoir vérifiées les mêmes conditions environnementales que la Supervision AD globale et se trouvent donc éga-

lement dans l'état *Activatable*. En revanche, étant donné que la Supervision AD globale n'est pas dans l'état *Active*, il n'est pas requis que la fonction AD-3 soit dans l'état *Available*. En outre, puisque l'événement d'alimentation électrique des Supervisions a eu lieu, comme en atteste l'état des autres Supervisions, la fonction AD-3 doit donc se trouver dans l'état *Not_ available* ;

— *État global : **MRM3*** : dans cet état, une défaillance a affecté la fonction de contrôle gérée par la fonction AD-3, alors que la conduite était entièrement déléguée au véhicule (donc la Supervision AD globale était dans l'état *Active*). En conséquence, cette dernière entre dans l'état *Silent*. La fonction Main_AD doit activer la fonction de contrôle réalisant les manœuvres de sécurité appropriées et passe pour ce faire dans l'état *MRM*. Enfin, la fonction Sub_AD doit empêcher une activation future, non désirée, de la fonction de contrôle qu'elle gère, et entre ainsi dans l'état *Blocked*.

État Supervision AD (global)	**État Main_AD (local)**	**État Sub_AD (local)**	**État AD-3 (local)**
Off	*Off*	*Off*	*Off*
Not_ available	*Not_ available*	*Not_ available*	*Not_ available*
Available	*Available*	*Available*	*Not_ available*
Activatable	***Activatable***	***Activatable***	***Not_ available***
Active	*Active*	*Active*	*Available*
MRMm	*Silent*	*MRM*	*Blocked*
MRMs	*MRM*	*Silent*	*Blocked*
MRM3	***MRM***	***Blocked***	***Silent***
MRMms	*Silent*	*Silent*	*MRM*
Blockedm	*Silent*	*Blocked*	*Blocked*
Blockeds	*Blocked*	*Silent*	*Blocked*
Blocked3	*Blocked*	*Blocked*	*Silent*
Blockedms	*Silent*	*Silent*	*Blocked*

TABLE 6.1 – Correspondance entre états globaux et états locaux

6.2.2.3 Comparaison des conditions et avis de l'expertise (A3.1.2, A4.1)

Le but de cette analyse est de mettre en regard les conditions définies par les exigences fonctionnelles de comportement retenues avec celles définies par les exigences fonctionnelles de SdF sélectionnées afin de déterminer l'éventualité de variables communes ou de réalités physiques (décrites par des variables portant des noms différents) similaires. Si cette situation se produit, il est pertinent de fusionner les variables en cause et de former ainsi un ensemble de variables partagées, à la fois par les AMS et les pilotes SdF.

Dans ce but, il convient d'étudier les deux Tables restituant les variables fonctionnelles et les variables de sûreté, situées en annexe (Tables 10.1 et 10.2). Après analyse des deux ensembles de variables, il ressort qu'aucune variable ne peut être fusionnée. En

effet, du point de vue des AMS, les variables correspondent à des données de l'environnement, mesurées par des capteurs ; tandis que, selon la perspective SdF, les variables sont relatives à des commandes émises par les fonctions de Contrôle AD. Ainsi, même si des variables semblent proches, comme la variable fonctionnelle *mean_accel* et les variables de sûreté *m_accel* et *s_accel*, elles diffèrent car *mean_accel* est la mesure de l'accélération moyenne du véhicule, alors que *m_accel* et *s_accel* sont les commandes d'accélération émises respectivement par la fonction Control_Main_Active et Control_Sub_Standby. Finalement, aucune hypothèse particulière n'est donc prise concernant les variables et les équations logiques (**HM13**, **HM14**).

En conclusion, les listes d'états attendus locaux et globaux, ainsi que la correspondance entre ces deux listes ont été soumises à la fois aux AMS et pilotes SdF durant l'activité **A4.1**. Ces nouvelles listes d'états attendus ont été jugées pertinentes, car elles permettent d'unifier le vocabulaire des deux métiers dans ce cas d'étude. Il a été toutefois noté l'absence d'homogénéité entre la définition des états *Blocked* (*Blockedm*, *Blockeds*, *Blocked3*, *Blockedms*) au niveau global et de l'état *Blocked* au niveau local. En effet, au niveau global, les états *Blocked* sont atteints uniquement lorsque la Supervision AD globale n'est pas dans l'état *Active* et suite à la défaillance d'une fonction de contrôle. Quant à l'état *Blocked* local, il peut être atteint par une Supervision locale même si la Supervision AD globale est dans l'état *Active*. Cette différence s'explique par le fait que lorsqu'une fonction de contrôle est défaillante alors que la Supervision AD globale est *Active*, les Supervisions locales qui ne gèrent ni la fonction de contrôle défaillante ni celle assurant les manœuvres de sécurité, doivent empêcher une activation future des fonctions de contrôle qu'elles gèrent. En effet, après l'occurrence d'une défaillance d'une fonction de contrôle, la fonction AD ne doit plus être proposée au conducteur avant réparation. C'est pourquoi ces Supervisions locales entrent dans l'état *Blocked*.

Par ailleurs, il a également été confirmé que les variables ne devaient pas être modifiées, malgré la potentielle lourdeur des modèles complets. Effectivement, puisqu'elles recoupent deux réalités différentes, elles ne peuvent pas être fusionnées.

À partir de ce vocabulaire unifié, en termes d'états et de conditions, il est désormais possible d'entreprendre les modifications des modèles d'état obtenus à l'issue des activités de construction (décrites dans le chapitre 5). Nous commençons donc par les Modèles d'État Fonctionnels, sur la base desquels nous bâtissons des Modèles Globaux cibles.

6.3 Détermination des Modèles Globaux cibles (A3.2)

Le but de cette activité est de définir des modèles de la Supervision AD globale cibles. Ces modèles constituent la spécification formelle du comportement attendu de la Supervision AD globale, à la fois selon le point de vue des AMS et selon celui des pilotes SdF. Puisque les Modèles d'État Fonctionnels spécifient le comportement de la Supervision AD globale, il est naturel de partir de ces modèles pour construire les Modèles Globaux cibles. En outre, ces modèles ont été construits dans un premier temps car nous disposons de l'opération de composition parallèle d'automates pour vérifier que les modifications apportées, lors de l'activité **A3.3**, aux Modèles d'État de Sûreté sont conformes aux Modèles Globaux (voir paragraphe 6.4.2).

6.3.1 Activités mises en jeu

Les activités déployées pour construire les Modèles Globaux cibles sont représentées sur la figure 6.5. Après avoir déterminé et fait valider la liste d'états attendus globaux lors de l'activité **A3.1**, il faut, pour construire les Modèles Globaux cibles, établir les transitions entre les états de cette liste en se fondant sur les Modèles d'État Fonctionnels résultants de l'activité **AF2**. Ceci est l'objet de l'activité **A3.2.1**. Ensuite, les Modèles Globaux ainsi obtenus sont exposés aux AMS et pilotes SdF qui décident si les modèles obtenus sont corrects ou non (**A4.2**). Enfin, la spécification, sous forme d'exigences, du comportement de la Supervision AD globale, doit être complétée au vu des modifications réalisées sur les Modèles d'État Fonctionnels. Ces exigences additionnelles, qui sont des exigences fonctionnelles de comportement de SdF, sont produites durant l'activité **A3.2.2**.

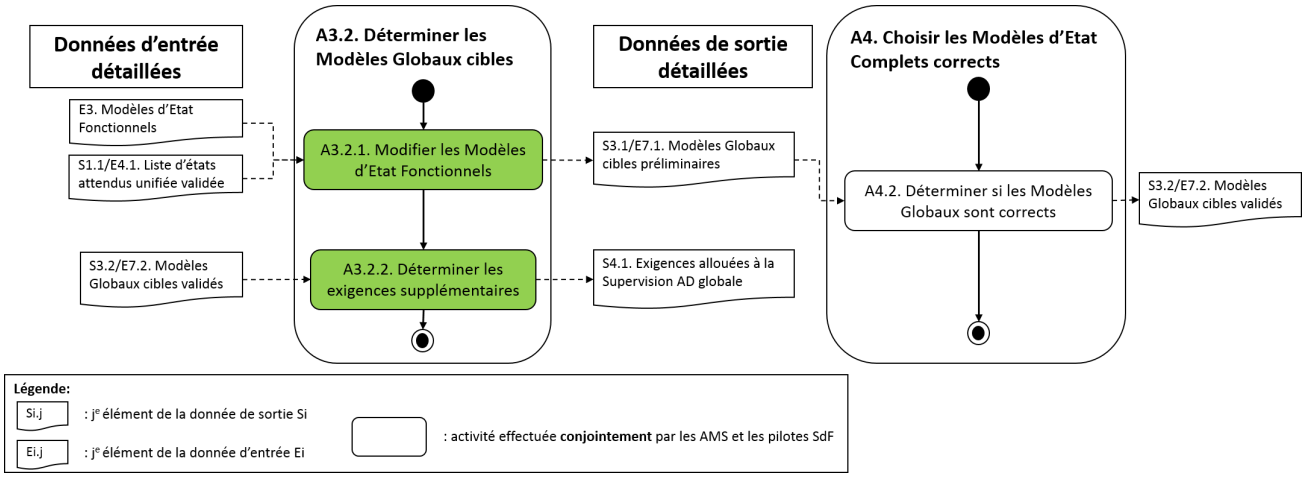


FIGURE 6.5 – Construction des Modèles Globaux cibles (**A3.2**)

6.3.2 Modification des Modèles d'État Fonctionnels (A3.2.1)

La liste d'états globaux attendus apporte 8 états supplémentaires par rapport à liste d'états attendus de la Supervision AD globale initiale : *Blockedm*, *Blockeds*, *Blocked3*, *Blockedms*, *MRMm*, *MRMs*, *MRM3* et *MRMms*. Pour aboutir aux Modèles Globaux à partir des Modèles d'État Fonctionnels, les deux points suivants doivent être examinés :

1 Existe-t-il des transitions entre les états supplémentaires introduits par la liste d'états globaux attendus ? Si oui, lesquelles ?

2 Quelles sont les transitions entre les états additionnels de la liste d'états globaux attendus et les états de la liste d'états attendus initiale ?

Pour le premier point, rien ne garantit *a priori* qu'il soit possible de passer d'un des huit états listés précédemment à un autre. Quant à la seconde question, il doit nécessairement exister au moins une transition entre un des états supplémentaires et les états des Modèles d'État Fonctionnels car tous les états doivent être atteignables. Nous traitons donc ces deux questions :

1 Il n'existe pas de transitions entre les états supplémentaires au vu de leurs définitions, données dans le paragraphe 6.2.2. En effet, tous ces états sont des états puits ;

2 S'agissant des transitions entre les états des Modèles d'État Fonctionnels et les états supplémentaires, il faut s'en référer à la définition des états additionnels ainsi qu'à la correspondance entre états globaux et locaux (voir Table 6.1). Voici les raisonnements tenus, pour chacun de ces états supplémentaires :

i. Les états **Blocked** (*Blockedm*, *Blockeds*, *Blocked3*, *Blockedms*) : ces état étant atteints suite à une défaillance d'une fonction de Contrôle AD lorsque la fonction AD est alimentée électriquement mais non active, ceci signifie que la Supervision AD peut alors être dans l'état *Not_available*, *Available* ou *Activatable*. Depuis chacun de ces états, l'événement de défaillance de la fonction :
   - Control_Main_Active mène à l'état *Blockdem* ;
   - Control_Sub_Standby mène à l'état *Blockeds* ;
   - Control_3_Standby mène à l'état *Blocked3* ;
   - « Control_Main_Active AND Control_Sub_Standby » (défaillance simultanée) mène à l'état *Blockedms*.

ii. **MRMm** : cet état est atteint suite à une défaillance de la fonction Control_Main_Active lorsque la fonction AD est alimentée électriquement et active, donc la Supervision AD se trouve dans l'état *Active*. La garde de la transition entre l'état *Active* et *MRMm* correspond logiquement à l'événement de défaillance de la fonction Control_Main_Active ;

iii. **MRMs** : cet état est atteint suite à une défaillance de la fonction Control_Sub_Standby lorsque la Supervision AD se trouve dans l'état *Active*. La garde de la transition entre l'état *Active* et *MRMs* correspond à l'événement de défaillance de la fonction Control_Sub_Standby ;

iv. **MRM3** : cet état est atteint suite à une défaillance de la fonction Control_3_Standby lorsque la Supervision AD se trouve dans l'état *Active*. La garde de la transition entre l'état *Active* et *MRM3* correspond à l'événement de défaillance de la fonction Control_3_Standby ;

v. **MRMms** : cet état est atteint suite à une défaillance simultanée des fonctions Control_Main_Active et Control_Sub_Standby lorsque la Supervision AD se trouve dans l'état *Active*. La garde de la transition entre l'état *Active* et *MRMms* correspond à l'événement de défaillance simultanée des fonctions Control_Main_Active et Control_Sub_Standby.

Cette détermination des transitions est valable et applicable pour les 6 Modèles d'État Fonctionnels étant donné qu'ils comportent tous les mêmes états (voir la figure 10.4 en annexe). La figure 6.6 offre une représentation graphique du Modèle Global cible préliminaire obtenu à partir du Modèle d'État Fonctionnel *SupAD_12*, représenté sur la figure 5.13. Les cinq autres Modèles Globaux, élaborés selon les mêmes principes, sont représentés sur la figure 10.8 en annexe. Sur ces modèles, les notations suivantes sont adoptées :

— *ctl_main_fail* correspond à une défaillance de la fonction Control_Main_Active ;
— *ctl_sub_fail* correspond à une défaillance de la fonction Control_Sub_Standby ;
— *clt_3_fail* correspond à une défaillance de la fonction Control_3_Standby ;
— *ctl_ms_fail* correspond une défaillance simultanée des fonctions Control_Main_Active et Control_Sub_Standby.

Remarque 6.3
*Le modèle apparaissant sur la figure 6.6 n'est délibérément pas doté d'une sémantique particulière. À l'instar des squelettes de modèles d'état, construits lors des activités précédentes (activité **AFS1.3.2**, voir paragraphe 5.5.4), ils constituent un support visuel facilitant le travail de validation des experts.* ♢

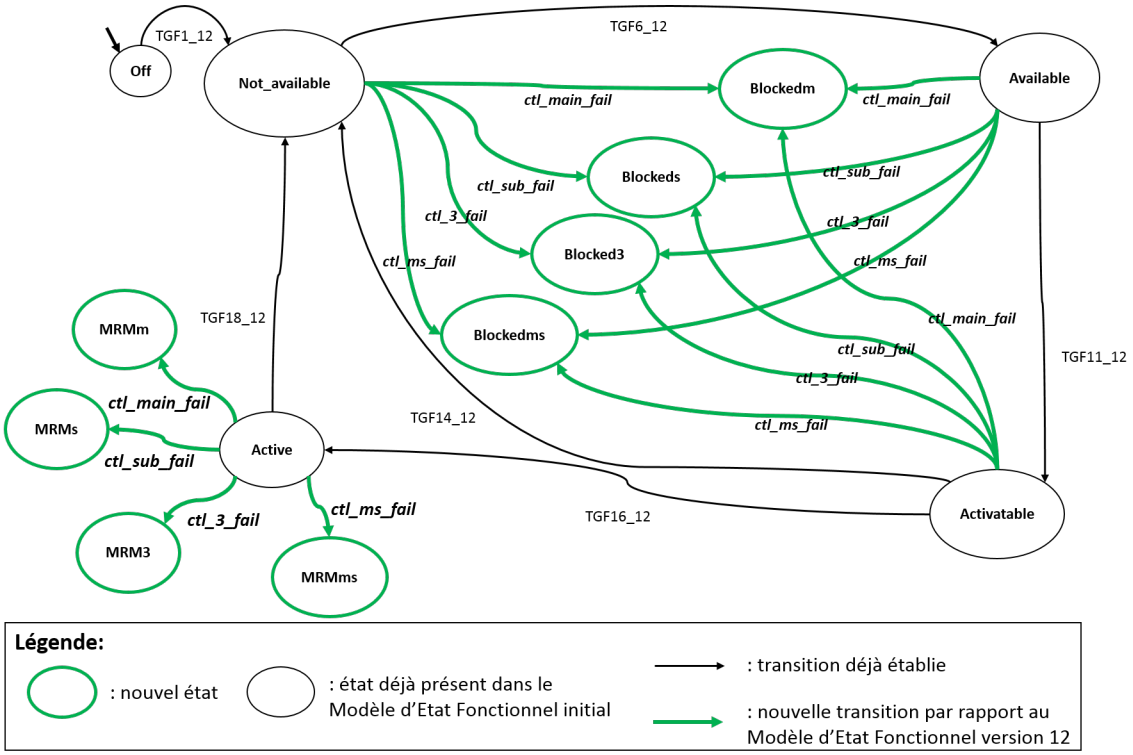


FIGURE 6.6 – Modèle Global cible préliminaire version 12

6.3.2.1 Validation des Modèles Globaux (A4.2)

Les représentations graphiques des Modèles Globaux préliminaires contribuent à la validation de ces modèles, durant l'activité **A4.2**. À la suite de la validation de la liste d'états globaux attendus, les modèles proposés ont également été validés par les deux ingénieries (voir fil conducteur, figure 6.1).

Néanmoins, l'approbation de ces différents modèles engendre la nécessité d'allouer des exigences supplémentaires à la Supervision AD globale. Contrairement aux activités précédentes (réalisées dans le chapitre 5), ce n'est pas le contenu des exigences fonctionnelles de comportement retenues qui est modifié, ni même l'ensemble de ces exigences, mais des exigences fonctionnelles de SdF supplémentaires qui sont allouées à la Supervision AD globale.

6.3.2.2 Exigences supplémentaires allouées à la Supervision AD globale (A3.2.2)

Nous procédons à la rédaction des exigences supplémentaires en examinant chacune des nouvelles transitions représentées sur les Modèles Globaux (figures 6.6 et 10.8 en annexe). En se rapportant à l'exemple du Modèle Global préliminaire 12, représenté sur la figure 6.6, nous illustrons le raisonnement adopté à l'aide de l'exigence **ESMBF6_1** (voir paragraphe 5.5.5). Cette exigence, initialement allouée à la fonction Main_AD, fait partie de la transition de groupe **TGSFM3_1**, associée à la transition de l'état *Active* à l'état *Off*, comme le montre la figure 5.17. Rappelons que l'exigence **ESMBF6_1** s'écrit initialement (voir paragraphe 5.5.5) :

ESMBF6_1 : *« If Main_AD is in the state **Active** and in case of a faulty deactivation of the AD function due to dysfunction of Main_AD function, then Main_AD shall switch to the state **Silent** »*

Remarque 6.4
*Dans l'énoncé de **ESMBF6_1**, l'état **Off** a été renommé en **Silent** conformément à la nouvelle liste d'états attendus, produite à l'issue de l'activité **A3.1.1**.* ♢

Puisque, nous l'avons vu, la Supervision AD globale passe de l'état *Active* à *MRMm* lorsque la fonction Main_AD passe de l'état *Active* à *Silent*, nous obtenons l'exigence supplémentaire suivante, allouée à la Supervision AD globale :

ESG1(ESMBF6_1) : *« If **AD Supervision** is in the state **Active** and in case of a faulty deactivation of the AD function due to dysfunction of Main_AD function, then **AD Supervision** shall switch to the state **MRMm** »*

où l'identifiant signifie qu'il s'agit de l'**E**xigence de **S**ûreté d'indice **1** allouée à la Supervision AD **G**lobale, issue de l'exigence **ESMBF6_1**. L'indice 1 peut être choisi pour cette exigence car aucune exigence de comportement de SdF n'était allouée à la Supervision AD globale.
Cet exemple illustre donc la ré-allocation d'exigences en termes de fonction. Le même principe est adopté pour toutes les nouvelles transitions des Modèles Globaux illustrés sur la figure 10.8 en annexe. Pour chaque modèle, il en résulte :

- **14 exigences** par transition suivante : ($Active \rightarrow MRMm$), ($Active \rightarrow MRMs$), ($Active \rightarrow MRM3$), ($Active \rightarrow MRMms$), soit un total de **56 (= 14.4) exigences** ;

— **14 exigences** par transition suivante : (*Not_ available* → *Blockedm*), (*Available* → *Blockedm*), (*Activatable* → *Blockedm*), (*Not_ available* → *Blockeds*), (*Available* → *Blockeds*), (*Activatable* → *Blockeds*), (*Not_ available* → *Blocked3*), (*Available* → *Blocked3*), (*Activatable* → *Blocked3*), (*Not_ available* → *Blockedms*), (*Available* → *Blockedms*), (*Activatable* → *Blockedms*), soit un total de **168 (= 14.12) exigences**.

L'enrichissement de l'ensemble d'exigences initialement allouées à la Supervision AD globale clôt donc l'activité de construction des Modèles Globaux cibles. Il convient désormais d'apporter les changement appropriés au Modèle d'État de Sûreté résultant de l'activité **AS2**.

6.4 Enrichissement des Modèles de Sûreté (A3.3)

Les 6 Modèles Globaux cibles définissent le comportement attendu de la Supervision AD globale selon les points de vue des AMS et des pilotes SdF. Or, les Modèles d'État de Sûreté issus des activités **AS1** et **AS2** n'intègrent pas d'aspects liés à la perspective des AMS. Les comportements locaux doivent toutefois être cohérents avec les comportements globaux. Nous proposons de vérifier cette cohérence en s'appuyant sur l'opération de composition parallèle des automates finis.

6.4.1 Aperçu des activités

Dans un premier temps (**A3.3.1** sur la figure 6.7), les Modèles d'État de Sûreté sont modifiés par l'ajout de transitions et d'états, à partir de la liste d'états locaux et des Modèles Globaux cibles. La composition parallèle des automates finis constituant les modèles locaux est ensuite entreprise, durant l'activité **A3.3.2**. Chaque Modèle Local (comprenant 3 automates) est construit selon une version d'un Modèle d'État Fonctionnel (donc d'un Modèle Global cible également). Pour une version donnée, un Modèle Local est considéré comme correct si l'automate résultant de la composition parallèle des 3 automates qui le constituent est identique (en termes d'états, transitions et événements) à l'automate du Modèle Global cible de la même version. Chaque Modèle d'État Complet obtenu à l'issue de l'activité **A3.3.2** est donc composé d'un Modèle Global et d'un Modèle Local, soit un total de quatre automates. Ces Modèles d'État Complets sont ensuite soumis aux AMS et aux pilotes SdF pour validation, durant l'activité **A4.3**. Enfin, au même titre que l'activité A3.2, les modifications apportées aux Modèles d'état de Sûreté causent la ré-allocation d'exigences supplémentaires allouées aux Supervisions locales (activité **A3.3.3**).

6.4.2 Modification des Modèles d'État de Sûreté (A3.3.1)

Afin de compléter chacun des automates constituant le Modèle d'État de Sûreté obtenu à l'issue de l'activité **AS2**, et représenté sur la figure 5.17, nous employons la même méthode que celle utilisée pour obtenir les Modèles Globaux à partir des Modèles d'État Fonctionnels (paragraphe 6.3.2). La figure 6.8 illustre le Modèle Local obtenu à partir de la version 12 de Modèle d'État Fonctionnel. Pour y parvenir, nous avons repris, en

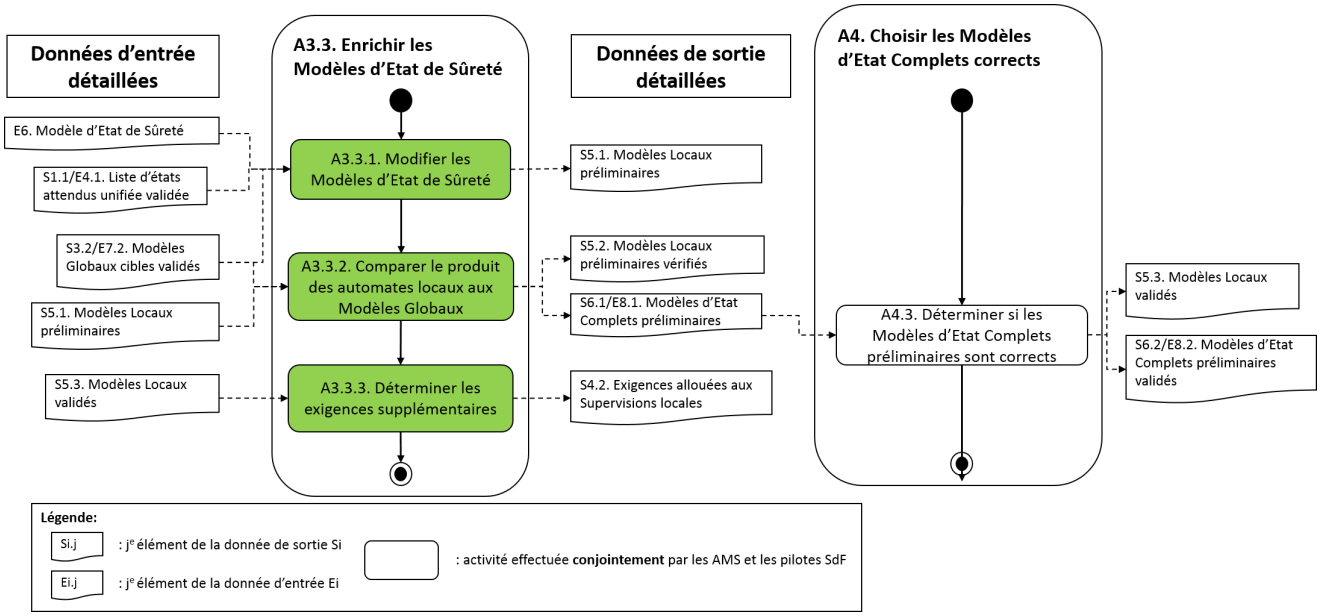


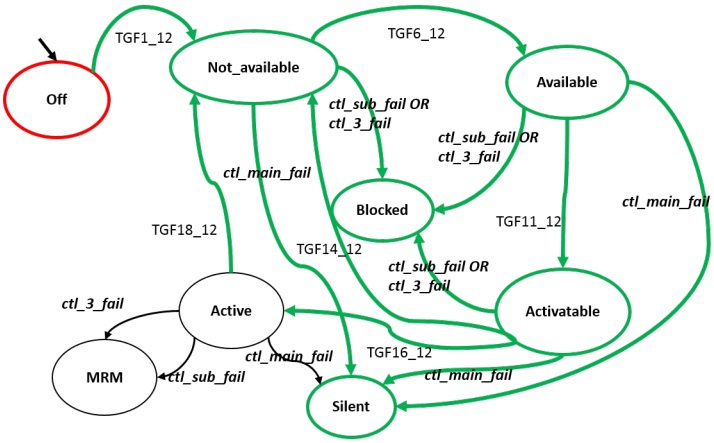
FIGURE 6.7 – Enrichissement des Modèles d'État de Sûreté (**A3.3**)

les adaptant, les deux points soulevés dans le paragraphe 6.3.2 relatifs aux transitions :

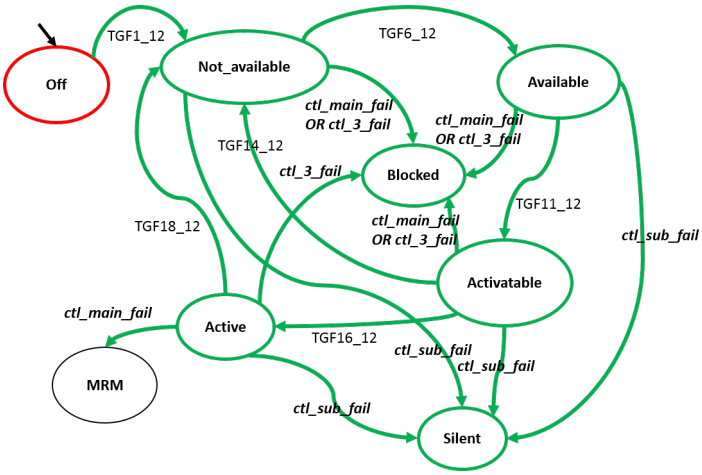
(a) ***Transitions entre les états supplémentaires introduits par la liste d'états locaux attendus ?***
Contrairement aux Modèles d'État Fonctionnels, il existe des transitions entre les états supplémentaires introduits par la liste d'états attendus locaux, c'est-à-dire, pour les fonctions Main_AD et Sub_AD, les états *Off*, *Not_available*, *Available*, *Activatable*, *Silent* et *Blocked*, et, pour la fonction AD-3, les états *Off*, *Available*, *Silent* et *Blocked*. Concernant les états *Off*, *Not_available*, *Available* et *Activatable*, les transitions entre ces états sont en fait déjà définies dans chacune des versions des six Modèles Globaux validés, produits de l'activité **A3.2**. Ainsi, six Modèles Locaux sont également élaborés, et, pour chacun de ces modèles, les transitions entre les états cités sont déduites de celles de la version du Modèle Global cible considéré. Par exemple, en prenant en compte un modèle global cible, noté *MG_ij* :

— *Pour les fonctions Main_AD et Sub_AD* : les transitions entre les états *Off*, *Not_available*, *Available* et *Activatable* sont celles du modèle MG_ij ;

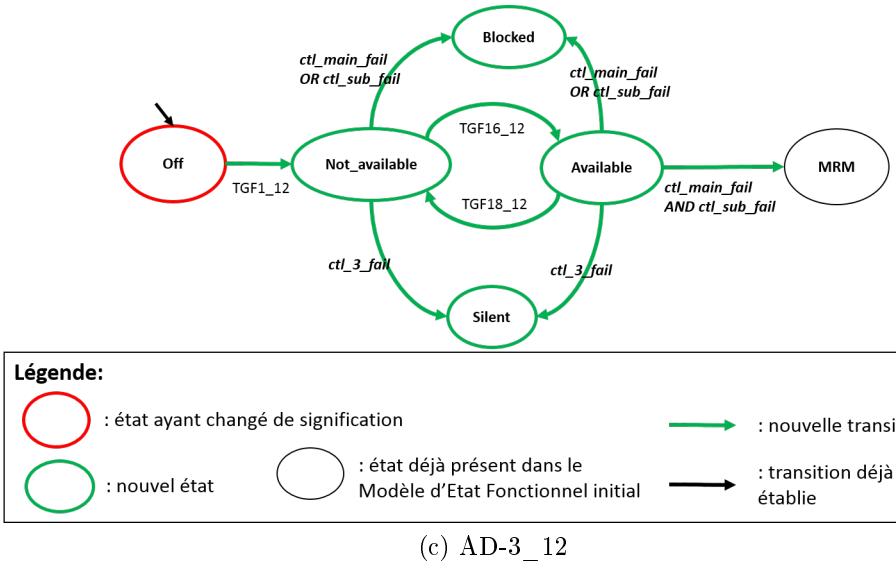
— *Pour la fonction AD-3* : la transition entre l'état *Off* et *Not_available* est celle du modèle MG_ij. En revanche, la transition entre l'état *Not_available* et *Available* correspond à la transition entre l'état *Activatable* et *Active* de MG_ij. Effectivement, lorsque la Supervision AD globale est dans l'état *Active*, la fonction AD-3 doit être dans l'état *Available* (voir Table 6.1). Quant à la transition entre l'état *Available* et l'état *Not_available* de la fonction AD-3, elle correspond, pour la même raison, à la transition entre l'état *Active* et l'état *Not_available* de MG_ij.



(a) Main_AD_12



(b) Sub_AD_12



(c) AD-3_12

FIGURE 6.8 – Modèle d'État Local obtenu préliminaire version 12

Les états ***Silent*** et ***Blocked*** ne peuvent, en revanche, être traités de la même manière. Effectivement, ces états sont liés à la disponibilité de fonctions de contrôle particulières. Les deux règles suivies pour construire les transitions permettant d'atteindre ces deux états (on ne peut en sortir car ce sont des états puits) sont les suivantes :

i. L'état *Silent* est atteint depuis n'importe quel état (sauf les états *Off* et *MRM*) de chaque sous-fonction de Supervision, suite à la défaillance de la fonction gérée par la Supervision en question (Control_Main_Active pour Main_AD, Control_Sub_Standby pour Sub_AD et Control_3_Standby pour AD-3) ;

ii. Les états *Blocked* sont atteints depuis :

   — Les états pour lesquels la Supervision AD globale est alimentée et non *Active* (*Not_ available*, *Available* et *Activatable* pour les fonctions Main_AD et Sub_AD et *Not_ available* pour AD-3) suite à la défaillance d'une fonction de contrôle non gérée par la Supervision locale considérée. Par exemple, la fonction Sub_AD entre dans l'état *Blocked* (depuis les états *Not_ available*, *Available* et *Activatable*) suite à une défaillance de la fonction Control_Main_Active ou Control_3_Standby ;

   — Les états pour lesquels la Supervision AD globale est alimentée et *Active* (*Active* pour Main_AD et Sub_AD et *Available* pour AD-3) suite à une défaillance d'une fonction de contrôle non gérée par la Supervision locale considérée et ne nécessitant pas un passage dans l'état *MRM* de cette même Supervision. Ainsi la fonction Sub_AD, lorsqu'elle est dans l'état *Active*, entre dans l'état *Blocked* seulement après une défaillance de la fonction Control_3_Standby. En effet, une défaillance de la fonction Control_Main_Active provoque une entrée dans l'état *MRM* de la fonction Sub_AD.

(b) ***Transitions entre les états supplémentaires introduits par la liste d'états locaux attendus et les états de la liste d'états de sûreté initiale ?***
En ce qui concerne ces transitions, il faut raisonner automate par automate. Pour un Modèle Global, noté MG_ij :

— **Main_AD** : les états initialement présents étaient : *Off*, *Active* et *MRM* :
   — *Off* : cet état a été remplacé par l'état *Silent*, que nous avons déjà traité ;
   — *Active* : puisque cet état est également défini du point de vue des AMS, les transitions entre l'état *Active* et les autres états de l'automate sont celles du modèle MG_ij considéré ;
   — *MRM* : cet état est atteint depuis l'état *Active* suite à la défaillance d'une fonction de contrôle non gérée par la fonction Main_AD (Control_Sub_Standby ou Control_3_Standby).

— **Sub_AD** : les états initialement présents étaient : *Off*, *Standby* et *MRM* :
   — *Off* : cet état a été remplacé par l'état *Silent*, que nous avons déjà traité ;
   — *Standby* : cet état ne fait plus partie des états locaux. Il a été « décomposé » selon les états *Not_ available*, *Available*, *Activatable* et *Active* ;

- *MRM* : cet état est atteint depuis l'état *Active* suite à la défaillance de la fontion de contrôle gérée par la fonction Main_AD (Control_Main_Active).

— **AD-3** : les états initialement présents étaient : *Off*, *Standby* et *MRM* :

- *Off* : cet état a été remplacé par l'état *Silent*, que nous avons déjà traité ;
- *Standby* : cet état ne fait plus partie des états locaux. Il a été « décomposé » selon les états *Not_available* et *Available* ;
- *MRM* : cet état est atteint depuis l'état *Available* suite à la défaillance simultanée des fonctions de contrôle gérées par les fonctions Main_AD et Sub_AD (Control_Main_Active et Control_Sub_Standby).

6.4.3 Vérification des Modèles Locaux par rapport aux Modèles Globaux cibles et validation (A3.3.2, A4.3)

6.4.3.1 Outil *Supremica*

Pour entreprendre la vérification de la cohérence entre les Modèles Locaux et les Modèles Globaux cibles, nous nous reposons sur l'opération de composition parallèle d'automates finis. En effet, cette opération offre la possibilité de générer le comportement global de plusieurs automates locaux concurrents. Nous disposons justement du comportement global objectif (les Modèles Globaux cibles) et des comportements locaux (les Modèles Locaux). Pour procéder à cette opération, nous avons utilisé l'outil Supremica, qui est, tout comme UPPAAL, un outil reconnu (Rohée et al. 2006 ; Markovski et Van de Mortel-Fronczak 2012 ; Mohajerani, Malik et Fabian 2017). Il convient, en premier lieu, de préciser l'usage que nous avons eu de cet outil car les objets manipulés par UPPAAL et Supremica ne sont pas les mêmes. En effet, dans l'outil Supremica, les automates manipulés sont des tuplets $A = (Q, q_i, Q_x, Q_m, \Sigma, \delta)$ (Akesson et al. 2003) où :

- Q est un ensemble fini d'états ;
- $q_i \in Q$ est l'état initial ;
- $Q_x \subset Q$, $q_i \notin Q_x$ est l'ensemble des états interdits ;
- $Q_m \subset Q$ est l'ensemble des états marqués ;
- Σ est l'alphabet fini des événements ;
- $\delta : Q \times \Sigma \times Q$ est la fonction de transition définie telle que $\delta(q_k, \sigma) = q_l$ avec $\sigma \in \Sigma$, q_k, $q_l \in Q$.

Les notions d'*état interdit* et, dans une moindre mesure, d'*état marqué*, font référence à la Théorie du Contrôle par Supervision, initiée par P.J.Ramadge et W.M.Wonham dès 1982 (Ramadge et Wonham 1982). Le principe de base de cette théorie consiste à générer, à partir de la description du comportement d'un système libre et de celle des contraintes (ou exigences) allouées à ce système, le tout sous forme d'automates finis, un ensemble de *Superviseurs*. Ces derniers permettent de spécifier un contrôleur autorisant le plus grand nombre de comportements possibles du système tout en respectant les spécifications. Le logiciel Supremica est premièrement dévolu à l'application de cette théorie.

Cependant, dans notre cas d'étude, nous avons eu recours à Supremica essentiellement pour le fait que, contrairement à UPPAAL (et, qui plus est, NuSMV), Supremica offre une vue graphique de l'automate résultant de la composition parallèle d'automates à états finis. Les comparaisons entre les automates résultants de la composition parallèle des automates constituant les Modèles Locaux avec les Modèles Globaux reposent justement sur ces représentations. En règle générale, les objets manipulés dans Supremica sont sémantiquement différents de ceux utilisés dans UPPAAL. (Bart Koolmees 2011) montre que la différence entre ces deux formalismes repose dans le traitement des *Événements* (Supremica).

Effectivement, un événement, dans Supremica, n'est possible uniquement s'il est autorisé dans chaque automate synchronisé. Néanmoins, dans l'outil UPPAAL, il existe deux types de canaux (*« channels »*) au travers desquels les automates peuvent communiquer : les canaux dits binaires (*« binary channels »*) et les canaux de diffusion (*« broadcast channels »*). La communication via un canal binaire n'est possible que si les deux automates impliqués (il ne peut pas y en avoir plus) l'autorisent, *i.e.* se trouvent dans un état dans lequel ils sont capables, l'un d'émettre, et l'autre de recevoir, le signal. En revanche, pour les canaux de diffusion, qui impliquent deux ou plusieurs automates, le signal partagé peut être émis même si tous les automates ne sont pas en état de le recevoir. Or, dans l'outil Supremica, un événement ne peut avoir lieu que s'il est autorisé dans chaque automate synchronisé. Toutefois, pour les automates que nous avons manipulés, seuls des canaux de communication binaires ont été employés. C'est pourquoi nous n'avons pas rencontré ces difficultés particulières de traduction.

Plus précisément, les objets que nous avons manipulés, dans l'outil Supremica, sont les automates : $A1 = (Q,\ q_i,\ \Sigma,\ \delta)$ et, dans UPPAAL, les tuplets : $A2 = (L, l_0, V, A, E)$ (voir paragraphe 5.6.3). Il est donc nécessaire d'établir des relations entre ces deux objets.

La traduction des places L (y compris la place initiale l_0) apparaissant dans le modèle UPPAAL en états dans le logiciel Supremica ne posent pas de problèmes particuliers. En effet, la place initiale $l_0 \in L$ dans UPPAAL correspond à l'état initial $q_i \in Q$ dans Supremica, tandis qu'une place quelconque $l \in L$ dans l'environnement UPPAAL sera traduite par un état $q \in Q$, également quelconque (portant, de préférence, le même nom). En revanche, la traduction des *Actions* (A) et des *Arcs* (E) cause plus de difficultés.

Rappelons (voir paragraphe 5.6.3), que dans notre cadre de travail, les *Actions* sont des signaux transitant entre les Supervisions locales. Par ailleurs, toujours pour notre cas d'étude, les *Arcs* contiennent des gardes, qui sont des équations logiques, des canaux de synchronisation, modélisant les événements de défaillance, et des mises à jour de variables. Les *Actions* et les gardes des transitions sont traduites, dans l'outil Supremica, par des *événements*. Ainsi, dans l'environnement initial (UPPAAL), les gardes des transitions peuvent être vraies pendant une certaine durée, alors que dans la modélisation de Supremica, ceci n'est pas possible car un événement n'a pas de durée. Dans Supremica, seul l'instant auquel la garde d'une transition devient vraie pourra donc être modélisé. Dans notre contexte, étant donné que nous sommes focalisés sur les changements d'état, cette distinction n'a pas d'incidences. Enfin, les *canaux de synchronisation*, définissant des supports de communication pour l'occurrence des *Actions*

dans UPPAAL, n'ont pas à être déterminés explicitement dans Supremica, car un événement peut s'y produire sans que sa source ne soit identifiée. De plus, puisque nous ne manipulons pas de variables dans Supremica, aucune mise à jour n'est requise.

En dernier lieu, il convient de noter un *changement de paradigme* entre l'environnement UPPAAL et Supremica : dans le premier, la modélisation est **synchrone** dans le sens où les valeurs des variables peuvent évoluer au même instant. C'est pourquoi, depuis une place, deux gardes de deux transitions différentes peuvent *a priori* être vraies simultanément. En revanche, dans l'environnement Supremica, la modélisation est **asynchrone** dans le sens où un seul événement peut avoir lieu à un instant donné. Or, comme nous l'avons vu dans le paragraphe 5.6.4, les gardes des transitions des modèles sont construites de telle sorte que deux (ou plus) gardes de deux transitions partant d'une même place ne puissent pas être vraies au même moment. Dans ce cadre, la traduction d'une garde de transition par un événement ne pose pas de problème relatif au changement de paradigme (asynchrone, synchrone) évoqué. La figure 6.9 illustre ce raisonnement à l'aide de la version 12 du Modèle d'État Fonctionnel, présentée dans le paragraphe 5.6.3.

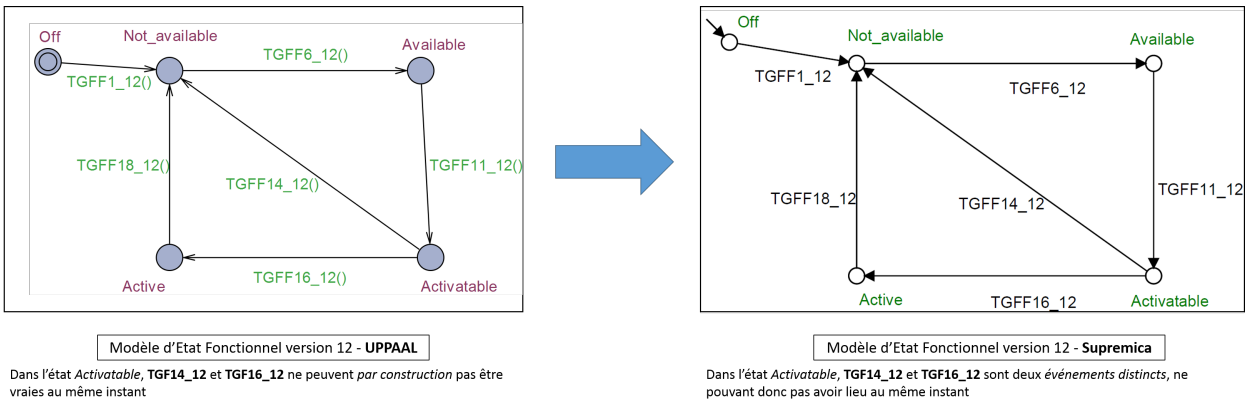


FIGURE 6.9 – Traduction de l'environnement UPPAAL vers Supremica

6.4.3.2 Mise en œuvre de la composition parallèle des automates des Modèles Locaux

Les Modèles Globaux cibles et les Modèles Locaux sont retranscrits dans l'outil Supremica. Nous avons représenté, respectivement sur les figures 6.10 et 6.11, la version 12 du Modèle Local (dénoté ML_12, dont la version préliminaire avait été fournie sur la figure 6.8) et la version 12 du Modèle Global cible (noté MG_12, dont la version préliminaire avait été donnée sur la figure 6.6). Par rapport au Modèle Local préliminaire, notons qu'un événement supplémentaire, noté *ctl_ms_fail*, a dû être ajouté dans les automates. En effet, nous avions indiqué, dans les automates préliminaires, la possibilité que les événements *ctl_main_fail* et *ctl_sub_fail* surviennent au même instant (défaillance de mode commun) par la garde suivante : *ctl_main_fail AND ctl_sub_fail* (voir figure 6.8). Or, comme nous l'avons vu plus haut, il n'est pas possible que deux événements distincts aient lieu au même moment dans Supremica.

Le Modèle Global cible version 12, illustré sur la figure 6.11, donne une représentation explicite du résultat que l'on doit obtenir (si le Modèle Local est correct) en procédant à la composition parallèle des automates du Modèle Local version 12. La figure 6.12 offre justement une vue de l'automate résultant de cette opération de composition parallèle (des trois automates représentés sur la figure 6.10, correspondant au Modèle Local version 12). Formellement, l'automate $A = (Q, q_i, \Sigma, \delta)$ résultant de la *composition parallèle* de deux automates $A^1 = (Q^1, {q_i}^1, \Sigma^1, \delta^1)$ and $A^2 = (Q^2, {q_i}^2, \Sigma^2, \delta^2)$ est ((Cassandras et Lafortune 2009)) :

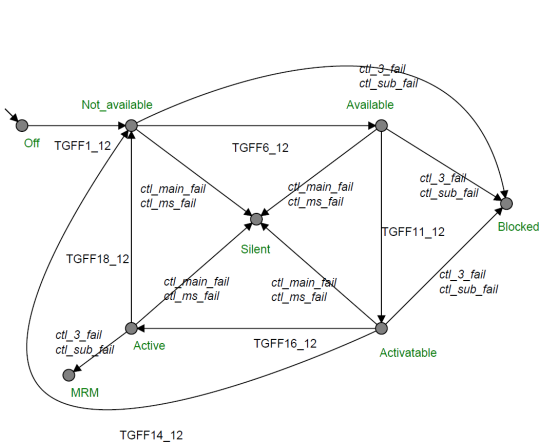
$$\boldsymbol{A = A^1 \| A^2 = Ac(Q^1 \times Q^2, (q_i^1, q_i^2), \Sigma^1 \cup \Sigma^2, \delta)} \quad avec :$$

$\mathbf{Q} = Q^1 \times Q^2$;
$\mathbf{q_i} = (q_i^1, q_i^2)$;
$\mathbf{\Sigma} = \Sigma^1 \cup \Sigma^2$;

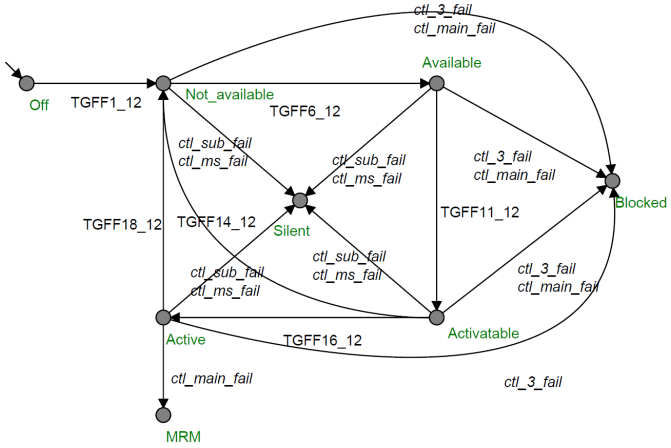
$$\boldsymbol{\delta((q^1, q^2), \sigma)} = \begin{cases} \boldsymbol{(\delta^1(q^1, \sigma), \delta^2(q^2, \sigma))} \\ si \quad \delta^1(q^1, \sigma)! \quad et \quad \delta^2(q^2, \sigma)! \\ \\ \boldsymbol{(\delta^1(q^1, \sigma), q^2)} \\ si \quad \delta^1(q^1, \sigma)! \quad et \quad \sigma \notin \Sigma^2 \\ \\ \boldsymbol{(q^1, \delta^2(q^2, \sigma))} \\ si \quad \delta^2(q^2, \sigma)! \quad et \quad \sigma \notin \Sigma^1 \\ \\ \textbf{non définie sinon} \end{cases}$$

Où *!* signifie « existe ».

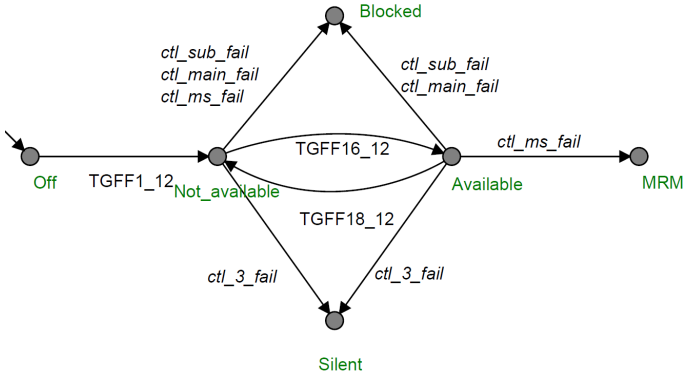
Sur la figure 6.12, les états globaux sont également représentés, en dessous de la combinaison d'états locaux auxquels ils correspondent. Cette association est établie au moyen de la Table 6.1, restituant les relations entre états globaux et locaux. En fin de compte, il apparaît que les automates de la figure 6.11, illustrant le Modèle Global cible version 12, et celui de la figure 6.12, produit de la composition parallèle des automates du Modèle Local version 12, sont identiques en termes d'états, de transitions et d'événements. La cohérence entre les vues locale et globale est, de ce fait, garantie. Nous n'avons présenté le travail que pour la version 12 du Modèle d'État Fonctionnel, obtenu à l'issue de l'activité **AF2**. Mais le même travail a été effectué pour chacune des cinq autres versions possibles de ce modèle. Chaque couple : (Modèle Global version *ij*, Modèle Local version *ij*) constitue un Modèle Complet version *ij*.



(a) Main_AD_12



(b) Sub_AD_12



(c) AD-3_12

FIGURE 6.10 – Modèle d'État Local version 12 (ML_12)

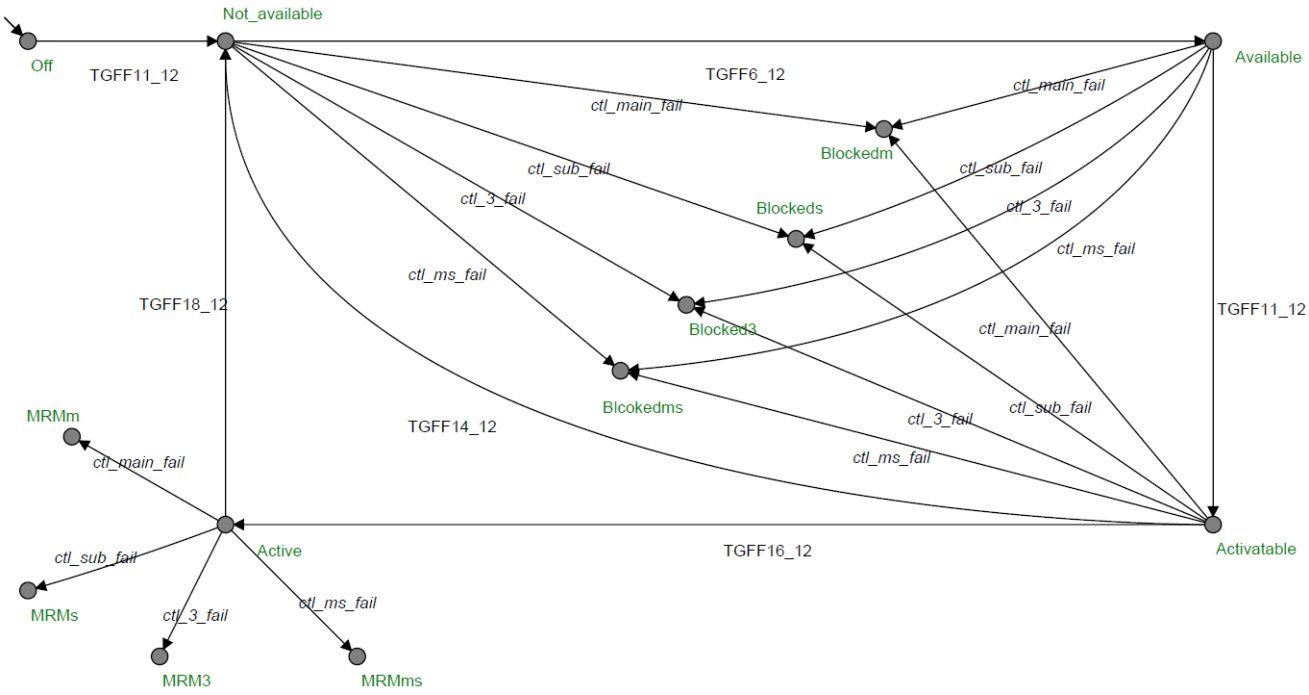


FIGURE 6.11 – Modèle d'État Global cible version 12 (MG_12)

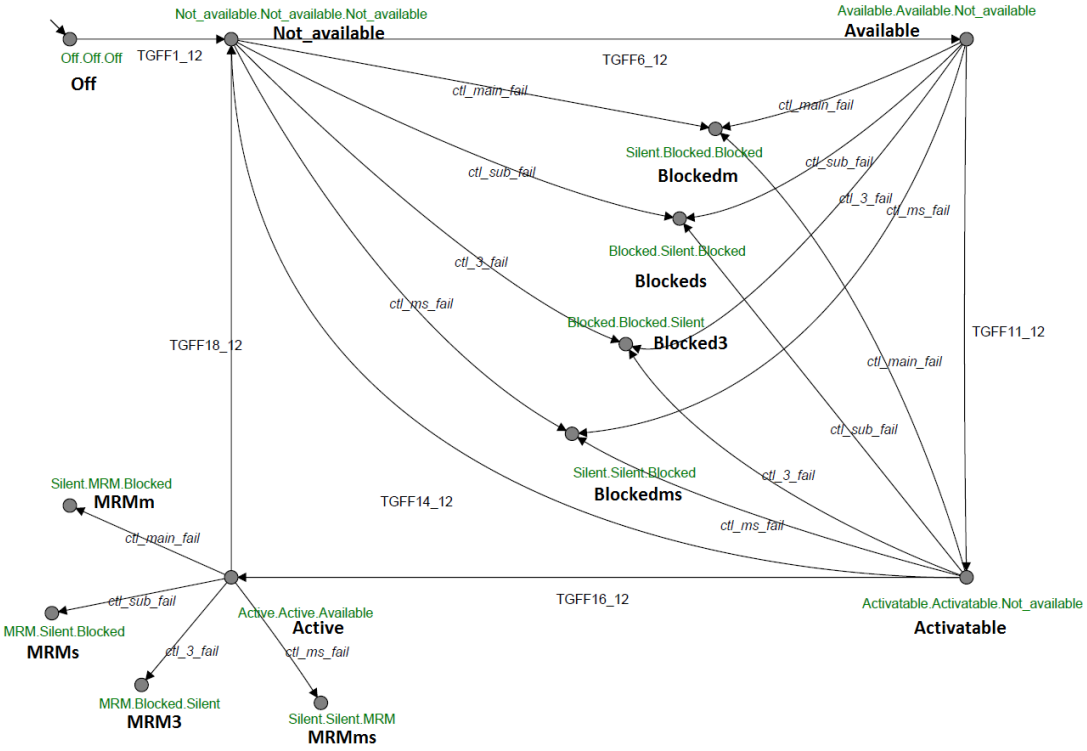


FIGURE 6.12 – Automate résultant de la composition parallèle des automates du Modèle Local version 12

6.4.3.3 Validation des modèles (A4.3)

Bien que la cohérence entre les vues locale et globale soit assurée par l'opération formelle de composition parallèle d'automates finis, les modifications apportées aux Modèles d'État de Sûreté sont importantes et méritent à ce titre d'être analysées par les deux acteurs métier. L'élément additionnel principal, par rapport à la liste d'états locaux attendus et à la correspondance entre états locaux et globaux, déjà validées lors de l'activité **A4.1**, correspond aux nouvelles transitions identifiées sur la figure 6.8. Ces transitions, bien que très nombreuses, découlent logiquement de la définition des états locaux supplémentaires ou de la nouvelle définition d'états de la liste d'états de sûreté attendus initiale. C'est la raison pour laquelle les six Modèles d'État Complets obtenus à l'issue de l'activité **A3.3.2** ont été validés par les AMS et les pilotes SdF.

6.4.4 Exigences supplémentaires allouées aux Supervisions locales (A3.3.3)

Pour former les exigences additionnelles allouées aux Supervisions locales et découlant des modifications effectuées sur les Modèles d'État de Sûreté, nous adoptons la même méthode que pour la Supervision AD globale. Les exigences supplémentaires sont donc déterminées en examinant, pour chaque automate d'une Supervision locale, les nouvelles transitions identifiées (voir figure 6.8), ainsi que la correspondance entre les transitions de groupe (fonctionnelles et de sûreté) et les groupes bien formés. Au même titre que pour les exigences supplémentaires allouées à la Supervision AD globale (voir paragraphe 6.3.2), nous illustrons également ce processus de ré-allocation à l'aide des deux exigences exemples du paragraphe 5.5.5. Nous considérons la version 33 du Modèle Local, dont la fonction Main_AD est illustrée sur la figure 6.13 :

1 L'exigence fonctionnelle de comportement bien formée **EFBF6_1**, illustre à nouveau la ré-allocation en termes de *fonction* ;

2 L'exigence fonctionnelle de SdF bien formée **ESMBF6_1**, qui met en en oeuvre une ré-allocation en termes d'*espace d'états*.

L'exigence **EFBF6_1** appartient à la transition de groupe **TGF6_33** du Modèle Global version 33 (voir figure 10.8), et son énoncé est le suivant :

EFBF6_1 : *« If AD Supervision is in the state* ***Not_available*** *at night without impact on Autonomous Vehicle performance, then AD SupervisionS shall be in the state* ***Available*** *»*

Étant donné que, lorsque la Supervision AD globale passe de l'état *Not_available* à l'état *Available*, la fonction Main_AD doit passer également de l'état *Not_available* à *Available* (voir Table 6.1) et que ces deux états ont la même définition pour les fonctions Supervision AD et Main_AD, on en déduit que le changement d'état en question s'opère sous les mêmes conditions. En conséquence, l'exigence supplémentaire, déduite de **EFBF6_1**, et allouée à la fonction Main_AD, s'écrit de la façon suivante :

EFM1(EFBF6_1) : *« If* ***Main_AD*** *is in the state* ***Not_available*** *at night without impact on Autonomous Vehicle performance, then* ***Main_AD*** *shall be in the*

*state **Available** »*

où l'identifiant signifie qu'il s'agit de l'**E**xigence **F**onctionnelle de comportement d'indice **1** allouée à la fonction **M**ain_AD, issue de l'exigence **EFBF6_1**. L'indice 1 peut être choisi pour cette exigence car aucune exigence fonctionnelle de comportement n'était initialement allouée à la fonction Main_AD.
De nouveau, un processus de ré-allocation *fonctionnelle* est ainsi illustré. En revanche, le traitement de l'exigence **ESMBF6_1** met en jeu une ré-allocation en termes d'espace d'états. En effet, rappelons tout d'abord l'énoncé de l'exigence **ESMBF6_1** :

ESMBF6_1 : *« If Main_AD is in the state **Active** and in case of a faulty deactivation of the AD function due to dysfunction of Main_AD function, then Main_AD shall switch to the state **Silent** »*

Puisqu'une réaction similaire (*i.e.* l'entrée dans l'état *Silent*) depuis les états *Not_available*, *Available*, *Activatable* suite à une défaillance de la fonction de contrôle gérée par la fonction Main_AD, a été définie, trois exigences supplémentaires, allouées à la fonction Main_AD, ont été produites :

- **ESM42(ESMBF6_1)** : *« If Main_AD is in the state **Not_available** and in case of a faulty deactivation of the AD function due to dysfunction of Main_AD function, then Main_AD shall switch to the state **Silent** »*
- **ESM43(ESMBF6_1)** : *« If Main_AD is in the state **Available** and in case of a faulty deactivation of the AD function due to dysfunction of Main_AD function, then Main_AD shall switch to the state **Silent** »*
- **ESM44(ESMBF6_1)** : *« If Main_AD is in the state **Activatable** and in case of a faulty deactivation of the AD function due to dysfunction of Main_AD function, then Main_AD shall switch to the state **Silent** »*

où les identifiants signifient qu'il s'agit des **E**xigences de **S**ûreté d'indices **42**, **43** et **44** allouées à la fonction **M**ain_AD, toutes issues de l'exigence **ESMBF6_1**. Puisque 41 exigences étaient, à l'issue de l'activité **AS2** (voir figure 4.8), allouées à la fonction Main_AD (voir paragraphe 5.6.4), le premier indice que peut prendre une exigence de comportement de SdF allouée à Main_AD est l'indice 42.
Le même principe est ensuite appliqué pour obtenir toutes les autres exigences supplémentaires. Voici la quantité d'exigence supplémentaires obtenues pour la fonction Main_AD de la version 33, représentée sur la figure 6.13, du Modèle Local (les relations entre les groupes d'exigences bien formés et les transitions de groupe ont été établies dans le paragraphe 5.6.3 pour cette version) :

- *(Off → Not_available)* : 1 exigence appartenant à la Transition de Groupe TGF1_33, donc au groupe bien formé GF3.1, est allouée à la fonction Main_AD ;
- *(Not_available → Available)* : 13 exigences appartenant à la Transition de Groupe TGF6_33, donc au groupe bien formé GF2.1, sont allouées à la fonction Main_AD ;
- *(Available → Not_available)* : 14 exigences appartenant à la Transition de Groupe TGF10_33, donc au groupe bien formé GF2.2, sont allouées à la fonction Main_AD ;

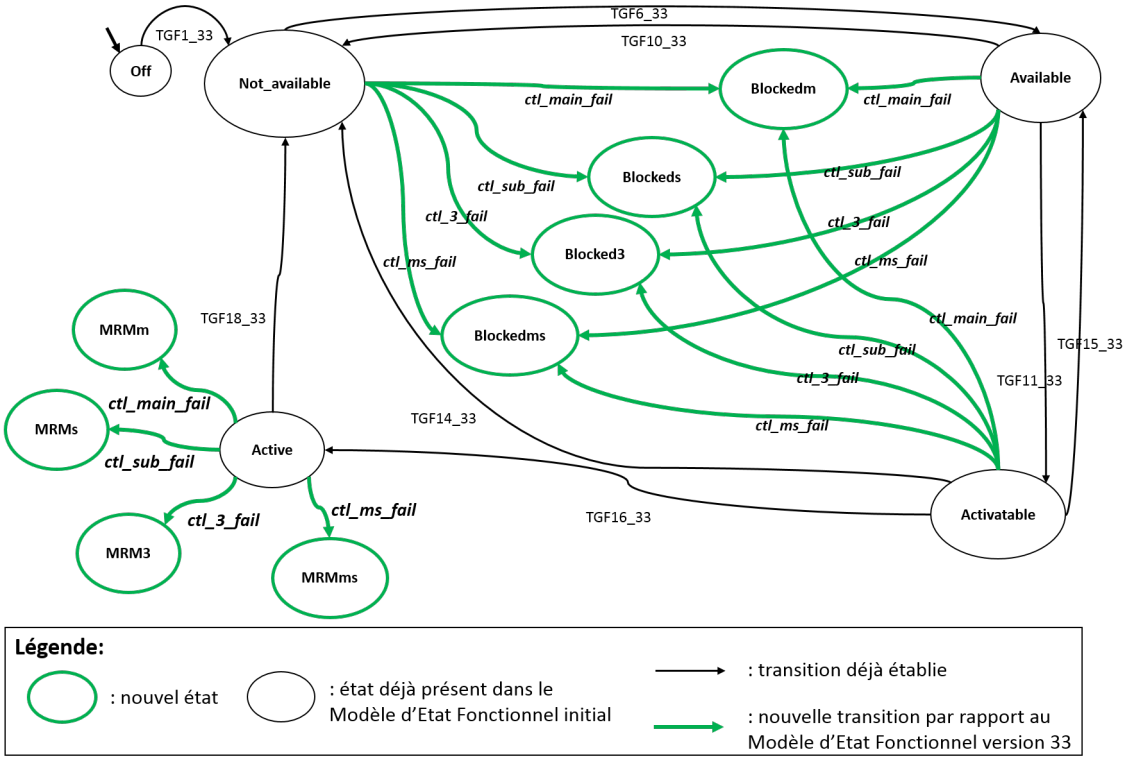


FIGURE 6.13 – Automate de la fonction Main_AD, version 33

— *(Available → Activatable)* : 1 exigence appartenant à la Transition de Groupe TGF11_33, donc au groupe bien formé GF6.1, est allouée à la fonction Main_AD ;

— *(Activatable → Available)* : 13 exigence appartenant à la Transition de Groupe TGF15_33, donc au groupe bien formé GF2.1, est allouée à la fonction Main_AD ;

— *(Activatable → Not_available)* : 14 exigences appartenant à la Transition de Groupe TGF14_33, donc au groupe bien formé GF2.2, sont allouées à la fonction Main_AD ;

— *(Activatable → Active)* : 1 exigence appartenant à la Transition de Groupe TGF16_33, donc au groupe bien formé GF5.1, est allouée à la fonction Main_AD ;

— *(Active → Not_available)* : 36 exigences appartenant à la Transition de Groupe TGF18_33, donc au groupe bien formé GF4.1, sont allouées à la fonction Main_AD ;

— *(Not_available → Silent)*, *(Available → Silent)*, *(Activatable → Silent)* : 14 exigences appartenant à la Transition de Groupe TGSM3_1, donc au groupe bien formé GSM2.1, sont allouées à la fonction Main_AD ;

— *(Not_available → Blocked)*, *(Available → Blocked)*, *(Activatable → Blocked)* : 13 exigences appartenant à la Transition de Groupe TGSS3_1, donc au groupe bien

formé GSS2.1, et 1 exigence appartenant à la Transition de Groupe TGS33, donc au groupe bien formé GS35.1, sont allouées à la fonction Main_AD ;

Ce travail est effectué pour toutes les Supervisions locales constituant chacun des 6 Modèles Locaux.

L'activité **A3.3** s'achève donc sur l'obtention de 6 Modèles d'État Complets, constitués chacun d'un Modèle Global et d'un Modèle Local, ainsi que, pour chaque modèle, deux ensembles d'exigences, allouées à la Supervision AD globale et aux Supervisions locales. Les Modèles d'État Complets obtenus sont, d'une part, validés par l'expertise, ce qui leur confère du crédit et du sens. D'autre part, la cohérence interne, entre vues locale et globale, a été vérifiée formellement, au moyen de l'opération de composition parallèle d'automates.

Cependant, les exigences supplémentaires que les activités de convergence ont générées n'ont pas encore été vérifiées formellement (de la même manière que dans le chapitre 5) sur les Modèles d'État Complets. Par ailleurs, certaines exigences, initialement vérifiées par les Modèles d'État Fonctionnels et de Sûreté dont les Modèles d'État Complets sont issus, ont été conservées et doivent être de nouveau vérifiées sur les modèles complets. Ces exigences sont relatives aux transitions communes entre les Modèles Locaux et Globaux et les Modèles d'État Fonctionnels et de Sûreté. Par exemple, les exigences spécifiant le passage de l'état *Not_ available* à *Available* dans le Modèle d'État Fonctionnel (d'une version donnée) doivent également être vérifiées sur le Modèle d'État Complet (de la même version).

6.5 Vérification des Modèles d'État Complets (A3.4)

6.5.1 Déroulement de l'activité de vérification

La figure 6.14 présente les activités réalisées pour vérifier les Modèles d'État Complets et finaliser ainsi les activités de convergence **A3** et **A4**. Les propriétés formelles à vérifier sur les Modèles d'État Complets sont déterminées à partir de la comparaison entre les Modèles d'État Fonctionnels et le Modèle d'État de Sûreté avec les Modèles Globaux cibles et les Modèles Locaux. Un raisonnement analogue à celui mené pour construire les exigences supplémentaires (voir paragraphes 6.3.2 et 6.4.4) est suivi. Ensuite, la vérification formelle des propriétés est mise en œuvre, dans l'environnement NuSMV (activité **A3.4.2**). Les Modèles d'État Complets ont dû subir de nouvelles modifications. Les modèles proposés, résultants de ces modifications, sont par conséquent soumis aux AMS et aux pilotes SdF pour une validation finale constituant l'activité **A4.4**.

6.5.2 Propriétés formelles à vérifier sur les Modèles d'État Complets (A3.4.1)

Afin d'établir les propriétés formelles qu'un Modèle d'État Complet version *ij* doit vérifier, les transitions de groupe (Fonctionnelles et de Sûreté) que le modèle étudié contient doivent être caractérisées. Deux cas de figure sont alors possibles :

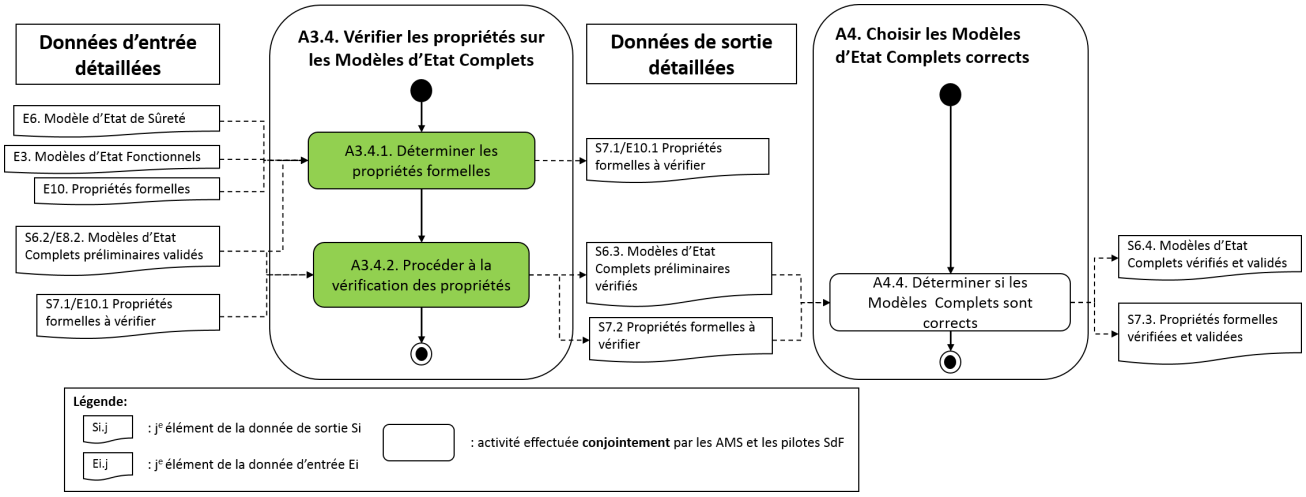


FIGURE 6.14 – Détails de l'activité de vérification des Modèles d'État Complets (**A3.4**)

1 La transition de groupe est associée à une transition préalablement définie dans :
   — Le Modèle d'État Fonctionnel dont est issu le Modèle Global du Modèle d'État Complet étudié ou ;
   — Le Modèle d'État de Sûreté dont est issu le Modèle Local du Modèle d'État Complet étudié.

2 La transition de groupe est associée à une transition qui n'est pas préalablement définie dans le Modèle d'État Fonctionnel (resp. le Modèle d'État de Sûreté) dont le Modèle Global (resp. le Modèle Local) du Modèle d'État Complet étudié est issu.

Dans le premier cas, il suffit de reprendre, pour chaque transition de groupe concernée, l'expression de la Propriété Formelle préalablement vérifiée sur le Modèle d'État Fonctionnel version *ij* et le Modèle d'État de Sûreté pour, respectivement, le Modèle Global version *ij* et le Modèle Local version *ij*. À titre illustratif, pour le Modèle Global version 33 (représenté sur la figure 10.8), il apparaît que la transition de groupe **TGF6_33**, entre l'état *Not_available* et *Available*, est déjà définie dans le Modèle d'État Fonctionnel dont le Modèle Global version 33 est issu. En conséquence, la Propriété Formelle Fonctionnelle Globale 6 du Modèle Global version 33 (notée **PFFG6_33**) est la même que la Propriété Formelle Fonctionnelle **PFF6_33**, c'est-à-dire (voir paragraphe 5.6.3) :

PFFG6_33(PFF6_33) : AG((state = activatable & TGFF6_33) → AF(state = available))

Où *TGFF6_33* : (((veh_pos==0 AND veh_dir==1) OR (veh_pos==6)) AND (forec_AD_dur==1) AND (veh_loca==0 OR veh_loca==1) AND (lan_width==1) AND ((weat_cond==0) OR (weat_cond==1) OR (weat_cond==2) OR (weat_cond==3) OR (weat_cond==4) OR (weat_cond==5) OR (weat_cond==6)))

Notons que pour identifier la provenance de cette exigence, la propriété source est, comme précédemment pour les exigences supplémentaires, annotée entre parenthèses dans l'identifiant de la propriété qui en est issue.

Pour le deuxième cas de figure, un raisonnement analogue à celui mené pour déterminer les exigences supplémentaires a été conduit. Pour l'automate de la fonction Main_AD du Modèle Local version 33, représenté sur la figure 6.13, la transition menant de l'état *Not_available* à *Available*, associée à la transition de groupe **TGF6_33**, est nouvelle par rapport à l'automate de la fonction Main_AD du Modèle d'État de Sûreté validé à l'issue de l'activité **AS2** (voir figure 5.17). Or, comme nous l'avions vu dans le paragraphe 6.4.4, lorsque la Supervision AD globale passe de l'état *Not_available* à *Available*, la fonction Main_AD passe également de l'état *Not_available* à *Available*, sous les mêmes conditions. Par conséquent, la Propriété Formelle Fonctionnelle Locale indice 6 version 33 (notée **PFFL6_33**), associée à la transition de groupe **TGF6_33**, s'exprime de la façon suivante :

PFFL6_33(PFF6_33) : AG((state_m = activatable_m & TGFF6_33) → AF(state_m = available_m))

Où *TGFF6_33* a la même expression que précédemment et *state_m* représente l'état courant de la fonction Main_AD. La propriété source de la propriété PFL6_33 est également indiquée entre parenthèses.

Pour le Modèle Global version 33 (figure 10.8), la nouvelle transition, par rapport au Modèle d'État Fonctionnel version 33 dont ce modèle est issu (représenté sur la figure 10.4), de l'état *Active* à *MRMm*, correspond à la transition de l'état *Active* à *Silent* de l'automate de la fonction Main_AD. En effet, lorsque la fonction Main_AD passe de l'état *Active* à *Silent*, la Supervision AD globale passe de l'état *Active* à *MRMm*. De plus le franchissement de cette transition se fait suite à un événement de défaillance de la fonction de Contrôle AD. Nous avons choisi de modéliser cet événement sous la forme d'une variable booléenne pour deux raisons :

1 Le détail relatif à la défaillance d'une fonction n'a pas d'intérêt dans le cadre de la vérification menée car il s'agit de s'assurer qu'en cas d'une défaillance (peu importe laquelle) la Supervision AD (aux niveaux global et local) ait la réaction appropriée. Or, chaque défaillance joue un rôle symétrique en ce sens que toutes les défaillances d'une même fonction de contrôle doivent causer la même réaction de la Supervision. Par exemple, que la défaillance de la fonction Control_Main_Active se manifeste par une accélération excessive ou une décélération insuffisante, les réactions des Supervisions locales et de la Supervision globale seront identiques ;

2 La prise en compte de l'ensemble des variables de sûreté cause une explosion combinatoire, non gérée par les *model checkers* utilisés (UPPAAL et NuSMV).

En fin de compte, la Propriété Formelle de Sûreté Globale d'indice 1 (**PFSG1**), issue de la propriété PFSM3 (voir paragraphe 5.6.4), s'exprime de la manière suivante :

PFSG1(PFSM3) : AG((state = active & ctl_main_fail) → AF(state = MRMm))

Où

— ctl_main_fail = 0 : pas de défaillance de la fonction Control_Main_Active ;

— ctl_main_fail = 1 : défaillance de la fonction Control_Main_Active.

En suivant un raisonnement similaire pour chaque version de Modèle d'État Complet, nous avons pu, pour chacune des six versions, déterminer les propriétés formelles qu'ils devaient vérifier. Les propriétés exprimées prennent les deux formes suivantes :

— AG((*état fonctionnel 1* & *TGFFk_ij*) → (*état fonctionnel 2*)), pour les propriétés fonctionnelles ;

— AG((*état fonctionnel 1* & *ctl_function_fail*) → (*état de sûreté 1*)), pour les propriétés de sûreté.

Où :

— *TGFFk_ij* est le k^{e} transition de groupe fonctionnelle formalisée de la version *ij* du Modèle d'État Complet ;

— *état fonctionnel 1* et *état fonctionnel 2* sont des états spécifiés par les AMS, *i.e.* l'un des états suivants : *Off*, *Not_available*, *Available*, *Activatable* ou *Active* ;

— *état de sûreté 1* est un état déterminé par les pilotes SdF, *i.e. Silent*, *MRM* ou *Blocked* ;

— *ctl_function_fail* représente l'une des quatre défaillances de fonction de contrôle envisagée.

6.5.3 Vérification des propriétés et validation (A3.4.2 et A4.4)

Pour mettre en œuvre la vérification formelle des exigences sur les six Modèles d'État Complets, nous avons été confrontés à la même difficulté que celle rencontrée lors de la vérification des exigences sur les Modèles d'État Fonctionnels : l'explosion combinatoire (voir paragraphe 5.6.4). En effet, les Modèles Complets intègrent les aspects fonctionnels. En conséquence, NuSMV a également été utilisé pour procéder à la vérification des propriétés formelles sur les modèles. Les six versions de Modèles d'État Complets ont donc été retranscrites dans NuSMV. À titre d'illustration, la version 12 est fournie en annexe (paragraphe 10.8).

Les propriétés formelles, de la forme décrite précédemment, sont toutes respectées sur les Modèles d'État Complets, tels qu'ils ont été construits dans NuSMV. Ainsi, aucune modification particulière n'a dû être apportée aux Modèles Complets suite à la vérification. Ces modèles, ainsi que les résultats obtenus ont été soumis aux AMS et aux pilotes SdF pour validation lors de l'activité **A4.4**.

Cependant, à l'instar d'UPPAAL, la modélisation dans l'environnement NuSMV est **synchrone**. Par conséquent, comme nous l'avions déjà souligné dans le paragraphe 6.4.3, toutes les variables peuvent changer de valeur au même instant. Or, puisque les défaillances, modélisées par des *événements* dans Supremica, correspondent à des variables (booléennes) dans NuSMV, il est *a priori* possible qu'une garde d'une transition entre deux états fonctionnels et une défaillance surviennent au même instant. Ceci n'est pas possible dans Supremica car la garde et la défaillance sont modélisées par deux événements distincts.

C'est pourquoi, à l'instar des modifications entreprises dans le paragraphe 5.6.4 sur la version 12 du Modèle d'État Fonctionnel, les gardes des transitions entre les états

fonctionnels ont été complétées pour signifier que l'entrée dans un état fonctionnel ne peut se faire qu'en l'absence de défaillance. Par exemple, la garde de la transition entre l'état *Not_ available* et *Available*, pour la Supervision AD globale, est désormais spécifiée de la manière suivante (voir le Modèle d'État Complet version 12 fourni en annexe, au paragraphe 10.8) :

— « state=not_available & TGFF6_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail | ctl_ms_fail) : available ; »

La seconde conséquence de la différence de modélisation entre Supremica et NuSMV concerne les défaillances. Deux défaillances indépendantes ne peuvent pas avoir lieu au même instant. Ceci se modélise aisément dans Supremica puisque les événements sont asynchrones. En revanche, il doit être spécifié explicitement, dans l'environnement NuSMV, que deux défaillances indépendantes ne peuvent être simultanées. De ce fait, les lignes de code suivant la mention « **Défaillances non simultanées** » ont été ajoutées dans les Modèles d'État Complets, comme l'illustre la version 12 (paragraphe 10.8).

6.6 Bilan de l'activité de convergence des points de vue

Nous avons illustré dans ce chapitre la **Contribution 3** (voir figure 4.2), relative à la convergence des points de vue des AMS et des pilotes SdF par l'emploi d'un formalisme commun. Les activités menées durant le chapitre 6 s'appuient essentiellement sur les deux expertises métier et la connaissance de la fonction à concevoir (la Supervision AD et, plus généralement, la fonction AD). Plus particulièrement, il s'agit du dialogue et de la confrontation des deux expertises qui mènent à la possibilité d'une spécification du comportement cohérente. Bien que l'expertise joue un rôle majeur, il n'en reste pas moins que l'apport des méthodes formelles et de l'utilisation du formalisme commun constitue une brique essentielle de ces activités pour les trois raisons suivantes :

1 *Aide à l'identification des points durs* : la représentation des deux points de vue sous un formalisme commun facilite la détermination des différences sémantiques entre les deux perspectives métier. En effet, les informations contenues dans les exigences initiales ayant été formalisées selon le même procédé, il devient alors plus simple de comparer les éventuels conflits de spécifications. L'exemple le plus significatif est l'état *Off*, apparaissant tant du point de vue des AMS que de celui des pilotes SdF mais revêtant deux sens différents ;

2 *Garantie formelle de cohérence entre vues locale et globale* : cette assurance est donnée par l'utilisation de l'opération mathématique de composition parallèle des automates finis. À partir d'un comportement global objectif, les automates locaux sont progressivement modifiés jusqu'à ce que leur produit soit identique à l'objectif ;

3 *Rôle de la vérification formelle* : en dernier lieu, la vérification formelle des propriétés a permis, comme ce fut le cas lors de l'activité **AFS1.4.5** (paragraphe 5.6.4), de déceler les dernières lacunes de spécification.

Par ailleurs, la question de l'utilité de la réécriture des exigences se pose dans notre cas de figure. Effectivement, si les deux données, exigences et modèles, représentent la même information, il est envisageable de n'utiliser plus qu'un seul support pour cette information : *a priori* les modèles. Nous avons cependant jugé nécessaire de mettre à jour les corpus d'exigences pour deux raisons :

1 Les exigences textuelles correspondent à la pratique courante chez Renault, qui sera conservée à l'avenir. Il n'est en effet pas question d'abandonner cette pratique, du moins en phase amont de conception dans laquelle les travaux de thèse se sont inscrits ;

2 Les exigences finales représentent, il est vrai, la même information que les modèles, mais au prix de la formulation d'un ensemble d'hypothèses (d'interprétation et de modélisation) dont l'importance doit être à nouveau soulignée. D'une part, sans ces hypothèses, les modèles ne sont pas constructibles. D'autre part, c'est sur ces hypothèses que repose en grande partie la traçabilité entre les exigences initiales (retenues) et les modèles obtenus.

Ces différents points sont relatifs uniquement aux apports revendiqués dans le chapitre 6. En guise de conclusion générale, nous résumerons, dans le chapitre 7, les principales questions abordées dans ces travaux. Nous mettrons également ces derniers en perspective dans le contexte de l'entreprise hôte et, plus général, en situant également ces perspectives par rapport aux recherches en cours. Les apports de la démarche et ses principales limitations seront décrits. Nous verrons, en dernier temps, quelles pistes opérationnelles et quelles problématiques dévoilent les travaux menés durant la thèse.

Troisième partie

Conclusions et perspectives

7

Conclusions générales et perspectives

Les travaux présentés dans ce mémoire ont traité essentiellement la problématique de la convergence des points de vue des Architectes Métier Système et des pilotes en Sûreté de Fonctionnement en phase amont de conception, dans un contexte industriel opérationnel. Bien que la littérature propose de nombreuses voies contribuant à résoudre ce problème (restituées dans le chapitre 3), aucune n'est pleinement adaptée à l'environnement industriel actuel (voir chapitre 2). Ainsi, en nous fondant sur cet état de l'art, nous avons construit et mis en œuvre une démarche formalisée et outillée de conception visant à mettre en cohérence les perspectives des AMS et de la SdF, au plus tôt dans le cycle de conception. Cette démarche repose sur la connaissance des spécificités du système conçu des deux experts métier et est appuyée par des méthodes formelles, qui garantissent rigueur et cohérence. Dans une perspective d'Ingénierie Système, un équilibre a été recherché entre interventions des experts (en termes de fréquence et d'investissement requis) et utilisation de méthodes formelles (formalisation, vérification formelle). Après avoir rappelé les principaux résultats obtenus, nous identifierons les limitations des travaux menés qui donnent lieu aux perspectives et suites à donner à cette thèse.

7.1 Bilan des travaux réalisés

7.1.1 Livrables et données obtenus

Le premier critère pour juger de la pertinence des travaux entrepris réside dans la comparaison entre les données initiales avec celles qui résultent de la démarche proposée. L'efficacité et l'utilité des travaux peuvent, à première vue, être évaluées à l'aune de la différence entre ces deux jeux de données. Comme le montre la figure 4.8, les données d'entrée sont constituées de **175** exigences fonctionnelles de comportement, extraites du System Technical Requirements (AMS Renault 2017b), allouées à la Supervision AD globale, et de **217** exigences fonctionnelles de SdF allouées aux Supervisions locales, issues du Functional Safety Concept (IAV 2016). La caractéristique commune à ces deux ensembles d'exigences réside dans leur rédaction en langage naturel, sans règles spécifiques ni contenus formalisés. À partir de ces données, nous avons construit :

— **6** Modèles d'État Complets qui spécifient le comportement de la Supervision AD globale (voir figure 2.12) et des trois fonctions de Supervision locales Main_AD, Sub_AD et AD-3 (figure 2.13) ;

- Entre **1034** et **1142** (selon les versions de Modèle d'État Complet) exigences allouées à la fonction de Supervision AD, plus précisément :
  - Entre **290** et **317** exigences allouées à la Supervision AD globale ;
  - Entre **290** et **317** exigences allouées à la fonction Main_AD ;
  - Entre **276** et **303** exigences allouées à la fonction Sub_AD ;
  - Entre **178** et **205** exigences allouées à la fonction AD-3.
- Entre **59** et **65** propriétés formelles vérifiées, selon la version du Modèle d'État Complet considéré.

Le processus de formalisation des exigences (décrit dans le chapitre 5), visant à exprimer chacune des exigences retenues sous une forme donnée a, en premier lieu, mené à la reformulation des exigences. Ces dernières ont été rendues plus précises. De plus, les corpus initiaux d'exigences ont également été modifiés, après consultation des deux expertises métier (AMS et pilotes SdF). Dans un second temps, les ré-allocations, en termes de fonction et d'espace d'états, opérées dans le chapitre 6, ont sensiblement augmenté le nombre d'exigences allouées à la Supervision AD globale (enrichie par les aspects dysfonctionnels) et aux Supervisions locales (augmentées par la vue fonctionnelle).

Si la comparaison entre les données d'entrée et de sortie offre un premier aperçu de l'impact de la démarche, il est nécessaire de le compléter par la considération des données intermédiaires qui ont été produites. Celles-ci illustrent, de façon plus éloquente que cette simple comparaison, la complexité d'un processus de formalisation engagé dans un contexte industriel et opérationnel réel. Ce processus et ces données apportent un éclairage sur les nombreuses difficultés rencontrées lors de la mise en œuvre de méthodes formelles dans l'industrie. À ce sujet, (Woodcock et al. 2009) offre un panorama complet, dressant le bilan de l'application des méthodes formelles dans des contextes industriels. Il y est notamment relevé, dans les projets opérationnels visant à implémenter une ou plusieurs méthodes formelles, que « the cost increase was largely due to the lack of precise, complete information about the required externally visible behavior of the software product. ». Preuve que ce problème est toujours d'actualité, ce constat est appuyé par une récente revue, centrée sur la Théorie du Contrôle par Supervision (Zaytoon et Riera 2017) : « There is a need to define a methodological framework , that could help the designer to progressively transform the user's requirements and the plant operations into formal specifications of the expected behavior ». Ceci correspond effectivement au principal écueil que nous avons rencontré. C'est pour le contourner que nous avons produit et manipulé de nombreuses données intermédiaires : *exigences bien formées*, *groupes bien formés*, *transitions de groupe formalisées*, *modèles d'état préliminaires*, *modèles d'état vérifiés*... Leur fonction essentielle est de faire intervenir régulièrement l'expertise dans la démarche proposée afin d'obtenir *in fine* des modèles vérifiés formellement, mais aussi, validés. Ainsi leur signification est comprise et approuvée par les deux expertises métier. Ces données intermédiaires permettent de faire passer progressivement de une expression du besoin informelle, imprécise, et ambigüe à plusieurs spécifications formelles plausibles.

Ces travaux justifient en conséquence que les méthodes formelles sont prioritairement à utiliser pour des systèmes qui nécessitent une grande rigueur dans leur conception. Il s'agit des systèmes complexes et critiques du point de vue de la sûreté. En effet,

des efforts importants sont requis pour mettre en œuvre ces méthodes : déploiement d'outils et de méthodes mais aussi adoption d'une organisation adéquate (équipes de travail, processus), acquisition ou consolidation de connaissances théoriques... S'ajoute à ces mesures, la formation nécessaire car les outils et méthodes ne sont que des moyens d'application. Seule l'appropriation et la compréhension des méthodes permettent, dans un premier temps, l'adhésion à ces nouvelles pratiques, et, dans un second temps, l'adoption et le déploiement.

7.1.2 Amélioration de la démarche de conception existante

Aux trois contributions illustrées sur la figure 4.8 s'ajoute un premier apport qui est l'élaboration de la démarche outillée de conception développée. Cette dernière, par le traitement des exigences, contribue à faire converger les points de vue des AMS et des pilotes SdF en phase amont de conception. Cette démarche est inscrite dans les processus de conception Renault existants, qui sont eux-mêmes conformes aux normes en vigueur : l'ISO 15288 :2008 pour l'Ingénierie Système et la norme ISO 26262 :2011 concernant la Sûreté de Fonctionnement. L'application de la démarche permet d'obtenir, à partir de deux ensembles d'exigences informelles spécifiant le comportement de la Supervision AD émanant des deux métiers, plusieurs possibilités de modèles d'état formels respectant l'ensemble des exigences. La traçabilité entre les exigences initiales et les modèles obtenus est garantie par la formulation d'hypothèses et l'adoption de notations appropriées. En outre, la validation des hypothèses au fil du déroulement de la démarche, par les deux acteurs métier, facilite la validation finale des modèles.

7.1.3 Apports scientifiques

Concernant les contributions scientifiques, nous résumons les solutions que nous avons mises en œuvre afin de lever les verrous identifiés dans l'introduction générale (voir paragraphe 1.4).

7.1.3.1 Processus de formalisation d'exigences informelles

Un processus opérationnel de formalisation d'exigences textuelles appuyé par les deux expertises métier a été mis au point. Les exigences traitées sont initialement rédigées sans règles spécifiques sur la structure, la syntaxe ou la sémantique et leur contenu n'est pas formalisé. Les exigences sont traduites en une ou plusieurs propriétés formelles, exprimées dans la logique temporelle *Computation Tree Logic (CTL)*. Pour ce faire, un critère de sélection filtrant les exigences de comportement relatives à un changement d'état a été formulé. Puis, les exigences ont été regroupées et complétées par les informations d'état manquantes, donnant lieu à des *hypothèses d'interprétation*.

Enfin, des *variables*, des *paramètres* ainsi que des *opérateurs logiques* ont été déterminés pour formaliser le contenu des exigences et finalement aboutir aux propriétés formelles cibles. L'ensemble de ces éléments (*hypothèses d'interprétation, critères, variables, paramètres, opérateurs logiques*) ont régulièrement été soumis aux AMS et pilotes SdF pour une validation effectuée en silos. Par ailleurs, les hypothèses d'interprétation, ainsi que l'adoption de notations adaptées, contribuent à assurer la *taçabilité* des exigences,

tout au long du processus de formalisation. En effet, les notations permettent, à partir d'un énoncé d'une exigence finale, de retrouver l'exigence source dont elle est issue.

7.1.3.2 Construction des modèles d'état et vérification formelle

L'approche proposée contient également une méthode de construction de modèles de comportement formels, qui sont plus précisément des automates à états finis. À partir des groupes d'exigences formés lors du processus de formalisation, des *squelettes de modèles d'état* sont premièrement élaborés. Ces représentations graphiques ne sont pas formalisées, en ce sens qu'ils ne constituent pas des objets formellement définis, manipulables dans un outil logiciel tel qu'UPPAAL par exemple. En revanche, ces représentations ont pour intérêt de réaliser un premier filtre. Effectivement, les deux acteurs métier peuvent rapidement y repérer de potentielles erreurs. Par la suite, et au prix d'hypothèses supplémentaires, ces squelettes servent de base à la construction des objets formels que sont les automates à états finis. Ces automates ont ensuite pu être vérifiés dans le *model checker* symbolique NuSMV au regard des propriétés formelles issues du processus de formalisation. Cette vérification finale garantit l'absence d'erreurs de formulation dans les énoncés des propriétés formelles et assure ainsi la cohérence entre les exigences et les modèles formels.

7.1.3.3 Méthode de convergence des points de vue

La dernière contribution scientifique est la mise au point d'une méthode de convergence des perspectives de l'AMS et du pilote SdF, fondée sur leur expertise conjointe et sur l'opération de composition parallèle d'automates finis. En premier lieu, les deux expertises sont convoquées pour enrichir les Modèles d'État Fonctionnels. Ces modèles (appelés Modèles Globaux cibles), consolidés par les aspects dysfonctionnels, spécifient le comportement attendu, à la fois par les AMS et par les pilotes SdF, de la Supervision AD globale. Ensuite, les Modèles d'État de Sûreté (locaux) sont également enrichis par des considérations provenant de la vue fonctionnelle. Enfin, les fonctions de transition de chaque automate local constituant les Modèles d'État de Sûreté sont déterminées de telle sorte que l'automate résultant de leur composition parallèle soit identique à l'automate du Modèle Global cible étudié. De cette manière, la cohérence entre les vues locales et la vue globale est formellement garantie.

7.1.4 Apports méthodologiques

7.1.4.1 Mise en évidence de solutions alternatives

La première contribution méthodologique identifiée sur la figure 4.8, est propre au processus de formalisation des exigences déployé. Cette contribution consiste en la mise en évidence de plusieurs modèles d'état formels possibles à partir d'un unique ensemble d'exigences informelles. L'objectif de ce processus est d'identifier, puis d'exploiter les ambigüités contenues dans les exigences initiales. Il ne s'agit donc pas de supprimer les ambigüités (comme par exemple dans (Lebeaupin 2017)) mais, par la formalisation, de mettre en évidence un ensemble de solutions possibles (*i.e* validées par l'expertise). Ces

solutions sont à l'origine masquées par le manque de précision et de clarté des exigences traitées. L'exhibition des formalisations crédibles requiert la formulation d'hypothèses, régulièrement soumises aux deux acteurs métier pour validation.

7.1.4.2 Rôle des experts métier

La seconde contribution est relative à la définition précise du rôle des deux experts métier, AMS et pilote SdF, dans le processus de formalisation des exigences et de construction des modèles d'état. En effet, chaque activité de la démarche nécessitant une prise d'hypothèse est clairement identifiée. Par la suite, les interventions des experts sont déterminées en termes de planning (à quel moment les experts sont-ils sollicités ?) et de résultats attendus (quelles sont les hypothèses à examiner, et pourquoi a-t-il été nécessaire de les formuler ?). La détermination des activités impliquant l'expertise métier constitue une approche pragmatique pour l'utilisation de méthodes formelles dans un contexte industriel automobile. Elle permet par ailleurs d'apprécier la difficulté de la mise en œuvre de telles méthodes dans l'industrie tout en donnant des pistes pour leurs futurs déploiements. Il convient de trouver un équilibre entre des interventions trop fréquentes de l'expertise, qui minimisent sensiblement l'intérêt des méthodes formelles, et une automatisation de ce processus, pouvant mener à une grande quantité de résultats non pertinents.

7.1.4.3 Mise en cohérence par un formalisme commun

Nous avons détaillé, dans le chapitre 6, la méthode proposée pour confronter les deux points de vue métier, dont les spécifications ont été préalablement formalisées selon une démarche similaire. Un des enseignements principaux montre que la formalisation des exigences selon les deux perspectives facilite considérablement l'identification des incohérences entre les deux métiers. La méthode de convergence repose sur l'application de la *composition parallèle d'automates finis* qui garantit la cohérence entre les vues locales, initialement spécifiées par les pilotes SdF, et la vue globale, déterminée à l'origine par les AMS. La seconde conclusion est l'implication nécessaire des deux expertises pour appuyer la méthode mise en œuvre et valider les résultats. Effectivement, les modifications apportées aux modèles d'état avant de procéder à l'opération de composition demeurent primordiales.

7.1.4.4 Distinction entre Processus et « Processus de Processus »

Étant donné le caractère novateur et exploratoire des travaux, dans l'entreprise hôte, il est pertinent de distinguer deux types principaux d'activités de la démarche présentée dans les chapitres 5 et 6 :

1 *Activités de processus* : il s'agit des activités opérationnelles, qui ont permis d'élaborer les modèles d'état à partir des exigences informelles initiales. Ce sont les activités que de futurs acteurs doivent suivre pour mettre en œuvre la démarche proposée ;

2 *Activités de « processus de processus »* : ce sont les activités déployées pour construire la démarche en elle-même. Si cette démarche est pérennisée, elles sont

vouées à ne plus être appliquées. En effet, elles donnent généralement lieu à des hypothèses. Par la suite, on considèrera ces hypothèses validées pour les systèmes auxquels la démarche peut être appliquée.

À titre illustratif, l'activité **AFS1.1.2** (voir figure 5.3), qui consiste à procéder à la sélection des exigences pertinentes (celles qui seront analysées et vérifiées par la suite), est une *activité de processus* car elle devra bien être réalisée si la démarche est pérennisée. Au contraire, l'activité **AFS1.1.1**, qui vise à formuler un critère de sélection des exigences, est une *activité de processus de processus*. En effet, si la démarche est réutilisée, les discussions relatives à la détermination de ce critère n'auront certainement plus lieu d'être.

7.1.5 Application à la Supervision de Conduite Autonome

Le dernier aspect important des travaux menés est leur caractère opérationnel, en accord avec la convention CIFRE dans lesquels ils s'inscrivent. Les données qui ont été traitées sont celles d'un projet d'innovation d'envergure en cours chez Renault : celui du véhicule autonome. De ce fait, l'approche proposée pour répondre à la problématique soulevée est très pragmatique et adaptée aux moyens, processus, outils, ressources et contraintes de l'entreprise hôte. Les résultats obtenus en termes de formalisation et consolidation des exigences, construction de modèles d'état formels, mise en cohérence de points de vue locaux et globaux ont suscité l'intérêt au sein de l'ingénierie Renault. Ceci s'est notamment concrétisé en fin de thèse par un stage centré sur la construction des modèles « en silos » à partir des exigences informelles, mené au sein de Renault Software Lab, à Nice Sophia Antipolis (voir paragraphe 7.2.2.1 pour plus de détails). Ce travail a notamment montré que certains critères de sélection des modèles pertinents sont automatisables (comme l'élimination des automates bloquants ou comprenant un état ou des états non atteignables). En revanche, des considérations plus fines, et plus spécifiques au cas d'étude, sont beaucoup plus difficiles à intégrer dans un outil (par exemple la modification de gardes de transition suite à l'analyse de leurs contenus).

7.2 Principales perspectives

Avant de dégager les perspectives les plus intéressantes qui ont été ouvertes par les travaux de thèse, il s'agit d'identifier et de caractériser les limitations de notre approche. Ces limitations peuvent être purement techniques, ou revêtent un aspect plus général. Dans le premier cas, des solutions opérationnelles peuvent être envisagées, tandis que le second cas donne lieu à des perspectives de nouveaux travaux de recherche.

7.2.1 Limitations

7.2.1.1 Processus complexe : réutilisabilité ?

Comme nous venons de l'évoquer, la question de la réutilisabilité de la démarche se pose avec d'autant plus d'acuité que cette démarche est complexe. Elle fait effectivement intervenir trois acteurs métier : l'Architecte Métier Système, le pilote Sûreté de

Fonctionnement et l'« ingénieur méthodes formelles » (procédant aux activités menées durant la thèse, voir paragraphe 4.2.4) qui doivent se coordonner entre eux. En effet le séquencement des activités doit être respecter au risque de compromettre l'efficacité de la démarche. En outre, les résultats attendus en fin de chaque activité, doivent être produits afin de ne pas bloquer le processus. Le bon déroulement de l'approche repose d'une part sur une connaissance fine de la fonction conçue, ce qui peut poser problème à l'« ingénieur méthodes formelles ». D'autre part les compétences requises en méthodes formelles (formalisme, signification des résultats des vérifications formelles, composition parallèle des automates finis) et dans les outils utilisés (UPPAAL, NuSMV, Supremica) sortent généralement du périmètre des AMS et des pilotes SdF.

Rappelons aussi que cette approche ne saurait être exhaustive car un type d'exigences peut être vérifié. Ainsi, même en cas de réutilisation, il est primordial de délimiter clairement les exigences qui pourront effectivement être vérifiées par notre approche, de celles qui devront l'être par d'autres moyens (simulations, tests, *design review*...). Enfin, la traçabilité des exigences réalisée dans les travaux de thèse, par des notations appropriées et la formulation d'hypothèses justifiant les diverses étapes de transformation, devra être consolidée au moyen des outils informatiques utilisés. Par exemple, des liens d'héritage de propriétés pourraient être créés entre les exigences mères (informelles) et les exigences filles (formelles). La traçabilité serait donc ainsi également garantie par les outils logiciels.

7.2.1.2 Intégration fine dans les processus Renault en évolution

Deux évolutions majeures en cours chez Renault se sont opposées à une analyse détaillée de l'intégration de la démarche proposée dans les processus existants :

1 Le déploiement d'une approche *Model Based Systems Engineering* outillée ;

2 La consolidation d'une Direction Software.

Ces deux chantiers sont le résultat de l'évolution automobile présentée dans le chapitre 2. Les voitures intégrent désormais une architecture électronique embarquée complexe, le métier logiciel et la qualité logicielle prennent désormais une importance considérable dans l'industrie automobile. Il s'agit en effet de maîtriser cette complexité, d'un point de vue purement fonctionnel (les performances attendues et annoncées doivent être atteintes), mais aussi du point de vue de la Sûreté de Fonctionnement. La conséquence directe de ces deux évolutions se concrétise par la modification, l'adaptation et l'enrichissement du Système de Conception Renault, succinctement présenté dans le paragraphe 2.2.2. Ainsi, certaines procédures ont subi des changements significatifs et d'autres ont été créées. De ce fait, les procédures dont nous disposions au moment de mener les travaux de thèse reflétaient une organisation en mutation. À la fin des travaux de thèse, ces modifications procédurales ont été en grande partie achevées. Il conviendra donc, si la démarche est adoptée, de déterminer en détail comment elle peut s'intégrer dans les nouveaux processus Renault en termes d'acteurs, de jalons, de livrables, d'outils, et de planning.

En complément des aspects procéduraux, il s'agit également d'évoquer les outils sur lesquels s'appuient les processus. Nous avons pour ces travaux, utilisé trois outils : UPPAAL, NuSMV et Supremica. Le recours à ces trois logiciels différents a été justifié

dans le cadre de la thèse. Toutefois, chaque changement d'environnement conduit à des questions d'équivalences ou de correspondances sémantiques entre les différents formalismes manipulés dans les outils. Si les hypothèses particulières prises pour le cas d'étude présenté n'ont pas causé de difficultés trop importantes, il faudra veiller à ce que l'application de la démarche à une autre fonction ne cause pas ce type de problèmes. Par ailleurs, dans le but d'outiller la démarche chez Renault, il est nécessaire de s'interroger sur l'intégration des outils dans les suites logicielles de l'entreprise, ou, au minimum, de leur compatibilité (voir paragraphe 7.2.2.1).

7.2.1.3 Considération de points de vue métier supplémentaires

La démarche élaborée se focalise sur deux points de vue : celui des Architectes Métier Système et celui des pilotes Sûreté de Fonctionnement. Ce choix s'explique opérationnellement car il s'agit des deux métiers avec lesquels nous nous sommes interfacés. Plus fondamentalement, les AMS synthétisent les besoins de la plupart des parties prenantes (législateurs, clients, actionnaires, métiers concernés par le système : génie électrique, mécanique, mécatronique, acoustique vibratoire...) en phase amont de conception. Chacune de ces parties prenantes apporte un point de vue différent sur le système à concevoir. Les exigences émises par les AMS doivent traduire les besoins de ces différents points de vue. Ainsi considérer la perspective des AMS revient à prendre en compte les différentes parties prenantes. Les pilotes SdF seront ensuite en charge de critiquer la ou les conceptions (prenant la forme d'exigences et d'architectures) proposées par les AMS.

Toutefois, ce raisonnement se heurte à la difficulté de prise en compte, dès les premières phases de conception, de l'ensemble des exigences des parties prenantes pesant sur le système conçu. Deux raisons expliquent ce phénomène. Premièrement il n'est pas acquis que les AMS parviennent, dès les phases amont de conception, à synthétiser tous les besoins. Deuxièmement, les contraintes induites par l'implémentation physique ne peuvent pas être ignorées mais elles sont constatées nécessairement plus tard (selon le cycle en V) que les architectures fonctionnelles, du moins en théorie (voir paragraphe 7.2.2).

7.2.2 Pistes de travail

Parmi les points précédemment évoqués, les aspects relatifs à l'intégration et la réutilisation de la démarche semblent prioritairement à traiter de manière opérationnelle au sein de l'entreprise. Certains points, comme la réorganisation et les modifications à apporter au cadre procédural, peuvent il est vrai relever de travaux plus scientifiques. En revanche, la prise en compte de points de vue métier additionnels constitue une perspective de recherche.

7.2.2.1 Intégration de la démarche de formalisation

Une des principales pistes à donner à la suite de mes travaux réside dans leur intégration dans les processus Renault existants, en vue de leur réutilisation. Pour ce faire, deux approches complémentaires peuvent être distinguées :

1 Une approche opérationnelle ;
2 Une approche organisationnelle.

Approche opérationnelle

La première approche, opérationnelle, se concrétise par le développement d'outils, voire d'algorithmes. Dans le contexte d'un groupe de la taille de Renault, il est préférable de développer un algorithme dans le but d'apporter une preuve de concept consolidée (en vue d'une adoption de l'outil par Renault et d'éventuelles futures certifications normatives). En effet, le choix des outils à l'échelle du groupe relève d'une stratégie globale de l'entreprise, dans laquelle les travaux doivent s'inscrire harmonieusement. La voie la plus simple, rapide et efficace pour implémenter un processus de formalisation des exigences analogue à la démarche de formalisation proposée, passe donc par le développement d'un algorithme. C'est précisément le travail qui a été entamé durant le stage mené par Pierre Scotti, au sein de Renault Software Labs, la filiale de Renault spécialisée dans le logiciel embarqué implantée à Nice Sophia Antipolis. À l'issue de ce stage, un algorithme a été conçu. Cet algorithme prend en données d'entrée les exigences informelles que nous avons traitées et est capable de générer les squelettes de modèles d'état qui résulteraient de l'activité **AFS1.3.2** (voir figure 5.5), si nous ne prenions pas en considération les estimations de *plausibilités*. Nous constatons donc (voir la Table 5.8), qu'un nombre élevé de squelettes sont ainsi générés. Certains principes sur lesquels repose l'estimation de plausibilité, peuvent néanmoins être rendus objectifs et prendre la forme de critères, intégrés dans un algorithme (exclusion des automates bloquants par exemple).

L'analyse menée durant les travaux de thèse, appuyée par le développement de cet algorithme, a soulevé chez Renault la question des relations entre les expressions informelles des exigences, restituées dans un outil de recueil d'exigences (comme *Rational Doors*) et les modèles formels ou semi-formels manipulés dans un modeleur SysML (type *MagicDraw*). Le problème se pose donc en ces termes : laisse-t-on une totale (ou quasi-totale) liberté dans la rédaction des exigences, ou restreint-on, en imposant par exemple des structures types, la liberté de rédaction ? La première voie complexifie considérablement la traçabilité entre les exigences initiales et les modèles manipulés. Le risque de la seconde solution est de réduire le pouvoir d'expressivité accordé aux spécificateurs. La réponse qu'apporte nos travaux à ce problème est nuancée. Nous avons pu constater que laisser une liberté totale dans l'expression des exigences constitue un obstacle considérable à la formalisation (au vu du nombre de possibilités obtenues, voir de nouveau la Table 5.8). Cependant, l'élimination d'un grand nombre de possibilités repose finalement sur des règles relativement simples, voire triviales. Toute la difficulté, si l'on veut développer un outil, est de déterminer quelle règle est triviale *uniquement pour un cas d'application*, auquel cas elle pourra être intégrée dans l'outil, et quelle autre règle dépend du cas étudié. Par exemple, le fait qu'un automate soit bloquant n'est pas conforme aux attentes des AMS. En revanche, ceci peut être acceptable pour les pilotes SdF (voir figure 5.17). Dans notre cas, nous sommes arrivés aux conclusions suivantes, qui prennent la forme de perspectives opérationnelles à court-terme, réalistes au regard du contexte industriel :

(a) Intégrer dans l'outil de recueil des exigences une ou plusieurs *structures types*, inspirées de la Proposition 5.1, pour les exigences de comportement *relatives à un changement d'état* ;

(b) Générer une première représentation à partir de ces exigences, sous forme de squelettes de modèles d'état, en laissant aux spécificateurs la possibilité d'appliquer un panel de règles adaptées à leurs besoins ;

(c) Établir un *lien formel* entre les exigences, ou des éléments d'exigences recensées dans le logiciel du recueil d'exigences, et des objets du modeleur SysML. Ainsi, par exemple, une modification d'un élément d'un modèle sera, ou bien répercuté directement dans les énoncés des exigences concernées, ou simplement remontée aux utilisateurs par une alerte.

En définitive, nous pensons que la potentielle intégration de la démarche passe plutôt par l'extension des possibilités des suites logicielles existantes (ou en cours de déploiement) que le développement d'un outil logiciel à part entière, qui posera nécessairement des problèmes d'interopérabilité (avec les autres logiciels).

Approche organisationnelle

La première approche, pragmatique fournit une preuve d'efficacité et d'utilité relative à l'application de la démarche, mais ne saurait être suffisante. Elle a pour rôle de convaincre les acteurs concernés de la pertinence de la méthode et de leur fournir sur le champ des moyens opérationnels pour sa mise en œuvre. Néanmoins, pour que de nouvelles pratiques soient réellement adoptées, elles doivent s'inscrire dans une réflexion à plus long terme et à plus grande échelle que l'approche opérationnelle. En conséquence, ce travail dépasse le cadre de cette thèse. De plus, comme évoqué dans le paragraphe 7.2.1, le Système de Conception Renault est en pleine évolution. À partir de l'étude des procédures (décrivant les processus, voir figure 2.7) actuelles et en cours d'élaboration, nous avons proposé les potentielles modifications et évolutions suivantes pour :

— **Les processus en lien direct avec les travaux de thèse** : rappelons que les activités de thèse s'intègrent dans deux processus Renault, décrits par les deux procédures suivantes : *« Concevoir le système »* et *« Maîtriser la Sûreté de Fonctionnement des systèmes mécatroniques »* (voir la Table 2.1) :

  — *Structures types des exigences* : des structures types pour les exigences de comportement doivent être décrites et *justifiées* et les liens entre ces exigences et les éléments de modèles d'état explicités ;

  — *Activités supplémentaires* : les activités supplémentaires introduites par les travaux de thèse par rapport aux activités déjà existantes (voir figure 4.4), doivent être intégrées, la séquentialité clairement définie, et surtout des *acteurs* désignés, ainsi que le planning établi.

— **L'élaboration de nouveaux processus** : dans le cadre de la mise en place et du développement d'une Direction Software (voir paragraphe 7.2.1), de nouvelles activités, relatives à la conception et à l'implémentation de logiciel embarqué (aussi bien la partie Hardware que Software), ont été intégrées chez Renault. Ces activités sont décrites et formalisées dans des procédures qui viennent enrichir l'ensemble documentaire constitué par le Système de Conception Renault. Étant donné les liens qu'entretiennent nos travaux avec les aspects logiciels, il sera nécessaire d'analyser l'intégration ou les interactions entre ces activités, centrées sur le logiciel embarqué, et les activités menées durant la thèse ;

— **La portée des processus** : de façon générale, les processus ont été jusqu'à présent focalisés sur les *systèmes*, *i.e.* chaque processus s'applique à la conception et au développement d'un système en particulier. Or les nouveaux enjeux critiques de Sûreté de Fonctionnement, liés notamment au Véhicule Autonome, sont soit à l'origine de nouveaux événements redoutés, soit accentuent la criticité d'événements redoutés pouvant également affecter des véhicules conventionnels. Puisque les causes de certains de ces événements sont *inter-systèmes*, il pourra être nécessaire de mettre au point des processus à une échelle inter-systèmes. Dans ce cadre, des techniques comme la composition parallèle d'automates finis, que nous avons utilisée, pourraient être employées pour vérifier la cohérence globale.

7.2.2.2 Prise en compte des contraintes de l'implémentation

Dans le paragraphe 7.2.1, nous soulignions le fait que les architectures physiques soient *théoriquement* connues après les architectures fonctionnelles. Dans les faits, le processus d'implémentation (voir figure 2.8) se déroule généralement en parallèle avec les processus d'analyse des exigences et de conception d'architectures. Ainsi, l'apparente linéarité du cycle en V de conception est remise en cause. Ceci est dû à une réalité du secteur automobile qui est l'importance du *carry over*. Ce principe de conception consiste, lors du lancement d'un nouveau véhicule ou d'une nouvelle gamme de véhicules, à réutiliser le maximum de pièces existantes plutôt que d'en développer de nouvelles. Cette pratique présente un double avantage : la réduction du coût de conception et l'augmentation de la sûreté du produit. En effet, la norme ISO 26262 préconise le recours à des *proven in use arguments* qui correspondent à des preuves de sûreté fondées sur les données de fiabilité des composants. Naturellement, plus les composants ont été utilisés par le passé, plus l'évaluation de leur fiabilité est précise.

Cette réalité soulève la question, centrale pour les démonstrations de sûreté, des impacts réciproques que peuvent avoir les modifications apportées au niveau de l'implémentation sur les exigences formulées en amont. Ainsi l'intégration de contraintes liées à l'implémentation est une perspective importante et nécessaire des travaux menés. Il existe des recherches, que nous avions déjà évoquées dans le paragraphe 4.2, visant à lier des modèles des diagrammes de comportement dans un formalisme SysML avec des modèles de simulation dans *Simulink* (Cressent, Idasiak et Kratz 2011 ; Kawahara et al. 2009). La piste que nous avons commencée à explorer au cours des travaux de thèse consiste à enrichir les modèles d'états obtenus, qui relèvent du point de vue *fonctionnel*, par des abstractions de contraintes physiques intégrables dans les modèles. En particulier, deux types de contraintes pourraient être ainsi analysés :

1 *Contraintes topologiques* : à partir d'une allocation des éléments de l'architecture fonctionnelle sur ceux de l'architecture physique et des liens entre la défaillance d'un composant et la perte de fonction(s), les Modèles d'État Complets pourraient être enrichis puis utilisés pour vérifier si une défaillance quelconque d'un composant entraîne bien le comportement fonctionnel attendu ;

2 *Défauts de communication* : les trois Supervisions locales Main_AD, Sub_AD et AD-3 reçoivent des signaux provenant d'un ensemble de capteurs partiellement distincts, mais surtout véhiculés par des canaux différents. Par conséquent, rien

ne garantit que ces signaux seront perçus au même instant par les trois Supervisions. Par exemple, l'événement **TGFF16_12** figurant sur les trois automates du Modèle Local version 12 (voir figure 6.10), n'est pas nécessairement simultané pour les trois automates, contrairement à l'hypothèse formulée jusqu'alors. Pour modéliser ce phénomène, des événements différents décrivant l'événement *perçu* par une des sous-fonctions peuvent être créés. En modifiant de la sorte les Modèles Locaux, le produit de la composition parallèle des automates qui les constituent sera nécessairement différent de celui, par exemple, représenté sur la figure 6.12. De nouvelles combinaisons d'états locaux, non répertoriées dans la Table 6.1, apparaîtront et la question sera alors de déterminer la dangerosité potentielle de ces nouveaux états globaux.

Ces problèmes constituent de réels questionnements de recherche et pourraient faire l'objet d'une continuation scientifique, sous la forme d'une autre thèse par exemple. La piste évoquée consiste à consolider les modèles d'état obtenus par certaines contraintes de l'implémentation. Néanmoins, cette voie reste à explorer et des questions supplémentaires se posent. Par exemple, est-il possible et pertinent de quantifier les retards potentiels entre les signaux ? Si cette possibilité est choisie, les modèles manipulés devront devenir temporisés (ce qui est possible dans UPPAAL). On pourrait alors déterminer, dans un premier temps, si des états non désirés sont atteignables à l'aide de modèles non temporisés (comme ceux de Supremica) en considérant les signaux comme des événements distincts. Puis, à l'aide d'automates temporisés et de quantification de retards entre signaux, vérifier l'atteignabilité effective, dans un intervalle de temps donné, de ces états non désirés. Ainsi des propriétés temporisées, et non plus temporelles, comme nous l'avons fait dans nos travaux, seraient vérifiées. D'importants efforts (caractérisation composants, détermination et modélisation dynamique des retards de signaux...) seraient à fournir pour suivre cette piste de travail.

Quatrième partie

Définitions

8
Glossaire

activité : ensemble d'actions qui consomment du temps et des ressources et dont l'exécution est nécessaire pour obtenir ou contribuer à la réalisation d'un ou de plusieurs résultats (d'après la norme ISO 15288:2008). 17

architecture fonctionnelle : arrangement de fonctions et de flux présenté sous un point de vue statique. 26

architecture physique : arrangement statique de composants, de flux physiques et d'interfaces. 27

boîte noire : représentation d'un système sans considérer son fonctionnement interne. 15

condition environnementale : une condition environnementale est relative à une évolution perceptible par les équipements embarqués dans le véhicule autonome de l'environnement. Ces changements sont principalement liés aux conditions extérieures (climat, type de route, signalétique...), l'état du véhicule (état des ouvrants, niveau d'huile moteur, etc), au comportement du conducteur (interactions avec l'IHM, les pédales, le volant...), ou encore à la disponibilité des fonctions de contrôle. 91

degré de définition d'un état : pour une exigence retenue, il s'agit de la quantité d'information disponible à propos d'un état. C'est-à-dire qu'un état peut être:

(a) **Entièrement défini (Df)**: l'intitulé de l'état apparaît de manière claire et non ambigüe dans l'énoncé de l'exigence considérée;

(b) **Non défini (NDf)**: l'intitulé de l'état n'apparaît pas du tout dans l'énoncé de l'exigence considérée;

(c) **Partiellement défini (PDf)**: l'intitulé de l'état apparaît mais sous une forme négative.

. 97

ensemble d'exigences : agrégation d'exigences (sélectionnées) contenant le ou les **même(s) type(s) d'état** caractérisés par le **même degré de définition**. 98

equation logique indépendante : deux équations logiques sont dites *indépendantes* si elles n'ont aucune variable commune. 122

etat d'arrivée : dans un contexte donné par une exigence définissant une ou plusieurs conditions environnementales et un ensemble d'états de départ, l'état d'arrivée est l'**unique** état dans lequel la fonction spécifiée doit se trouver suite à l'occurrence des conditions environnementales. L'état d'arrivée ne peut pas être un élément de l'ensemble des états de départ. 91

etat de départ : dans un contexte donné par une exigence, l'ensemble d'états de départ est l'ensemble des états de l'automate pour lequel l'exigence s'applique. Cet ensemble peut comprendre (ou non) l'état initial de l'automate. 91

état sûr : état spécifié par le métier de Sûreté de Fonctionnement atteint suite à l'occurrence d'une ou plusieurs défaillance(s) dans lequel le véhicule est mis en sécurité. 37

exigence : énoncé qui prescrit une fonction, une aptitude, une caractéristique ou une limitation à laquelle doit satisfaire un produit ou un processus dans des conditions d'environnement données. 26

exigence bien formée : une exigence est dite bien formée si les trois éléments de la Proposition 5.1 se retrouvent dans son énoncé, à savoir, au moins un **état de départ** et au moins une **condition environnementale** ainsi qu'un (unique) **état d'arrivée**. 92

exigence de comportement : exigence fonctionnelle spécifiant le comportement attendu d'un élément de l'architecture fonctionnelle ou de l'architecture physique. 39

exigence de comportement quantifiée : exigence de comportement qui fait référence explicite à des valeurs quantifiées telles que des seuils de probabilité ou des durées maximales d'exécution par exemple. 39

exigence de processus : énoncé spécifiant comment le produit doit être développé (activités, livrables...). 39

exigence de transition : exigence qui détermine les conditions sous lesquelles un changement d'état de l'élément spécifié doit se produire. Elles sont allouées à la fonction de Supervision. 39

exigence d'état :exigence qui définit les actions particulières à effectuer dans un état donné de l'élément spécifié. Ces exigences sont allouées à la fonction de Contrôle AD. 39

exigence fonctionnelle : énoncé spécifiant ce que le système doit faire en précisant les informations attendues et les conditions pour atteindre un but donné. 39

exigence fonctionnelle de sûreté de fonctionnement : spécification d'un comportement sûr ou d'une mesure de sécurité indépendants d'une implémentation donnée (d'après ISO 26262 part 1, définition 1.53). Ce type d'exigence est exprimée au niveau fonctionnel, c'est-à-dire qu'elle porte sur ou plusieurs éléments de l'architecture fonctionnelle du système étudié. 29

exigence technique de sûreté de fonctionnement : exigence dérivée d'une exigence fonctionnelle de sûreté de fonctionnement spécifique à une implémentation donnée (d'après ISO 26262 part 1, définition 1.133). Ce type d'exigence est exprimé au niveau physique, elle porte sur un ou plusieurs éléments de l'architecture physique. 30

flux : circulation de matière, d'énergie ou d'information. Un flux a toujours une fonction d'origine et une fonction de destination. 26

groupe bien formé : un groupe bien formé est un groupe qui ne contient que des exigences bien formées partageant le/les mêmes états de départ et le même état d'arrivée. 102

groupe d'exigences : agrégation des exigences appartenant à un même *ensemble d'exigences* et partageant le ou les mêmes état(s). 98

hypothèse de modélisation : hypothèses portant sur la modélisation en elle-même. Il s'agit par exemple de règles pour regrouper les exigences ou de la définition de variables et de paramètres pour la formalisation. Il revient aux experts de choisir parmi ces hypothèses lesquelles sont correctes et d'exclure celles qui ne le sont pas. 87

hypothèse d'interprétation : il s'agit d'hypothèses émises pour compléter une exigence (ou un groupe d'exigence) qui n'est pas de la forme attendue. Cette hypothèse implique la modification ou la création d'exigences. La confrontation avec les experts permet de trancher entre les hypothèses et seules les interprétations les plus crédibles sont retenues. 87

logique temporelle : extension de la logique propositionnelle intégrant des opérateurs temporels. 53

matrice de conformité : ce document permet de tracer l'application et la justification de toutes les exigences issues des concepts de sécurité (FSC, TSC). 27

niveau de contrainte d'état un état A est dit plus contraint qu'un état B si un nombre moins élevé de conditions doit être rempli pour passer de l'état A à l'état *actif* que pour passer de l'état B à l'état *actif*. 121

ontologie : ensemble structuré de termes et de concepts représentant le sens d'un champ d'information. 61

partie prenante : partie intéressée par l'utilisation et l'exploitation du système (voire par ses impacts sur son environnement), mais aussi participant à sa conception, sa production, son déploiement, sa commercialisation, son maintien en condition opérationnelle et son retrait de service. 17

processus : ensemble d'activités corrélées ou en interaction qui utilise des éléments d'entrée pour produire un résultat escompté (d'après la norme ISO 9000:2015). 17

processus de management : processus qui contribue à la détermination de la politique et au déploiement des objectifs de l'Ingénierie Produit. 24

processus opérationnel : processus constituant l'ossature de la logique de développement et décrivant l'enchaînement et la synchronisation des activités métiers ainsi que leur contribution aux livrables de l'ingénierie. 23

processus support : processus qui assure les ressources internes et externes nécessaires au bon déroulement des processus de l'Ingénierie Produit. 24

safety goal : exigence de sécurité de haut niveau issue de l'APR. 30

squelette de modèles d'état : graphe d'état sur lequel n'apparait que le libellé des états et les transitions orientées possibles entre ces états, dont les gardes ne sont pas définies. 106

système : un système est un arrangement d'éléments en interaction, organisé pour atteindre un ou plusieurs objectifs définis. 26

transition de groupe : une transition de groupe est une agrégation d'exigences qui ne contient que des exigences ayant le **même et unique état de départ** et le même état d'arrivée. Une transition de groupe est donc associée à une transition d'un modèle d'état. 119

transition de groupe contrainte : une transition de groupe est dite contrainte si son état de départ est un état moins contraint que son état d'arrivée. 122

transition de groupe formalisée : une transition de groupe est dite formalisée s'il existe des opérateurs logiques entre toutes les équations logiques associées aux exigences qui la constitue. 120

type d'état : pour une exigence donnée, le type d'état peut être *état de départ* ou *état d'arrivée*. 92

9
Acronymes

AADL Architecture Analysis and Design Language. 46

ABS AntiBlockierSystem. 13

AD Autonomous Driving. 33

AdD Arbres de Défaillances. 30

AFIS Association Française d'Ingénierie Système. 16

AMDEC Analyse des Modes de Défaillance de leurs Effets et de leurs Criticités. 29

AMS Architecte Métier Système. 26

APR Analyse Préliminaire des Risques. 29

ASIL Automotive Safety Integrity Level. 61, 94

ATESST Advancing Traffic Efficiency and Safety through Software Technology. 45

AUTOSAR AUTomotive Open System ARchitecture. 21

CIFRE Convention Industrielle de Formation par la REcherche. 42

COMPASS Correctness Modeling and Performance of Aerospace Systems. 46

CTL* Full Computation Tree Logic. 54

DAL Development Assurance Level. 20

DSML Domain Specific Modeling Language. 48

E/E Électroniques Embarqués. 13

EAST-ADL Embedded electronic Architecture and STudy - Architecture Description Language. 45

EIC Evénements Indésirés Client. 29

ERS Evénements Redoutés Système. 29

ESP Electronic Stability Program. 13

FSC Functional Safety Concept. 27

GPS Global Positioning System. 13

HIP-HOPS Hierarchically Performed Hazard Origin and Propagation Studies. 46

IEEE Institute of Electrical and Electronics Engineers. 17

IMOFIS Ingénierie des MOdèles de FonctIons Sécuritaires. 48

INCOSE International Council On Systems Engineering. 16

IS Ingénierie Système. 16

ISO Organisation internationale de normalisation. 20

LAAS-CNRS Laboratoire d'Analyse et d'Architecture des Systèmes - CNRS. 60

MAENAD Model-based Analysis and Engineering of Novel Architectures for Dependable Electric Vehicles. 45

MARTE Modeling and Analysis of Real Time and Embedded systems. 60

MBSA Model Based Safety Analysis/Assessment. 21

MBSE Model Based Systems Engineering. 19

MéDISIS Méthode d'Intégration des analyses de Sûreté de Fonctionnement au processus d'Ingénierie Systèmes. 45

MEMVATEX MEthode de Modelisation pour la VAlidation et la Tracabilité des EXigences. 60

MRM Minimal Risk Manoeuver. 36

NASA National Aeronautics and Space Administration. 16

NHTSA National Highway Traffic Safety Administration. 33

NLP Natural Language Processing. 53

OICA Organisation Internationale des Constructeurs Automobiles. 33

OMG Object Management Group. 19

SAE Society of Automotive Engineers. 20

SCR Système de Conception Renault. 23

SDD System Design Document. 27

SdF Sûreté de Fonctionnement. 15, 16

STR System Technical Requirements. 27

SysML Systems Modeling Language. 19

TSC Technical Safety Concept. 27

UCE Unités de Commande Électronique. 14

UML Unified Modeling Language. 60

UNECE United Nations Economic Commission for Europe. 33

USAF United States Air Force. 16

V&V Vérification & Validation. 61

Cinquième partie

Bibliographie

Bibliographie

Akesson, K. et al. (2003). « Supremica—a tool for verification and synthesis of discrete event supervisors ». In : *11th mediterranean conference on control and automation*.

Albinet, A., S. Begoc et al. (2008). « The MeMVaTEx methodology : from requirements to models in automotive application design ». In : *4th European congress ERTS (embedded real time software), Toulouse, France*.

Albinet, A., J.-L. Boulanger et al. (2007a). « Model-Based Methodology for Requirements Traceability in Embedded Systems ». In : *Proceedings of 3rd European Conference on Model Driven Architecture Foundations and Applications, ECMDA'07*.

— (2007b). « Model-based methodology for requirements traceability in embedded systems ». In : *Proceedings of 3rd European Conference on Model Driven Architecture Foundations and Applications, ECMDA'07*.

Almeida Ferreira, D. de et A. R. da Silva (2009). « A controlled natural language approach for integrating requirements and model-driven engineering ». In : *Software Engineering Advances. ICSEA'09. Fourth International Conference on*. IEEE, p. 518-523.

AMS Renault (juillet 2017a). *System Design Document AD System AD2.1*.

— (août 2017b). *System Technical Requirements AD System V1.2*.

Apvrille, L. et A. Becoulet (2012). « Prototyping an embedded automotive system from its UML/SysML models ». In : *ERTSS'2012*.

Badreau, S. et J.-L. Boulanger (2014). *Ingénierie des exigences : Méthodes et bonnes pratiques pour construire et maintenir un référentiel*. Dunod.

Baier, C. et J.-P. Katoen (2008). *Principles of model checking*. MIT press.

Bart Koolmees Michel Reniers, J. M. (2011). *Validation of model behavior using UPPAAL*.

Behere, S. et M. Törngren (2016). « A functional reference architecture for autonomous driving ». In : *Information and Software Technology* 73, p. 136-150.

Behm, P. et al. (1999). « METEOR : A successful application of B in a large project ». In : *International Symposium on Formal Methods*. Springer, p. 369-387.

Behrmann, G., A. David et K. G. Larsen (2004). « A tutorial on uppaal ». In : *Formal methods for the design of real-time systems*. Springer, p. 200-236.

Belmonte, F. et E. Soubiran (2012). « A Model Based Approach for Safety Analysis ». In : *SAFECOMP Workshop*.

Biggs, G., T. Sakamoto et T. Kotoku (2016). « A profile and tool for modelling safety information with design information in SysML ». In : *Software & Systems Modeling* 15.1, p. 147-178.

Blom, H. et al. (2016). « EAST-ADL : An Architecture Description Language for Automotive Software-intensive Systems in the Light of Recent use and Research ». In : *International Journal of System Dynamics Applications (IJSDA)* 5.3, p. 1-20.

Boehm, B. W. (1988). « A spiral model of software development and enhancement ». In : *Computer* 21.5, p. 61-72.

Boiteau, M. et al. (2006). « The AltaRica data-flow language in use : modeling of production availability of a multi-state system ». In : *Reliability Engineering & System Safety* 91.7, p. 747-755.

Boulanger, J.-L. (2014). *Formal methods applied to complex systems : implementation of the B method*. John Wiley & Sons.

Bozzano, M. et al. (2010). « Formal verification and validation of AADL models ». In : *ERTS 2010 - Embedded Real Time Software and Systems*.

Cabral, G. et A. Sampaio (2008). « Formal specification generation from requirement documents ». In : *Electronic Notes in Theoretical Computer Science* 195, p. 171-188.

Cassandras, C. G. et S. Lafortune (2009). *Introduction to discrete event systems*. Springer Science & Business Media.

Chanaron, J.-J. (1995). « Constructeurs/Fournisseurs : spécificités et dynamique d'évolution des modes relationnels ». In : *Actes du GERPISA* 14, p. 9-22.

Chen, D. et al. (2011). « Integrated safety and architecture modeling for automotive embedded systems ». In : *Elektrotechnik und Informationstechnik* 128.6, p. 196-202.

Chong, J. (2016). *Véhicules autonomes et connectés : état d'avancement de la technologie et principaux enjeux stratégiques pour les pouvoirs publics au Canada*. Bibliothèque du Parlement= Library of Parliament.

Clarke, E. M. et E. A. Emerson (1981). « Design and synthesis of synchronization skeletons using branching time temporal logic ». In : *Workshop on Logic of Programs*. Springer, p. 52-71.

Cressent, R. (2012). « Valorisation de l'Ingénierie Système à Base de Modèles, pour l'analyse de sûreté de fonctionnement des systèmes complexes critiques intégrant des COTS ». Thèse de doctorat dirigée par Kratz, Frédéric Sciences et technologies industrielles Orléans 2012. Thèse de doct.

Cressent, R., P. David et al. (2013). « Designing the database for a reliability aware Model-Based System Engineering process ». In : *Reliability Engineering & System Safety* 111, p. 171-182.

Cressent, R., V. Idasiak et F. Kratz (2011). « Prise en compte des analyses de la sûreté de fonctionnement dans l'ingénierie de système dirigée par les modèles SysML ». In : *Génie logiciel*, p33-39.

Cressent, R., V. Idasiak, F. Kratz et P. David (2011). « Mastering safety and reliability in a model based process ». In : *Reliability and Maintainability Symposium (RAMS), 2011 Proceedings-Annual*. IEEE, p. 1-6.

David, P. (nov. 2009). « Contribution to reliability analysis of complex systems during their design phase : application to the evaluation of human sensors networks missions ». Theses. Université d'Orléans.

David, P., V. Idasiak et F. Kratz (2010). « Reliability study of complex physical systems using SysML ». In : *Reliability Engineering & System Safety* 95.4, p. 431-450.

Djoudi, B., C. Bouanaka et N. Zeghib (2016). « A formal framework for context-aware systems specification and verification ». In : *Journal of Systems and Software* 122, p. 445-462. issn : 0164-1212.

Dumitrescu, C. et al. (2013). « Bridging the gap between product lines and systems engineering : an experience in variability management for automotive model based systems engineering ». In : *Proceedings of the 17th International Software Product Line Conference*. ACM, p. 254-263.

Dumont, J., F. Sadmi et F. Vallee (2011). « Safety Architect© : un outil d'analyse de risques s' inscrivant dans les processus d'ingénierie de systèmes complexes ». In : *Genie logiciel* 98, p. 27-33.

Dwyer, M. B., G. S. Avrunin et J. C. Corbett (1998). « Property specification patterns for finite-state verification ». In : *Proceedings of the second workshop on Formal methods in software practice*. ACM, p. 7-15.

Electronic Industries Alliance (sept. 2003). *EIA-632 :2003 Processes for Engineering a System*.

Ernst, E. (2003). « Separation of concerns ». In : *Proceedings of the AOSD 2003 Workshop on Software-Engineering Properties of Languages for Aspect Technologies (SPLAT), Boston, MA, USA*.

Evrot, D. (2008). « Contribution à la vérification d'exigences de sécurité : application au domaine de la machine industrielle ». Thèse de doct. Université Henri Poincaré-Nancy I.

Fagnant, D. J. et K. Kockelman (2015). « Preparing a nation for autonomous vehicles : opportunities, barriers and policy recommendations ». In : *Transportation Research Part A : Policy and Practice* 77, p. 167-181. issn : 0965-8564.

Fennel, H. et al. (2006). *Achievements and exploitation of the AUTOSAR development partnership*. Rapp. tech. SAE Technical Paper.

Filax, M., T. Gonschorek et F. Ortmeier (2016). « Correct formalization of requirement specifications : a v-model for building formal models ». In : *International Conference on Reliability, Safety and Security of Railway Systems*. Springer, p. 106-122.

Fiorèse, S. et J.-P. Meinadier (2012). « Découvrir et comprendre l'ingénierie système ». In : *CEPADUES Editions, ISBN* 978.36493.005, p. 6.

Florian, M. et al. (2014). « Multi-view modeling in SysML : thematic structuring for multiple thematic views ». In : *Procedia Computer Science* 28, p. 531-538.

Forsberg, K., H. Mooz et H. Cotterman (2005). *Visualizing project management : models and frameworks for mastering complex systems*. John Wiley & Sons.

Fürst, S. et al. (2009). « AUTOSAR–A Worldwide Standard is on the Road ». In : *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden*. T. 62, p. 5.

Gaudré, T. (février 2016). *Formation Renault : Ingénierie des Exigences (IE)*.

Genius, D. et L. Apvrille (2016). « Virtual yet precise prototyping : An automotive case study ». In : *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, p. 691-700.

Gervasi, V. et B. Nuseibeh (2002). « Lightweight validation of natural language requirements ». In : *Software : Practice and Experience* 32.2, p. 113-133.

Ghazel, M., J. Yang et E.-M. El-Koursi (2015). « A pattern-based method for refining and formalizing informal specifications in critical control systems ». In : *Journal of Innovation in Digital Ecosystems* 2.1, p. 32-44. ISSN : 2352-6645.

Góngora, H. G. C., T. Gaudré et S. Tucci-Piergiovanni (2013). « Towards an architectural design framework for automotive systems development ». In : *Complex Systems Design & Management*. Springer, p. 241-258.

Groupe Renault - Direction de la Qualité et de la Satisfaction Client (Avril 2016). *V3P-2 mémento de référence*.

— (jan. 2018). *Présentation du Système de Conception Renault*.

Groupe Renault - Direction de la Qualité/ Sûreté de Fonctionnement - Sécurité générale du Produit (Décembre 2017). *Procédure : Maîtriser la sûreté de fonctionnement des systèmes mécatroniques*.

Groupe Renault - Direction Méthodes et Standards de Conception (Avril 2018). *Procédure : Concevoir le système*.

Güdemann, M. (2011). « Qualitative and quantitative formal model-based safety analysis : push the safety button ». Theses. Magdeburg, Univ., Fak. für Informatik.

Guillerm, R. (juin 2011). « Integration of Dependability in System Engineering Processes ». Theses. Université de Toulouse.

Guychard, C. et al. (2013). « Conceptual interoperability through models federation ». In : *Semantic Information Federation Community Workshop*.

Hautiere, N., H. Tattegrain et M. Guilbot (2017). « Véhicules connectés et autonomes : quels enjeux technologiques, juridiques et de sécurité routière ? » In : *Hygiène et Sécurité du Travail* 246, pp-100.

Hürsch, W. L. et C. V. Lopes (1995). *Separation of Concerns*. Rapp. tech.

IAV (sept. 2016). *RSA AD Functional Safety Concept*.

Idasiak, V. et R. Kajdan (2014). « Ingénierie système basée modèle et analyse des défaillances : retour d'expérience ». In : *3C-Ingénierie dirigée par les modèles : SysML,...*

IEEE (jan. 2005). *IEEE 1220 :2005 Application and management of the systems engineering process*.

ISO (2011). *ISO 26262 Road Vehicles - Functional Safety*.

— (oct. 2015). *ISO 9000 :2015 Systèmes de management de la qualité - Principes essentiels et vocabulaire*.

ISO/CEI (2000). *ISO/CEI 61508 :2000 Functional safety of electrical/ electronic/ programmable electronic safety-related,*

— (2008). *ISO/CEI/IEEE 15288 :2008 Systems and Software Engineering - Systems Life Cycle Process*.

— (Avril 2014). *ISO/CEI Guide 51 :2014 Aspects liés à la sécurité – Principes directeurs pour les inclure dans les normes*.

Jo, K. et al. (2015). « Development of autonomous car—Part II : A case study on the implementation of an autonomous driving system based on distributed architecture ». In : *IEEE Transactions on Industrial Electronics* 62.8, p. 5119-5132.

Joshi, A. et M. P. Heimdahl (2005). « Model-based safety analysis of simulink models using SCADE design verifier ». In : *International Conference on Computer Safety, Reliability, and Security*. Springer, p. 122-135.

Joshi, A., M. Whalen et M. P. Heimdahl (2006). *Model-Based Safety Analysis Final Report*. Citeseer.

Judalet, V. (avr. 2016). « Robust architecture for the shared control of by-wire vehicles ». Theses. Université Paris-Saclay.

Kaiser, B. et al. (2010). « Integrating system modelling with safety activities ». In : *International Conference on Computer Safety, Reliability, and Security*. Springer, p. 452-465.

Kalra, N. et S. M. Paddock (2016). « Driving to safety : How many miles of driving would it take to demonstrate autonomous vehicle reliability ? » In : *Transportation Research Part A : Policy and Practice* 94, p. 182-193. issn : 0965-8564.

Kang, E.-Y. et al. (2013). « A methodology for formal analysis and verification of EAST-ADL models ». In : *Reliability Engineering & System Safety* 120, p. 127-138.

Kawahara, R. et al. (2009). « Verification of embedded system's specification using collaborative simulation of SysML and simulink models ». In : *Model-Based Systems Engineering, 2009. MBSE'09. International Conference on*. IEEE, p. 21-28.

l'Express (mar. 2018). *Uber suspend son programme de voitures autonomes après un accident mortel*. Article consulté en ligne le 05/04/2018.

Lasalle, J., F. Peureux et F. Fondement (2011). « Development of an automated MBT toolchain from UML/SysML models ». In : *Innovations in Systems and Software Engineering* 7.4, p. 247-256.

Le Monde (décembre 2017). *A la suite d'une erreur de calcul, la Russie perd un satellite*. Article consulté en ligne le 12/04/2018.

Lebeaupin, B. (2015). « A language for writing system specifications in an aeronautical context ». In : *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*. IEEE, p. 406-411.

— (2017). « Vers un langage de haut niveau pour une ingénierie des exigences agile dans le domaine des systèmes embarqués avioniques ». Thèse de doct. Paris Saclay.

Lebeaupin, B., A. Rauzy et J.-M. Roussel (2017). « A language proposition for system requirements ». In : *Systems Conference (SysCon), 2017 Annual IEEE International*. IEEE, p. 1-8.

Legendre, A. (déc. 2017). « System engineering and dependability : methodology synchronization of models ». Theses. Université Paris-Saclay.

Legendre, A., A. Lanusse et A. Rauzy (2017). « Toward Model Synchronization Between Safety Analysis and System Architecture Design in Industrial Contexts ». In : *Model-Based Safety and Assessment*. Sous la dir. d'Y. P. Marco Bozzano. T. 10437. Proceedings of the 5th International Symposium, IMBSA 2017, Trento, Italy, September 11–13, 2017, Springer, p. 35-49.

Leibinger, R. (2015). « Software architectures for advanced driver assistance systems (ADAS) ». In : *Agenda : Short overview of Elektrobit automotive*.

Leveson, N. G. (2002). « An approach to designing safe embedded software ». In : *International Workshop on Embedded Software*. Springer, p. 15-29.

Lions, J.-L. et al. (1996). *Ariane 5 flight 501 failure*.

Liu, X. et Z. Zhu (2011). « Construct Aspectual Models from Requirement Documents for Model-driven Development of Automotive Software ». In : *Electronic Notes in Theoretical Computer Science* 274, p. 33-50.

Marinescu, R. et al. (2014). « Analyzing industrial architectural models by simulation and model-checking ». In : *International Workshop on Formal Techniques for Safety-Critical Systems*. Springer, p. 189-205.

Markovski, J. et J. Van de Mortel-Fronczak (2012). « Modeling for safety in a synthesis-centric systems engineering framework ». In : *International Conference on Computer Safety, Reliability, and Security*. Springer, p. 36-49.

Mauborgne, P. (mai 2016). « Towards a Safe Systems Engineering ». Theses. Université de Lorraine.

Mauborgne, P., S. Deniaud, E. Levrat, E. Bonjour, P. Lamothe et al. (2013). « Comment relier l'ingénierie système et la sûreté de fonctionnement ? » In : *10e Congrès international de Génie Industriel, CIGI'2013*.

Mauborgne, P., S. Deniaud, E. Levrat, E. Bonjour, J.-P. Micaëlli et al. (2016). « Operational and system hazard analysis in a safe systems requirement engineering process–Application to automotive industry ». In : *Safety science* 87, p. 256-268.

Maurer, M. et H. Winner (2013). *Automotive Systems Engineering*. Springer Publishing Company, Incorporated.

Mckinley, B. (jan. 2016). *Voitures connectées : les nouveaux défis du test électronique*. Article en ligne, consulté le 05/04/2018.

Mécatronique, A. (mai 2014). *Automobile et mécatronique : l'histoire d'une évolution parallèle*. Article en ligne, consulté le 05/04/2018.

Mhenni, F. (déc. 2014). « Safety analysis integration in a systems engineering approach for mechatronic systems design ». Theses. Ecole Centrale Paris.

Micouin, P. (2011). « Exigences en ingénierie systèmes basée modèles ». In : *9 ième Congrès International de Génie Industriel*.

Mihal, A. et al. (2002). « Developing architectural platforms : A disciplined approach ». In : *IEEE Design & Test of Computers* 19.6, p. 6-16.

Moati, P. (2001). « Organiser les marchés dans une économie fondée sur la connaissance : le rôle clé des «intégrateurs» ». In : *Revue d'économie industrielle* 97.1, p. 123-138.

Mohajerani, S., R. Malik et M. Fabian (2017). « Compositional synthesis of supervisors in the form of state machines and state maps ». In : *Automatica* 76, p. 277-281.

Moncelet, G. (1998). « Application des réseaux de Petri a l'évaluation de la sûrete de fonctionnement des systèmes mécatroniques du monde automobile ». Thèse de doct. Université Henri Poincaré-Nancy I, p. 158.

New York Times (sept. 2016). *Autopilot Cited in Death of Chinese Tesla Driver*. Article consulté en ligne le 05/04/2018.

NHTSA (juin 2016). *Failure Report Summary, Autonomous vehicle control systems*. Article consulté en ligne le 05/04/2018.

— (sept. 2017). *AUTOMATED DRIVING SYSTEMS A vision for Safety*.

Nouacer, R. et al. (2016). « EQUITAS : A tool-chain for functional safety and reliability improvement in automotive systems ». In : *Microprocessors and Microsystems* 47, p. 252-261.

OICA (mar. 2014). *Automated Driving : Definition for Levels of Automation*.

Papadopoulos, Y. et al. (2011). « Engineering failure analysis and design optimisation with HiP-HOPS ». In : *Engineering Failure Analysis* 18.2, p. 590-608.

Peres, F., J. Yang et M. Ghazel (2012). « A formal framework for the formalization of informal requirements ». In : *The International Journal of Soft Computing and Software Engineering* 2.8, p14-27.

Pétin, J.-F. et al. (2010). « Combining SysML and formal methods for safety requirements verification ». In : *22nd International Conference on Software & Systems Engineering and their Applications*, CDROM.

Piques, J. et E. Andrianarison (2011). « SysML for embedded automotive systems : lessons learned ». In : *Interfaces* 3, 3b.

Pnueli, A. (1977). « The temporal logic of programs ». In : *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE, p. 46-57.

Pohl, K. (2010). *Requirements engineering : fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated.

Promoneuve Opinion Way (sept. 2016). *Les Français et les voitures neuves*.

Ramadge, P. et W. M. Wonham (1982). « Supervision of discrete event processes ». In : *Decision and Control, 1982 21st IEEE Conference on*. T. 21. IEEE, p. 1228-1229.

Rohée, B. et al. (juin 2006). « A methodology to design and check a plant model ». In : *3rd IFAC Workshop on Discrete-Event System Design (DESDes'06)*. Rydzyna, Poland, pp. 246-250.

Roussel, J.-M. et B. Denis (2002). « Safety properties verification of ladder diagram programs ». In : *Journal Européen des Systemes Automatisés (JESA)* 36.7, pp-905.

RTCA-Inc (Mai 2012). *DO 178C :2012 Software considerations in airborne systems and equipment certification*.

SAE International (Décembre 1996). *ARP4761 :1996 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*.

— (Décembre 2010). *ARP4754 :2011 Guidelines For Development Of Civil Aircraft and Systems*.

SEBoK (2017). *Guide to the Systems Engineering Body of Knowledge (SEBoK) — SEBoK*. [Online ; accessed 23-May-2018].

Sharvia, S. et Y. Papadopoulos (2015). « Integrating model checking with HiP-HOPS in model-based safety analysis ». In : *Reliability Engineering & System Safety* 135, p. 64-80.

Simonot-Lion, F. et N. Navet (2006). *Les réseaux temps réel embarqués dans les véhicules*.

Standards Council of Canada (2002). *Systems Engineering - Guide for ISO/IEC 15288 (Sytem Life Cycle Processes)*. Rapp. tech. ISO.

Stineau, J.-Y. (nov. 2015). *Sensibilisation à l'architecture électrique et électronique*. Formation interne Renault.

Taofifenua, O. (juil. 2012). « Ontology centric design process : Sharing a conceptualization ». Theses. Conservatoire national des arts et metiers - CNAM.

Taş, Ö. Ş. et al. (2016). « Functional system architectures towards fully automated driving ». In : *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE, p. 304-309.

TNS Sofres (mai 2014). *Les Français et l'Automobile - vague 5*.

Tucci-Piergiovanni, S. et al. (2014). « Model-Based Analysis and Engineering of Automotive Architectures with EAST-ADL ». In : *Handbook of Research on Embedded Systems Design*, p. 242-282.

UNECE (mar. 2016). *La UNECE ouvre la voie à la conduite automatisée en modifiant la Convention de Vienne sur la circulation routière*. Article consulté en ligne le 19/04/2018.

US Department of Transportation (2008). *National Motor Vehicle Crash Causation Survey Report DOT HS 811 059*. Rapp. tech. National Highway Traffic Safety Administration.

Verries, J. (2010). « Approche pour la conception de systèmes aéronautiques innovants en vue d'optimiser l'architecture. Application au système portes passager ». Thèse de doct. Université Paul Sabatier-Toulouse III.

Villemeur, A. (1988). « Sureté de fonctionnement des systèmes industriels : fiabilité-facteurs humains, informatisation ». In :

Weissnegger, R. et al. (2015). « A novel method to speed-up the evaluation of cyber-physical systems (ISO 26262) ». In : *Intelligent Solutions in Embedded Systems (WISES), 2015 12th International Workshop on*. IEEE, p. 109-114.

Woodcock, J. et al. (2009). « Formal methods : Practice and experience ». In : *ACM computing surveys (CSUR)* 41.4, p. 19.

Wu, W. et T. Kelly (2006). « Deriving safety requirements as part of system architecture definition ». In : *Proceedings of 24th International System Safety Conference, System Safety Society*.

Yakymets, N., S. Dhouib et al. (2013). « Model-driven safety assessment of robotic systems ». In : *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, p. 1137-1142.

Yakymets, N., M. Perin et A. Lanusse (oct. 2014). « Modélisation des réseaux en AltaRica 3.0 ». In : *19e Congrès de Maîtrise des Risques et Sûreté de Fonctionnement*.

— (2015). « Model-driven multi-level safety analysis of critical systems ». In : *Systems Conference (SysCon), 2015 9th Annual IEEE International*. IEEE, p. 570-577.

Zaytoon, J. et B. Riera (2017). « Synthesis and implementation of logic controllers–a review ». In : *Annual reviews in control* 43, p. 152-168.

Zhang, W. et al. (2012). « Towards tool integration through artifacts and roles ». In : *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific*. T. 1. IEEE, p. 603-613.

Zhioua, Z., Y. Roudier et R. Ameur-Boulifa (2017). « Formal specification of security guidelines for program certification ». In : *Theoretical Aspects of Software Engineering (TASE), 2017 International Symposium on*. IEEE, p. 1-8.

Zhioua, Z., Y. Roudier et R. B. Ameur (2017). « Formal specification and verification of security guidelines ». In : *Dependable Computing (PRDC), 2017 IEEE 22nd Pacific Rim International Symposium on*. IEEE, p. 267-273.

Sixième partie

Annexes

10
Figures et Tables supplémentaires

Sommaire

10.1 Squelettes de modèles d'état

10.1.1 Squelettes fonctionnels résultants de l'analyse groupe par groupe

Suite à l'analyse de plausibilité groupe par groupe, 9 squelettes de modèles d'état de plausibilité forte sont obtenus :

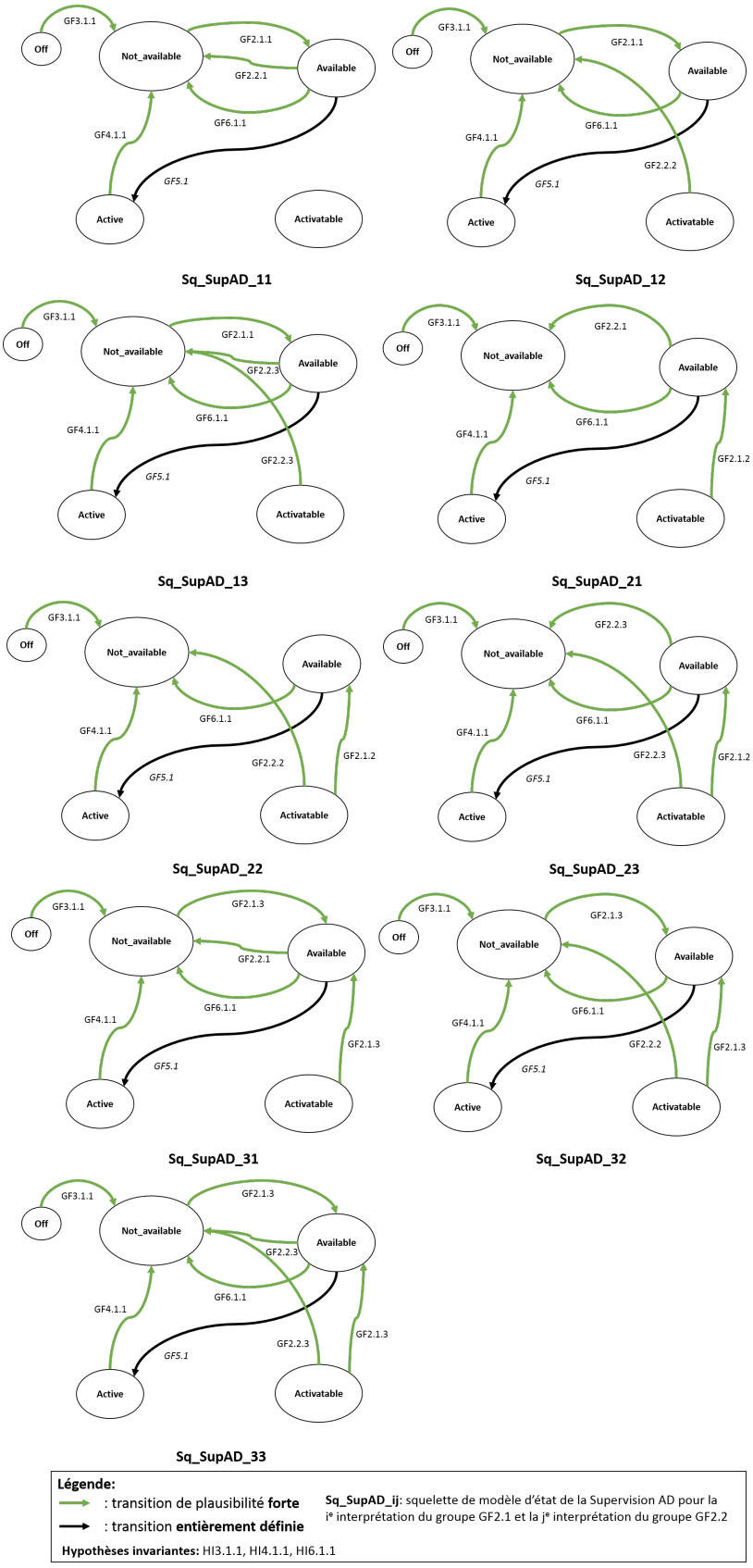


FIGURE 10.1 – Squelettes de modèles d'état fonctionnels de plausiblité forte : analyse groupe par groupe

10.1.2 Squelettes résultants de l'analyse « inter-groupes »

En complétant l'analyse groupe par groupe par une analyse dite « inter-groupes », c'est-à-dire une analyse des squelettes précédemment obtenus, nous obtenons les 6 squelettes de modèles d'état de la figure 10.2.

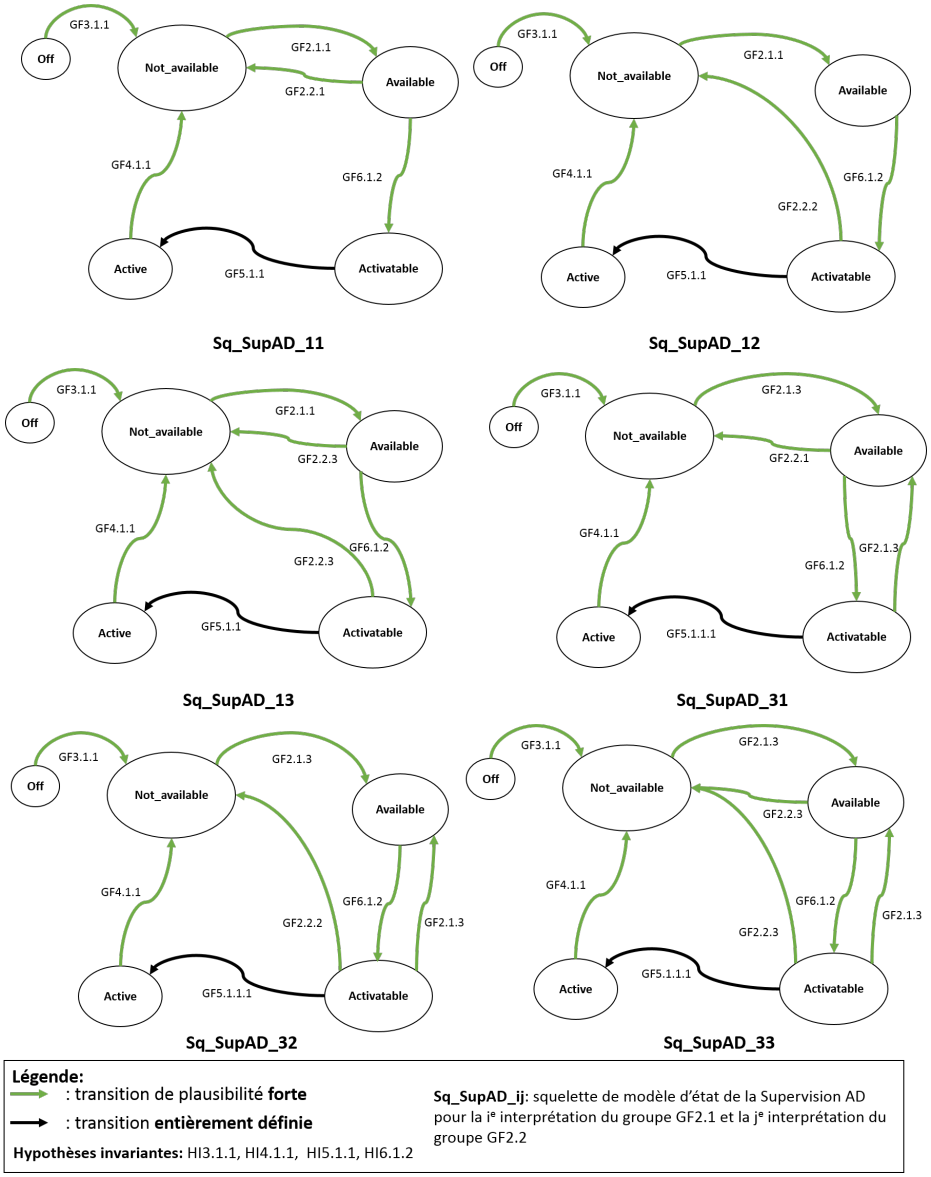


FIGURE 10.2 – Squelettes de modèles d'état fonctionnels de plausiblité forte : analyse « inter-groupes »

10.1.3 Squelettes de modèles d'état de sûreté retenus

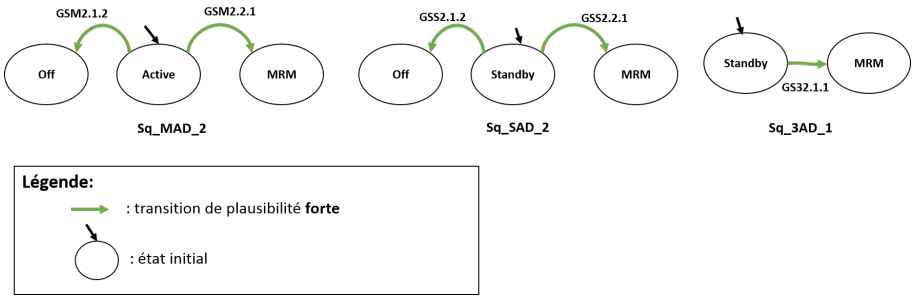


FIGURE 10.3 – Squelettes de modèles d'état de sûreté retenus par les pilotes SdF

10.2 Variables

10.2.1 Variables fonctionnelles

Dénomination	Description	Type
igni	« Ignition » : alimentation électrique	Booléen
road_sec	Section de route	Booléen
veh_pos	Position du véhicule	Entier $\in [0, 6]$
veh_dir	Direction du véhicule	Booléen
forec_AD_dur	Durée prévue de l'activation de la fonction AD	Booléen
veh_loc	Position globale du véhicule	Entier $\in [0, 2]$
lan_width	Largeur de la voie courante	Entier $\in [0, 2]$
weat_cond	Conditions climatiques	Entier $\in [0, 8]$
lane_typ	Type de ligne courante	Entier $\in [0, 13]$
adj_la_typ	Type de la ligne adjacente	Booléen
cstr_area	Zone de chantier	Booléen
tfc_sign	Panneaux de signalisation	Booléen
veloc	Vitesse du véhicule	Entier $\in [0, 3]$
open_part	État des ouvrants	Entier $\in [0, 3]$
AD_activ	Activation de la fonction AD	Booléen
AD_button	État du bouton d'activation	Booléen
gear_lev	Vitesse enclenchée	Entier $\in [0, 2]$
lat_dist	Distance latérale moyenne entre le centre du véhicule et les lignes	Booléen
vel_reduc	Réduction de la vitesse	Booléen
forec_end	Fin prévue de l'activation de la fonction AD	Entier $\in [0, 3]$
dist_ped	Distance d'un piéton détecté	Booléen
tfc_info	Informations sur le trafic	Booléen
AE_inter	Durée d'intervention de l'ABS ou ESC	Booléen
mean_accel	Valeur de l'accélération moyenne	Booléen
pos_dur	Durée du maintien d'une position véhicule	Entier $\in [0, 3]$
rte_def	Définition d'un itinéraire	Entier $\in [0, 2]$
adj_L_lane	Existence d'une ligne adjacente gauche	Booléen
val_adj_R_lane	Différentiel entre la vitesse du véhicule et celle de vehicule sur la voie ajacente de droite	Booléen
adj_R_lane	Existence d'une ligne adjacente droite	Booléen
val_adj_L_lane	Différentiel entre la vitesse du véhicule et celle de vehicule sur la voie ajacente de gauche	Booléen
R_gap_spd	Différence entre la vitesse moyenne de la voie de droite et la vitesse véhicule	Booléen
safe_lane	Voie cible sûre	Booléen
tqe_steer	Couple actuel sur la direction	Entier $\in [0, 3]$

tqe_dur	Durée du couple sur la direction	Entier $\in [0, 3]$
foot_ped	Pied sur une pédale	Booléen
hand_steer	Main sur le volant	Booléen
pres_ac_ped	Pression courante sur la pédale d'accélération	Booléen
pres_ac_dur	Durée de la pression sur la pédale d'accélération	Booléen
foot_ac_ped	Pied sur la pédale d'accélération	Booléen
ac_ped_thru	Poussée sur la pédale d'accélération	Booléen
pres_br_ped	Pression courante sur la pédale de frein	Entier $\in [0, 3]$
thru_ac_dur	Durée de la poussée sur la pédale d'accélération	Booléen
pres_br_dur	Durée de la pression sur la pédale de frein	Booléen
foot_br_ped	Pied sur la pédale de frein	Booléen
turn_indic	État des clignotants	Booléen
spd_veh	Vitesse du véhicule	Booléen
dist_obj	Distance du véhicule à un objet	Booléen
perc_cap	Capacités actuelles de perception	Booléen
dif_pos	Différence entre position requise et position effective	Booléen
time_activ	Durée d'activation de la fonction AD	Booléen
veh_decel	Décélération du véhicule	Booléen
L_gap_spd	Différentiel entre la vitesse du véhicule et celle des vehicules sur la voie ajacente de gauche	Booléen

TABLE 10.1 – Variables fonctionnelles

10.2.2 Variables de sûreté

Dénomination	Description	Type
m_acti	Activation de la fonction AD par Control_Main_Active	Booléen
m_deac	Désactivation de la fonction AD par Control_Main_Active	Booléen
m_stay	Non désactivation de la fonction AD due à Control_Main_Active	Booléen
m_notif	Notifications envoyées par Control_Main_Active	Booléen
m_accel	Accélération commandée par Control_Main_Active	Entier $\in [0, 3]$
m_decel	Décélération commandée par Control_Main_Active	Entier $\in [0, 3]$
m_lat_accel	Accélération latérale commandée par Control_Main_Active	Entier $\in [0, 3]$
m_visib	Informations lumineuses aux autres véhicules émises par Control_Main_Active	Entier $\in [0, 3]$
m_info	Informations émises au conducteur par Control_Main_Active	Entier $\in [0, 3]$
s_acti	Activation de la fonction AD par Control_Sub_Standby	Booléen
s_deac	Désactivation de la fonction AD par Control_Sub_Standby	Booléen
s_stay	Non désactivation de la fonction AD due à Control_Sub_Standby	Booléen
s_notif	Notifications envoyées par Control_Sub_Standby	Booléen
s_accel	Accélération commandée par Control_Sub_Standby	Entier $\in [0, 3]$
s_decel	Décélération commandée par Control_Sub_Standby	Entier $\in [0, 3]$
s_lat_accel	Accélération latérale commandée par Control_Sub_Standby	Entier $\in [0, 3]$
s_visib	Informations lumineuses aux autres véhicules émises par Control_Sub_Standby	Entier $\in [0, 3]$
s_info	Informations émises au conducteur par Control_Sub_Standby	Entier $\in [0, 3]$
AD3_loss	Disponibilité de la fonction Control_3_Standby	Booléen

TABLE 10.2 – Variables de sûreté

10.3 Transitions de groupe

10.3.1 Transitions de Groupe Fonctionnelles

La Table 10.3 liste l'ensemble des transitions de groupe possibles, du point de vue des AMS.

Transitions de Groupe Fonctionnelles (TGF)	État de départ	État d'arrivée	**Contrainte (C) Non Contrainte (NC)**
TGF1	Off	Not_available	NC
TGF2	Off	Available	NC
TGF3	Off	Activatable	NC
TGF4	Off	Active	NC
TGF5	Not_available	Off	C
TGF6	Not_available	Available	NC
TGF7	Not_available	Activatable	NC
TGF8	Not_available	Active	NC
TGF9	Available	Off	C
TGF10	Available	Not_available	C
TGF11	Available	Activatable	NC
TGF12	Available	Active	NC
TGF13	Activatable	Off	C
TGF14	Activatable	Not_available	C
TGF15	Activatable	Available	C
TGF16	Activatable	Active	NC
TGF17	Active	Off	C
TGF18	Active	Not_available	C
TGF19	Active	Available	C
TGF20	Active	Activatable	C

TABLE 10.3 – Transitions de Groupe Fonctionnelles

10.3.2 Transitions de Groupe de Sûreté

Fonction Concernée	**Transitions de Groupe de Sûreté (TGS)**	État de départ	État d'arrivée	**Contrainte (C) Non Contrainte (NC) - : Non pertinent**
Main_AD	TGSM1	Off	Active	-
	TGSM2	Off	MRM	-
	TGSM3	Active	Off	NC
	TGSM4	Active	MRM	NC
	TGSM5	MRM	Off	-
	TGSM6	MRM	Active	-
Sub_AD	TGSS1	Off	Standby	-
	TGSS2	Off	MRM	-
	TGSM3	Standby	Off	NC
	TGSS4	Standby	MRM	NC
	TGSS5	MRM	Off	-
	TGSS6	MRM	Standby	-
AD-3	TGS31	Off	Standby	-
	TGS32	Off	MRM	-
	TGSM3	Standby	Off	NC
	TGS34	Standby	MRM	NC
	TGS35	MRM	Off	-
	TGS36	MRM	Standby	-

TABLE 10.4 – Transitions de Groupe de Sûreté

10.4 Modèles d'État Fonctionnels préliminaires

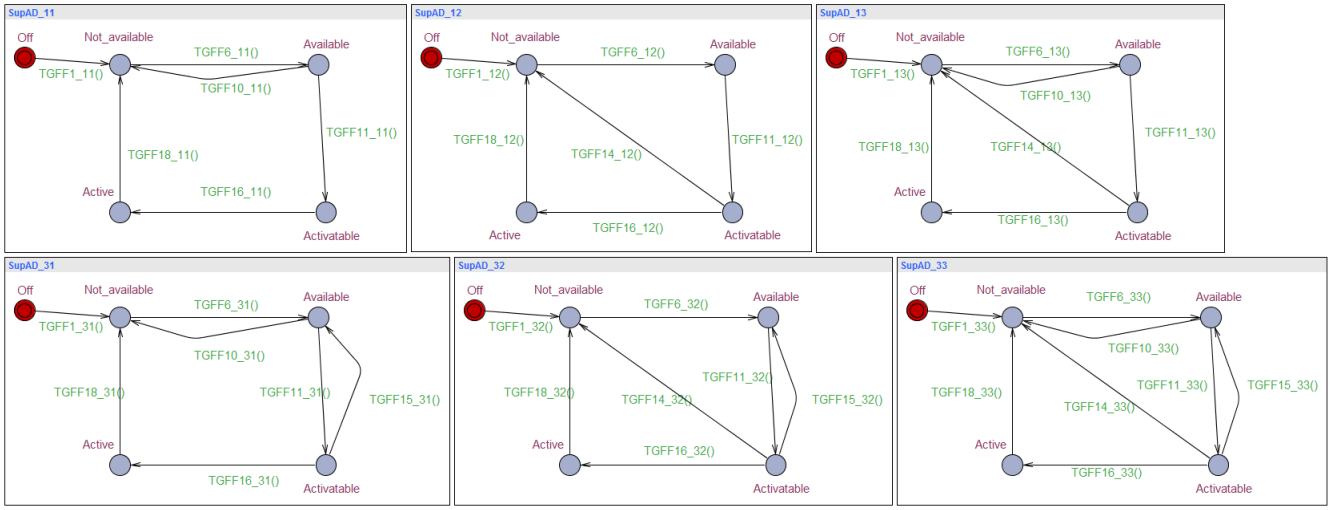


FIGURE 10.4 – Modèles d'état préliminaires fonctionnels

10.5 Modèles d'État Fonctionnels vérifiés

10.5.1 Version UPPAAL

La figure 10.5 représente la modélisation de l'évolution de l'environnement dans UPPAAL. Le premier automate, *TOP*, envoie à une fréquence arbitraire un signal reçu par un automate qui modélise l'évolution des variables. Cet automate affecte, à la réception de ce signal, à une variable quelconque une valeur aléatoirement choisie dans son domaine de définition. Ceci permet de modéliser l'environnement de manière aléatoire. La figure 10.5 montre l'automate qui envoie les signaux aléatoires et un automate modélisant l'évolution aléatoire de la variable booléenne *igni*. Ensuite, nous reprenons les modèles déjà représentés sur la figure 10.4, en y adjoignant simplement un signal *top ?* de réception de manière à forcer le passage d'une garde dans un automate lorsqu'elle est vérifiée (ce qui n'est pas le comportement par défaut d'UPPAAL).

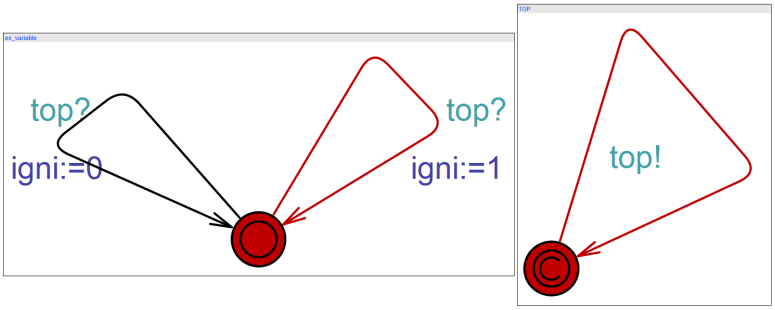


FIGURE 10.5 – Automates modélisant l'évolution de l'environnement

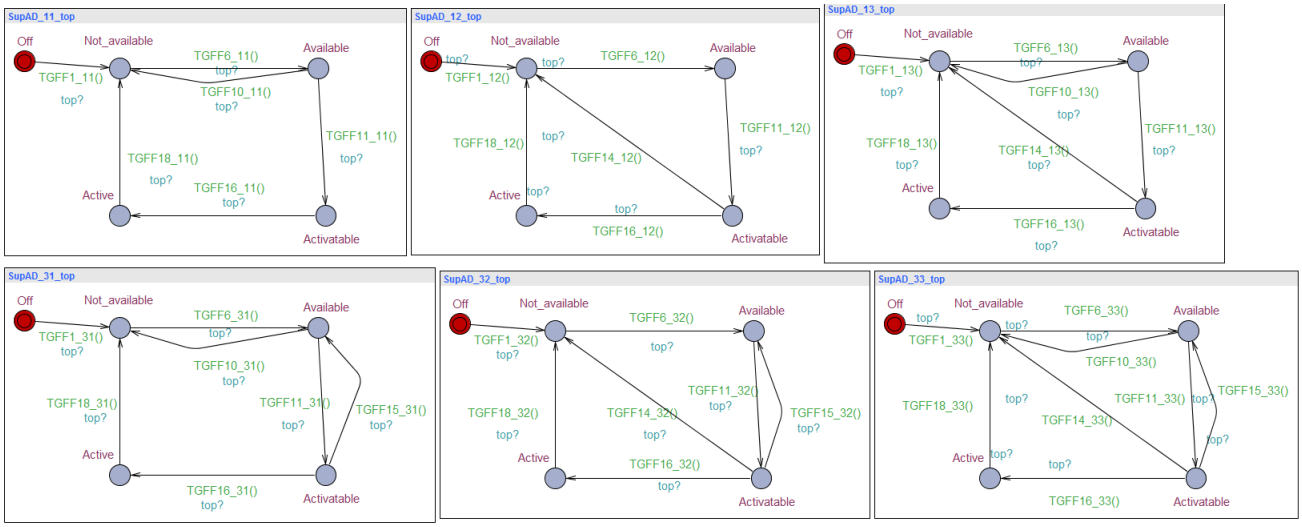


FIGURE 10.6 – Modèles d'état fonctionnels adpatés à la vérification

10.5.2 Version NuSMV

10.5.2.1 Première version de *SupAD_12*

– **SupAD_12**
MODULE main
– **Déclaration des variables**

```
VAR
igni : boolean;
road_sec : boolean;
veh_dir : boolean;
forec_AD_dur : boolean;
adj_la_typ : boolean;
cstr_area : boolean;
tfc_sign : boolean;
AD_activ : boolean;
```

AD_button : boolean ;
lat_dist : boolean ;
vel_reduc : boolean ;
dist_ped : boolean ;
tfc_info : boolean ;
AE_inter : boolean ;
mean_accel : boolean ;
adj_L_lane : boolean ;
val_adj_R_lane : boolean ;
adj_R_lane : boolean ;
val_adj_L_lane : boolean ;
R_gap_spd : boolean ;
safe_lane : boolean ;
foot_ped : boolean ;
hand_steer : boolean ;
pres_ac_ped : boolean ;
pres_ac_dur : boolean ;
foot_ac_ped : boolean ;
ac_ped_thru : boolean ;
thru_ac_dur : boolean ;
pres_br_dur : boolean ;
foot_br_ped : boolean ;
turn_indic : boolean ;
spd_veh : boolean ;
dist_obj : boolean ;
perc_cap : boolean ;
dif_pos : boolean ;
time_activ : boolean ;
veh_decel : boolean ;
L_gap_spd : boolean ;
TGFF1_12 : boolean ;
TGFF6_12 : boolean ;
TGFF14_12 : boolean ;
TGFF11_12 : boolean ;
TGFF16_12 : boolean ;
TGFF18_12 : boolean ;
veh_loc : {0, 1, 2} ;
lan_width : {0, 1, 2} ;
gear_lev : {0, 1, 2} ;
rte_def : {0, 1, 2} ;
veloc : {0, 1, 2, 3} ;
open_part : {0, 1, 2, 3} ;
forec_end : {0, 1, 2, 3} ;
pos_dur : {0, 1, 2, 3} ;
tqe_steer : {0, 1, 2, 3} ;
tqe_dur : {0, 1, 2, 3} ;

pres_br_ped : {0, 1, 2, 3} ;
veh_pos : {0, 1, 2, 3, 4, 5, 6} ;
weat_cond : {0, 1, 2, 3, 4, 5, 6, 7, 8} ;
lane_typ : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13} ;
state : {off, not_available, available, active, activatable} ;

– Gardes des groupes de transition formalisés

ASSIGN
TGFF1_12 := igni ;

TGFF6_12 := (!road_sec & ((veh_pos=0 & veh_dir) | veh_pos=6) & forec_AD_dur & (veh_loc=0 | veh_loc=1) & lan_width=1 & (weat_cond=0 | weat_cond=1 | weat_cond=2 | weat_cond=3 | weat_cond=4 | weat_cond=5 | weat_cond=6)) ;

TGFF14_12 := (adj_la_typ | lane_typ=0 | lane_typ=1 | (lane_typ=2 | lane_typ=3) | lane_typ=4 | lane_typ=5 | cstr_area | lane_typ=6 | !tfc_sign | (veloc=2 | veloc=3) | (veloc=1 | veloc=3) | weat_cond=7 | weat_cond=8 | (open_part=0 | open_part=2)) ;

TGFF11_12 := (!(rte_def=0) | !(turn_indic & spd_veh) | !(dist_obj | veh_pos=5) | !(!perc_cap)) ;

TGFF16_12 := (AD_activ & AD_button & gear_lev=1) ;

TGFF18_12 :=((lat_dist & vel_reduc) | (lane_typ=7 & forec_end=1) | (veh_pos=1 & !dist_ped) | (road_sec & forec_end=2) | (road_sec & forec_end=3) | lane_typ=6 | weat_cond=8 | (lane_typ=9 & tfc_info) | (lane_typ=9 & !tfc_info) | (AE_inter & !mean_accel) | (veh_pos=1 & veh_pos=3 & (pos_dur=1 | pos_dur=3)) | (veh_pos=1 & veh_pos=4 & (pos_dur=2 | pos_dur=3)) | rte_def=2 | (lane_typ=10 & lane_typ=11) | (lan_width=2) | (lan_width=0 & AE_inter) | (lan_width=0 & !tfc_info) | lane_typ=0 | weat_cond=3 | weat_cond=8 | (lane_typ=12 & dif_pos) | ((adj_L_lane & val_adj_L_lane) | (adj_R_lane & val_adj_R_lane)) | (R_gap_spd | L_gap_spd) | (lane_typ=13 & !safe_lane) | ((tqe_steer=1 | tqe_steer=3) & (tqe_dur=2 | tqe_dur=3) & foot_ped) | ((tqe_steer=1 | tqe_steer=3) & !foot_ped) | (hand_steer & (tqe_steer=1 | tqe_steer=3) & time_activ) | (hand_steer & (tqe_steer=2 | tqe_steer=3) & time_activ & veh_decel) | (hand_steer & (tqe_steer=2 | tqe_steer=0) & (tqe_dur=1 | tqe_dur=3)) | (pres_ac_ped & pres_ac_dur & hand_steer) | (pres_ac_ped & pres_ac_dur & hand_steer) | (foot_ac_ped & !ac_ped_thru & pres_br_ped=1) | ((pres_br_ped=1 | pres_br_ped=3) & thru_ac_dur & hand_steer) | (foot_br_ped & (pres_br_ped=2 | pres_br_ped=3)) | (veh_pos=2 & (open_part=0 | open_part=2)) | (!igni & veh_pos=2)) ;

init(state) :=off ;

– Déclaration de l'automate *SupAD_12*

```
next(state) :=
case
state=off & TGFF1_12 : not_available;
state=not_available & TGFF6_12 : available;
state=available & TGFF11_12 : activatable;
state=activatable & (TGFF14_12 & TGFF16_12) : {not_available, active};
state=activatable & TGFF14_12 : not_available;
state=activatable & TGFF16_12 : active;
state=active & TGFF18_12 : not_available;
TRUE : state;
esac;
```

– Contraintes d'équité

```
FAIRNESS TGFF1_12
FAIRNESS TGFF6_12
FAIRNESS TGFF11_12
FAIRNESS TGFF14_12
FAIRNESS TGFF16_12
FAIRNESS TGFF18_12
```

– Propriétés Formelles Fonctionnelles à vérifier

```
SPEC AG(((state=off) & TGFF1_12) -> AF (state=not_available));
SPEC AG(((state=not_available) & TGFF6_12)-> AF(state=available));
SPEC AG(((state = available) & TGFF11_12) -> AF (state = activatable));
SPEC AG((state = activatable & TGFF14_12) -> AF state = not_available);
SPEC AG((state = activatable & TGFF16_12) -> AF state = active);
SPEC AG((state = active & TGFF18_12) -> AF state = not_available);
```

10.5.2.2 Version corrigée de *SupAD_12*

Pour la version corrigée suite à la déterminisation de l'automate, seules les gardes de GTF14_12 et GTF16_12 changent, comme le montre l'extrait du modèle NuSMV suivant :

```
ASSIGN

TGFF1_12 := igni;
TGFF6_12 := (!road_sec & ((veh_pos=0 & veh_dir) | veh_pos=6) & fo-
rec_AD_dur & (veh_loc=0 | veh_loc=1) & lan_width=1 & (weat_cond=0 |
weat_cond=1 | weat_cond=2 | weat_cond=3 | weat_cond=4 | weat_cond=5 |
weat_cond=6));
```

TGFF14_12 := (adj_la_typ | lane_typ=0 | lane_typ=1 | (lane_typ=2 | lane_typ=3) | lane_typ=4 | lane_typ=5 | cstr_area | lane_typ=6 | !tfc_sign | (veloc=2 | veloc=3) | (veloc=1 | veloc=3) | weat_cond=7 | weat_cond=8 | (open_part=0 | open_part=2))& **!(AD_activ & AD_button & gear_lev=1)**;

TGFF11_12 := (!(rte_def=0) | !(turn_indic & spd_veh) | !(dist_obj | veh_pos=5) | !(!perc_cap));

TGFF16_12 := (AD_activ & AD_button & gear_lev=1) & **!(adj_la_typ | lane_typ=0 | lane_typ=1 | (lane_typ=2 | lane_typ=3) | lane_typ=4 | lane_typ=5 | cstr_area | lane_typ=6 | !tfc_sign | (veloc=2 | veloc=3) | (veloc=1 | veloc=3) | weat_cond=7 | weat_cond=8 | (open_part=0 | open_part=2))**;

TGFF18_12 := ((lat_dist & vel_reduc) | (lane_typ=7 & forec_end=1) | (veh_pos=1 & !dist_ped) | (road_sec & forec_end=2) | (road_sec & forec_end=3) | lane_typ=6 | weat_cond=8 | (lane_typ=9 & tfc_info) | (lane_typ=9 & !tfc_info) | (AE_inter & !mean_accel) | (veh_pos=1 & veh_pos=3 & (pos_dur=1 | pos_dur=3)) | (veh_pos=1 & veh_pos=4 & (pos_dur=2 | pos_dur=3)) | rte_def=2 | (lane_typ=10 & lane_typ=11) | (lan_width=2) | (lan_width=0 & AE_inter) | (lan_width=0 & !tfc_info) | lane_typ=0 | weat_cond=3 | weat_cond=8 | (lane_typ=12 & dif_pos) | ((adj_L_lane & val_adj_L_lane) | (adj_R_lane & val_adj_R_lane)) | (R_gap_spd | L_gap_spd) | (lane_typ=13 & !safe_lane) | ((tqe_steer=1 | tqe_steer=3) & (tqe_dur=2 | tqe_dur=3) & foot_ped) | ((tqe_steer=1 | tqe_steer=3) & !foot_ped) | (hand_steer & (tqe_steer=1 | tqe_steer=3) & time_activ) | (hand_steer & (tqe_steer=2 | tqe_steer=3) & time_activ & veh_decel) | (hand_steer & (tqe_steer=2 | tqe_steer=0) & (tqe_dur=1 | tqe_dur=3)) | (pres_ac_ped & pres_ac_dur & hand_steer) | (pres_ac_ped & pres_ac_dur & hand_steer) | (foot_ac_ped & !ac_ped_thru & pres_br_ped=1) | ((pres_br_ped=1 | pres_br_ped=3) & thru_ac_dur & hand_steer) | (foot_br_ped & (pres_br_ped=2 | pres_br_ped=3)) | (veh_pos=2 & (open_part=0 | open_part=2)) | (!igni & veh_pos=2));

init(state) :=off;

10.6 Modèle d'État de Sûreté vérfié

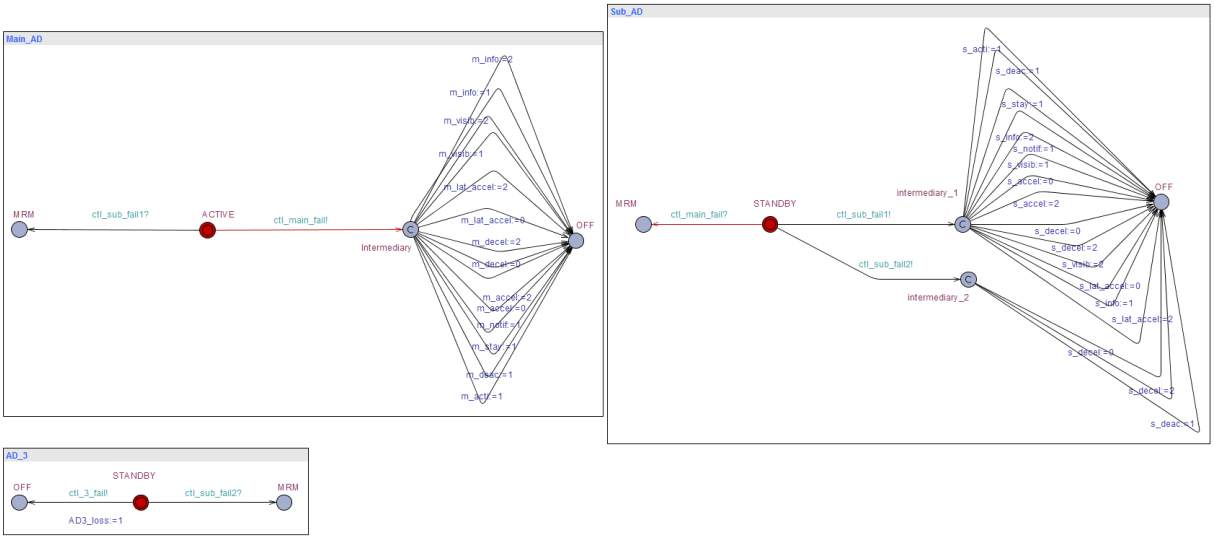


FIGURE 10.7 – Modèle d'état de sûreté vérifié

10.7 Modèles Globaux cibles

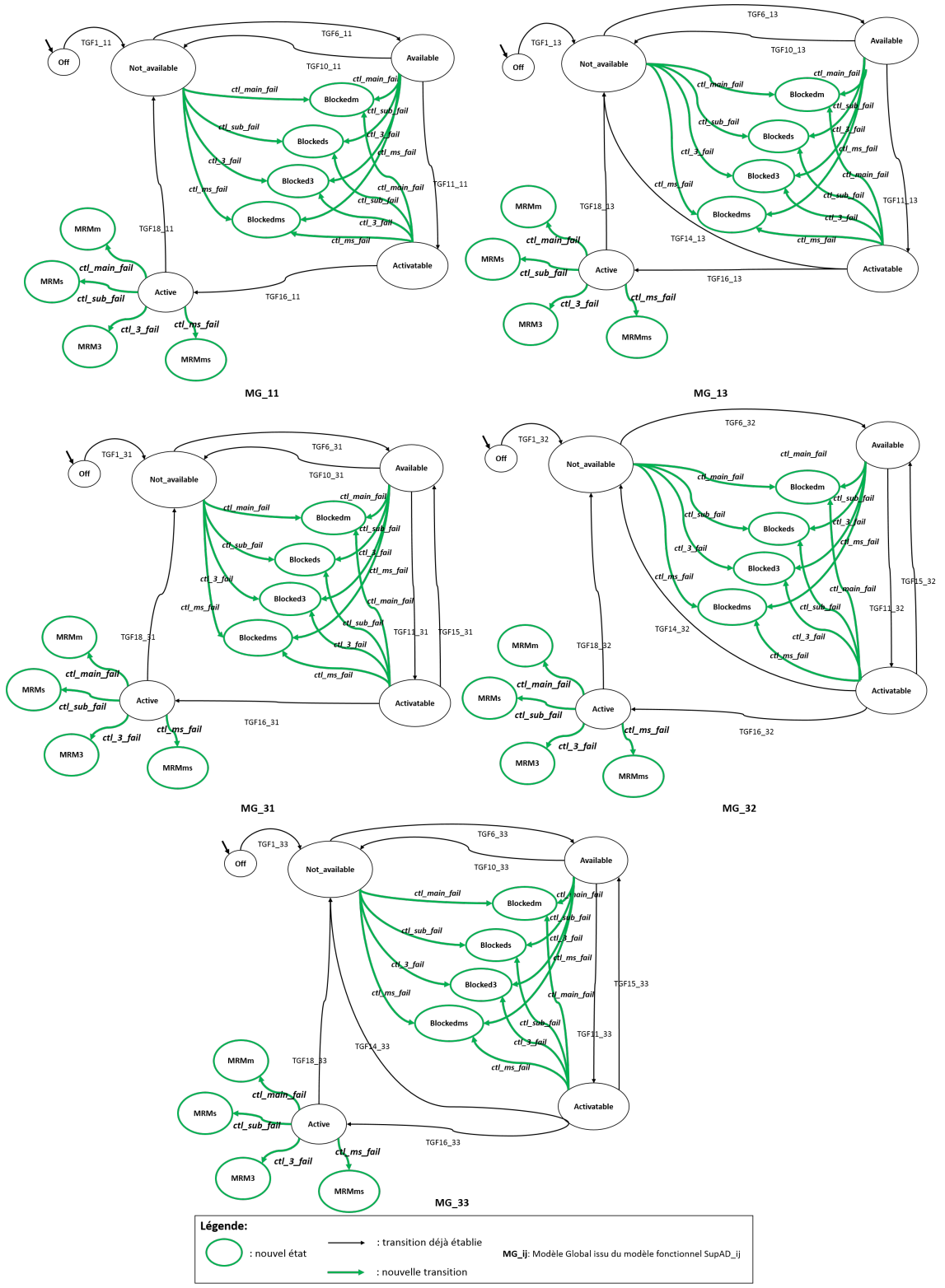


FIGURE 10.8 – Modèles Globaux cibles

10.8 Modèle d'État Complet version 12

– **Modèle d'Etat Complet version 12**

MODULE main
– **Déclaration des variables**

VAR

igni : boolean ;
road_sec : boolean ;
veh_dir : boolean ;
forec_AD_dur : boolean ;
adj_la_typ : boolean ;
cstr_area : boolean ;
tfc_sign : boolean ;
AD_activ : boolean ;
AD_button : boolean ;
lat_dist : boolean ;
vel_reduc : boolean ;
dist_ped : boolean ;
tfc_info : boolean ;
AE_inter : boolean ;
mean_accel : boolean ;
adj_L_lane : boolean ;
val_adj_R_lane : boolean ;
adj_R_lane : boolean ;
val_adj_L_lane : boolean ;
R_gap_spd : boolean ;
safe_lane : boolean ;
foot_ped : boolean ;
hand_steer : boolean ;
pres_ac_ped : boolean ;
pres_ac_dur : boolean ;
foot_ac_ped : boolean ;
ac_ped_thru : boolean ;
thru_ac_dur : boolean ;
pres_br_dur : boolean ;
foot_br_ped : boolean ;
turn_indic : boolean ;
spd_veh : boolean ;
dist_obj : boolean ;
perc_cap : boolean ;
dif_pos : boolean ;
time_activ : boolean ;

veh_decel : boolean ;
L_gap_spd : boolean ;
TGFF1_12 : boolean ;
TGFF6_12 : boolean ;
TGFF14_12 : boolean ;
TGFF11_12 : boolean ;
TGFF16_12 : boolean ;
TGFF18_12 : boolean ;
ctl_main_fail : boolean ;
ctl_sub_fail : boolean ;
ctl_3_fail : boolean ;
ctl_ms_fail : boolean ;
veh_loc : {0, 1, 2} ;
lan_width : {0, 1, 2} ;
gear_lev : {0, 1, 2} ;
rte_def : {0, 1, 2} ;
veloc : {0, 1, 2, 3} ;
open_part : {0, 1, 2, 3} ;
forec_end : {0, 1, 2, 3} ;
pos_dur : {0, 1, 2, 3} ;
tqe_steer : {0, 1, 2, 3} ;
tqe_dur : {0, 1, 2, 3} ;
pres_br_ped : {0, 1, 2, 3} ;
veh_pos : {0, 1, 2, 3, 4, 5, 6} ;
weat_cond : {0, 1, 2, 3, 4, 5, 6, 7, 8} ;
lane_typ : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13} ;

– *Etats de la Supervision AD globale*

state : {off, not_available, available, active, activatable, MRMm, MRMs, MRM3, MRMms, blockedm, blockeds, blocked3, blockedms} ;

– *Etats de la fonction Main_AD*

state_m : {off_m, not_available_m, available_m, activatable_m, active_m, silent_m, blocked_m, MRM_m} ;

– *Etats de la fonction Sub_AD*

state_s : {off_s, not_available_s, available_s, activatable_s, active_s, silent_s, blocked_s, MRM_s} ;

– *Etats de la fonction AD-3*

state_3 : {off_3, not_available_3, available_3, silent_3, blocked_3, MRM_3} ;

– Gardes des groupes de transition formalisés

ASSIGN

TGFF1_12 := igni ;

TGFF6_12 := (!road_sec & ((veh_pos=0 & veh_dir) | veh_pos=6) & forec_AD_dur & (veh_loc=0 | veh_loc=1) & lan_width=1 & (weat_cond=0 | weat_cond=1 | weat_cond=2 | weat_cond=3 | weat_cond=4 | weat_cond=5 | weat_cond=6)) ;

TGFF14_12 := (adj_la_typ | lane_typ=0 | lane_typ=1 | (lane_typ=2 | lane_typ=3) | lane_typ=4 | lane_typ=5 | cstr_area | lane_typ=6 | !tfc_sign | (veloc=2 | veloc=3) | (veloc=1 | veloc=3) | weat_cond=7 | weat_cond=8 | (open_part=0 | open_part=2)) & !(AD_activ & AD_button & gear_lev=1) ;

TGFF11_12 := (!(rte_def=0) |!(turn_indic & spd_veh) |!(dist_obj | veh_pos=5) |!(!perc_cap)) ;

TGFF16_12 := (AD_activ & AD_button & gear_lev=1) & !(adj_la_typ | lane_typ=0 | lane_typ=1 | (lane_typ=2 | lane_typ=3) | lane_typ=4 | lane_typ=5 | cstr_area | lane_typ=6 |!tfc_sign | (veloc=2 | veloc=3) | (veloc=1 | veloc=3) | weat_cond=7 | weat_cond=8 | (open_part=0 | open_part=2)) ;

TGFF18_12 := ((lat_dist & vel_reduc) | (lane_typ=7 & forec_end=1) | (veh_pos=1 & !dist_ped) | (road_sec & forec_end=2) | (road_sec & forec_end=3) | lane_typ=6 | weat_cond=8 | (lane_typ=9 & tfc_info) | (lane_typ=9 & !tfc_info) | (AE_inter & !mean_accel) | (veh_pos=1 & veh_pos=3 & (pos_dur=1 | pos_dur=3)) | (veh_pos=1 & veh_pos=4 & (pos_dur=2 | pos_dur=3)) | rte_def=2 | (lane_typ=10 & lane_typ=11) | (lan_width=2) | (lan_width=0 & AE_inter) | (lan_width=0 & !tfc_info) | lane_typ=0 | weat_cond=3 | weat_cond=8 | (lane_typ=12 & dif_pos) | ((adj_L_lane & val_adj_L_lane) | (adj_R_lane & val_adj_R_lane)) | (R_gap_spd | L_gap_spd) | (lane_typ=13 & !safe_lane) | ((tqe_steer=1 | tqe_steer=3) & (tqe_dur=2 | tqe_dur=3) & foot_ped) | ((tqe_steer=1 | tqe_steer=3) & !foot_ped) | (hand_steer & (tqe_steer=1 | tqe_steer=3) & time_activ) | (hand_steer & (tqe_steer=2 | tqe_steer=3) & time_activ & veh_decel) | (hand_steer & (tqe_steer=2 | tqe_steer=0) & (tqe_dur=1 | tqe_dur=3)) | (pres_ac_ped & pres_ac_dur & hand_steer) | (pres_ac_ped & pres_ac_dur & hand_steer) | (foot_ac_ped & !ac_ped_thru & pres_br_ped=1) | ((pres_br_ped=1 | pres_br_ped=3) & thru_ac_dur & hand_steer) | (foot_br_ped & (pres_br_ped=2 | pres_br_ped=3)) | (veh_pos=2 & (open_part=0 | open_part=2)) | (!igni & veh_pos=2)) ;

– Initialisation des variables

– Initialisation des variables de sûreté : pas de défaillance à l'état initial

init(ctl_main_fail) :=FALSE ; init(ctl_sub_fail) :=FALSE ;
init(ctl_3_fail) :=FALSE ; init(ctl_ms_fail) :=FALSE ;

– *Initialisation des variables des états*

init(state) :=off ;
init(state_m) := off_m ;
init(state_s) := off_s ;
init(state_3) := off_3 ;

– **Défaillances non simultanées**
next(ctl_main_fail) := case
(ctl_sub_fail | ctl_3_fail | ctl_ms_fail) : !ctl_main_fail ;
TRUE : ctl_main_fail ;
esac ;

next(ctl_sub_fail) := case
(ctl_main_fail | ctl_3_fail | ctl_ms_fail) : !ctl_sub_fail ;
TRUE : ctl_sub_fail ;
esac ;

next(ctl_3_fail) := case
(ctl_sub_fail | ctl_main_fail | ctl_ms_fail) : !ctl_3_fail ;
TRUE : ctl_3_fail ;
esac ;

next(ctl_ms_fail) := case
(ctl_sub_fail | ctl_3_fail | ctl_main_fail) : !ctl_ms_fail ;
TRUE : ctl_ms_fail ;
esac ;

– **Modèle Global version 12**
– *Déclaration de l'automate de la Supervision AD globale*

next(state) :=
case
state=off & TGFF1_12 : not_available ;
state=not_available & TGFF6_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail | ctl_ms_fail) : available ;
state=not_available & ctl_main_fail : blockedm ;
state=not_available & ctl_sub_fail : blockeds ;
state=not_available & ctl_3_fail : blocked3 ;
state=not_available & ctl_ms_fail : blockedms ;

```
state=available & TGFF11_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : activatable;
state=available & ctl_main_fail : blockedm;
state=available & ctl_sub_fail : blockeds;
state=available & ctl_3_fail : blocked3;
state=available & ctl_ms_fail : blockedms;
state=activatable & ((TGFF14_12 & TGFF16_12) &!(ctl_main_fail | ctl_sub_fail
| ctl_3_fail | ctl_ms_fail)) : {not_available, active};
state=activatable & TGFF14_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : not_available;
state=activatable & TGFF16_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : active;
state=activatable & ctl_main_fail : blockedm;
state=activatable & ctl_sub_fail : blockeds;
state=activatable & ctl_3_fail : blocked3;
state=activatable & ctl_ms_fail : blockedms;
state=active & TGFF18_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : not_available;
state=active & ctl_main_fail : MRMm;
state=active & ctl_sub_fail : MRMs;
state=active & ctl_3_fail : MRM3;
state=active & ctl_ms_fail : MRMms;
TRUE : state;
esac;
```

– **Modèle Local version 12**

– *Déclaration de l'automate de la fonction Main_ AD*

```
next(state_m) :=
case
state_m=off_m & TGFF1_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : not_available_m;
state_m=not_available_m & TGFF6_12 &!(ctl_main_fail | ctl_sub_fail |
ctl_3_fail | ctl_ms_fail) : available_m;
state_m=not_available_m & (ctl_main_fail | ctl_ms_fail) : silent_m;
state_m=not_available_m & (ctl_3_fail | ctl_sub_fail) : blocked_m;
state_m=available_m & TGFF11_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : activatable_m;
state_m=available_m & (ctl_main_fail | ctl_ms_fail) : silent_m;
state_m=available_m & (ctl_3_fail | ctl_sub_fail) : blocked_m;
state_m=activatable_m & ((TGFF14_12 & TGFF16_12) &!(ctl_main_fail |
ctl_sub_fail | ctl_3_fail | ctl_ms_fail)) : {not_available_m, active_m};
state_m=activatable_m & TGFF14_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail
| ctl_ms_fail) : not_available_m;
```

```
state_m=activatable_m & TGFF16_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail
| ctl_ms_fail) : active_m;
state_m=activatable_m & (ctl_main_fail | ctl_ms_fail) : silent_m;
state_m=activatable_m & (ctl_3_fail | ctl_sub_fail) : blocked_m;
state_m=active_m & TGFF18_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : not_available_m;
state_m=active_m & (ctl_main_fail | ctl_ms_fail) : silent_m;
state_m=active_m & (ctl_3_fail | ctl_sub_fail) : MRM_m;
TRUE : state_m;
esac;
```

– *Déclaration de l'automate de la fonction Sub_AD*

```
next(state_s) :=
case
state_s=off_s & TGFF1_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : not_available_s;
state_s=not_available_s & TGFF6_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail
| ctl_ms_fail) : available_s;
state_s=not_available_s & (ctl_sub_fail | ctl_ms_fail) : silent_s;
state_s=not_available_s & (ctl_3_fail | ctl_main_fail) : blocked_s;
state_s=available_s & TGFF11_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : activatable_s;
state_s=available_s & (ctl_sub_fail | ctl_ms_fail) : silent_s;
state_s=available_s & (ctl_3_fail | ctl_main_fail) : blocked_s;
state_s=activatable_s & ((TGFF14_12 & TGFF16_12) & !(ctl_main_fail |
ctl_sub_fail | ctl_3_fail | ctl_ms_fail)) : {not_available_s, active_s};
state_s=activatable_s & TGFF14_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : not_available_s;
state_s=activatable_s & TGFF16_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : active_s;
state_s=activatable_s & (ctl_sub_fail | ctl_ms_fail) : silent_s;
state_s=activatable_s & (ctl_3_fail | ctl_main_fail) : blocked_s;
state_s=active_s & TGFF18_12 & !(ctl_main_fail | ctl_sub_fail | ctl_3_fail |
ctl_ms_fail) : not_available_s;
state_s=active_s & (ctl_sub_fail | ctl_ms_fail) : silent_s;
state_s=active_s & ctl_3_fail : blocked_s;
state_s=active_s & ctl_main_fail : MRM_s;
TRUE : state_s;
esac;
```

– *Déclaration de l'automate de la fonction AD-3*

```
next(state_3) :=
case
```

state_3=off_3 & TGFF1_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail | ctl_ms_fail) : not_available_3 ;
state_3=not_available_3 & TGFF16_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail | ctl_ms_fail) : available_3 ;
state_3=not_available_3 & (ctl_sub_fail | ctl_main_fail | ctl_ms_fail) : blocked_3 ;
state_3=not_available_3 & ctl_3_fail : silent_3 ;
state_3=available_3 & TGFF18_12 &!(ctl_main_fail | ctl_sub_fail | ctl_3_fail | ctl_ms_fail) : not_available_3 ;
state_3=available_3 & (ctl_sub_fail | ctl_main_fail | ctl_ms_fail) : blocked_3 ;
state_3=available_3 & ctl_3_fail : silent_3 ;
state_3=available_3 & ctl_ms_fail : MRM_3 ;
TRUE : state_3 ;
esac ;

– **Contraintes d'équité**

FAIRNESS TGFF1_12
FAIRNESS TGFF6_12
FAIRNESS TGFF11_12
FAIRNESS TGFF14_12
FAIRNESS TGFF16_12
FAIRNESS TGFF18_12

– **Exigences portant sur la Supervision AD globale**
– *Exigences fonctionnelles*

SPEC AG((state=off & TGFF1_12) -> AF (state=not_available)) ;
SPEC AG((state=not_available & TGFF6_12)-> AF(state=available)) ;
SPEC AG(((state = available) & TGFF11_12) -> AF (state = activatable)) ;
SPEC AG((state = activatable & TGFF14_12) -> AF (state = not_available)) ;
SPEC AG((state = activatable & TGFF16_12) -> AF (state = active)) ;
SPEC AG((state = active & TGFF18_12) -> AF (state = not_available)) ;

– *Exigences de sûreté*

SPEC AG((state = not_available & ctl_main_fail) -> AF(state = blockedm)) ;
SPEC AG((state = not_available & ctl_sub_fail) -> AF(state = blockeds)) ;
SPEC AG((state = not_available & ctl_3_fail) -> AF(state = blocked3)) ;
SPEC AG((state = not_available & ctl_ms_fail) -> AF(state = blockedms)) ;
SPEC AG((state = available & ctl_main_fail) -> AF(state = blockedm)) ;
SPEC AG((state = available & ctl_sub_fail) -> AF(state = blockeds)) ;
SPEC AG((state = available & ctl_3_fail) -> AF(state = blocked3)) ;
SPEC AG((state = available & ctl_ms_fail) -> AF(state = blockedms)) ;
SPEC AG((state = activatable & ctl_main_fail) -> AF(state = blockedm)) ;
SPEC AG((state = activatable & ctl_sub_fail) -> AF(state = blockeds)) ;

```
SPEC AG((state = activatable & ctl_3_fail) -> AF(state = blocked3));
SPEC AG((state = activatable & ctl_ms_fail) -> AF(state = blockedms));
SPEC AG((state = active & ctl_main_fail) -> AF(state = MRMm));
SPEC AG((state = active & ctl_sub_fail) -> AF(state = MRMs));
SPEC AG((state = active & ctl_3_fail) -> AF(state = MRM3));
SPEC AG((state = active & ctl_ms_fail) -> AF(state = MRMms));
```

– Exigences portant sur la fonction Main_AD

– Exigences fonctionnelles

```
SPEC AG((state_m=off_m & TGFF1_12) -> AF(state_m=not_available_m));
SPEC AG((state_m=not_available_m & TGFF6_12)-> AF(state_m=available_m));
SPEC AG((state_m = available_m & TGFF11_12) -> AF(state_m = activatable_m));
SPEC AG((state_m = activatable_m & TGFF14_12) -> AF(state_m = not_available_m));
SPEC AG((state_m = activatable_m & TGFF16_12) -> AF(state_m = active_m));
SPEC AG((state_m = active_m & TGFF18_12) -> AF(state_m = not_available_m));
```

– Exigences de sûreté

```
SPEC AG((state_m = not_available_m & (ctl_main_fail | ctl_ms_fail)) -> AF(state_m = silent_m));
SPEC AG((state_m = available_m & (ctl_main_fail | ctl_ms_fail)) -> AF(state_m = silent_m));
SPEC AG((state_m = activatable_m & (ctl_main_fail | ctl_ms_fail)) -> AF(state_m = silent_m));
SPEC AG((state_m = active_m & (ctl_main_fail | ctl_ms_fail)) -> AF(state_m = silent_m));
SPEC AG((state_m = not_available_m & (ctl_3_fail | ctl_sub_fail)) -> AF(state_m = blocked_m));
SPEC AG((state_m = available_m & (ctl_3_fail | ctl_sub_fail)) -> AF (state_m = blocked_m));
SPEC AG((state_m = activatable_m & (ctl_3_fail | ctl_sub_fail)) -> AF(state_m = blocked_m));
SPEC AG((state_m = active_m & (ctl_3_fail | ctl_sub_fail)) -> AF(state_m = MRM_m));
```

– Exigences portant sur la fonction Sub_AD

– Exigences fonctionnelles

```
SPEC AG((state_s=off_s & TGFF1_12) -> AF(state_s=not_available_s));
SPEC AG((state_s=not_available_s & TGFF6_12)-> AF(state_s=available_s));
SPEC AG((state_s = available_s & TGFF11_12) -> AF(state_s = activatable_s));
```

```
SPEC AG((state_s = activatable_s & TGFF14_12) -> AF(state_s = not_available_s));
SPEC AG((state_s = activatable_s & TGFF16_12)-> AF(state_s = active_s));
SPEC AG((state_s = active_s & TGFF18_12)-> AF(state_s = not_available_s));
```

– *Exigences de sûreté*

```
SPEC AG((state_s = not_available_s & (ctl_sub_fail | ctl_ms_fail)) -> AF(state_s = silent_s));
SPEC AG((state_s = available_s & (ctl_sub_fail | ctl_ms_fail)) -> AF(state_s = silent_s));
SPEC AG((state_s = activatable_s & (ctl_sub_fail | ctl_ms_fail)) -> AF(state_s = silent_s));
SPEC AG((state_s = active_s & (ctl_sub_fail | ctl_ms_fail)) -> AF(state_s = silent_s));
SPEC AG((state_s = not_available_s & (ctl_3_fail | ctl_main_fail)) -> AF(state_s = blocked_s));
SPEC AG((state_s = available_s & (ctl_3_fail | ctl_main_fail)) -> AF(state_s = blocked_s));
SPEC AG((state_s = activatable_s & (ctl_3_fail | ctl_main_fail)) -> AF(state_s = blocked_s));
SPEC AG((state_s = active_s & ctl_3_fail) -> AF(state_s = blocked_s));
SPEC AG((state_s = active_s & ctl_main_fail) -> AF(state_s = MRM_s));
```

– **Exigences portant sur la fonction AD-3**

– *Exigences fonctionnelles*

```
SPEC AG((state_3 = off_3 & TGFF1_12) -> AF(state_3 = not_available_3));
SPEC AG((state_3 = not_available_3 & TGFF16_12) -> AF(state_3 = available_3));
SPEC AG((state_3 = available_3 & TGFF18_12) -> AF(state_3 = not_available_3));
```

– *Exigences de sûreté*

```
SPEC AG((state_3 = not_available_3 & (ctl_sub_fail | ctl_main_fail | ctl_ms_fail)) -> AF(state_3 = blocked_3));
SPEC AG((state_3 = available_3 & (ctl_sub_fail | ctl_main_fail)) -> AF(state_3 = blocked_3));
SPEC AG((state_3 = not_available_3 & ctl_3_fail) -> AF(state_3 = silent_3));
SPEC AG((state_3 = available_3 & ctl_3_fail) -> AF(state_3 = silent_3));
SPEC AG((state_3 = available_3 & ctl_ms_fail) -> AF(state_3 = MRM_3));
```

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : Cuer

DATE de SOUTENANCE : 23/11/2018

Prénoms : Romain

TITRE : Démarche de conception sûre de la Supervision de la fonction de Conduite Autonome

NATURE : Doctorat

Numéro d'ordre : 2018LYSEIXXXX

Ecole doctorale : Electronique Electrotechnique Automatique (EEA)

Spécialité : Automatique

RESUME :
Le véhicule autonome est un véhicule qui se conduira, à terme, sans aucune intervention du conducteur, quelle que soit la situation de conduite. Ce véhicule comprend une nouvelle fonction, nommée fonction Autonomous Driving (AD), chargée de la conduite autonome. Cette fonction peut se trouver dans des états différents, gérés par la Supervision AD. Le principal objet de ces travaux consiste à garantir que la fonction AD se trouve constamment dans un état sûr. Ceci revient à s'assurer que la Supervision AD respecte l'ensemble des exigences fonctionnelles et de sûreté qui spécifient son comportement. Ces deux types d'exigences sont émis par deux métiers: l'Architecte Métier Système (AMS) et le pilote Sûreté de Fonctionnement (SdF), qui se distinguent en de nombreux points (objectifs, contraintes, planning...). La mise en cohérence de ces deux perspectives métier, au plus tôt dans le cycle de conception et dans un contexte industriel, est la problématique centrale traitée.

Afin de répondre à ces préoccupations, nous avons proposé une démarche outillée et collaborative de conception sûre de la Supervision AD. Cette démarche est intégrée dans les processus normatifs en vigueur (normes ISO 15288 et ISO 26262) ainsi que dans les processus de conception internes chez Renault. Elle est fondée sur la vérification formelle par *model checking*, la composition parallèle d'automates finis et l'expertise métier. Cette démarche prône l'utilisation d'un même formalisme par les deux métiers pour mener à bien des activités partageant un objectif de modélisation commun : la vérification d'exigences de comportement en phase amont de conception. Les contributions principales de ces travaux sont : la mise en évidence de plusieurs spécifications formelles crédibles (i.e. validées par les experts) à partir d'exigences informelles ; et la définition précise du rôle des deux expertises métier dans le processus de formalisation des exigences et de construction des automates.

MOTS-CLÉS : Sûreté de Fonctionnement, Véhicule Autonome, Ingénierie Système, Vérification formelle, Formalisation

Laboratoire (s) de recherche : Ampère

Directeur de thèse: Éric Niel
Co-directeur de thèse : Laurent Piétrac

Président de jury :

Composition du jury : Frédéric Kratz, Éric Levrat, Agnes Lanusse, Mohamed Ghazel, Éric Niel, Laurent Piétrac
Invités : Christophe Dang-Van-Nhan, Emil Dumitrescu