Conception et génération de tests par altération de données pour les systèmes de contrôle et de surveillance des transports : application aux domaines aérien et maritime

Aymeric Cretin

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

UNIVERSITÉ DE FRANCHE-COMTÉ

INSTITUT DE RECHERCHE
femto-st
SCIENCES & TECHNOLOGIES

THÈSE DE DOCTORAT

DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE FRANCHE-COMTÉ

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

par

AYMERIC CRETIN

Conception et génération de tests par altération de données pour les systèmes de contrôle et de surveillance des transports.

Application aux domaines aérien et maritime.

Thèse présentée et soutenue à Besançon, le 15 mars 2021

Composition du Jury :

MARIE-LAURE POTET	Professeur à l'Université Grenoble Alpes	Rapporteur
ALAIN BOUJU	Professeur à l'Université de La Rochelle	Rapporteur
ARNAUD GOTLIEB	Professeur à Simula Research Laboratory	Examinateur
YVES LE TRAON	Maître de Conférence à l'Université du Luxembourg	Examinateur
BRUNO LEGEARD	Professeur à l'Université de Franche-Comté	Directeur de thèse
FABIEN PEUREUX	Maître de Conférence à l'Université de Franche-Comté	Co-encadrant de thèse
ALEXANDRE VERNOTTE	Postdoc à l'Université de Franche-Comté	Co-encadrant de thèse

N° | X | X | X |

REMERCIEMENTS

Dans un premier temps je tiens à remercier mes encadrants de thèse, tout d'abord Bruno Legeard pour m'avoir permis de travailler sur ce sujet, ainsi que pour son encadrement et la confiance qu'il m'a accordé durant ces trois années. Ensuite Fabien Peureux pour ses nombreux conseils et le temps passé à faire ses relectures avisées, notamment concernant ce manuscrit de thèse. Enfin, je remercie Alexandre Vernotte qui a activement participer à ces travaux, tant par sa participation aux réflexions que par son soutient technique important sur de nombreux aspects.

Je remercie sincèrement Marie-Laure Potet et Alain Bouju pour avoir accepté le rôle de rapporteurs pour mon manuscrit de thèse et l'attention portée à mes travaux.

Je remercie également Yves Le Traon pour m'avoir fait l'honneur de présider mon jury de thèse, ainsi que Arnaud Gotlieb pour avoir accepté le rôle d'examinateur.

Mes remerciements vont également à mes trois collègues doctorants, Antoine Chevrot, Ralph Karam et Pierre Bernabé qui ont acceptés volontier de collaborer avec moi dans le cadre d'expérimentations de mes travaux. Je tiens également à remercier Christophe Sauvage de Smartesting qui a été un grand soutien technique au tout début de ma thèse.

SOMMAIRE

I Contextes et motivations 1

1 Problématique et positionnement de la thèse 3

1.1 Enjeux de la surveillance aérienne et maritime 3

1.2 Cas d'application des travaux 11

1.3 Objectif de la thèse et questions de recherche 16

1.4 Contextes et publications de la thèse 17

1.5 Plan de la thèse 20

2 Zoom sur les surveillances aérienne et maritime 21

2.1 La surveillance de l'espace aérien 21

2.2 La surveillance de l'espace maritime 28

2.3 Les vulnérabilités des protocoles 32

3 État de l'art 39

3.1 Attaque par injection de fausses données (FDIA) 39

3.2 La génération de données de test synthétiques 45

3.3 Utilisation de langages dédiés pour la spécification des tests 50

II Contributions 55

4 Démarche générale pour la génération de données 57

4.1 Architecture multi-domaine 57

4.2 Spécialisation aux domaines d'application 64

4.3 Synthèse 71

5 Génération de données altérées 73

5.1 Le processus d'altération 73
5.2 Les moteurs d'altération 74
5.3 Validation expérimentale 86

6 Langages dédiés à la scénarisation 95
6.1 La décision : pourquoi développer un langage dédié ? 95
6.2 L'analyse du domaine 97
6.3 La conception du langage dédié 112
6.4 Implémentation du DSL 122
6.5 Validation des langages dédiés 129
6.6 Synthèse 137

III Implémentation et contextes d'usage 139

7 FDI-T : Chaîne outillée 141
7.1 FDI-T : *False Data Injection Testing* 142
7.2 L'intégration des trois langages dédiés 145
7.3 L'intégration de la génération de données altérées 148
7.4 Synthèse 150

8 Expérimentations dans les domaines aérien et maritime 153
8.1 Technique d'apprentissage supervisé dans le domaine aérien 154
8.2 Technique d'apprentissage non-supervisé dans le domaine aérien 159
8.3 Technique d'apprentissage supervisé dans le domaine maritime 164
8.4 Synthèse 168

Conclusion et perspectives 169

A Annexes 185
A.1 Arbre d'héritage de l'ontologie 185

TABLE DES FIGURES

1.1 Couverture de l'espace aérien Australien par les technologies radar et ADS-B [29] 5

1.2 Deux navires éloignés utilisent la même identité au même moment. 7

1.3 Localisation aux États-Unis des stations au sol chargées de transmettre l'ADS-B aux centres de contrôle en 2014. 10

2.1 Division de la surveillance aérienne. 22

2.2 Aperçu de la chaîne de surveillance d'EUROCONTROL 27

2.3 Fonctionnement de la technologie AIS. 30

3.1 Résultats annuels de Google Scholar pour le terme *False Data Injection Attacks*. 41

4.1 Exemple d'un réseau multi-niveaux. 59

4.2 Génération de données : processus générique 60

4.3 Exemple de conversion : découpage d'un enregistrement en 3 sous-enregistrements. 61

5.1 Vue d'ensemble du processus d'altération 75

5.2 Processus partagé par les moteurs d'altération 76

5.3 Score d'anomalie sur l'anomalie **DRIFT** 91

5.4 Exemple de l'anomalie **DT** appliquée à un vol Moscou-Madrid 92

5.5 Score d'anomalie sur l'anomalie **DT** 93

6.1 EBNF de la Base Expression Grammar (BEG) 114

6.2 EBNF de la Property Evaluation Grammar (PEG) 116

6.3 La grammaire PEG est étendue avec des opérateurs temporels. 117

6.4 La grammaire PEG étendue pour les déclencheurs 118

6.5 Résultat de l'évaluation d'un déclencheur sur quatre avions. 119
6.6 Conjonction et disjonction de déclencheurs. 120
6.7 Grammaire du langage de schéma d'altération 121
6.8 Du schéma abstrait aux schémas concrets 127
6.9 Obtentions des directives d'altération à partir d'un schéma concret 128
6.10 Schéma d'altération de *Aircraft Flooding* 130
6.11 Schéma d'altération de *False Alarm* 131
6.12 Schéma d'altération de *Aircraft Disappearance* 131
6.13 Schéma d'altération de *Ghost Aircraft Injection* 132
6.14 Schéma d'altération de *Trajectory Modification* 132
6.15 Premier schéma : *False Alarm Attack* 134
6.16 Deuxième schéma : *Aircraft Disappearance Attack* 135
6.17 Extrait d'un fichier XML de cas de test. 136

7.1 Éditeur graphique de scénario de FDI-T pour le domaine aérien. 143
7.2 Éditeur d'exécution de FDI-T. 143
7.3 Éditeur graphique de scénario de FDI-T pour le domaine maritime. 144
7.4 Éditeur de schéma de FDI-T. 146
7.5 Création d'un scénario textuel par la combinaison de deux schémas d'altération. 146
7.6 Éditeur de filtre de FDI-T. 147
7.7 Éditeur de déclencheurs de FDI-T. 148
7.8 Exemple de fichier de configuration pour le mode console. 150

8.1 Architecture d'apprentissage supervisé pour la détection d'anomalies. . . . 156
8.2 Décalage de latitude appliqué à un vol entre Londres et Varsovie. 157
8.3 Scénario du décalage graduel de l'altitude. 157
8.4 Scénario du décalage graduel de la vitesse au sol. 157
8.5 Scénario du décalage graduel de la vitesse au sol. 158
8.6 Scénarios des décalages graduels de la latitude et de la longitude. 158
8.7 Huit vols complets aux États-Unis. 158

8.8 Architecture d'apprentissage non-supervisé pour la détection d'anomalies. 160

8.9 Scénario abstrait du décalage positif de latitude pour le vol Londres-Varsovie. 161

8.10 Exemple d'un scénario concret du décalage positif de latitude pour le vol Londres-Varsovie. 162

8.11 Seize exemples de modifications de trajectoires sur le vol Londres-Varsovie. 162

8.12 Architecture de détection non-supervisée dans les données ADS-B. 163

8.13 Représentation du trafic maritime et son utilisation par différentes tâches. . 165

8.14 Exemple de scénario abstrait de modification de trajectoire avec décalage de latitude positif. 166

8.15 Exemple de scénario abstrait de modification de trajectoire avec décalage de latitude positif. 167

A.1 Arbre d'héritage de l'ontologie 1/3. 186

A.2 Arbre d'héritage de l'ontologie 2/3. 187

A.3 Arbre d'héritage de l'ontologie 3/3. 188

LISTE DES TABLES

2.1 Taux de transmission selon la classe du transpondeur AIS. 31

3.1 Liste des différents schémas dans les domaines aérien et maritime. 44

4.1 Exigences pour la réalisation d'une FDIA. 63
4.2 Exigence pour la faisabilité de l'approche. 64
4.3 Exigences pour le domaine aérien. 65
4.4 Exigences pour le domaine maritime. 68

6.1 Effet des portées sur les événements. 111

7.1 Calcul du masque pour la labellisation. 149

TABLE DES ACRONYMES

ADS-B	Automatic Dependent Surveillance-Broadcast
AIS	Automatic Identification System
AIVDM	Automatic Identification Vessel Data Message
ATC	Air Traffic Control
ATS	Air Traffic Services
BEG	Base Expression Grammar
DSL	Domain Specific Language
EBNF	Extended Backus-Naur Form
FAA	Federal Aviation Administration
FDIA	False Data Injection Attack
GNSS	Global Navigation Satellite System
IA	Intelligence Artificielle
ICAO	International Civil Aviation Organization
IMO	International Maritime Organization
LSTM	Long Short-Term Memory
LTL	Linear Temporal Logic
MLAT	Multilatération
NMEA	National Marine Electronics Association
OWL	Web Ontology Language
PEG	Property Evaluation Grammar
PSR	Primary Surveillance Radar
RAP	Recognized Air Picture
SAG	Situation Aérienne Générale
SCADA	Supervisory Control And Data Acquisition
SDPS	Surveillance Data Processing and Distribution System
SSR	Secondary Surveillance Radar
SUT	System Under Test
TCAS	Traffic Collision Avoidance System
VHF	Very High Frequency
VTS	Vessel Traffic Services

I

CONTEXTES ET MOTIVATIONS

1

PROBLÉMATIQUE ET POSITIONNEMENT DE LA THÈSE

Le doctorat s'est déroulé au sein de l'équipe VESONTIO, du Département Informatique des Systèmes Complexes (DISC) de l'Institut Femto-ST - UMR 6174 de l'Université de Franche-Comté [1]. Les travaux présentés dans ce document répondent à des besoins de test identifiés dans les domaines des transports aérien et maritime, notamment autour des nouvelles technologies de surveillance déployées ces dernières années afin d'améliorer la localisation des appareils en circulation. Ils consistent à la création d'une chaîne outillée permettant de tester des systèmes de contrôle par injection de fausses données. Les travaux présentés dans ce document s'inscrivent dans le projet ISITE-BFC SARCoS [2] pour la partie conception des scénarios / génération des données de test et dans le projet ANR ASTRID GeLeaD [3] pour les aspects liés à la production des données pour des composants d'apprentissage automatique de détection d'anomalies.

1.1/ ENJEUX DE LA SURVEILLANCE AÉRIENNE ET MARITIME

L'utilisation croissante cette dernière décennie de technologies fondées sur la géolocalisation des appareils pour améliorer les systèmes de surveillance entraîne avec elle son lot de menaces par rapport à l'intégrité des données. La communauté scientifique s'est emparée de ces problématiques et propose des solutions pour réduire le risque lié à la cybersécurité. Cette section présente le cas de deux technologies dites "ouvertes", qui ont fait apparaître, avec leur déploiement massif, de nombreuses vulnérabilités dans les systèmes de surveillances. La première sous-section traite de la technologie ADS-B, utilisée dans le domaine de la surveillance aérienne. La deuxième sous-section concerne

1. https://www.femto-st.fr/fr/Departements-de-recherche/DISC/Presentation [Dernière visite : Novembre 2020]
2. https://projects.femto-st.fr/sarcos/fr [Dernière visite : novembre 2020]
3. https://projects.femto-st.fr/gelead/fr [Dernière visite : novembre 2020]

le système AIS, utilisé dans le domaine de la surveillance maritime. Enfin, la dernière sous-section discute différents moyens à disposition des attaquants pour effectuer leurs attaques.

1.1.1/ CAS DU DOMAINE AÉRIEN : LA RÉSILIENCE FACE AUX CYBERATTAQUES

Avec l'augmentation constante du nombre d'avions en circulation [4], le monde des transports aériens fait face à de nouveaux défis. L'espace aérien devenant de plus en plus surchargé, le Contrôle de la Circulation Aérienne, ou *Air Traffic Control* (ATC), nécessite des technologies capables de tolérer la surveillance simultanée d'un nombre d'aéronefs toujours grandissant tout en garantissant une estimation précise de leur localisation. Dans cette optique, le protocole ADS-B [28] (*Automatic dependent surveillance-broadcast*) a été déployé, entraînant une réduction des coûts de surveillance en parallèle de l'augmentation de la précision dans la localisation des avions [103]. La technologie ADS-B a également aidé certains pays, comme l'Australie, à améliorer la surveillance globale de leur espace aérien en permettant de couvrir des zones inaccessibles pour les technologies radars plus conventionnelles, cet exemple est illustré dans la figure 1.1. En effet, la configuration de l'Australie rend difficile l'installation de radars primaires (à onde électromagnétique) dans toute la partie centrale du pays. Le système de surveillance ADS-B a permis, par le faible coût et la facilité d'installation des antennes, d'augmenter drastiquement la couverture de l'espace aérien de ces régions.

Les communications utilisant le protocole ADS-B requièrent que les transpondeurs des avions diffusent périodiquement leur position complétée par d'autres informations (vitesse au sol, direction, identité, etc.). La mise en place du protocole ADS-B a démarré au début des années 2010 et la quasi-totalité des transpondeurs doivent être capables d'émettre via ADS-B en 2020 [23]. L'arrivée à grande échelle de cette technologie a entraîné un changement de paradigme dans la manière de surveiller le ciel. Historiquement la surveillance de l'espace aérien était principalement basée sur des technologies dites indépendantes et non coopératives, comme le PSR (radar primaire). Le SSR (radar secondaire) marque le début d'une transition qui aboutit finalement à la technologie ADS-B, dite coopérative et dépendante. Dans ce nouveau contexte, les stations au sol ont besoin de la coopération des avions (par envoi de message ADS-B) qui eux-mêmes sont dépendants du système GNSS [16] (*Global Navigation Satellite System*) pour déterminer leur position. On peut simplifier ce changement de paradigme par une évolution de la surveillance aérienne à partir de techniques qui consistaient historiquement à "observer" le ciel, vers une surveillance consistant à "écouter" le ciel. Ce changement fondamental dans la manière de surveiller l'espace

4. http://www.boeing.com/commercial/market/current-market-outlook-2017/ [Dernière visite : Octobre 2020]

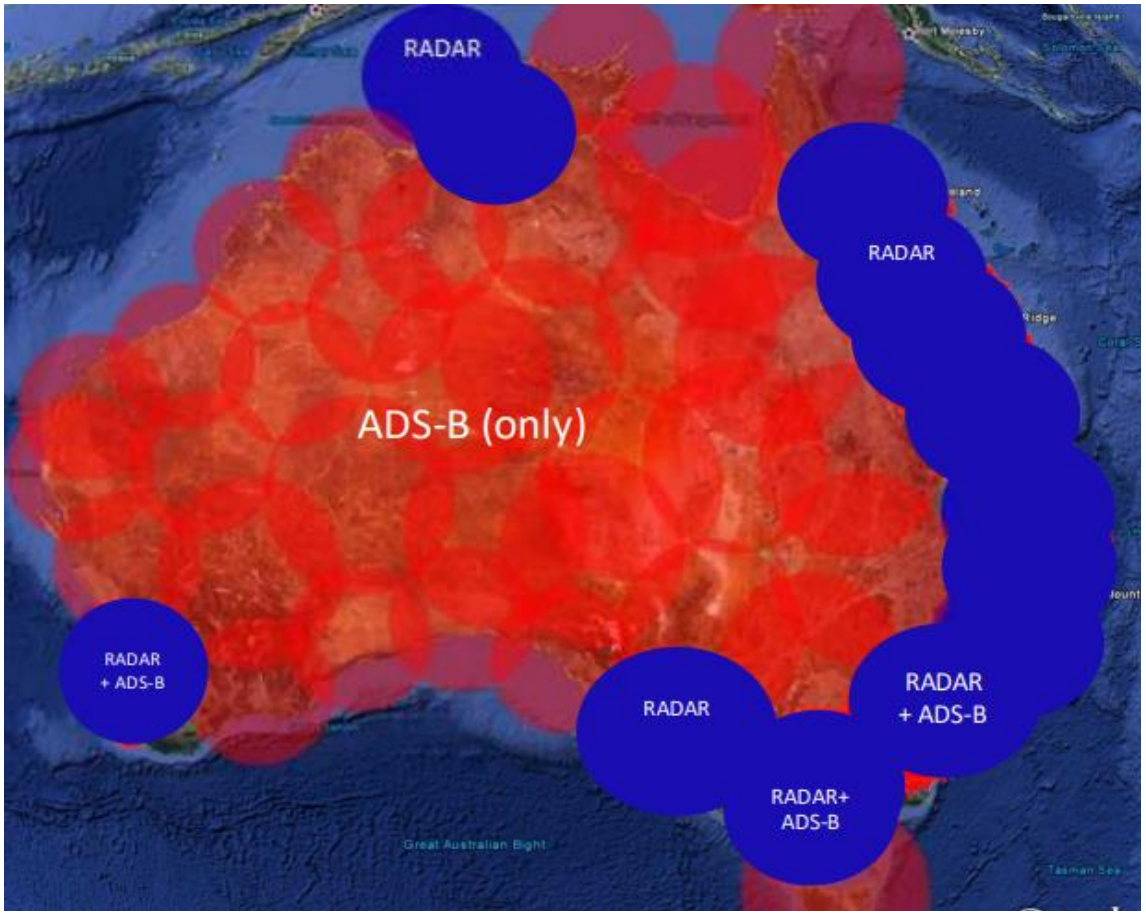


FIGURE 1.1 – Couverture de l'espace aérien Australien par les technologies radar et ADS-B [29]

aérien n'a malheureusement pas été accompagné des réflexions suffisantes en matière de cybersécurité et l'apparition de nouvelles menaces n'a pas été anticipée. En effet, l'ADS-B, dont la spécification remonte aux années 1990 [77], n'utilise pas de chiffrement [69] dans la transmission des messages et n'impose pas d'authentification de la part de l'émetteur. Ainsi, avec un équipement relativement peu coûteux, il est possible de recevoir et d'émettre des messages ADS-B, comme le montre l'existence de produits permettant d'équiper les avions des particuliers d'un transpondeur ADS-B à partir d'un smartphone et d'un dongle [5]. La liberté totale d'émettre ou de recevoir des messages ADS-B rend le protocole vulnérable à différents types d'attaques : *spoofing* (usurpation d'identité), fausses alertes, déni de service, etc. Plus généralement, le protocole est sensible à une classe d'attaques nommées Attaques par Injection de Fausses Données, ou *False Data Injection Attacks* (FDIA). Les FDIA combinent suppression de messages originaux et émission de messages méticuleusement fabriqués de toute pièce. Un attaquant peut alors duper la vigilance des contrôleurs aériens en faisant apparaître de

5. https://www.uavionix.com/products/skybeacon/ [Dernière visite : Septembre 2020]

fausses situations sur leur écran de contrôle. On peut ainsi imaginer pouvoir perturber le contrôle aérien en émettant des messages d'urgence ou en faisant croire au survol d'une zone interdite temporaire par un ou plusieurs aéronefs.

D'autres protocoles sont utilisés en complément de l'ADS-B et des technologies radar PSR et SSR, comme le système de communications entre contrôleur et pilote par liaison de données, basé sur les échanges vocaux, ou le système *Aircraft Communication Addressing and Reporting System*. Néanmoins l'ADS-B joue un rôle central vis-à-vis de la manière dont sont déterminées les positions des avions. Si elles étaient initialement obtenues grâce aux technologies radars, elles le sont désormais grâce au GNSS, notamment en dehors des phases de décollage et d'approche. Son rôle est important au point de pouvoir causer l'annulation de tous les vols d'une zone en cas de dysfonctionnement[6]. Il existe donc un besoin fort de renforcer la sécurité globale du protocole ADS-B. Néanmoins, du fait des propriétés inhérentes à ce protocole, les solutions visant à le renforcer directement entraînent un coût important en temps et en moyens car elles nécessitent la modification des systèmes existants [105]. De ce fait, la tendance est plutôt de renforcer les capacités d'analyse des communications ADS-B. Mais la différenciation entre les attaques et les situations réelles reste encore aujourd'hui un challenge auquel se confronte la communauté ATC, même si, actuellement, de nombreuses solutions basées sur la détection de fausses données ou sur des contrôles d'intégrité sont en déploiement ou à l'étude. Ce besoin d'analyse constitue une des motivations de ces travaux de thèse, comme cela est détaillé dans la section 1.2, car le développement des capacités de détection d'altération nécessite des données de test, c'est à dire des données altérées suivant des scénarios précis.

1.1.2/ Cas du domaine maritime : la prévention de comportements illégaux

Comme le ciel, les mers et les océans ne sont pas épargnés par l'augmentation du trafic [7] : marchandises, plaisance, pêche, exploitation pétrolière, opérations militaires, etc. De nos jours les navires sont spécialisés dans des activités variées et leur équipage jouit d'une grande liberté dans les eaux internationales. Ces dernières représentent 64% de la surface des océans et, bien qu'elles offrent une liberté totale de mouvement, il existe tout de même des interdictions comprenant : le transport d'esclaves, la piraterie, le trafic illicite de stupéfiants et certaines émissions radioélectriques. La pluralité des activités et la flexibilité dans les mouvements des navires produisent un fourmillement incessant dans l'ensemble des mers et océans qui s'avère alors difficile à surveiller. Depuis 2002, dans l'optique d'améliorer la sécurité de la navigation, les navires dont la jauge

6. https://hackaday.com/2019/06/09/gps-and-ads-b-problems-cause-cancelled-flights/ [Dernière visite : Octobre 2020]

brute excède les 300 tonneaux ainsi que tous les navires de plus de douze passagers doivent obligatoirement être équipés de l'*Automatic Identification System* (AIS). Dans son fonctionnement, le parallèle avec la technologie ADS-B est évident. En effet, avec le système AIS, les bateaux déterminent leur position grâce aux systèmes GNSS et la diffuse couplée à d'autres informations (comme le type du navire, sa direction, sa vitesse sur fond, etc.) sur la bande des très hautes fréquences (VHF pour *Very High Frequency*). C'est un système de communication automatisé permettant aux navires et systèmes de surveillance d'établir une estimation du trafic maritime en temps réel dans des zones plus ou moins étendues. L'avancée technologique qu'apporte le système AIS bénéficie ainsi à de nombreux pans du domaine maritime : surveillance du trafic [111], évitement des collisions [114], opération de recherche et de secours, investigation d'incident [41], aide à la navigation, etc.

À plus bas niveau, le système d'échange AIS est basé sur le protocole AIVDM [47], ce dernier ne présente aucun chiffrement, ni authentification de l'émetteur, mettant une fois de plus en avant un parallèle avec le protocole ADS-B. Les messages AIVDM peuvent donc être émis par quiconque dispose d'un encodeur et d'un émetteur VHF. On retrouve donc dans le système AIS, une vulnérabilité à la même classe de cyberattaques qui menacent le protocole ADS-B : les FDIA. La technique pour réaliser ces attaques reste basée sur la modification, la suppression et la création de messages. Ces attaques peuvent avoir différents objectifs qui sont identifiés dans la littérature [10] : *spoofing* (usurpation d'identité), fausse alerte de collision imminente, modification des informations météo, etc.

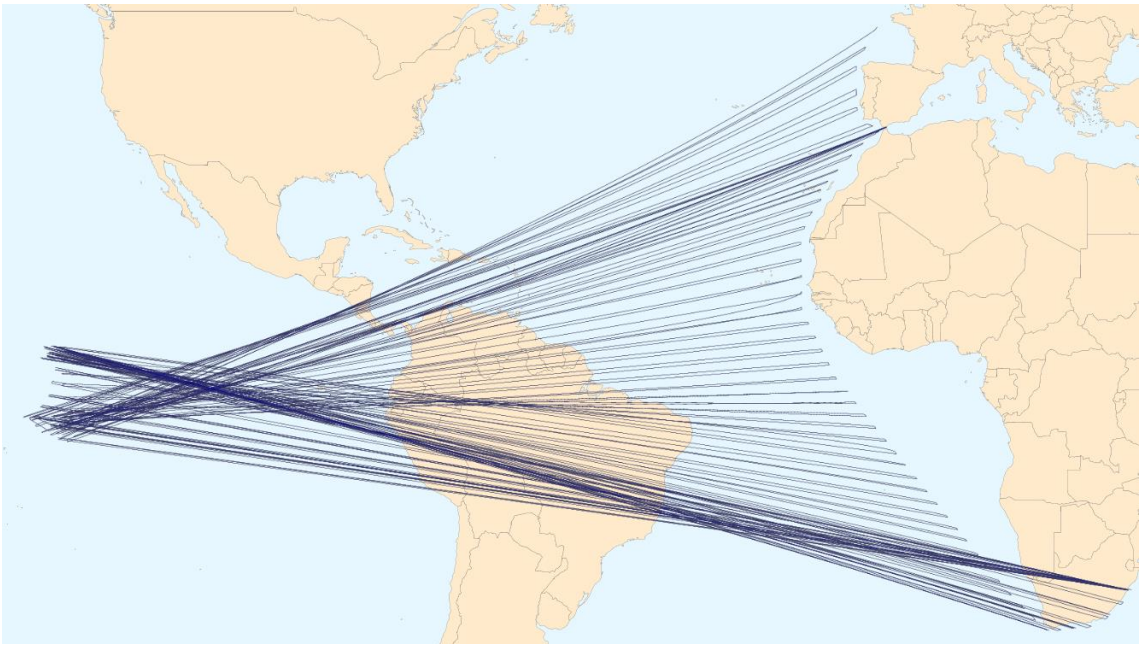


FIGURE 1.2 – Deux navires éloignés utilisent la même identité au même moment.

Si le risque de cyberattaques sur le système AIS est bien réel, ce ne sont pas les attaquants extérieurs à un navire qui préoccupent les experts du domaine. En effet, de nombreux cas de *spoofing* comme celui visible sur la figure 1.2 proviennent de l'équipage du navire lui-même. L'image présente un extrait de messages AIVDM émis par deux bateaux différents utilisant la même identité. Le système interprète les deux trajectoires comme celle d'un seul et même bateau, ce qui produit ces allers-retours entre le milieu du Pacifique et la côte Est de l'Atlantique. Ces incidents sont imputables soit à une mauvaise manipulation du transpondeur, soit à la volonté de dissimuler des actions illégales [90] comme de la pêche en zone interdite. Pour camoufler leurs agissements, les délinquants peuvent également avoir recourt à la coupure pure et simple du transpondeur AIS afin de ne plus être tracé. Pour détecter ce genre de comportement, des moyens originaux sont mis en place comme l'utilisation d'albatros [7]. Ces oiseaux ayant l'habitude de suivre les navires pendant des heures sont équipés d'un émetteur permettant aux autorités de repérer les pêcheurs illégaux. Si cette solution est efficace pour détecter ce type de comportement illicite, la tendance au sein de la communauté scientifique penche plutôt vers l'utilisation de solutions basées sur l'analyse de la logique des données fournies par le système AIS afin de détecter différents types de situations anormales, comme le *spoofing* [90] ou également la pêche illégale [65, 125].

1.1.3/ DES MOYENS DIFFÉRENTS D'UN DOMAINE À L'AUTRE

Globalement, on remarque que les deux domaines que sont la surveillance aérienne et la surveillance maritime possèdent des vulnérabilités face aux FDIA causées par l'utilisation de technologies ouvertes comme ADS-B et AIS. Mais ces deux domaines de la surveillance de systèmes de transport présentent des enjeux et des caractéristiques différents.

S'il est difficile de déclarer qu'un domaine est plus critique que l'autre, force est de constater que le trafic aérien est, à l'heure actuelle, beaucoup plus encadré que le trafic maritime, que cela soit au niveau de la surveillance des appareils, de l'accès à leur bord ou plus simplement de la possibilité de les approcher. Cela se justifie par le nombre important d'attaques ciblant les avions depuis le début des vols commerciaux, de la première recensée (une attaque à la bombe visant le vol *United Airlines 32* le 10 octobre 1932) jusqu'au point marquant la mise en place d'une sécurité maximale, notamment au niveau des aéroports (les événements du 11 septembre 2001). Si cette sécurité renforcée n'est pas justifiée par des cyberattaques, elle limite néanmoins les moyens à disposition des attaquants. Côté maritime, moins d'attaques, ou tentatives, impliquant des vies humaines sont recensées par rapport à l'aviation, donc moins de

7. https://www.rts.ch/info/sciences-tech/11052821-des-albatros-recrutes-pour-detecter-les-pecheurs-illegaux.html [Dernière visite : Octobre 2020]

faits ayant incité à la mise en place d'une sécurité forte[8]. De plus, le statut des eaux internationales offre une liberté d'action importante aux personnes mal intentionnées : de ce fait, la sécurité globale est plus importante sur les trajets entre aéroports que sur les trajets entre ports.

Trois moyens majeurs à disposition des attaquants pour faciliter la réalisation de FDIA sont identifiés : l'accès physique aux transpondeurs, l'accès aux appareils émetteurs et enfin l'accès aux récepteurs.

Accès physique aux transpondeurs :
Dans les avions de ligne, le transpondeur est situé dans le cockpit à côté de ses utilisateurs que sont les pilotes et le commandant de bord. Potentiellement, tout le personnel de bord (hôtesses et stewards compris), dont le nombre est limité, peut avoir un accès physique à l'appareil. Un protocole sécuritaire est en place pour empêcher les passagers d'accéder au cockpit (digicode, temps de déverrouillage contrôlé, ouverture depuis l'intérieur, etc.) et donc au transpondeur. L'accès au transpondeur AIS est plus libre. En effet la timonerie[9] d'un navire est le compartiment où se trouve la barre, le système AIS et les autres outils de navigation. Aucun standard n'impose son verrouillage comme c'est le cas pour les cockpits. Sur la majorité des navires, cette salle est accessible à l'intégralité de l'équipage ainsi qu'aux passagers, bien qu'il soit possible que ces derniers aient à déjouer la vigilance de l'équipage. À noter que la durée de certains trajets (jusqu'à plusieurs mois) augmente la probabilité d'existence d'une possibilité d'accès au transpondeur par un passager non autorisé.

Accès aux appareils émetteurs :
La régulation de l'espace aérien (autorisation nécessaire au décollage), l'altitude (dépassant parfois les 40000 pieds) et la vitesse (frôlant les 400 nœuds) des avions de ligne rendent difficile toute approche visant à corrompre les données à la source. Il est toutefois imaginable que du matériel électronique soit camouflé pour assister l'attaquant dans une future cyberattaque. Ce matériel devra néanmoins échapper à la vérification minutieuse et systématique de l'appareil avant son décollage. En opposition, la lenteur des bateaux et leur libre circulation dans les eaux internationales permettent aux attaquants de facilement se tenir proche d'un navire émetteur de messages AIVDM dont la portée en mer est d'environ 40 milles nautiques [19].

Accès aux récepteurs :
Dans les deux domaines, les appareils sont à la fois émetteurs et récepteurs de messages. Ainsi, en ce qui concerne les appreils récepteurs, on aboutit au même constat que pour l'accès aux appareils émetteurs. En revanche, pour le cas des récepteurs au

8. Malgré cela, il est à noter les événements du 11 septembre 2001 ont également mis en lumière le manque de sécurité dans l'espace maritime et ont ainsi influencé la mise en place du système AIS.

9. Compartiment d'un navire où l'on effectue la navigation appelé *passerelle* dans la marine militaire et *timonerie* dans la marine marchande.

sol qui composent l'architecture des deux systèmes, le constat est différent.

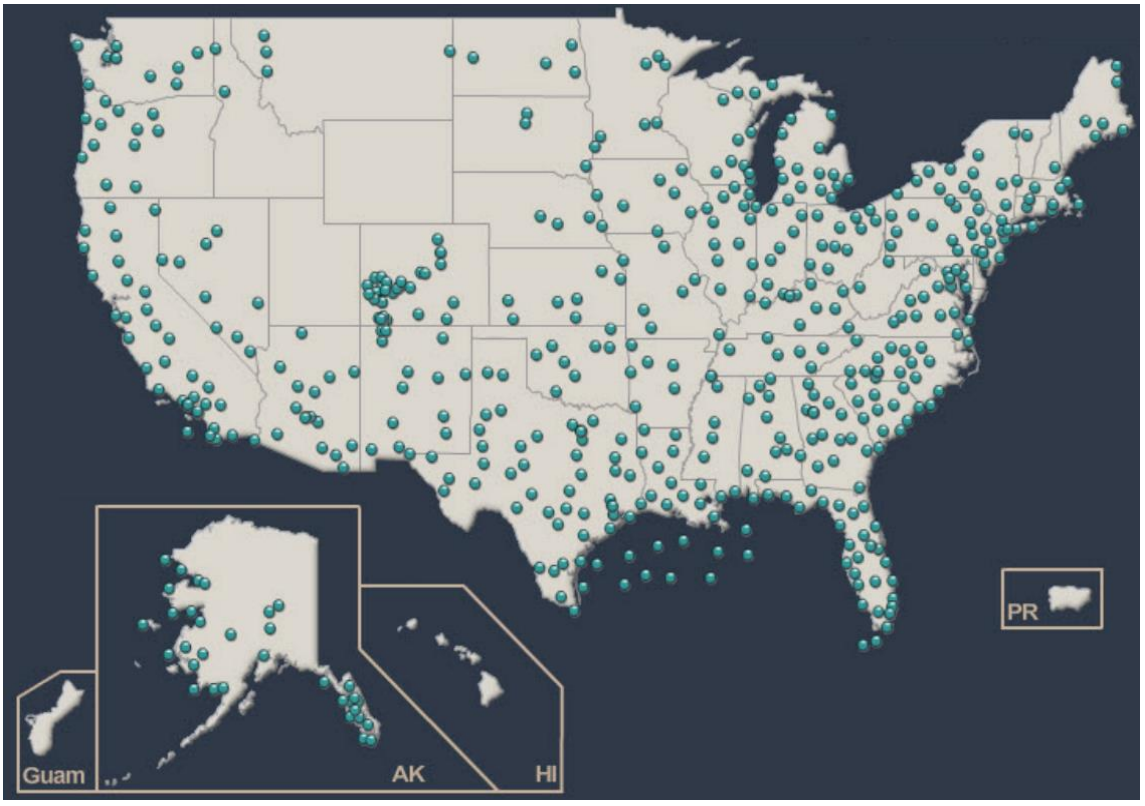


FIGURE 1.3 – Localisation aux États-Unis des stations au sol chargées de transmettre l'ADS-B aux centres de contrôle en 2014.

Selon l'agence gouvernementale chargée des réglementations et des contrôles concernant l'aviation civile aux États-Unis, la *Federal Aviation Administration* (FAA), la localisation des stations au sol (ou GBT pour *Ground Based Transceiver*) n'est pas tenue secrète [10]. La figure 1.3 montre la distribution d'environ 800 récepteurs ADS-B sur le sol des États-Unis. Il existe même un site web [11] qui les recense sur une carte interactive. La portée des messages ADS-B étant approximativement de 180 à 370 kilomètres [23], on imagine facilement qu'un message émanant d'un seul avion peut être reçu par plusieurs antennes. Pour ce qui est du système AIS, les principaux récepteurs de messages direct sont des centres de surveillance établis sur les côtes. En France on les appelle les sémaphores et les Centre régional opérationnel de surveillance et de sauvetage (CROSS). La rotondité de la terre limitant fortement la portée des messages AIVDM, les centres côtiers reçoivent peu de messages directement. Ils sont tout de même accessibles dans des rayons de quelques centaines de mètres à quelques kilomètres et leur emplacement est public. Pour combler cette portée très limitée, les exploitants du

10. https://www.faa.gov/air_traffic/flight_info/aeronav/acf/media/Presentations/13-02-RD261_Charting-ADS-B-GBT-JCollins.pdf [Dernière visite : Octobre 2020]
11. http://towers.stratux.me/ [Dernière visite : Octobre 2020]

système AIS préfèrent s'appuyer sur des constellations de satellites afin d'obtenir une couverture plus globale du trafic maritime [19], ce qui rend de fait ces récepteurs très difficilement accessibles.

À la vue des différences entre ces trois moyens au sein de chacun des domaines, on peut établir le constat suivant : dans le cas de cyberattaques de type FDIA visant la technologie ADS-B, le risque principal vient d'un attaquant au sol situé à proximité des antennes réceptrices des messages ADS-B. À l'inverse, dans le système AIS, le risque principal vient plutôt d'un attaquant situé en mer, à proximité des transpondeurs émetteurs de messages AIVDM. Ce constat est en accord avec le contenu de la section 1.1.2 p. 6 en ce qui concerne le domaine maritime et la propension de l'équipage lui-même à falsifier les données AIS émises. Si les moyens mis à la disposition des équipages permettent de comprendre à quels niveaux se situent les sources probables d'attaques, il est également important de prendre en compte la réalité du domaine. En effet, si dans les vols commerciaux le personnel de bord a peu d'intérêt personnel évident à fausser les informations émises par leur avion, les équipages de bateau peuvent avoir des raisons pécuniaires d'agir, par exemple dans un contexte de pêche illégale, de piraterie ou de contrebande où la discrétion est de mise et peut amener à falsifier son identité pour se dissimuler.

L'utilisation de technologies ouvertes comme ADS-B et AIS pose en définitive des problématiques propres à chacun des deux domaines. Ces problématiques, bien que différentes, présentent de fortes similarités au niveau des solutions proposant d'y répondre, fondée sur l'analyse des communications pour identifiés des anomalies.

Dans le cadre de nos travaux de thèse portant sur la conception et génération de tests par altération de données, l'application aux domaines aérien et maritime a permis de définir des concepts et techniques plus génériques qu'en étant restreint à un seul domaine d'application. Les similarités et différences entre les domaines ont été un atout pour définir une approche plus générique, et potentiellement utilisable pour d'autres domaines de la surveillance des systèmes de transport (comme le véhicule terrestre par exemple).

1.2/ CAS D'APPLICATION DES TRAVAUX

Les solutions qui visent à réduire le risque lié aux cyberattaques demandent à être testées en profondeur et de manière efficace. Cette section présente plusieurs problématiques du test des systèmes de surveillance qui ont motivé nos travaux. La première sous-section s'intéresse aux tests de sécurité dans le domaine aérien. La deuxième sous-section se focalise sur l'utilisation de techniques de détection d'anomalies, notamment à base d'intelligence artificielle. Enfin la troisième et dernière

section identifie ce qui peut être une approche commune pour les deux domaines de l'aérien et du maritime et en souligne les aspects clés développés ensuite dans le corps de la thèse.

1.2.1/ TESTS DES SYSTÈMES DE CONTRÔLE ORIENTÉS CYBERSÉCURITÉ

La communauté de la recherche dans les transports aériens a mené de nombreuses recherches sur la vulnérabilité des protocoles face aux cyberattaques, notamment autour du protocole ADS-B. Des analyses de sécurité montrent ainsi que l'utilisation indépendante de ce protocole entraîne des risques en matière de cybersécurité, avec l'existence de vulnérabilités, en particulier l'absence d'authentification et de chiffrement [23], qui permettent la création, suppression ou modification de messages [69]. Afin d'augmenter la résilience des systèmes basés sur le protocole ADS-B contre les cyberattaques, des solutions sont en cours de déploiement ou même déjà exploitées [12]. Ces solutions s'apparentent à des contrôles d'intégrité et peuvent être divisées en trois catégories :

— Basées sur les propriétés physique du signal : Multilatération (MLAT) et multilatération étendue intégrées à l'ADS-B [13], mesure de l'angle d'arrivée du message, corrélation entre la puissance du signal et la distance de l'émetteur, etc.

— Basées sur la logique des données : Contrôle du changement de position par rapport à la vitesse, analyse comportemental des avions, etc.

— Basées sur l'agrégation de données : Source de données multiples (plusieurs radars pour une zone), fusion de données, etc.

Afin d'évaluer le risque d'une cyberattaque, il est impératif de mettre en relation sa probabilité d'occurrence avec sa criticité. Dans le cadre des systèmes de contrôle du trafic aérien, bien que les FDIA requièrent une haute technicité pour être effectuées correctement, elles n'en demeurent pas moins réalisables (voir section 3.1 p. 39). La probabilité qu'une FDIA soit menée à bien est donc assez faible, néanmoins son impact peut être très important, ce qui peut motiver des acteurs à gros budget (comme des états par exemple, on parle alors de *State-sponsored hacker groups*), et classe ce type d'attaque dans les attaques à haut risque. Cette absence de droit à l'erreur implique une totale confiance envers les différentes solutions qui sont mises en place pour contrer

12. Le programme de recherche SESAR vise à mettre en place des solutions de modernisation des systèmes de gestion du trafic aérien européen, certaines de ces solutions cible la sûreté et la sécurité autour du protocole ADS-B : https://www.sesarju.eu/sites/default/files/documents/reports/SESAR_Solutions_Catalogue_2019_web.pdf [Dernière visite : Octobre 2020]

13. La multilatération est une technique de navigation basée sur les temps de réception des messages, elle est expliquée dans la section 2.1.4 p. 26

les FDIA. Il faut s'assurer que chacune d'entre elles remplit son rôle, pouvant aller de la détection à la réduction de l'impact, en passant par la récupération d'un système.

Pour être testées, les solutions basées sur les propriétés physiques du signal nécessitent par définition une infrastructure physique et potentiellement coûteuse à mettre en place. À l'inverse, les solutions basées sur la logique des données ou sur leur agrégation peuvent être testées numériquement en se focalisant sur la sémantique de la donnée. En effet, une manière de valider ces solutions est de les confronter à des scénarios s'apparentant à des cyberattaques, et plus précisément à des FDIA. C'est précisément les tests en lien avec la sémantique de l'information qui nous intéressent dans le cadre de cette thèse.

La validation de ces composants logiques s'appuie sur l'identification de scénarios à risque et leur traduction en cas de test, c'est-à-dire en données à injecter dans les Systèmes Sous Test (SUT pour *System Under Test*). Un scénario est dit "à risque" s'il est susceptible de compromettre le comportement du composant testé de quelque manière que ce soit. La criticité du domaine aérien entraîne une certaine opacité autour de ces aspects de cybersécurité, notamment en ce qui concerne les scénarios à risque. En effet, les experts sécurité du domaine chargés d'identifier ces scénarios concrets sont peu enclins à les communiquer, en particulier dans le domaine militaire (ces scénarios d'attaques sont généralement classifiés Confidentiel Défense).

Dans le cadre de nos travaux, nous avons rencontré cette limite de la confidentialité dans l'accès aux scénarios d'attaques que les experts souhaiteraient tester. Mais l'analyse de l'état de l'art, les remontées des incidents de cybersécurité dans les forums spécialisés et les discussions avec les experts des domaines considérés, nous ont permis d'appréhender la taxonomie des scénarios FDIA identifiés actuellement (cf. section 2.3.1 p. 33).

1.2.2/ LA DÉTECTION D'ANOMALIES

Dans tous les domaines, les études liées aux données font face à des valeurs possédant un écart avec la norme. Quand cet écart excède un certain seuil bien défini, appelé seuil d'anomalie, on parle alors d'anomalie. Dans des espaces qui comprennent peu de dimensions, ce seuil peut être calculé de manière assez triviale, en revanche dans le cas de données avec un nombre de dimensions plus important, la norme est plus difficile à définir. Le *Machine Learning* (apprentissage automatique ou ML) a contribué de manière significative à améliorer les capacités de détection de ce type d'anomalies dans de nombreux domaines d'application : détection de cellules anormales en imagerie médicale [93, 98], détection d'intrus dans un réseau à partir des journaux (*logs*) [123] ou encore détection d'incohérences dans le trafic via les trajectoires des bus [54].

Un inconvénient majeur pour utiliser des techniques de *Machine Learning* pour la

détection d'anomalies est le manque de jeux de données anormales qui répondent pleinement aux besoins d'entraînement, de test et de validation des modèles. En effet, les données anormales sont par nature rares dans les historiques de données des différents domaines. Elles peuvent même être inexistantes quand elles sont le résultat de cyberattaques dans des systèmes critiques où leur indisponibilité est la conséquence de leur caractère exceptionnel et de la confidentialité qui les entoure. On distingue deux grands types de modèle de détection :

— Les modèles supervisés sont entraînés avec des données normales et anormales, les données sont labellisées, c'est-à-dire que leur nature (normale ou anormale) est une information annotée sur la donnée et donc connue par le modèle lors de son apprentissage.

— Les modèles non-supervisés, à l'inverse, ne connaissent pas la nature des données car elles ne sont pas labellisées. L'entraînement de ces modèles consiste donc à découvrir les structures implicites de ces données. En détection d'anomalie, cela implique souvent d'entraîner les modèles seulement avec des données normales, cela a, par exemple, déjà été fait sur le protocole ADS-B [39], mais de pouvoir tester la validité du modèle entrainé avec des données anormales.

Le manque de données anormales contraint donc la création de modèles de détection supervisés et les modèles présentent souvent des résultats médiocres, tandis que les modèles non supervisés présentent de bons scores de détection sur des jeux de données déséquilibrés [81, 17] entre données normales et anormales.

Pour ce qui est de la détection d'anomalies utilisant le *Machine Learning* dans le protocole ADS-B, quelques expérimentations peuvent être trouvées dans la littérature [39, 4]. En l'absence d'un historique de messages ADS-B certifiés comme anormaux, les modèles de détection utilisent exclusivement de l'apprentissage non supervisé. L'évaluation de ces modèles repose souvent sur des anomalies générées via des scripts ad-hoc.

On identifie deux besoins liés à la difficulté de se procurer des jeux de données anormales dans de la détection d'anomalies à base de *Machine Learning*. Le premier besoin est celui de pouvoir obtenir facilement des données de test afin d'évaluer l'efficacité d'une Intelligence Artificielle (IA) de détection. Le deuxième besoin est celui de disposer de données d'entraînement pour permettre l'utilisation de modèles supervisés. Dans les deux cas, on attend des données anormales générées un haut degré de réalisme. En effet, l'utilisation de techniques d'IA réside avant tout dans la détection de données anormales difficilement repérables "à l'œil nu" pour les contrôleurs, car noyées dans un trafic aérien dense, ou trop réalistes pour attirer l'attention. De plus, pour valider une IA de détection, il est utile de disposer d'un *continuum* entre anomalies brutes et réalistes afin d'y situer la limite des capacités du composant testé. Une quantité importante de données est également essentielle à une validation ou un entraînement de

qualité. La génération massive de données altérées est donc nécessaire à l'élaboration de modèles de détection, et doit être accompagnée par une scalabilité permettant la création de données de test ou d'entraînement dans des temps acceptables. Enfin, l'entraînement de modèles d'apprentissage supervisé implique de disposer de données labellisées, c'est-à-dire que chaque message ADS-B embarque des informations sur les modifications qu'il a éventuellement subi.

Dans le domaine maritime, sur le système AIS, on peut trouver un certain nombre de résultats de recherche visant à mettre en œuvre des moyens de détecter des anomalies dans les données : fusion avec des données complémentaires [52], analyse statistique [92, 57], ou encore *Machine Learning* [116]. Les besoins de validation de ces cas d'utilisation sont semblables aux besoins identifiés pour le domaine aérien dans les sections 1.2.1 p. 12 et 1.2.2 p. 13. Une différence importante déjà mentionnée est que dans le domaine maritime (sections 1.1.2 p. 6 et 1.1.3 p. 8) les FDIA sont souvent liées à différents comportements illégaux. De ce fait, en plus de la détection d'anomalies, on note également dans la littérature l'existence de travaux focalisés directement sur la détection de ces comportements dans les cas où les contrevenants ne prennent pas les mesures nécessaires pour masquer leurs activités comme la plongée illégale [6] ou la pêche en zone interdite [56].

1.2.3/ DES BESOINS CONVERGEANTS ENTRE LES DOMAINES D'APPLICATION

Si la spécification de tests orientés cybersécurité (section 1.2.1 p. 12) pose la question du pouvoir d'expression, c'est-à-dire la possibilité de spécifier des scénarios diversifiés et répondant aux objectifs de tests des experts cybersécurité, le cas de détection d'anomalies via le *Machine Learning* insiste plutôt sur la quantité et la pertinence des données générées (section 1.2.2 p. 13). En combinant ces deux problématiques et en concevant un moyen de test pour y répondre, nous avons développé une chaîne logicielle visant dans un premier temps à spécifier n'importe quelle FDIA identifiée dans la taxonomie grâce à un ensemble de langages dédiés (DSL pour *Domaine Specific Languages*). Pour répondre aux besoins de tests de cybersécurité ou pour valider / entraîner un composant de détection d'anomalies par apprentissage automatique, il s'agit de générer des données anormales, s'apparentant alors à des cas de test, reproduisant l'attaque spécifiée.

Finalement il apparaît que l'utilisation d'un tel outil ne se limiterait pas seulement à la création de test de sécurité, ou de composant IA, mais pourrait être étendue à la création d'autres types de test. Par exemple, dans la famille des tests de robustesse, ceux vérifiant les propriétés de sûreté peuvent être adressés avec un tel outil permettant de produire massivement des anomalies (données sémantiquement incorrectes), les tests

de scalabilité apparaissent également concernés par un tel outil de par sa nature à créer de la donnée synthétique et ainsi augmenter le volume de données entrant dans un système à un instant t.

La conception et le développement d'un environnement permettant de tester ces solutions constitue l'objectif de cette thèse. Ces travaux ont donné lieu, dans un premier temps, à la création d'un environnement de test pour les systèmes aériens basés sur l'ADS-B : *FDI-T*, pour *False Data Injection Testing*. Une version pour le domaine maritime a également été développée dans un second temps, permettant de valider le passage des concepts d'un domaine à l'autre. Cette version pour le maritime présente néanmoins un niveau de maturité moins avancé, c'est pourquoi on se focalisera d'avantage sur le domaine aérien au cours de ce manuscrit. FDI-T permet aux experts métier de définir des scénarios de tests sur des données réelles issues d'enregistrements du trafic aérien. Ces scénarios sont ensuite exécutés sur les systèmes de surveillance afin d'en évaluer la capacité de détection face à des anomalies de sécurité ou de sûreté potentielles liées aux FDIA. La réalisation de cette chaîne outillée constitue une contribution principale de cette thèse. Elle a en outre nécessité la mise en place d'approches dédiées pour répondre aux problématiques du pouvoir d'expression pour la spécification de tests, de la pertinence des données générées, et enfin pour la généralisation de l'approche.

1.3/ OBJECTIF DE LA THÈSE ET QUESTIONS DE RECHERCHE

En se basant sur les observations de la section 1.2.3 p. 15, l'objectif de la thèse peut être formuler de la manière suivante :

RO : Dans quelle mesure une approche générique à plusieurs domaines, peut-elle permettre grâce à l'utilisation d'un ensemble de langages dédiés la génération automatique de données altérées selon des scénarios conçus par les experts du domaine ?

Si l'objectif de recherche est bien identifié, avec la création de l'environnement de test, il est également nécessaire de définir la manière dont cet objectif peut être atteint. De fait, les travaux de thèse sont axés autour de plusieurs questions de recherche :

RQ-1 : Dans quelle mesure un ensemble de langages dédiés assure-t-il la couverture de la taxonomie des FDIA ?

Le pouvoir d'expression détermine la capacité à concevoir un éventail de scénarios aussi large que possible. Si l'objectif est principalement de couvrir la taxonomie des attaques identifiées dans l'état de l'art, il est également important de permettre la scénarisation de FDIA n'ayant pas encore été clairement identifiées dans l'état de l'art.

RQ-2 : Dans quelle mesure est-il possible de garantir le réalisme de données simulant des FDIA, tout en les générant de manière automatique et massive ?

La qualité des données générées est essentielle pour la majorité des cas d'utilisation identifiés. De la détection de fautes à l'entraînement ou à la validation de modèle de détection, le réalisme des données doit être assuré à un certain degré. L'obtention de données réalistes n'est pas un problème à part entière, la difficulté vient surtout de la génération de gros volumes de données réalistes de manière automatisée.

RQ-3 : Dans quelle mesure l'approche peut elle être générique à plusieurs domaines ?

Ici, l'objectif est d'identifier les parties spécifiques et génériques de l'approche. Pour faire cela, il faut implémenter l'approche dans deux domaines d'applications, après quoi il est possible de proposer un processus permettant le portage de l'approche d'un domaine à un autre.

Chacune des trois questions de recherche identifiées ici est traitée dans un des chapitres de la partie II.

1.4/ CONTEXTES ET PUBLICATIONS DE LA THÈSE

Ces travaux de recherche ont donné lieu à un ensemble de publications qui sont présentées dans cette section. Il s'agit plus précisément de publications en journal et conférence, de présentations effectuées dans des *workshops*, de livrables réalisés dans le cadre de projets collaboratifs, et enfin de contributions logicielles, accessibles sur Github.

Journal international :

- A Domain Specific Language to Design False Data Injection Tests for Air Traffic Control Systems (2020) :
  - **Journal** : *International Journal on Software Tools for Technology Transfer* (STTT)
  - **Auteurs** : Alexandre Vernotte, Aymeric Cretin, Bruno Legeard et Fabien Peureux.
  - **Description** : L'article porte sur les langages dédiés à la scénarisation des tests FDIA par altération de données ADS-B pour le domaine aérien et AIS pour le domaine maritime. Cet article est axé sur le développement de ces langages qui est détaillé en plusieurs étapes : décision, analyse du domaine,

conception, implémentation et validation.

Conférences et Workshops internationaux :

- Improved Testing of AI-based Anomaly Detection System using Synthetic Surveillance Data (2020) :
  - **Workshop** : 8th OpenSky Symposium (2020)
  - **Auteurs** : Antoine Chevrot, Alexandre Vernotte, Aymeric Cretin, Fabien Peureux et Bruno Legeard.
  - **Description** : L'article présente les apports de la génération de données de tests réalistes pour la validation d'un modèle de Machine Learning de détection d'anomalies sur le protocole ADS-B. Notamment grâce à la possibilité de générer des anomalies pour tester ce modèle de détection.
- Test Data Generation for False Data Injection Attack Testing in Air Traffic Surveillance (2020) :
  - **Workshop** : *ITEQS 2020, 4th International Workshop on Testing Extra-Functional Properties and Quality Characteristics of Software Testing*
  - **Auteurs** : Aymeric Cretin, Alexandre Vernotte, Antoine Chevrot, Bruno Legeard et Fabien Peureux.
  - **Description** : *Best Paper Award*. L'article porte sur la génération de données de tests, il inclut la description des algorithmes implémentés et une validation effectuée sur un module de détection de données par Machine Learning.

Doctoral symposium :

- Increasing the Resilience of ATC systems against False Data Injection Attacks using DSL-based Testing (2018) :
  - **Conférence** : 8th International Conference on Research in Air Transportation (ICRAT'18)
  - **Auteurs** : Aymeric Cretin, Bruno Legeard, Fabien Peureux et Alexandre Vernotte.
  - **Description** : Présentation des principes de la génération de données altérées dans le contexte ADS-B.

Présentations :

- FDI-T : A Testing Framework for Air Traffic Control Systems Based on False Data Injection
  - **Workshop** : 6th OpenSky Workshop (2018) : https://workshop.opensky-network.org/2018
  - **Description** : Présentation de la chaîne outillée FDI-T à la communauté OpenSky (fournisseur de données ADS-B)
- FDI-T : A Testing Framework for Air Traffic Control Systems
  - **Événement interne** : Poster pour les Journée FEMTO (Juin 2019)
  - **Description** : Présentation de la chaîne outillée FDI-T

Logiciels :

- Composant de scénarisation des cas de test
  - **Description** : Une interface graphique permet de créer des spécifications de tests sur la base de trois langages dédiés :
    - Un langage de scénarisation spécifiant l'attaque à réaliser.
    - Un langage de sélection spécifiant les avions à ciblés sur la basé sur la logique temporelle linéaire (LTL).
    - Un langage de déclencheurs basé sur la création d'événements.
  - **Dépôt** : https://github.com/aymeric-cr/dsl-scenario
- Composant de génération des données de test
  - **Description** : Sur la base des spécifications générées par le module de scénarisation, ce module applique des modifications à un extrait de communication ADS-B.
  - **Dépôt** : https://github.com/aymeric-cr/sbs-generation
- Intégration des deux modules à une chaîne outillée multi-domaine (FDI-T)
  - Initialement une version pour le domaine du contrôle aérien, réalisé avec des partenaires industriels : https://www.kereval.com
  - Portage vers une version pour le domaine de la surveillance maritime.

Livrables de projets collaboratifs

- ANR-ASTRID-GeLeaD_D1.1_A-data-generation-tool-for-AI-testing-and-validation
  - **Projet ANR ASTRID GeLeaD** : https://projects.femto-st.fr/gelead/fr

  - *Description* : Contribution à la description des langages dédiés à la scénarisation.

- D4.1_SARCoS_ArchitectureDocument (10/2019)
  - **Projet ISITE-BFC SARCoS** : https://projects.femto-st.fr/sarcos/fr
  - **Description** : Contribution à la description de l'architecture générique de la chaîne outillée.

1.5/ PLAN DE LA THÈSE

La thèse se décompose en trois parties, chacune divisée en trois chapitres.

La partie I aborde le contexte et les motivations de la thèse avec en premier lieu ce chapitre 1. Il présente les enjeux et les cas d'application des travaux, ainsi que les questions de recherche associées et les différentes contributions auxquelles ils ont mené. Les sujets du chapitre 2 sont le fonctionnement, les vulnérabilités et les améliorations existantes en terme de cybersécurité des protocoles de surveillance sur lesquels portent les travaux de cette thèse. Enfin, le dernier chapitre de cette partie, le chapitre 3 présente l'état de l'art du domaine, avec trois sections dédiées respectivement : aux FDIA, aux approches de test existantes autour des protocoles ADS-B et AIS, et enfin à l'utilisation de langages dédiés pour la spécification de tests.

Les contributions techniques de la thèse sont détaillées dans la partie II. Elle se compose de trois chapitres, soit un pour chaque contribution technique. Ainsi, le chapitre 4 présente l'architecture générique sur laquelle s'appuient la génération de données de test qui est traitée dans le chapitre 5, et l'ensemble des langages dédiés à la spécification des tests que nous avons conçu, dont la présentation est divisée étape par étape dans le chapitre 6.

La partie III aborde l'implantation des travaux de thèse au sein d'une chaîne outillée présentée dans le chapitre 7. Le dernier chapitre, le chapitre 8 présente ensuite l'utilisation de cette chaîne outillée pour la génération de données d'entraînement et de validation pour des composants de détection d'anomalies dans le domaine aérien et dans domaine maritime.

Finalement, une conclusion reprend les questions de recherche et discute les résultats obtenus au fur et à mesure de ce manuscrit.

2

ZOOM SUR LES SURVEILLANCES AÉRIENNE ET MARITIME

Depuis plusieurs années, la réponse aux enjeux de cybersécurité constitue un axe majeur de la modernisation des technologies dans les domaines de la surveillance aérienne et de la surveillance maritime. Alors que le chapitre 1 a posé les bases des problèmes de sécurité touchant les technologies ADS-B et AIS, ce présent chapitre offre une vision plus détaillée de chacun de ces deux domaines, et du rôle que jouent l'ADS-B et l'AIS au sein de ceux-ci. Ainsi, la première section traite de la surveillance dans le domaine aérien, tandis que la deuxième section se focalise sur son homologue côté maritime. Enfin la troisième et dernière section met en avant les vulnérabilités de chaque technologie et liste les attaques qui les menacent.

2.1/ LA SURVEILLANCE DE L'ESPACE AÉRIEN

La surveillance de l'espace aérien est un processus complexe et critique dont le but est de détecter, localiser et identifier tous les avions actifs dans le ciel à un instant donné. Le contrôle de la circulation aérienne (ou ATC pour *Air Traffic Control*) s'appuie sur la surveillance afin d'assurer la sécurité des avions en garantissant, entre autres, une distance de séparation minimale entre eux. Par exemple, cette séparation est de 3 milles marin (nm) à l'approche d'un aéroport, et 50 milles marin pour les routes au-dessus des océans où les moyens de surveillance sont réduits. Les systèmes de surveillance détectent les avions et envoient des informations détaillées aux systèmes de contrôle, permettant ainsi aux contrôleurs de guider les avions en sécurité. Dans les zones où la densité du trafic est faible, la séparation des avions peut être assurée par des reports de position manuels (vocaux ou textuels) entre les pilotes et les contrôleurs. En revanche, dans les zones denses, qui sont de plus en plus nombreuses, le contrôle du trafic n'est pas réalisable sans la mise en œuvre de systèmes de surveillance automatiques.

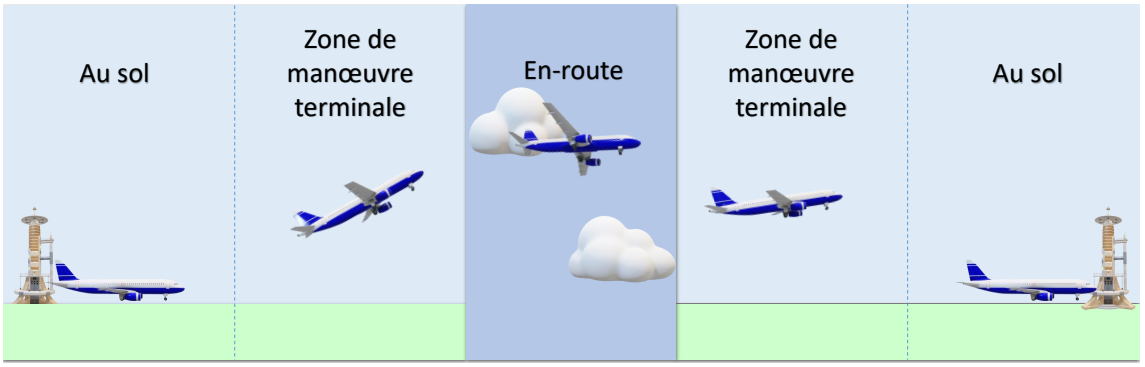


FIGURE 2.1 – Division de la surveillance aérienne.

On distingue trois divisions dans l'ATC, déterminées selon la proximité de l'avion avec ses aéroports de départ et d'arrivée. Cette division est illustrée dans la figure 2.1 et détaillée ci-dessous :

— La surveillance au sol : Elle concerne les avions qui circulent au sol dans les différentes zones des aéroports, comme par exemple les voies de circulation, les pistes ou les aires de stationnement. Ces différentes zones sont gérées par les contrôleurs dans la tour de contrôle à laquelle les technologies de surveillance au sol de l'aéroport fournissent une situation générale précise.

— La zone de manœuvre terminale (ou *Terminal Manoeuvring Area*) : Elle se focalise sur les avions sur le point de décoller ou d'atterrir. Cette division est gérée par les contrôleurs dans les tours de contrôle et plus précisément dans la vigie de la tour ou dans une salle d'approche dédiée. Différentes technologies assistent les contrôleurs dans leur tâche.

— La surveillance En-Route : Elle englobe les avions qui volent à moyenne ou haute altitude, loin des aéroports. Les contrôleurs qui gèrent cette surveillance se situent dans des centres de contrôle régionaux (ou *Area Control Centers*). La surveillance En-Route a pour but d'établir la Situation Aérienne Générale [1] (SAG) précise du secteur de l'espace aérien qu'elle couvre.

Dans ces trois divisions, les systèmes de surveillance s'appuient sur des technologies qui sont historiquement des radars dont l'utilisation est couplée à d'autres techniques plus récentes, comme la technologie ADS-B ou la multilatération.

2.1.1/ LE RADAR PRIMAIRE

Aussi appelé le radar de veille ou *Primary Surveillance Radar* (PSR), le radar primaire fait partie de la première génération de radar. Développé dans les années 1940, il reste

1. La Situation Aérienne Générale (SAG) fait référence à l'état du trafic (la localisation de chaque avion) à un instant t. En anglais on utilise le terme *Recognized Air Picture* (RAP).

toujours utilisé pour les phases d'approche et parfois pour la surveillance En-Route. Le fonctionnement du radar primaire est basé sur la détection de signal par écho : une antenne rotative émet une impulsion électromagnétique dans une large portion de l'espace aérien qui est reflétée sur les avions se trouvant dans cet espace. À la manière d'un écho, l'onde est reçue par le radar après avoir reflété sur un avion, ce qui permet de déterminer la position de ce dernier de manière continue à partir de l'angle de réception de la réflection et le temps de trajet aller-retour de chaque pulsation [102]. Le radar primaire est une technologie indépendante et non-coopérative car les avions sont passifs durant le processus de surveillance : ils n'apportent aucune aide au-delà de la réflection des ondes radar sur leur carlingue.

La surveillance non-coopérative et indépendante présente l'avantage de détecter tous les avions, y compris ceux ne souhaitant pas volontairement communiquer leur position, comme cela peut arriver en temps de guerre ou plus globalement dans un contexte militaire. Pour cette raison, ce type de surveillance est continuellement requis dans les zones où le trafic est dense, non seulement pour des questions de sécurité évidentes, mais également pour s'assurer d'avoir une estimation complète de la SAG. De plus, l'information fournie par ce type de surveillance, en plus d'être complète, possède un haut niveau d'intégrité tant il est complexe pour un attaquant de falsifier secrètement des impulsions électromagnétiques.

En dépit de ces nombreux avantages, le radar primaire possède tout de même des inconvénients. En effet, s'il permet de détecter les avions indépendamment de leur équipement, un radar primaire détecte également tout autre objet physique. Ainsi les bâtiments, les montagnes, ou même les nuages entre autres peuvent venir perturber le fonctionnement des radars primaires. Les éoliennes ont, par exemple, déjà posé des problèmes aux radars primaires par le passé [101]. Tous les objets autres que des avions se trouvant dans la portée du radar génèrent ainsi du bruit. Aussi, la puissance de rayonnement requise est très élevée car les échos doivent être facilement reconnaissables, faisant du radar primaire une technologie coûteuse. De plus, le radar primaire ne permet pas d'identifier les avions, sa portée est limitée et son taux de rafraîchissement est faible. Enfin, utilisé seul, il ne fournit pas l'altitude des avions sous surveillance, seulement leur vitesse et leur position 2D (longitude et latitude).

2.1.2/ LE RADAR SECONDAIRE

Aussi appelé *Secondary Surveillance Radar* (SSR), le radar secondaire constitue la deuxième génération de radars utilisés dans les surveillances d'approche et En-Route. À l'inverse du radar primaire, le radar secondaire est une technologie coopérative : sous la forme de messages numériques, les stations au sol diffusent des interrogations vers

les transpondeurs des avions sur la fréquence 1030 MHz, ces derniers répondent alors en fournissant l'information demandée sur la fréquence 1090 MHz [112]. De manière similaire au radar primaire, le radar secondaire est une technologie indépendante dans le sens où la position et la vitesse des avions sont déterminées via l'angle de réception du message ainsi que son temps de parcours. Historiquement, deux modes d'interrogation existaient : le mode A pour l'identification et le mode C pour l'altitude. Ils ont par la suite été substitués par le mode S (pour *selective*) qui permet l'interrogation d'un seul avion et ainsi l'allègement du canal 1090 MHz. En effet, initialement les interrogations n'étaient pas ciblées, ainsi tous les avions répondaient à la même interrogation. Les zones couvertes par plusieurs radars, et où le trafic est important, étaient alors sujettes à des interférences mutuelles, causées par la multiplication des interrogations *all-call*, entraînant ainsi la saturation du canal 1090 MHz [8]. L'interrogation sélective nécessite de pouvoir identifier un avion parmi les autres, un identifiant à quatre chiffres, le *squawk*, était initialement utilisé pour le mode d'interrogation A mais il était trop sensible à la collision d'identités. Actuellement, le *squawk* est remplacé [2] par un identifiant de transpondeur unique au niveau mondiale : l'adresse ICAO 24 bits, du nom anglais de l'organisation de l'aviation civile internationale (*International Civil Aviation Organization*). Cette adresse ICAO est utilisée pour les interrogations sélectives (mode S), elle remplace. Le mode S permet également l'échange d'information supplémentaire par rapport aux modes A et C.

Comparé au radar primaire, le radar secondaire est moins sensible aux interférences et couvre des zones plus importantes. Néanmoins, il partage avec son aîné une latence importante et un taux de rafraîchissement faible (en moyenne un message toutes les 6 à 12 secondes par avion) dépendant de la vitesse de rotation de l'antenne. Le niveau d'intégrité des données du radar secondaire est inférieur à celui du radar primaire car il est basé sur l'échange d'information. Enfin, le radar secondaire assure une précision de la localisation des avions de 1 à 2 milles marins ce qui, étant donné son taux de rafraîchissement, permet une séparation de 3 milles marins entre avions [117].

2.1.3/ LE SYSTÈME ADS-B

En complément de l'utilisation des radars primaires et secondaires, les communications utilisant le système ADS-B consistent à l'utilisation, par les avions, du *Global Navigation Satellite System* (GNSS) afin de déterminer leur position et de la diffuser périodiquement sans aucune sollicitation extérieure. Outre leur position, les avions diffusent d'autres informations issues des systèmes de bord : altitude, vitesse au sol, identité, direction, etc. Des stations au sol réceptionnent les messages, les traitent, puis envoient les

2. À noter que le *squawk* reste utilisé aujourd'hui pour communiquer le statut de l'avion ou des situations anormales comme un détournement (7500), une panne radioélectrique (7600) ou une situation de détresse (7700)

informations aux centres de contrôles concernés. La liaison de données de l'ADS-B se trouve généralement sur la fréquence 1090 MHz *Extended Squitter* (1090ES), la même fréquence utilisée par le mode S. Toutefois, il existe un nouveau standard pour la liaison de données dédiée à des protocoles tels que ADS-B : *Universal Access Transceiver*, mais il requiert une mise à jour matériel et n'est pas très répandu pour le moment.

La technologie de surveillance ADS-B est coopérative car un transpondeur est requis par les avions, et dépendante car elle dépend des données collectées par les appareils de bord des avions. L'apparition de cette technologie constitue un changement fondamental dans le contrôle du trafic aérien. Effectivement, les antennes réceptrices de messages ADS-B ne nécessitent pas d'être positionnées avec un angle et une direction spécifiques pour être efficaces, facilitant de fait leur mise en place. Les avions sont également capables, en étant équipés de récepteur ADS-B, de recevoir des informations d'autres avions. Cette communication entre avions offre aux pilotes une meilleure connaissance de la situation aérienne qui les entoure. Cela a également profité au système d'alerte de trafic et d'évitement de collision (TCAS pour *Traffic Collision Avoidance System*). En effet, depuis le TCAS-II (deuxième génération) les données ADS-B reçu par les avions sont prises en compte ce qui a amélioré l'efficacité du système grâce à l'acquisition de données en temps réel et plus précises.

L'introduction de la technologie ADS-B a aussi amélioré la connaissance de la SAG du côté des contrôleurs dans la surveillance En-Route, notamment du côté des zones non couvertes (*Non Radars Areas*). Théoriquement, l'ADS-B offre la possibilité de réduire la séparation minimale des avions par rapport à celle requise dans les procédures actuelles [1]. En effet, la technologie offre une précision de 0.05 milles marins dans la localisation et de 19.4 milles marin par heure dans le calcul de la vitesse, et un taux de rafraîchissement d'une à deux mises à jour par seconde. Concrètement, cette technologie a été conçue pour réduire la séparation latérale des avions de 90 à 20 milles marins et la séparation longitudinale de 80 à 5 milles marins dans les zones non couvertes. Et de 5 à 3 milles marins dans les zones couvertes [117]. L'ADS-B présente également l'avantage d'être une technologie peu coûteuse de par son infrastructure minimale requise. Par exemple, un récepteur l'ADS-B peut être acheté sur internet pour quelques centaines d'euros [3]. Comme mentionné précédemment, l'ADS-B permet une localisation précise des avions avec une taux de rafraîchissement important et une latence faible. Les inconvénients majeurs de cette technologie sont d'ordre sécuritaire : ils concernent principalement l'absence de chiffrement et d'authentification dans l'envoi des messages. Ces problèmes sont discutés en section 2.3.1 p. 33.

3. https://flywithscout.com [Dernière visite : novembre 2020]

2.1.4/ La multilatération

Désignée par l'abréviation MLAT, la multilatération est une technique de surveillance coopérative et indépendante principalement utilisée pour la surveillance au sol et en approche. Pour la surveillance En-Route, le terme adapté est multilatération étendue (*Wide Area Multilateration*). La multilatération est différente des technologies précédentes dans le sens où ce n'est pas un protocole à part entière mais plutôt une technique mathématique basée sur la pluralité des récepteurs issus des autres technologies (notamment radar secondaire et ADS-B). En effet un même signal reçu par différents récepteurs permet le calcul des différences de temps de propagation (mode hyperbolique [20]), ou de la somme des temps d'arrivée (mode circulaire [94]). Avec suffisamment de mesures venant de sources séparées (au moins quatre), il est possible de déterminer l'origine d'un signal et d'en déduire la position de l'avion émetteur [95]. Comparée aux radars primaire et secondaire, la multilatération possède un taux de rafraîchissement, une fiabilité, une flexibilité, une stabilité et une précision plus importantes. Néanmoins le taux de rafraîchissement est directement lié au protocole de surveillance sur laquelle se greffe la MLAT. La technique est en théorie peu coûteuse, aussi bien pour les équipements au sol que dans les avions. Toutefois, la mise en place de la multilatération est complexe car c'est un système distribué qui possède de fortes contraintes de synchronisation et peut devenir coûteux à déployer sur de larges régions.

2.1.5/ La chaîne de surveillance

Indépendamment de la technologie, le but des équipements de surveillance est de capturer le trafic aérien et d'envoyer des rapports aux différents services de la circulation aérienne (ou ATS pour *Air Traffic Services*) utilisés par les contrôleurs. Ces derniers peuvent ainsi s'assurer que les avions se trouvent dans l'espace qui leur est dédié. Avant d'atteindre les différents ATS, les rapports de surveillance convergent vers un système de traitement ou de distribution des données de surveillance (SDPD ou SDPS pour *Surveillance Data Processing and Distribution System*). La première fonction d'un SDPS est d'analyser les rapports de surveillance, de les rassembler en pistes (une piste représentant un avion avec son identité, sa trajectoire, et toute autre information complémentaire), puis de transmettre ces pistes à des ATS. Le SDPS principal en Europe, et le plus récent, est proposé par EUROCONTROL : il s'agit du système ARTAS, pour *ATM Surveillance Tracker and Server*, qui est actuellement en déploiement [4].

Un aperçu simplifié de la chaîne de surveillance d'EUROCONTROL [5] est donné dans la figure 2.2. Il apparaît que le SDPS reçoit des rapports depuis différentes sources

4. https://www.eurocontrol.int/product/artas [Dernière visite : novembre 2020]
5. https://www.eurocontrol.int [Dernière visite : décembre 2020]

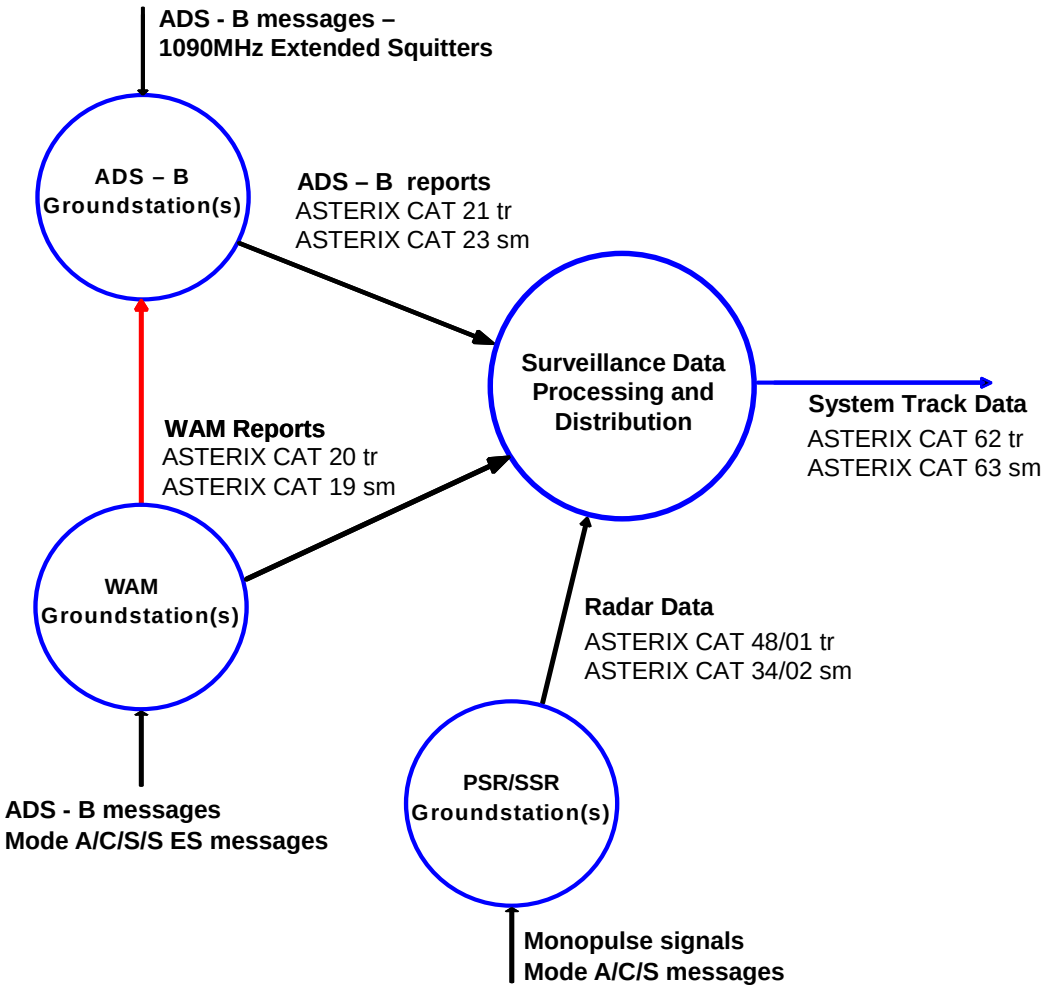


FIGURE 2.2 – Aperçu de la chaîne de surveillance d'EUROCONTROL

et les fusionne pour produire les données de pistes (*System Track Data*) à fournir aux différents services. Les échanges de données entre toutes les entités de cette chaîne (stations au sol, SDPS, différents ATS) sont rendus possible grâce à l'ASTERIX (*All Purpose Structured EUROCONTROL Surveillance Information Exchange*), une spécification créée par EUROCONTROL et dont l'objectif est de faciliter les échanges en harmonisant les informations de surveillance entre les différents systèmes et les différents pays européens [6].

6. https://www.eurocontrol.int/asterix [Dernière visite : novembre 2020]

2.2/ LA SURVEILLANCE DE L'ESPACE MARITIME

La surveillance de l'espace maritime est un terme très vaste qui englobe une importante variété de missions. L'origine de cette diversité est liée à la diversité des activités des navires sur les mers et océans, et à la liberté offerte par le statut des eaux internationales. Les différentes missions qui rentrent dans le cadre de la surveillance maritime sont également très dépendantes de l'emplacement géographique. En France, on identifie par exemple plusieurs catégories de missions :

— Secours et sauvetage : cela comprend la veille sur les fréquences de détresse, la surveillance du plan d'eau, la diffusion d'alerte ou de bulletin météo, ou encore la coordination des missions de sauvetage.

— Régulation du trafic et de la pêche : elle englobe la réglementation de la pêche, la surveillance des infractions de navigation, le contrôle de la séparation du trafic, notamment dans les ports de commerce, ou la prévention des pollutions.

— Surveillance du territoire : elle consiste à identifier les navires, à signaler les navires suspects aux autorités ou à surveiller les approches des ports militaires.

Les missions de la surveillance maritime varient non seulement d'un pays à l'autre, mais également d'un poste de surveillance à un autre au sein d'un même pays. Par exemple, tandis que certains garde-côtes français opérant dans les zones littorales à risque ont pour mission la surveillance des feux de forêts, d'autres sont affectés à la surveillance de sites archéologiques sous-marins.

2.2.1/ LES *Vessel Traffic Services*

Le terme de *Vessel Traffic Services* (VTS) est répandu dans le domaine de la surveillance maritime. Dans une optique d'uniformisation entre les pays, l'Organisation Maritime Internationale (ou IMO pour *International Maritime Organization*) définit les VTS en 1985 dans ses lignes directrices comme "Tout service, mis en place par une autorité compétente[7], conçu pour améliorer la sécurité, l'efficacité et la protection de l'environnement. Cela peut aller de la diffusion de simples messages d'information à la gestion du trafic dans un port ou une voie de navigation" [45]. Ces lignes directrices ont été complétées au fil du temps par des mises à jour appelées des "résolutions".

Les VTS dans leur définition originale sont assez comparables à la surveillance en zone d'approche pour le contrôle aérien dans le sens où ils couvraient essentiellement les ports et les approches immédiates des côtes [124] (de quelque milles marins à

7. Dans les résolutions suivantes, l'autorité compétente est précisée comme une autorité : "Fait responsable, par le gouvernement, de la sécurité et l'efficacité du trafic maritime, ainsi que de la protection environnementale".

quelques centaines). Effectivement, les technologies datant d'avant les années 2000 ne permettaient pas la surveillance de secteur plus étendus. Entre les résolutions de 1997 et de 2018 de l'IMO, d'importants changements ont vu le jour, notamment au niveau européen avec : la mise en place de l'AIS, la structuration de la chaîne de données avec SafeSeaNet[8], la création de l'Agence Européenne pour la Sécurité Maritime[9] (EMSA pour *European Maritime Safety Agency*), et l'apparition du concept de e-Navigation.

Avant 2019, trois services sont identifiés comme étant des VTS :

— **Information Service** : Le service d'information consiste pour un opérateur VTS à apporter des informations aux navires afin de les aider dans la prise de décision. Les opérateurs VTS doivent être en mesure de connaître la situation autour du navire et de communiquer avec lui via une liaison VHF.

— **Traffic Organisation Service** : Ce service fournit la même prestation que le service d'information, tout en exerçant une autorité sur les navires pour prévenir les collisions. Les opérateurs VTS ont les mêmes prérequis que les opérateur du service d'information, avec en plus des outils d'évaluation des risques, comprenant par exemple, des outils de détection de collision ou de contrôle de vitesse.

— **Navigation Assistance Service** : Il s'agit d'un service d'assistance à la navigation qui met à disposition, et sur demande des navires, diverses informations sur une situation particulière ou inhabituelle.

Depuis 2019, ces trois services, finalement jugés comme sources de confusion, sont fusionnés en un seul service qui "consiste à l'information, la supervision, la gestion, l'organisation du trafic maritime, ainsi que l'assistance à la navigation" [46]. Cette volonté de moderniser la surveillance maritime s'accompagne d'avancées technologiques qui permettent de diversifier les missions mais également de surveiller des zones plus étendues, notamment grâce à l'apparition de technologies satellites.

Comme pour la surveillance de l'espace aérien, on peut aujourd'hui identifier trois divisions dans la surveillance maritime, chacune de ces divisions étant supportée par différents systèmes de surveillance et de communication : la surveillance côtière, la surveillance portuaire et la surveillance au large. Si les deux premières sont assurées par des outils comme des radars (équivalents aux radars primaires dans l'aviation), des capteurs AIS, des jumelles ou des télescopes, la surveillance au large est très limitée du fait de la rotondité de la terre. Pour augmenter la portée de la surveillance, l'utilisation des technologies satellites s'avère essentielle : on peut citer pas exemple ARGOS (géolocalisation), COSPAS-SARSAT (sauvetage), INMARSAT (communication), CLEANSEANET (surveillance des pollutions) ou encore S-AIS (satellites récepteurs de messages AIS).

8. http://www.emsa.europa.eu/ssn-main/ssn-how-it-works.html [Dernière visite : novembre 2020]
9. http://www.emsa.europa.eu/ [Dernière visite : novembre 2020]

2.2.2/ LE SYSTÈME D'IDENTIFICATION AUTOMATIQUE

Apparue dans les années 1990, la technologie AIS est largement utilisée depuis près de 20 ans. Elle a initialement été développée pour offrir aux navires et postes de surveillance une estimation en temps réel du trafic proche (de l'ordre de la dizaine de kilomètres). Le but était l'amélioration de la sécurité de la navigation, notamment l'évitement des collisions accidentelles en favorisant, pour les bateaux, la capacité d'anticiper les trajectoires de leurs voisins. L'équipement AIS est obligatoire depuis 2004 pour les bateaux dont le tonnage excède les 300 tonneaux ou de plus de 12 passagers.

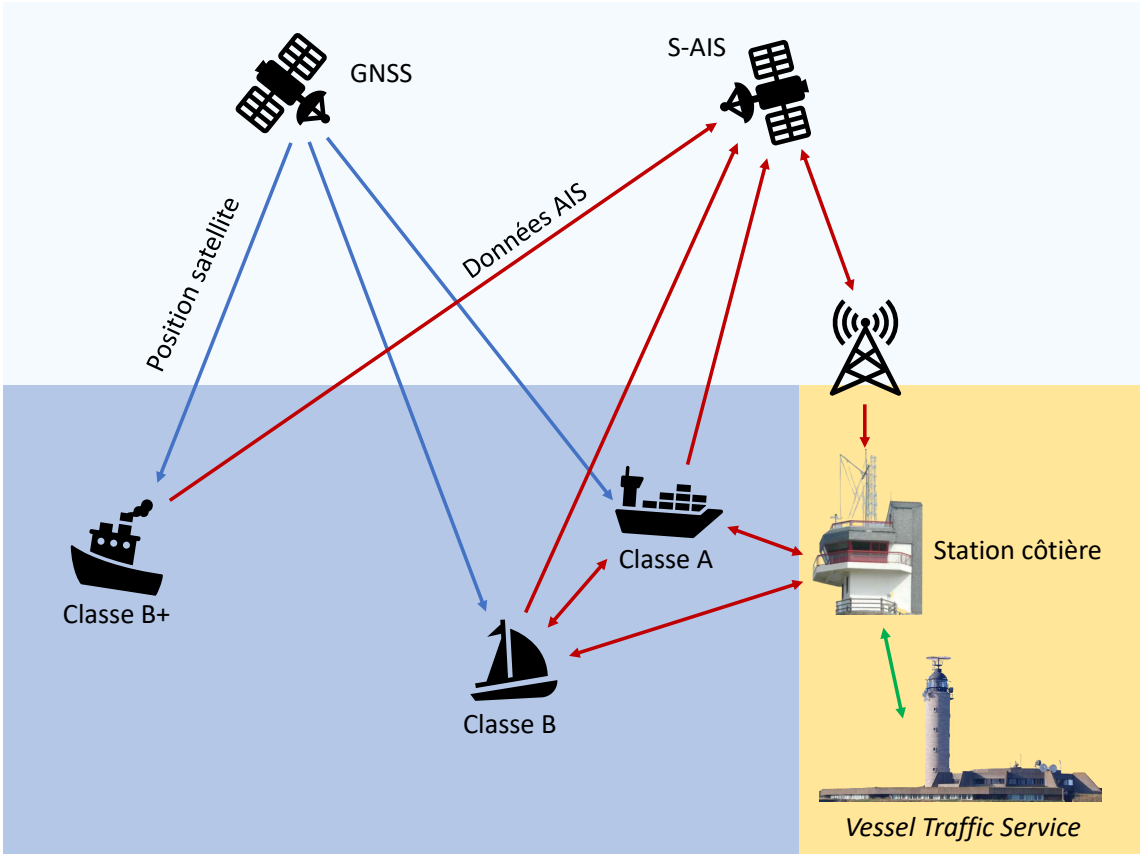


FIGURE 2.3 – Fonctionnement de la technologie AIS.

Le fonctionnement du système AIS est illustré dans la figure 2.3, Les bateaux déterminent leur position grâce au GNSS avec une diffusion couplée à d'autres informations sur la bande VHF. Les messages sont au format AIVDM [47] et ne sont ni chiffrés, ni authentifiés. La diffusion est effectuée périodiquement et la fréquence d'émission s'adapte à la situation du bateau. Cette diffusion va de quelques secondes à quelques minutes selon la classe du transpondeur. L'AIS est en effet divisé en plusieurs classes, selon le gabarit du bateau qui héberge le transpondeur AIS :

— **Classe A** : Elle concerne les bateaux qui sont obligatoirement équipés de l'AIS. Les

messages émis contiennent alors des informations supplémentaires par rapport aux autres classes, comme par exemple le type de cargaison, le tirant d'eau, la destination ou l'estimation de l'heure d'arrivée. Les messages sont émis au minimum toutes les 2 à 3 secondes.

— **Classe B** : Cette classe est dédiée aux navires de plaisance ou de pêche de moins de 15 mètres. Les messages sont émis au minimum toutes les 30 secondes, toujours selon la vitesse sur fond du bateau.

— **Classe B+** : Aussi appelée classe B SOTDMA, c'est la classe la plus récente. Elle comble l'écart entre les classes A et B et les messages sont émis au minimum toutes les 5 secondes. L'installation d'un système AIS sur les navires des classes B et B+ se fait sur la base du volontariat.

Le tableau 2.1 résume les différentes fréquences d'émission entre les différentes classes selon la situation des navires.

TABLE 2.1 – Taux de transmission selon la classe du transpondeur AIS.

Situation	**Classe A**	**Classe B+**	**Classe B**
Ancré ou amarré	3 minutes	3 minutes	3 minutes
Vitesse entre 0 et 2 nœuds	10 secondes	3 minutes	3 minutes
Vitesse entre 2 et 14 nœuds	10 secondes	30 secondes	30 secondes
Vitesse entre 2 et 14 nœuds + changement de cap	3.3 secondes	30 secondes	30 secondes
Vitesse entre 14 et 23 nœuds	6 secondes	15 secondes	30 secondes
Vitesse entre 14 et 23 nœuds + changement de cap	2 secondes	15 secondes	30 secondes
Vitesse supérieure à 23 nœuds + changement de cap	2 secondes	5 secondes	30 secondes
Information statiques (identité, dimensions, etc.)	6 minutes	6 minutes	6 minutes

Initialement, les récepteurs AIS étaient placés sur les côtes et sur les bateaux, ce qui a permis aux gardes-côtes et aux bateaux eux-mêmes de connaître, en temps réel, la positions des bateaux équipés du système AIS à proximité. Toutefois, dans cette configuration, les données étaient difficiles à collecter de par la courte portée du système. En 2008, l'introduction de satellites équipés de récepteurs AIS (S-AIS) a permis une couverture mondiale et la constitution d'historiques de données centralisés conséquents. La mise en place d'une couverture satellite répond donc au besoin d'une surveillance à très longue distance, permettant ainsi de surveiller les cargos de matières dangereuses, et les potentiels comportements illicites en haute mer [19]. L'utilisation de satellites a donné lieu à de nombreuses applications [33] autour des données produites par le système AIS, comme par exemple : la surveillance des zones de pêches [70], la détection des déversements d'hydrocarbure [31], la prévention des collision entre cétacés

et bateaux [119] ou encore la simulation de trafic pour l'entraînement des équipages [59].

2.3/ LES VULNÉRABILITÉS DES PROTOCOLES

Le passage progressif de technologies non-coopératives et indépendantes (radars primaires) vers des technologies coopératives et dépendantes (ADS-B et AIS) a créé par extension une dépendance des systèmes de contrôle envers des entités extérieures. Autant pour les avions que les bateaux, les systèmes de contrôle font désormais confiance à des systèmes qui peuvent présenter des vulnérabilités, comme par exemple le GNSS [36], ou directement aux systèmes de bords des véhicules, pour déterminer la positions de ces derniers. Ces nouvelles dépendances, combinées à l'introduction de liaisons de données ouvertes et non protégées, ont entraîné des problèmes de cybersécurité alarmants.

Ainsi, les caractéristiques de sécurité partagées entre les technologies ADS-B et AIS permettent aux attaquants la réalisation d'opérations malicieuses telles que :

— **L'injection (ou création) de message** : Elle consiste à émettre des messages illégitimes mais qui respectent la syntaxe du protocole. Cette opération est permise par l'absence d'authentification au sein des protocoles, empêchant ainsi la vérification logique de la source des messages.

— **La suppression de message** : Elle consiste à supprimer les messages émis par une cible. On distingue la suppression du message d'un brouillage dans le sens où elle ne vise qu'un sous-ensemble déterminé de messages, ce qui la rend plus difficile à réaliser que son homologue car elle nécessite une haute synchronisation.

— **La modification de message** : Elle consiste à modifier le contenu des messages. Cette modification peut se faire grâce à différentes techniques comme la combinaison d'une suppression et d'une injection. Il est également possible d'utiliser la superposition d'un signal sur le signal original, agissant ainsi comme un masque, afin d'effectuer des modifications directement au niveau des bits transmis. Une modification réussie se doit de respecter la syntaxe du protocole afin que le message soit accepté par les systèmes de contrôle (dans le cas inverse, elle s'apparente alors à une suppression puisque le message ne sera pas pris en compte par le récepteur).

Le premier objectif de cette section est de s'appuyer sur différentes analyses de vulnérabilité afin de montrer la faisabilité des trois opérations malicieuses identifiées ci-dessus. La première sous-section traite des vulnérabilités et des attaques réalisables au sein du protocole ADS-B, tandis que la deuxième sous-section se focalise sur son homologue du domaine maritime. Finalement, l'objectif de la troisième et dernière

sous-section est d'unifier le vocabulaire utilisé au sein des deux domaines en utilisant une classification à trois niveaux qui se décompose en opérations, en attaques et en scénarios.

2.3.1/ CAS DU PROTOCOLE ADS-B

En 2010, la *Federal Aviation Administration* (FAA) publie un rapport d'analyse [2] sur plusieurs aspects de la cybersécurité autour de la technologie ADS-B. Le rapport stipule que, malgré le caractère confidentiel de ces analyses empêchant la divulgation de leur résultat, la FAA est en mesure de confirmer que "l'utilisation de données ADS-B n'expose pas un avion à un risque accru par rapport au risque encouru aujourd'hui (N.D.R. 2010)". En dépit de ce constat, des analyses de sécurité menées sur l'ADS-B par des chercheurs extérieurs à l'agence américaine [71, 23, 87, 96] ont mis en lumière des failles de sécurité communes entre les transmissions du radar secondaire et celles de l'ADS-B, en précisant toutefois que la probabilité d'exploitation de ces caractéristique à des fins malicieuses ne sont pas les mêmes.

2.3.1.1/ LES VULNÉRABILITÉS DU PROTOCOLE ADS-B

Le rapport de la FAA [2] met en avant la fusion entre les données fournies par les radars primaires et celles fournies par la technologie ADS-B comme protection contre les cyberattaques. Dans leur analyse [71], McCallie et al. réfutent cet argument en rappelant que, selon le planning NextGen [10], l'ADS-B est identifié comme le futur système de surveillance principal. De plus, comme l'a montré l'exemple de l'Australie avec la figure 1.1, de nombreuses portions de l'espace aérien sont couvertes uniquement par la technologie ADS-B, ce qui était d'ailleurs précisément une des raisons de sa mise en place.

Que cela soit le radar secondaire, avec les modes A, C et S, ou l'ADS-B, aucune des deux technologies n'utilise de mécanisme de chiffrement de la donnée. De plus les communications se font sur une fréquence connue (1090 MHz). Cela permet à quiconque de recevoir les messages circulant sur cette fréquence, violant ainsi la caractéristique de confidentialité. C'est par ce mécanisme qu'il est possible de trouver des sites, comme OpenSky Network [11] ou FlightRadar24 [12], qui partagent en temps réel la SAG à des échelles continentales.

De plus, la caractéristique d'authentification de l'émetteur du message n'est également pas respectée. La conséquence de ce manque de sécurité autour de la donnée est qu'il

10. https://www.faa.gov/nextgen/ [Dernière visite : novembre 2020]
11. https://opensky-network.org/ [Dernière visite : novembre 2020]
12. https://www.flightradar24.com/ [Dernière visite : novembre 2020]

est possible, avec l'équipement approprié, de se faire passer pour un avion. Dans le cas du radar secondaire, cette capacité d'émettre clandestinement représente un risque moindre car l'utilisation d'antennes à faisceau (directionnelles) implique que le calcul de la position d'un avion se base sur des propriétés physiques du signal, compliquant ainsi l'injection de faux messages. Toutefois, dans le cas du protocole ADS-B, le calcul de la position à bord des avions permet l'utilisation d'antennes omnidirectionnelles. Ces antennes offrent davantage de flexibilité aux attaquants dans le positionnement de leurs émetteurs [23], ce qui facilite la réalisation technique des attaques et viole la caractéristique d'authenticité.

La violation des trois caractéristiques (authentification, confidentialité et authenticité) conduit donc à la possibilité de réaliser différentes attaques combinant des créations, des suppressions et des modifications de messages.

2.3.1.2/ LES ATTAQUES SUR LE PROTOCOLE ADS-B

Afin de prouver la faisabilité de ces attaques, certaines d'entre elles ont été réalisées de manière expérimentale à l'aide d'un émetteur-récepteur SDR USRP N210 et d'une antenne SBS-3 [13]. Le rôle du SDR USRP N210 consiste à empêcher la réception de messages issus du trafic réel par l'antenne SBS-3, et d'émettre de faux messages à la place [96]. Différentes attaques sont ainsi identifiées dans la littérature :

— ***Ghost Aircraft Injection*** : La création d'avion fantôme consiste à créer des messages ADS-B de toutes pièces puis à les diffuser afin de feindre l'existence d'un avion.

— ***Ghost Aircraft Flooding*** : Cette attaque est similaire à la précédente, à la différence qu'elle consiste à la création de multiples avions fantômes de manière simultanée dans le but de saturer la SAG et ainsi de générer un déni de service sur les systèmes de surveillance.

— ***Virtual Trajectory Modification*** : La modification virtuelle de trajectoire consiste à modifier les positions contenues dans les messages émis par un avion afin de feindre une trajectoire différente de la trajectoire réelle.

— ***False Alarm Attack*** : La création d'une fausse alarme est basée sur la même technique que l'attaque précédente, à savoir la modification de messages. Cette attaque consiste à identifier les messages contenant le code de statut de l'avion (squawk) et de les modifier en remplaçant cette identifiant par un code d'urgence : 7500 pour un détournement, 7600 pour un problème lié au communication et 7700 pour une alerte générale.

13. https://jp.wimo.eu/sbs-3-virtual-radar_e.html [Dernière visite : novembre 2020]

— ***Aircraft Disappearance*** : La disparition d'avion est réalisée par la suppression ciblée des messages émis par un sous-ensemble spécifique d'avion. Cela peut mener, par exemple, au dysfonctionnement du système d'évitement des collisions, ou à l'atterrissage forcé d'un avion pour des vérifications de sécurité.

— ***Aircraft Spoofing*** : L'usurpation de l'identité d'un avion consiste à modifier les messages d'un avion en remplaçant son ICAO ainsi que son indicatif d'appel (*callsign*) par d'autres empruntés à un autre appareil. Cela peut permettre par exemple le survol de zones non autorisées grâce à l'utilisation d'une identité habilitée.

À noter que ces attaques sont classées dans la catégorie des attaques par injection de fausses données (FDIA pour *False Data Injection Attacks*). Apparues initialement dans le domaine des réseaux électriques intelligents, la section 3.1 p. 39 leur est dédiée et en propose une définition multi-domaine plus générale.

2.3.2/ Cas du système AIS

Comme pour l'ADS-B, les organismes officiels communiquent assez peu en ce qui concerne la sécurité de la technologie AIS. Si l'association internationale de signalisation maritime (aussi appelée IALA pour *International Association of Marine Aids to Navigation and Lighthouse Authorities*) a effectué des mises en garde sur les vulnérabilités du protocole [14, 15], il est plus difficile de trouver l'équivalent chez son homologue de l'IMO. Une fois de plus, il faut compter sur des chercheurs indépendants pour la production d'analyses de sécurité [37, 10, 40]. Comme pour l'ADS-B, ces analyses mettent en avant le non-respect des caractéristiques de sécurité pour empêcher la manipulation des messages AIS.

2.3.2.1/ Les vulnérabilités du système AIS

En effet, ces analyses relèvent l'absence de mécanisme de chiffrement au sein des communications [40]. De plus, les communications utilisent des fréquences connues et accessibles par tous (161.975 MHz et 182.025 MHz). Comme pour le domaine aérien, il existe des sites partageant en temps réel la position d'un très grand nombre de navires à l'échelle mondiale, comme par exemple MarineTraffic [16] ou VesselFinder [17], ces sites

14. http://www.iainav.org/News/nws0456-ais-vulnerability.pdf [Dernière visite : novembre 2020]
15. https://www.navcen.uscg.gov/pdf/IALA_Guideline_1082_An_Overview_of_AIS.pdf [Dernière visite : novembre 2020]
16. https://www.marinetraffic.com/fr/ais/home/centerx:58.4/centery:-10.1/zoom:3 [Dernière visite : novembre 2020]
17. https://www.vesselfinder.com/fr [Dernière visite : novembre 2020]

ayant même accès à des données satellites (S-AIS). Ici aussi, la caractéristique de confidentialité n'est pas assurée.

Le système AIS ne propose pas de mécanisme qui permet l'identification formelle de la source d'un message du fait de l'absence d'authentification. Ainsi, n'importe qui est en mesure de créer de faux messages et de les diffuser sans avoir à justifier de son identité. Concernant la caractéristique d'authenticité, on a vu dans la section 1.1.3 p. 8 que, dans le domaine maritime, les attaquants peuvent facilement se placer à portée d'émission des transpondeurs et ainsi altérer les signaux émis par ces derniers par des techniques de manipulation des ondes radios.

Le non-respect des caractéristiques de confidentialité, d'authentification et d'authenticité conduit ici également à la possibilité de créer, supprimer et modifier les données AIS.

2.3.2.2/ LES ATTAQUES SUR LE SYSTÈME AIS

Dans la littérature [10, 37], on retrouve des attaques communes avec celles identifiées sur le protocole ADS-B. La particularité est ici la présence d'attaques visant, pour un attaquant, à se faire passer pour des éléments de l'architecture de l'AIS, comme des bouée d'aide à la navigation ou des balises de détresse. La faisabilité de certaines attaques est, par exemple, démontrée par une expérimentation [10] effectuée directement au niveau des ondes à l'aide d'une radio logicielle classique. Dans cette expérimentation, le but est de déclencher une alerte du système d'évitement des collisions par la création d'un bateau fantôme en émettant un signal AIS factice.

Les différentes attaques recensées sont les suivantes :

— ***Collision Prevention Assist Spoofing*** : Le système d'évitement des collisions (*Collision Prevention Assist*) sur les navires se base sur les signaux AIS émis par les bateaux des environs proches afin de déterminer l'imminence des collisions. Lorsque que cette imminence passe un certain seuil le système émet une alerte. Le but de cette attaque est de créer un bateau fantôme afin de déclencher le système d'évitement des collisions d'un navire voisin.

— ***AIS-SART Spoofing*** : Le but ici est de leurrer les bateaux ciblés dans une zone voulue grâce à la génération de message en utilisant l'identité d'une balise de détresse.

— **Faux bulletin météo** : Les autorités peuvent utiliser le système AIS pour informer des conditions météo. Ici le but est d'usurper l'identité des autorités pour émettre de fausses informations concernant les conditions de navigation.

— ***Availability disruption*** : En fonction de l'attaquant, on identifie trois manières d'empêcher les communications AIS d'un bateau. La première manière consiste

à simplement éteindre le transpondeur, cette technique est notamment utilisée par les équipages souhaitant se camoufler. La deuxième technique se base sur l'usurpation de l'identité d'une autorité pour ordonner un changement de fréquence à un navire (*frequency hopping*), cette technique est particulièrement efficace sur les transpondeurs de classe B car ces derniers n'offrent pas la possibilité de modifier manuellement la fréquence d'émission sans en avoir reçu l'ordre par une autorité. Enfin la troisième technique est le déni de service par envoi massif de requêtes aux stations. Elle utilise une fois de plus l'usurpation de l'identité d'une autorité.

— ***Ship Spoofing*** : Cette attaque fait référence à la création d'un bateau en usurpant une identité existante.

— ***Aid to Navigation spoofing*** : Cette attaque consiste à créer de faux messages en usurpant l'identité des systèmes d'aide à la navigation (AtoN) pour pousser les navires à effectuer de fausses manœvres.

— ***AIS Hijacking*** : Cette attaque consiste à modifier des messages AIVDM par l'envoi de signaux plus puissants. Elle désigne ainsi toute attaque visant à modifier le contenu des messages.

Ces attaques peuvent, comme celles identifiées dans le domaine aérien, être classées en tant que FDIA. On note toutefois une certaine confusion entre les différentes attaques présentes au sein des deux domaines. Par exemple, on remarque que certaines attaques listées pour l'ADS-B utilisent le même procédé que celles listées pour l'AIS mais sont identifiées sous un nom différent, comme *Ghost Aircraft Injection* et *Ship Spoofing* qui décrivent une attaque identique. À l'inverse, certaines attaques portent le même nom mais utilisent un procédé différent, comme *Aircraft Spoofing* et *Ship Spoofing*. De plus, certaines attaques sont présentes dans un seul domaine alors qu'elles sont parfaitement transposables à l'autre, comme la *Virtual Trajectory Modification* identifiée seulement côté aérien. Enfin, il semble nécessaire de faire une distinction entre certains types d'attaque. En effet, en prenant l'exemple du *Collision Prevention Assist Spoofing*, on constate finalement qu'il se rapproche d'un *Ship Spoofing* où l'attention est portée d'avantage vers le calcul de la trajectoire que sur le choix de l'identité, le *Ship Spoofing* étant lui même une *Ghost Aircraft Injection* qui utilise une identité bien spécifique.

2.3.3/ SYNTHÈSE

Les technologies ADS-B et AIS présentent les mêmes vulnérabilités au niveau de leur sécurité. L'absence d'authentification, de confidentialité et d'authenticité garantie des messages envoyés via ces systèmes de communication permettent de la réalisation d'opérations malicieuses sur les données qui transitent. La combinaison de ces

opérations (ajout, suppression et modification de messages) permet la réalisation d'attaques nommées FDIA pour *False Data Injection Attacks*.

Dans l'optique de proposer une approche qui soit générique aux deux domaines d'application, il est essentiel de formaliser les différentes attaques listées dans les sous-sections précédentes. En outre, et cela est l'objet du chapitre suivant, il convient de donner une définition multi-domaine des FDIA en s'appuyant sur les similarités qu'elles présentent au travers des deux domaines aérien et maritime, ainsi que des autres domaines dans lesquelles elles sont identifiées dans la littérature scientifique.

3

ÉTAT DE L'ART

La compréhension des travaux de thèse requiert l'introduction et la définition de plusieurs notions. Ce chapitre répond à ce prérequis en présentant l'état de l'art des travaux sur lesquels s'appuie la thèse. Pour commencer, la première section propose une définition de ce que l'on trouve dans la littérature sous l'appellation *False Data Injection Attack*. La deuxième section liste les différentes recherches effectuées autour des approches de test et de simulation pour les domaines aérien et maritime en lien avec les protocoles ADS-B et AIS respectivement. Enfin, en vue de la présentation d'un ensemble de langages dédiés (DSL) (voir travaux présentés dans le chapitre 6), la dernière section introduit le concept de DSL et passe en revue différents langages utilisés par la scénarisation de test.

3.1/ ATTAQUE PAR INJECTION DE FAUSSES DONNÉES (FDIA)

On trouve, au sein de plusieurs domaines Métier, la présence de cyberattaques qui présentent des caractéristiques communes de type FDIA, sans nécessairement être identifiées sous la même appellation. En parallèle, le terme FDIA est utilisé dans différents domaines pour définir des attaques similaires, sans toutefois répondre à une définition générale, c'est-à-dire qui ne soit pas spécifique à un domaine précis.

L'objectif de cette section est de proposer une définition multi-domaine des FDIA. Pour cela, une première section retrace l'historique du terme à travers divers travaux scientifiques. La deuxième section fait ensuite l'analogie entre tous les domaines identifiés afin de proposer une définition générique des FDIA. Enfin, la dernière section propose une taxonomie des différentes FDIA couvrant les domaines aérien et maritime.

3.1.1/ HISTORIQUE DES FDIA

L'appellation *False Data Injection Attacks*, ou FDIA, a initialement été introduite dans le domaine des réseaux de capteurs sans fil. Un tel réseau est composé d'un ensemble de capteurs qui transmettent des rapports de suivi de données à une ou plusieurs stations de collecte. Ces stations traitent les rapports afin d'établir une estimation de l'état global du système sous surveillance. Dans un article [66] de 2004, une FDIA est décrite à travers un scénario type, sans toutefois être réellement définie. Dans un réseau de capteur sans fil, ce scénario consiste pour un attaquant, dans un premier temps, à pénétrer le réseau en compromettant un ou plusieurs nœuds pour ensuite produire et envoyer des faux rapports vers les stations. Cela peut, par exemple, mener à la production de fausses alarmes, au gaspillage de ressources réseau ou même à un endommagement physique du matériel.

Après l'apparition du terme *False Data Injection Attacks* dans le domaine des réseaux de capteurs, il réapparaît dans divers articles [126, 80, 127] centrés sur l'amélioration de la résistance à ces attaques. Toutefois, le domaine cible reste le même et les auteurs ne fournissent pas de définition plus précise.

En 2009, Liu et al. publient un article [64] qui introduit de manière formelle les FDIA dans le secteur énergétique, notamment dans le domaine des *Electric Power Grids*, connus désormais sous l'appellation de *smart grids*. L'article est cité des milliers de fois et popularise le terme dans ce domaine. La figure 3.1 représente un graphique du nombre de résultats annuels trouvés pour le terme "*False Data Injection Attack*" avec le moteur de recherche Google Scholar. On remarque en effet une augmentation significative du nombre d'occurrences du terme à partir de 2009.

Dans les *smart grids*, des compteurs relèvent la consommation électrique à un instant t et à un endroit donné sur le réseau d'acheminement de l'énergie électrique. Ces mesures sont ensuite transmises à des centres de contrôle qui produisent alors une estimation de l'état du système, de sorte à ajuster précisément la production de l'électricité en fonction de la demande. Le descriptif des FDIA proposé par Liu et al. se base sur leur réalisation d'un point de vue technique. Celle-ci implique de compromettre un certain nombre de compteurs afin de remplacer leurs mesures originales par des données malicieuses. La conséquence de cette pratique est l'altération de l'estimation de l'état du système produite par les centres de contrôles qui traitent les mesures compromises, conduisant à un déséquilibre entre production et consommation. Une étude montre que ce type d'attaque peut, malgré de nombreux contrôles d'intégrité, entraîner des coupures électriques à grande échelle ou des perturbations des marchés de l'électricité [120].

Mo et al. proposent un modèle de FDIA plus générique dans un article [76] de 2009. En effet, les auteurs étendent aux systèmes cyber-physiques cette classe d'attaque qui se limitait alors à ce qu'ils définissent comme "*des système statiques et reposant*

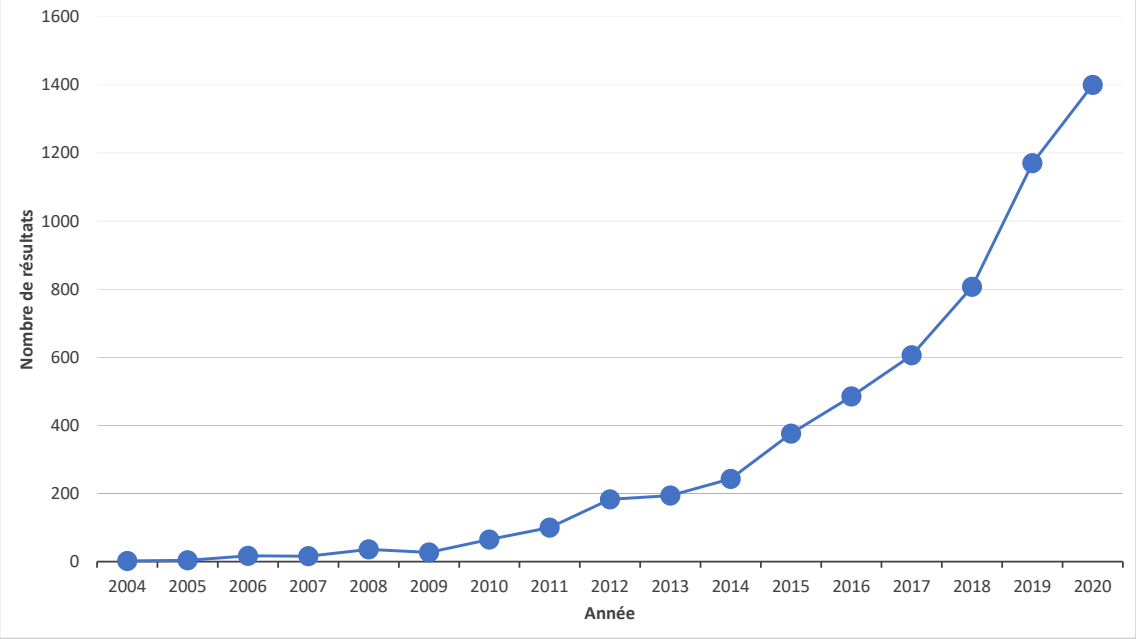


FIGURE 3.1 – Résultats annuels de Google Scholar pour le terme *False Data Injection Attacks*.

exclusivement sur des mesures de capteurs de courant". Une FDIA est décrit comme la déstabilisation d'un système par la compromission d'un sous-ensemble de capteurs et l'envoi de fausses mesures aux unités d'estimation de l'état d'un système. Depuis cette généralisation des FDIA, on trouve très sporadiquement des articles qui évoquent le terme dans des domaines précis, autres que celui des réseaux de capteurs sans fil ou celui des *smart grids*. On peut par exemple noter quelques travaux utilisant le terme dans le domaine aérien [100, 104], ou même dans le domaine du tatouage numérique [109], mais cela reste marginal. À l'inverse on peut trouver en 2015 un article [90] de Ray et al. faisant référence à ce type d'attaque mais sans l'associer directement au terme FDIA. Plus globalement, on note, depuis environ 2013, une utilisation plus fréquente du terme FDIA en tant qu'attaque ciblant de manière plus générique les systèmes cyber-physiques, comme peuvent en témoigner de nombreux articles [38, 61, 122].

On trouve dans les travaux les plus récents, une volonté de généralisation encore plus forte. C'est par exemple le cas d'un article [3] qui propose d'élargir le terme FDIA à des domaines reposant sur des données dites "non structurées", comme les images. Par opposition, une donnée structurée est organisée via des structures de données prédéfinies, typiquement stockées dans des tableaux. Pour ces travaux de thèse, on se cantonne à l'utilisation de données structurées comme le sont les messages ADS-B ou AIVDM dans les domaines aérien et maritime. En revanche, on note l'introduction par Sedjelmaci et al. du terme "consensus". Les auteurs abordent les FDIA comme des attaques qui ciblent des consensus distribués entre les différentes unités qui estiment

l'état d'un système [49].

Enfin, le caractère discret et subtile de ce type d'attaque est souvent mis en avant [62, 25]. L'attaquant est généralement décrit comme voulant retarder le moment où son attaque est détectée, contrairement à des attaques plus brutales comme le sont les dénis de service. Cette discrétion requiert de fait une connaissance profonde du système cible comme le suggèrent Mo et al. [76]. Au regard de tous ces éléments, il est possible de proposer une définition multi-domaine des FDIA.

3.1.2/ UNE DÉFINITION MULTI-DOMAINE DES FDIA

À partir de l'historique de l'utilisation du terme FDIA à travers différents domaines, il est possible de dégager quelques tendances fortes :

- Une FDIA a lieu avec la compromission d'un sous-ensemble de capteurs, (ou compteurs dans le cas des *smart grids*).
- Une FDIA consiste à fausser les données émises par ces capteurs compromis.
- Les domaines ciblés disposent d'unités d'estimation de l'état du système (*State Estimators*) qui se nourrissent des données récoltées par les capteurs.
- Le but d'une FDIA est finalement d'altérer l'estimation de l'état du système. Cette altération peut servir divers objectifs précis et liés au domaine.
- Les FDIA ont vocation à être discrètes impliquant une connaissance profonde du système.

Ces seules tendances ne sont toutefois pas suffisantes pour donner une définition multi-domaine englobante car on ne dispose pas d'un vocabulaire commun à tous les domaines potentiellement concernés par les FDIA. L'objectif premier de cette section étant d'inclure les domaines maritime et aérien à la définition de cette classe d'attaque, prenons l'exemple de ce dernier. C'est le transpondeur de l'avion qui transmet directement les données récoltés par les systèmes de bords, est-il alors correct de le placer sous le terme "capteur" ? La définition fournie par Mo et al., limitée aux systèmes cyber-physiques, peut s'appliquer aux domaines aérien et maritime présentés en chapitre 2 puisque ceux-ci comportent des systèmes de ce type. Toutefois, cette limitation aux systèmes mesurant des grandeurs physiques n'apparaît pas comme nécessaire puisqu'il est tout à fait concevable que des FDIA puissent cibler des systèmes purement logiques, et il serait précipité d'exclure ce type de systèmes de notre définition.

Que l'on parle de systèmes cyber-physiques, de *smart grids*, ou de transpondeurs et de stations au sol, leur principal point commun est avant tout de s'organiser en réseaux composés de nœuds. Certains de ces nœuds étant des capteurs, d'autres des unités de collecte et d'estimation, d'autres des compteurs électriques, etc. À partir de ce constat, voici la définition proposée pour les FDIA :

Une attaque par injection de fausses données (FDIA) est une combinaison d'opérations basiques illégales (suppression, modification et création de messages) altérant le consensus partagé par les nœuds d'un réseau, tout en préservant la cohérence du flux de communication.

Dans cette définition, le terme "message" fait référence à un envoi de données structurées, tandis que le terme "consensus" fait référence au traitement des données collectées, e.g., à l'estimation de l'état du système d'énergie dans les *smart grids* ou la Situation Aérienne Générale (SAG) dans le domaine aérien. Ce sont des "instantanés" de ce que le système surveille à un instant t. La mise en action d'une FDIA, peu importe le domaine, vise toujours à altérer ce consensus avec un objectif final bien précis : fausse alarme, distraction des opérateurs, invalidation des données, etc.

3.1.3/ UNIFORMISATION DES ATTAQUES DES DOMAINES AÉRIEN ET MARITIME

Pour la suite des travaux de thèse, il est important de disposer d'une taxonomie des attaques que l'on souhaite reproduire dans notre approche de test. Cette approche se voulant générique, la taxonomie doit être au moins commune aux deux domaines d'application, à savoir les domaines aérien et maritime. On remarque que certaines attaques listées pour l'ADS-B (voir section 2.3.1.2 p. 34) utilisent le même procédé que celles listées pour l'AIS (voir section 2.3.1.2) mais sont identifiées sous un nom différent, comme *Ghost Aircraft Injection* et *Ship Spoofing* qui décrivent une attaque identique. À l'inverse, certaines attaques portent le même nom mais utilisent un procédé différent, comme *Aircraft Spoofing* et *Ship Spoofing*. De plus, certaines attaques sont présentes dans un seul domaine alors qu'elle sont parfaitement transposables à l'autre, comme la *Virtual Trajectory Modification* identifiée seulement côté aérien. Enfin, il semble nécessaire de faire une distinction entre certains types d'attaque. En effet, en prenant l'exemple du *Collision Prevention Assist Spoofing*, on constate finalement qu'il se rapproche d'un *Ship Spoofing* où l'attention est portée davantage vers le calcul de la trajectoire que sur le choix de l'identité. Le *Ship Spoofing* étant lui-même une *Ghost Aircraft Injection* qui utilise une identité bien spécifique.

L'objectif de cette section est d'uniformiser la nomenclature de ces attaques en faisant la distinction entre les procédés qu'elles utilisent et leurs objectifs. Pour cela, on utilise un découpage à trois niveaux :

— **Les opérations** : Elles constituent les actions de bases à effectuer pour la réalisation d'une attaque. Listées en introduction de la section 2.3 p. 32, il s'agit de la création, la suppression et la modification de messages.

— **Les schémas** : Ils se rapprochent de ce qui a précédemment été listé comme étant des attaques, ce sont des combinaisons d'opérations menant à une altération de la

situation estimée par les systèmes de surveillance. Ils comprennent les différentes créations de nouveaux objets dans le système surveillé comme : *Aid to Navigation spoofing*, *Ship Spoofing*, ou encore *Ghost Aircraft Injection*.

— **Les scénarios** : Ils se basent sur une combinaison de schémas afin d'atteindre un objectif précis. Selon cette classification, certaines attaques évoquées dans les sections précédentes sont des scénarios. Par exemple, le *Collision Prevention Assist Spoofing Spoofing* est un scénario utilisant un seul schéma : la création d'un nouvel objet.

De ce découpage à trois niveaux se dégage un processus de conception et de réalisation des FDIA organisé en pyramide inversée : une infinité de scénarios, dont le nombre n'a de limite qu'à travers l'inventivité des personnes qui les imaginent, sont construits à partir d'un ensemble commun fini et réduit de schémas, eux-mêmes contruits à partir de seulement trois opérations de base. Ce découpage permet de ne pas avoir à formaliser l'infinité de scénarios possibles. Ainsi, le tableau 3.1 donne la liste des schémas utilisés par différents scénarios issus de la littérature, ainsi que les opérations (+ = création, - = suppression, ± = modification) requises pour les mener à bien.

Schéma	Opérations	Domaines	Description
Altération	±	Les deux	Modification du contenu des messages.
Injection d'appareil	+	Les deux	Création d'un nouvel appareil (avion ou bateau) à partir de zéro.
Saturation	+	Les deux	Création multiple d'appareils à partir de zéro.
Injection d'architecture	+	Maritime	Création d'un élément de l'architecture.
Disparition d'appareil	-	Les deux	Suppression des messages d'un appareil spécifique.
Disparition d'architecture	-	Maritime	Suppression des messages d'un élément spécifique de l'architecture
Modification de trajectoire	+ / - / ±	Les deux	Modification de la trajectoire d'un appareil.

TABLE 3.1 – Liste des différents schémas dans les domaines aérien et maritime.

À partir d'un schéma ou d'une combinaison de schémas, il est possible de créer n'importe quel scénario d'attaque. Par exemple, un scénario qui vise à déclencher le système d'évitement des collisions combine deux modifications de trajectoire ciblant chacune un appareil. L'avantage de cette taxonomie est de pouvoir proposer une méthode de scénarisation des attaques qui se base sur un nombre limité de schémas. Le chapitre 6

présente cette méthode de scénarisation qui s'appuie sur trois langages dédiés.

3.2/ LA GÉNÉRATION DE DONNÉES DE TEST SYNTHÉTIQUES

L'un des principaux axes de recherche de cette thèse concerne la génération de données métier altérées, principalement pour le test des différents systèmes basés sur les protocoles ADS-B et AIS. Les données initiales d'un scénario de test prennent la forme d'un enregistrement, c'est-à-dire un ensemble de messages de surveillance issus du trafic réel. Afin de répondre aux besoins du scénario de test, certains messages sont modifiés ou supprimés, tandis que d'autres sont créés de toutes pièces puis injectés parmi les messages authentiques. Les modifications dictées par un scénario de test sur un enregistrement authentique reproduisent ainsi l'impact qu'aurait la FDIA correspondante au niveau du flux de données (ajouts, modifications et suppressions de messages) et donc, par extension, le désordre créé au niveau du système de contrôle utilisant ces données. Pour cette raison, l'approche de test issue de cette thèse peut être classée comme approche de "test par injection de fausses données", ou "*false data injection testing*".

Pourtant, ce type d'approches – à différencier des approches de "*fault injection testing*" qui consistent à injecter des bogues dans des composants matériels ou logiciels – est quasiment absent de la littérature. Une approche a été identifiée [118] traitant de la calibration d'un ensemble de capteurs TGS2602-B00 [1] pour minimiser les imprécisions dans les mesures, à l'aide de filtre de Kalman. Afin de valider leurs travaux, les auteurs définissent des "*false data injection scenarios*", consistant effectivement à modifier la valeur de certaines mesures au milieu d'un ensemble de mesures authentiques. Cette approche correspond à ce que l'on a identifié, en début de section, comme étant du test par injection de fausses données. D'ailleurs, dans leur conclusion, les auteurs utilisent directement le terme de "*false data injection testing*". À notre connaissance, c'est la seule instance de cette terminologie dans la littérature. Effectivement, nos recherches bibliographiques ne faisant pas ressortir de travaux similaires aux nôtres, nous ne sommes pas en mesure de produire un état de l'art classique, où l'on défini les limites d'un certain type d'approche. Dans cette section le but sera plutôt de lister des travaux connexes aux nôtres afin de tirer des enseignements sur des techniques permettant d'atteindre notre objectif de recherche.

Dans nos travaux, le processus de test n'implique pas la mise en place réelle de FDIA mais la modification d'enregistrements, cette approche partage certaines similarités avec les approches de test basées sur la simulation, sans toutefois avoir la même complexité. Ce type d'approches étant davantage représenté dans la littérature, cette

1. Ce modèle de capteur est utilisé pour mesurer la qualité de l'air

section possède deux objectifs : le premier est de lister les travaux proposant des techniques de test par simulation, tous domaines confondus. Le deuxième objectif est de réaliser un tour d'horizon des approches de test par simulation dans le domaine aérien. Une section est dédiée à chacun de ces objectifs, tandis que la dernière section dresse le bilan des techniques pouvant contribuer à la réalisation de notre objectif de recherche.

3.2.1/ LE TEST PAR SIMULATION À TRAVERS PLUSIEURS DOMAINES

Le développement des logiciels du véhicule autonome constitue un domaine actif pour les tests basés sur la simulation. Ces tests consistent à simuler l'environnement dans lequel évolue le véhicule, et à y reproduire des situations afin d'observer la réaction du système de conduite autonome. Ces situations correspondent en fait à des cas de tests issus de scénarios possibles, comme illustré par Khastgir et al [53]. En effet, les auteurs utilisent un simulateur de conduite pour produire massivement des données de test pour le système de freinage d'urgence. Le cas de test initial est une voiture qui suit un chemin en mode de conduite autonome. Afin de tester le système de freinage d'urgence, un véhicule est injecté dans la simulation pour venir couper la trajectoire du véhicule sous test selon différents paramètres (angle d'intersection, vitesse, etc.). Les valeurs de ces paramètres sont ensuite échantillonnées aléatoirement par un logiciel dédié. Il est possible de faire un léger parallèle avec ce qu'on décrit comme du test par injection de fausses données. Toutefois, dans l'article cité précédemment, c'est davantage la technique de massification qui attire notre attention, et qui apparaît utilisable dans nos travaux de recherche.

Si on remarque que le véhicule autonome est très présent dans les approches de test basées sur la simulation, l'apparition des systèmes de navigation autonome introduit, dans le domaine maritime, de nouveaux cas d'utilisation de cette technique de test. Ainsi, dans le domaine maritime, on trouve un exemple de cas d'utilisation de données AIS pour la simulation d'environnement de test. Pedersen et al. [84] présentent le prototype d'un système de test basé sur la simulation afin de tester la navigation autonome. Les capteurs de navigation sont reliés à un environnement simulé et le bateau qu'ils dirigent est reproduit dans cet environnement. Ce dernier modélise les vents, les vagues, les courants marins, la position géographique, ainsi que le trafic aux alentours à partir d'historiques de données AIS (les bateaux sont ainsi simulés grâce au rejeu de données AIS antérieures). Dans un mode de simulation plus interactif, ces mêmes données AIS sont utilisées pour créer des bateaux eux-mêmes pilotés.

Comme évoqué en section 1.2.2 p. 13, les approches de détection d'anomalies manquent souvent de données de test, ou d'entraînement, quand il est question de modèle de *Machine Learning*. Schneider et al. [99] constatent effectivement le manque de jeux

de données (labellisés ou non) contenant des anomalies. Ils remarquent que pour la validation de certains travaux [60], les chercheurs ont généré leurs jeux de données avec des scripts développés en interne. Schneider et al. proposent de pallier le problème en utilisant des générateurs de données tels que ceux proposés pour les systèmes industriels de type cyber-physiques ou SCADA (système de contrôle et d'acquisition de données). Leur approche pour générer des données anormales se base tout d'abord sur la simulation de l'environnement. Premièrement, un simulateur est utilisé pour générer des données métiers réalistes. Ce simulateur s'appuie sur des modèles physiques et mathématiques propres au domaine. Le processus physique global est modélisé et ses entrées/sorties, principalement des capteurs, sont liés à des périphériques réseaux virtuels connectés dans une topologie bus. Chacun de ces périphériques virtuels est un nœud au sein de la simulation d'un réseau réalisée avec le simulateur NS-3 [2]. Pour scénariser la simulation, le langage de modélisation Modelica [3] est souvent utilisé. À partir de cette simulation, les auteurs sont en mesure d'effectuer plusieurs attaques comme la manipulation des données des capteurs, tandis que les données sont enregistrées pendant la simulation pour produire les jeux de test. Afin de valider leur approche, les auteurs simulent un système de chaudière de récupération sur lequel ils appliquent des attaques et entraînent, grâce aux données générées, des composants de détection à base de *Deep Learning*.

Cette section a présenté trois approches de test par simulation au sein de trois domaines différents. La section suivante se consacre à ce type d'approche dans le domaine d'application principal de cette thèse : le domaine aérien.

3.2.2/ LES APPROCHES DE TEST PAR SIMULATION DANS LE DOMAINE AÉRIEN

Un *framework* de test proposé par Barreto et al. [11] permet la simulation intégrale de l'environnement du trafic aérien : les avions, les relais radio, les infrastructures réseau, etc. Cette simulation est faite à partir de deux produits informatiques standards : MÄK VR-Forces [4] pour la simulation de la partie physique, et EXata Cyber [5] pour la simulation des réseaux de communication. Les cyberattaques sont effectuées dans l'environnement en utilisant une librairie d'attaque intégrée à EXata Cyber. Cette librairie proposant seulement la réalisation d'attaques génériques telles que du déni de service ou du brouillage, les auteurs ont également développé leur propre générateur d'attaques. Le but de l'expérimentation est d'évaluer l'impact des attaques sur chacun des composants de l'environnement simulé. L'approche peut apporter des informations importantes sur la manière dont ces composants réagissent aux FDIA. Toutefois, la mise en œuvre de

2. https://www.nsnam.org/ [Dernière visite : décembre 2020]
3. https://www.modelica.org/ [Dernière visite : décembre 2020]
4. https://www.mak.com/products/simulate/vr-forces [Dernière visite : décembre 2020]
5. https://www.scalable-networks.com/exatacyber [Dernière visite : décembre 2020]

tous les comportements réseau d'un scénario est très complexe, l'obtention de licence pour les deux librairies standards représente un coût certain, et l'approche ne permet pas l'exécution des attaques simulées sur des systèmes réels. De plus, aucune information n'est fournie sur le générateur d'attaques ou sur la nature des attaques qu'il est en mesure de générer. En effet, les seules attaques expérimentées dans l'article sont des dénis de service.

D'une manière similaire, Manesh et al. [68] étudient les effets de l'injection de faux messages dans le trafic aérien de manière à simuler une situation de rencontre entre deux avions. Une situation de rencontre est une violation de la séparation minimale qui nécessite, en conditions réelles, une intervention afin de rétablir cette séparation. Pour reproduire cette situation à risque, les auteurs utilisent une technique de simulation dite *Hardware-in-the-Loop*. Cette simulation implique un système réel de pilotage automatique embarqué sur un drone et une station de contrôle portable. Le but est d'observer le comportement de l'affichage du trafic destiné au pilote lorsqu'un avion fictif violant la séparation minimale est simulé par l'injection de messages ADS-B. Les résultats montrent qu'une alerte est déclenchée avec une mise en évidence en rouge de l'avion injecté sur l'affichage du trafic, ce qui signifie effectivement une violation de la séparation minimale. Toutefois, les auteurs ne détaillent pas la manière dont les faux messages sont créés.

Paielli [82] présente une méthode de test pour les logiciels de résolution des conflits dans le trafic aérien, comme par exemple le logiciel TSAFE (*Tactical Separation-Assisted Flight Environment*). Cette méthode permet de générer de manière automatique des situations de rencontre entre avions. Les algorithmes de résolution de conflits proposent des manœuvres à effectuer (généralement par des changements d'altitude et de direction) quand ils prédisent une violation de la séparation d'ici un certain laps de temps (dans le cas de TSAFE ce laps est de deux minutes). La génération de tests s'appuie sur un langage de script de trajectoire. Les utilisateurs spécifient plusieurs avions et, pour chacun d'eux, définissent une trajectoire composée de plusieurs segments (lignes droites, virages, montées, etc.). Il est également possible de générer des situations de rencontre en spécifiant des paramètres additionnels tels que l'angle du croisement ou la séparation minimale souhaitée. La méthode permet de tester une grande variété de type de conflits en comparant la réaction des algorithmes avec leur réaction attendue. Toutefois, cette méthode n'est pas capable de générer des données ADS-B et ne permet pas de tester la réaction des systèmes de surveillances face aux FDIA.

3.2.3/ SYNTHÈSE

Les exemples vus dans cette section mettent en avant le caractère actif des travaux sur les tests par simulation dans le domaine des systèmes de contrôle et de surveillance, en particulier du fait de l'augmentation de la complexité de ces systèmes. Chacun des exemples cité précédemment utilise des simulateurs existants qui proposent un niveau de réalisme très élevé. Dans le cas du domaine maritime, le simulateur utilisé par Pedersen et al. modélise l'environnement jusqu'aux vagues de l'océan, ce qui s'avère nécessaire au vu de leur objectif de test. Dans le cas de la génération de données pour les systèmes cyber-physiques de Schneider et al., le simulateur réseau utilisé permet de simuler les latences du réseau. Enfin, les travaux de Barreto et al. illustrent également ce niveau de détail avec la simulation de l'architecture complète des avions au réseau. Reprenons l'objectif de recherche de la thèse : **RO : Dans quelle mesure une approche générique à plusieurs domaines peut-elle permettre, grâce à l'utilisation d'un ensemble de langages dédiés, la génération automatique de données altérées selon des scénarios conçus par les experts du domaine ?** On constate qu'aucune des approches ne répond à cet objectif. Si elles permettent pour la plupart de générer des données, altérées ou non, aucune n'est réellement générique hormis celle proposée dans les travaux de Schneider et al. qui ciblent les systèmes cyber-physiques, se limitant toutefois à des topologie réseau en bus. Le manque de généricité de ces approches est principalement dû à cet aspect simulation contextualisé dans le souci du détail et nécessairement très spécifique à un domaine. Si l'approche de cette thèse partage des points communs avec les approches de test par simulation, elle ne peut pas être qualifiée de la même manière.

Le terme "test par injection de fausses données", bien que très peu rencontré dans la littérature, convient donc mieux à ces travaux de thèse car il s'agit pour nous de modifier des fichiers de données réelles enregistrées préalablement plutôt que de créer des données de toute pièce avec un simulateur.

Malgré la nature différente du test par injection de fausses données et du test par simulation, certaines techniques utilisées au sein des travaux listés dans cette section peuvent contribuer à atteindre l'objectif de recherche de la thèse. Premièrement, dans les travaux de Khastgir et al. (test du freinage d'urgence, voir section 3.2.1 p. 46), la technique de massification des cas de test peut servir d'appui pour répondre à la question de recherche **RQ-2 : Dans quelle mesure est-il possible de garantir le réalisme de données simulant des FDIA, tout en les générant de manière automatique et massive ?** En effet, l'utilisation d'ensemble de valeurs plutôt que de valeurs discrètes est un point abordé dans les travaux de cette thèse (voir section 6.2.13 p. 111). Dans les travaux de Pedersen et al. (test des systèmes de navigation automatique, voir section 3.2.1 p. 46), le rejeu de données AIS réelles est une technique réutilisée dans nos

travaux et permet de garantir le réalisme de la donnée. Enfin, les générations de données proposées par Schneider et al. (test des système cyber-physique, voir section 3.2.1 p. 46) et par Paielli (test du logiciel TSAFE, voir section précédente) s'appuient sur l'utilisation de langages pour la conception des scénarios de test. Cette technique est directement en lien avec la question de recherche **RQ-1 : Dans quelle mesure un ensemble de langages dédiés assure-t-il la couverture de la taxonomie des FDIA ?** L'utilisation de langages pour la scénarisation des test constitue en effet une contribution principale de la thèse (voir chapitre 6) à travers la définition et l'utilisation de langages dédiés, ou *Domain Specific Language* (DSL).

3.3/ UTILISATION DE LANGAGES DÉDIÉS POUR LA SPÉCIFICATION DES TESTS

Les langages dédiés (DSL) répondent aux exigences d'un domaine précis. Ils sont opposés aux langages de programmation conventionnels qui traitent un ensemble de domaines. Une contribution principale de la thèse étant la production d'un ensemble de DSL, il convient de traiter le sujet des DSL dans cette section dédiée. Selon la question de recherche **RQ-1 : Dans quelle mesure un ensemble de langages dédiés assure-t-il la couverture de la taxonomie des FDIA ?**, il apparaît que l'objectif des DSL dont il est question dans ces travaux est de scénariser des injections de fausses données. Pour cette raison, cette section se concentre sur des DSL utilisés pour scénariser diverses situations, en lien ou non avec les domaines aérien et maritime. La première sous-section aborde la question du développement de DSL. La deuxième sous-section liste plusieurs DSL dont le but est la création de scénarios. La dernière sous-section dresse le bilan des travaux listés et des concepts et techniques sur lesquels s'appuient les travaux de thèse.

3.3.1/ LE DÉVELOPPEMENT D'UN DSL

D'une manière générale, les DSL sont des langages haut-niveau qui fournissent une couche d'abstraction et des notations adaptées à un domaine d'application cible [115]. Ils permettent un gain substantiel d'expressivité et d'utilisabilité comparés aux langages de programmation à usage général (aussi appelés GPL pour *General-purpose Programming Languages*) comme Java ou C, ou à des langages multi-domaine comme UML [75]. D'autre part, les DSL permettent de faire le pont entre les experts du domaine d'application et les développeurs au travers d'une terminologie et d'une sémantique unifiées. Il existe de nombreux exemples de DSL à succès dans une grande variété de domaines : la gestion de système de gestion de base de données avec le langage SQL, la mise en page des sites web avec HTML et CSS, l'automatisation des *builds*

dans le développement logiciel avec Make, la description de matériel avec VHDL, et bien d'autres.

Les DSL ne sont néanmoins pas dépourvus d'inconvénients, le principal étant le coût de développement élevé (implémentation, création d'une documentation, maintenance, etc.), en particulier pour les DSL externes, ceux qui n'ont aucun lien avec un langages de programmation existant. De plus, le développement d'un DSL en réponse à un problème n'apparaît que rarement comme une évidence. Afin d'appuyer les développeurs dans leur décision, Mernik et al. [75] ont formalisé le processus de développement d'un DSL selon sept activités séquentielles :

1. **La décision** vise à déterminer si la création d'un DSL est une solution adéquate au problème courant et si le coût de développement est justifié.
2. **L'analyse du domaine** a pour but de modéliser le domaine d'application, c'est-à-dire définir un vocabulaire qui représente les objets et concepts du domaine ainsi que les relations qui peuvent exister entre ces différentes entités.
3. **La conception** d'un langage consiste à définir sa grammaire et sa sémantique.
4. **L'implémentation** concerne le choix de l'approche la plus adaptée pour la mise en place technique d'un langage : langage interprété, compilé, traduit dans un autre langage, etc. Il est également question des technologies à utiliser.
5. **Le test** définit la phase d'évaluation du langage. Cette activité peut reposer sur la vérification de son bon fonctionnement ou l'estimation de ses bénéfices, par exemple le gain de productivité constaté.
6. **Le déploiement** inclut l'installation et les différents usages du DSL.
7. **La maintenance** précise comment étendre la grammaire du langage pour répondre à de nouvelles exigences.

Dans nos travaux de thèse, trois langages dédiés complémentaires ont été développés. Le chapitre 6 présente cet ensemble de DSL au travers de cinq des sept activités identifiées par Mernik et al. Les activités de déploiement et de maintenance ne sont volontairement pas abordées car elles font référence à des phases de développement trop avancées pour un contexte scientifique.

3.3.2/ LES LANGAGES DÉDIÉS AUX SCÉNARIOS DE TEST ET DE SIMULATION

L'utilisation de langages dédiés (ou DSL pour *Domain Specific Language*) pour la définition de cas de test est une pratique répandue. Par exemple, Cucumber [6] est un outil de test qui utilise le DSL Gherkin [7] pour la définition de cas de test au travers de règles

6. https://cucumber.io/ [Dernière visite : décembre 2020]
7. https://cucumber.io/docs/gherkin/ [Dernière visite : décembre 2020]

Given-When-Then. Silva et al. proposent le *Test Specification Language* [67] qui produit des cas de test écrits avec le langage Gherkin lorsqu'il est interprété. Ce processus consiste en une transformation du *Test Specification Language* vers un autre DSL (Gherkin) avec un niveau d'abstraction différent. Selon cette méthode, on peut dire qu'une spécification de test abstraite écrite en *Test Specification Language* est concrétisée en plusieurs cas de test. En effet, une spécification contient des paramètres dont la valeur est abstraite et qui doivent être concrétisés avant qu'elle puisse être convertie en cas de test. La concrétisation consiste à convertir un ensemble de valeurs, continues ou énumérées, en valeurs discrètes.

Une approche similaire est proposée dans le domaine du développement de système de conduite autonome. Menzel et al. [74] proposent d'utiliser trois niveaux d'abstraction pour la création de cas de test. Le premier, et le plus haut niveau d'abstraction, est proche de l'anglais naturel et vise à être utilisé par les experts du domaine pour définir des scénarios "à risque" de manière informelle. Le deuxième niveau d'abstraction formalise les termes utilisés dans le premier niveau. Par exemple, s'il est question de routes et de véhicules dans le premier niveau, alors leurs dimensions et leur type sont spécifiés dans le deuxième niveau. Un DSL est alors utilisé pour écrire les scénarios du deuxième niveau. D'une façon similaire aux spécifications abstraites que l'on retrouve dans le travail de Silva et al., un scénario du deuxième niveau n'utilise pas directement des valeurs discrètes, mais plutôt des ensembles de valeurs continues ou des listes. Le troisième niveau, et le niveau d'abstraction le plus bas, concrétise les scénarios du deuxième niveau. Le troisième niveau consiste en des cas de test concrets qui sont généralement écrits dans un format lisible par une machine, par exemple, en XML ou en JSON. En s'appuyant sur le travail de Menzel et al., Queiroz et al. [88] proposent un DSL, nommé *GeoScenario*, basé sur un langage issu d'OpenStreetMap similaire au XML. Ce DSL correspond au troisième niveau d'abstraction proposé par Menzel et al. Le but de ce langage est d'être traité par plusieurs simulateurs de système de conduite autonome afin de générer des données de test correspondant aux scénarios spécifiés. Toutefois, le DSL de troisième niveau proposé par Queiroz et al. est certes générique d'un simulateur à un autre, mais sa syntaxe proche du XML l'éloigne du langage naturel et le destine plutôt à être utilisé par les développeurs que par les experts qui conçoivent les scénarios. Sans langage de plus haut niveau pour le générer, il perd ainsi de son intérêt.

Dans le domaine du test de performance dans les réseaux, un DSL nommé coNCePTuaL a été développé [83]. Ce DSL permet la conception de cas de test de performance que les auteurs décrivent comme des "modèles" mais qui correspondent en fait à la définition de "scénarios" des travaux cités plus haut. En effet, le résultat de l'interprétation d'un programme écrit en coNCePTuaL est un ensemble de machines abstraites qui échangent des messages entre elles. Scénarios ou modèles, tous deux décrivent les comportements d'entités spécifiques au domaine. Les données correspondantes

sont finalement générées par des simulateurs capables d'interpréter ces scénarios ou modèles.

Il existe des DSL servant à décrire des scénarios liés au domaine aérien. Par exemple, Jafer et al. [48] proposent un DSL nommé ASDL (*Aviation Scenario Definition Language*) et permettant de scénariser des vols du décollage à l'atterrissage. Avec ce DSL, les auteurs cherchent à fournir un langage bien défini qui permettrait la réutilisation des méthodes de génération de scénarios entre différents simulateurs. Toutefois, les auteurs ne précisent pas explicitement ce qui est produit par l'interprétation d'un scénario. Pour le développement de leur DSL, les auteurs réalisent l'analyse du domaine via une ontologie avec l'outil Protégé [8], et la conception via un méta-modèle réalisé avec l'*Eclipse Modeling Framwork*[9] (EMF). Par la suite, le langage ASDL a été étendu [21] pour inclure des terminologie permettant de scénariser les communications entre les avions et les centres de contrôle régionaux. Le langage ASDL emprunte sa syntaxe au XML. Grâce à cela il a l'avantage d'être facile à interpréter (de nombreuses librairies existent pour lire le format XML) et d'offrir un grand pouvoir d'expression, mais cela est aussi au détriment de sa lisibilité.

Il existe également des DSL dans le domaine de la cyber-sécurité utilisé pour concevoir des scénarios d'attaque. Par exemple, le *Meta Attack Language* [50] est un DSL créé par Johnson et al. qui permet de créer des scénarios d'attaque. L'interprétation du *Meta Attack Language* produit des graphes d'attaques qui sont ensuite pris en entrée par des simulateurs d'attaques. La syntaxe de DSL, comme les précédents, est proche de Java avec des structures de classe et est plutôt destinée à des développeurs.

3.3.3/ SYNTHÈSE

Les différentes travaux abordés dans cette section offrent un point de vue global sur les approches de spécification des scénarios de test. Cette étude de la littérature permet de tirer deux enseignements majeurs vis-à-vis de notre objectif de recherche.

Le premier enseignement à tirer concerne la conversion d'ensemble de valeurs (continues ou énumérées) en valeurs discrètes au travers d'un processus de concrétisation, comme l'ont présentés Silva et al. ainsi que Menzel et al. Cette technique semble en effet être essentielle dans la mise en œuvre d'une capacité à massifier les données générées, ce qui nous ramène directement à la question de recherche **RQ-2 : Dans quelle mesure est-il possible de garantir le réalisme de données simulant des FDIA, tout en les générant de manière automatique et massive ?**. De plus, on y voit un lien clair avec les travaux de Pedersen et al., présentés en section 3.2.1 p. 46, dans

8. https://protege.stanford.edu/ [Dernière visite : Décembre 2020]
9. https://www.eclipse.org/modeling/emf/ [Dernière visite : décembre 2020]

lesquels la même technique de massification est utilisée.

A notre connaissance, les DSL proposant la modification de données réelles semblent globalement absents de la littérature. En effet, la plupart des DSL pour la scénarisation fonctionnent de la même manière : une fois interprétés, ils produisent des données de plus bas niveau, destinées à être prises en entrée par un simulateur ou générateur de données. Toutefois, aucune des approches étudiées ne propose de langage dont le niveau d'abstraction est supérieur, voire équivalent, au deuxième niveau d'abstraction identifié par Menzel et al. Ce niveau d'abstraction fait pourtant le pont entre les développeurs et les experts qui conçoivent les scénarios. Un DSL proposant ce niveau d'abstraction peut facilement être manipulé par un utilisateur aux compétences de programmation limitées tout en épargnant du travail aux développeurs chargés de générer les données d'entrée pour les générateurs. Finalement, le manque de généricité des DSL présents dans la littérature, couplé à un niveau d'abstraction souvent trop faible, nous a poussé à développer notre propre ensemble de DSL (qui sont présentés au chapitre 6).

Afin de répondre au mieux à notre objectif de recherche, les trois DSL proposés dans ces travaux de thèse se veulent d'un niveau d'abstraction proche du langage naturel, tout en proposant un mécanisme permettant la massification des données et un pouvoir d'expression assurant la couverture de la taxonomie des attaques.

II

CONTRIBUTIONS

4

DÉMARCHE GÉNÉRALE POUR LA GÉNÉRATION DE DONNÉES

Ce chapitre vise à présenter l'architecture générique sur laquelle s'appuient les contributions de cette thèse. Chaque contribution, qu'il s'agisse du langage dédié à la création des scénarios, de la génération des données de test, ou du module de sélection et de priorisation, suit une approche multi-domaine. Les domaines aérien et maritime, présentés dans le chapitre 2, sont utilisés pour mettre en évidence la généricité de l'approche et illustrent le processus de spécialisation à mettre en place pour passer d'un domaine à l'autre.

4.1/ ARCHITECTURE MULTI-DOMAINE

Considérons un réseau physique formé d'un ensemble d'objets connectés et de bases d'acquisition ou de décision. Des échanges, régis par un protocole de communication, ont lieu depuis les objets vers les bases. Le rôle d'une base consiste à réaliser des actions en fonction de l'information reçue. Dans ce contexte, nous nous proposons de tester le bon fonctionnement d'une telle base, cette dernière faisant alors office de Système Sous Test, ou *System Under Test* (SUT). Pour cela, l'approche multi-domaine au centre de cette thèse vise à générer des données anormales, mais réalistes : il s'agit de modifier la sémantique de l'information tout en respectant la syntaxe et les règles fixées par le protocole utilisé au sein d'un réseau. Les données anormales générées sont construites à partir d'un extrait de communication du réseau cible ayant été préalablement enregistrées. Cet extrait est ainsi modifié afin d'en produire une version falsifiée dont certaines données sont anormales. Un langage dédié, ou *Domaine Specific Language* (DSL) permet, en outre, de décrire finement la manière dont ces données sont générées.

Cette section pose les bases de l'architecture générique sur laquelle s'appuie l'approche générale étudiée dans cette thèse. Pour cela, une première sous-section donne les

définitions nécessaires à la bonne compréhension de l'approche tandis que la sous-section suivante présente le processus d'altération et d'injection des données. Ensuite, les exigences pour la mise en place de cette architecture sont listées. Finalement, la dernière sous-section décrit le processus de spécialisation permettant de transposer l'approche d'un domaine d'application à un autre.

4.1.1/ DÉFINITIONS DES CONCEPTS CLÉS

Avant de présenter en détail l'architecture et ses composants, il est nécessaire de définir certains termes utilisés dans le cadre de cette architecture, et plus généralement, dans l'ensemble des contributions de cette thèse :

- Un **objet connecté** est l'entité à la base des échanges dans le réseau. Leur rôle est de récupérer de l'information sur leur environnement ou sur eux-mêmes et de la diffuser au sein du réseau.
- Une **propriété** est un type d'information intrinsèque ou extrinsèque à l'objet. Les messages d'un enregistrement peuvent ainsi contenir plusieurs propriétés. On dit d'une propriété qu'elle est "statique" si sa valeur reste fixe au cours du temps (e.g., identifiant, nom, etc.), et au contraire "dynamique" si sa valeur évolue au cours du temps (e.g., position d'un objet en déplacement, température, etc.).
- Une **base** est un récepteur de message dans le réseau physique. Elle se caractérise généralement par une portée, permettant de déterminer les messages qu'elle reçoit ou non. On distingue deux types de base :
  - Les bases d'acquisition qui reçoivent l'information venant des objets et la relaient vers une d'autres bases.
  - Les bases de décision qui effectuent des actions en fonctions des informations reçues (e.g., émettre une alerte, transmettre un rapport, etc.).

  La capacité des bases d'acquisition à envoyer de l'information vers d'autres bases permet la modélisation d'architectures multi-niveaux. Il est alors possible de modéliser un réseau sur plusieurs niveaux, où n'importe quelle base peut potentiellement faire office de SUT. À noter finalement qu'une base d'acquisition se trouve toujours au niveau le plus bas.

  La figure 4.1 donne un exemple de réseau qu'il est ainsi possible de modéliser. Deux bases d'acquisition de niveau 1 reçoivent des informations provenant des objets à portée puis les relaient vers une base d'acquisition de niveau supérieur. La base de décision de niveau 2 a pour rôle d'analyser les données venant des bases d'acquisition et de transmettre un rapport à une autre base de décision de niveau supérieur. Dans l'exemple de la figure 4.1, la base de décision de niveau 3 se trouve au niveau le plus haut et fait donc fait office de SUT, mais il est parfaitement

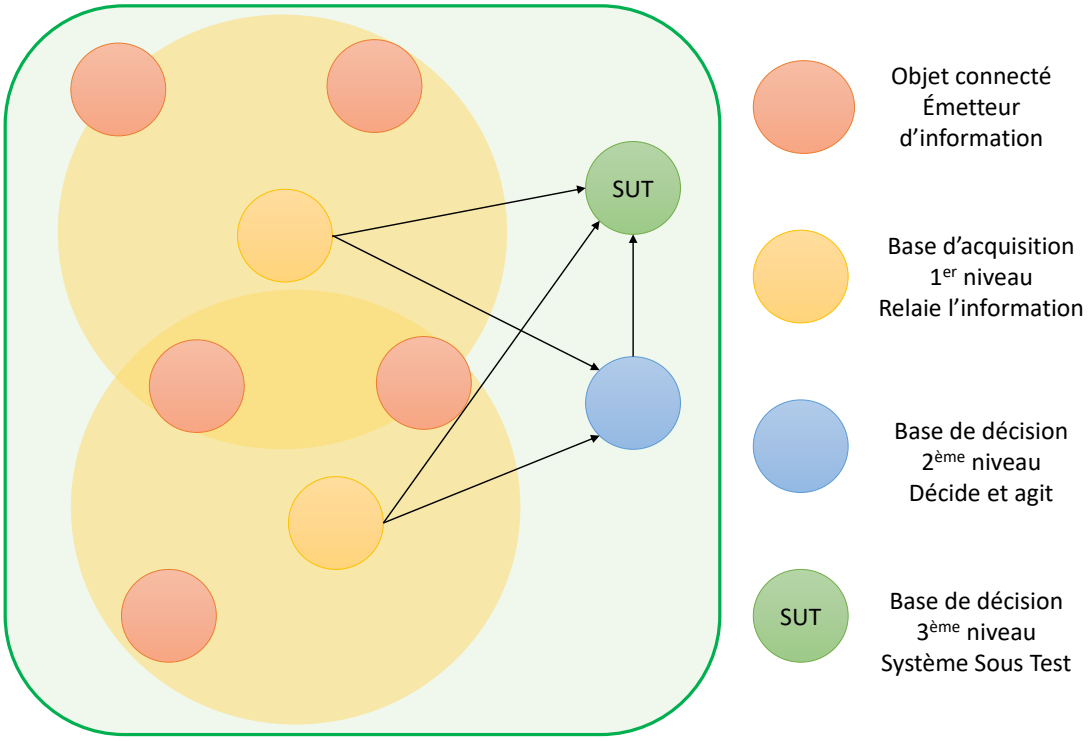


FIGURE 4.1 – Exemple d'un réseau multi-niveaux.

imaginable d'ajouter d'autres bases de niveaux supérieurs si cela est requis par le réseau à modéliser.

La nécessité de pouvoir modéliser des réseaux multi-niveaux se justifie par le comportement des bases dans les couches intermédiaires. En effet, pour émettre leurs propres messages, les bases peuvent dépendre des messages émis depuis les couches inférieures, en particulier lorsque leur rôle est d'agréger des données afin de transmettre des données unifiées. L'utilisation de plusieurs niveaux de base permet de simuler cette dépendance.

— Un **enregistrement** est un ensemble de messages émis par des objets connectés et récupérés grâce à l'écoute de leurs canaux de communication. Chaque message horodaté contient de l'information provenant d'un objet (intrinsèque) ou de son environnement (extrinsèque) ainsi que l'identité de celui-ci.

— Un **scénario** est une combinaison de modifications, créations et suppressions de messages, ciblant des objets au sein d'un enregistrement, sur une plage de temps définie manuellement ou via des événements. Un scénario reproduit une situation réelle ou altérée dans laquelle évolue les objets, cette situation vise à être confrontée à un SUT.

Tous ces éléments interagissent entre-eux dans un processus visant, à partir d'un

enregistrement réel, à altérer certaines données en vue de les injecter dans un SUT afin d'en observer la réaction.

4.1.2/ PROCESSUS D'INJECTION DE DONNÉES ALTÉRÉES

Ce processus d'injection de données altérées décrit dans la figure 4.2 peut être divisé en six modules :

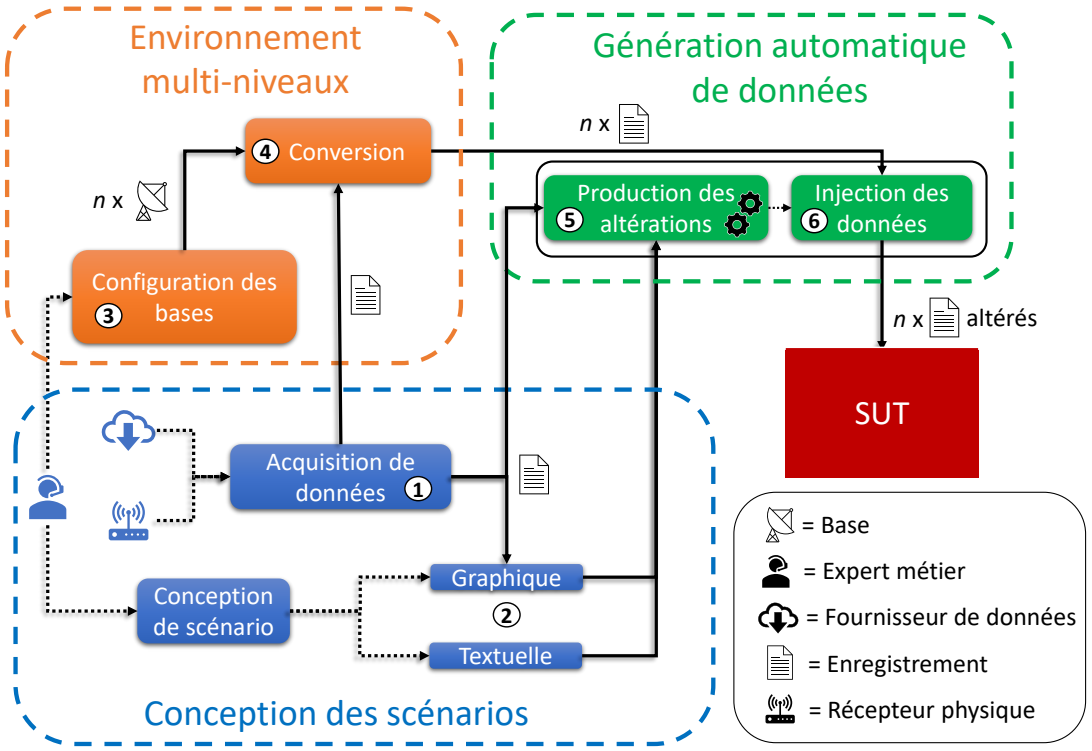


FIGURE 4.2 – Génération de données : processus générique

① **Acquisition de données**. Ce module a pour rôle la récupération de données nominales, i.e. exemptes de toutes falsifications. Selon le domaine d'application, les données peuvent être obtenues directement via un récepteur compatible avec la fréquence d'émission utilisée par le protocole cible, ou trouvées dans des bases de données en accès libre sur internet [1] [2].

② **Conception de scénario**. La conception de scénario est destinée à être réalisée par un expert du domaine car la définition d'un scénario cohérent (qui met en évidence les capacités du SUT à adopter le bon comportement face à une anomalie donnée) requiert une connaissance profonde du métier. Un scénario étant un ensemble de schémas d'altérations combinés, l'expert les définit un par un. Il peut utiliser une méthode

1. https://opensky-network.org/ [Dernière visite : Septembre 2020]
2. https://www.marinetraffic.com/ [Dernière visite : Septembre 2020]

graphique s'appuyant sur l'enregistrement récupéré par le module d'acquisition, ou définir textuellement un modèle de scénario applicable à n'importe quel enregistrement. Cette dernière méthode s'appuie sur trois langages dédiés (DSL) qui sont présentés et détaillés en chapitre 6.

③ **Configuration des bases**. La configuration des bases est également destinée à un expert métier puisque cette étape consiste à modéliser l'architecture physique dans laquelle évolue le SUT.

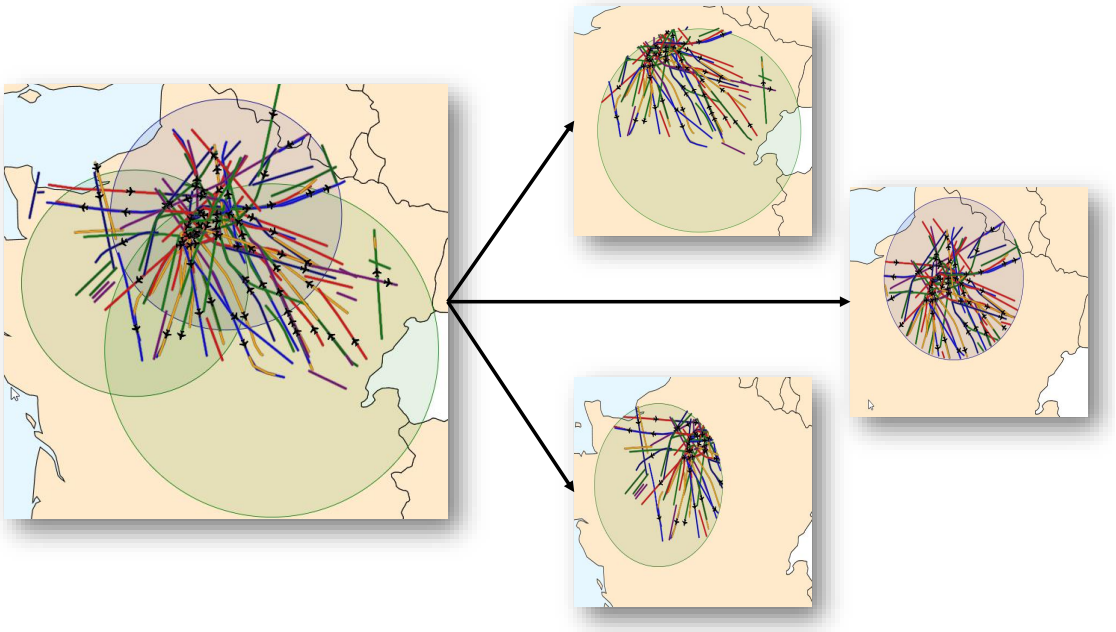


FIGURE 4.3 – Exemple de conversion : découpage d'un enregistrement en 3 sous-enregistrements.

④ **Conversion**. La conversion est une étape qui vise à diviser l'enregistrement initial en plusieurs sous-enregistrements, dont le nombre correspond au nombre de bases d'acquisition modélisé dans l'activité ③. Pour *n* bases de premier niveau, la conversion génère *n* sous-enregistrements. Chaque sous-enregistrement est associé à une base et correspond à l'ensemble des messages reçus par cette dernière, c'est-à-dire les messages émis par les objets à portée de la base. Il est possible que plusieurs sous-enregistrements contiennent les mêmes messages. Par exemple, sur la figure 4.1, deux objets sont à portée de deux bases d'acquisition : les messages émis par ces objets se retrouvent donc dans les deux sous-enregistrements correspondant à ces bases. Pendant la phase de conception de scénario, l'utilisateur peut appliquer des altérations à des sous-ensembles de bases. Ainsi, seules les bases ciblées par des altérations voient leurs sous-enregistrements altérés. Cela permet d'évaluer le comportement d'un SUT face à la réception d'informations hétérogènes. Un exemple de conversion appliquée à

un enregistrement de messages ADS-B est visible en figure 4.3.

⑤ **Production des altérations**. Ce module produit un ensemble de directives d'altérations à appliquer sur chaque sous-enregistrement reçu en entrée. Tandis que pour les scénarios graphiques les directives d'altérations sont définies manuellement par l'utilisateur, dans le cadre d'un scénario textuel, les directives d'altération sont obtenues par l'interprétation des trois DSL. Cette interprétation inclut le choix des cibles des altérations, la définition des dates de déclenchement de ces dernières et l'application de stratégies de test combinatoire (e.g., Pairwise). Ce processus d'interprétation est détaillé dans le chapitre 6.

⑥ **Injection des données**. La première étape de l'injection de données est la génération de données altérées par l'application des directives générées par le module précédent à chaque sous-enregistrement. Cette génération de données altérées requiert plus ou moins de réalisme selon le domaine : par exemple, dans le domaine aérien, il peut être nécessaire de respecter les propriétés physiques de l'appareil dans le cadre d'une modification de trajectoire. Cette étape est détaillée dans le chapitre 5. La deuxième étape de ce module est la création d'un flux de données vers le SUT pour chaque bases d'acquisition (et potentiellement pour les bases de décision émettant des rapports) dans le scénario. Les messages altérés sont naturellement envoyés à travers ces flux de données en respectant leur horodatage.

4.1.3/ EXIGENCES DE L'APPROCHE

L'approche présentée précédemment porte sur la capacité à simuler des FDIA par l'altération de données enregistrées. Cette approche de test de sécurité vise donc des domaines d'application sensibles aux attaques de type FDIA. Dans cette section nous analysons différentes caractéristiques des domaines d'application pour notre approche, à la fois vis à vis de la vulnérabilité aux FDIA et sur la nature des données de contrôle sur lesquels portent les altérations.

Si un domaine satisfait les quatre exigences listées dans le tableau 4.1, cela signifie qu'il lui est possible d'être la cible de FDIA telles qu'elles ont été identifiées dans la section 3.1 p. 39 et que celui-ci s'expose à un risque réel de cyberattaque. La vulnérabilité aux FDIA seule ne permet pas de déterminer si l'approche présentée dans la section 4.1.1 p. 58 est applicable à un domaine. En effet, une autre exigence relative à l'accès aux données doit être satisfaite, elle est présentée dans le tableau 4.2.

Le tableau 4.2 liste l'unique exigence portant sur l'accès aux données dans un domaine d'application. Quand toutes les exigences **FDIA** et **DATA** sont satisfaites, cela signifie qu'il est possible de mettre en place l'approche dans un domaine afin d'évaluer la résistance des systèmes qui le compose face aux FDIA. Le fait que différents domaines

ID	Exigence	Description
FDIA-1	Le SUT doit se situer dans un réseau.	Dans ce réseau les nœuds s'apparentent aux objets et aux bases présentés précédemment.
FDIA-2	Les objets doivent transmettre périodiquement leurs propriétés.	Les informations diffusées par les objets doivent l'être à intervalles de temps régulier.
FDIA-3	Les objets doivent avoir des propriétés évolutives.	Chaque objet doit posséder des propriétés intrinsèques et dynamiques (e.g., un bateau a une vitesse qui varie au cours du temps), ou son environnement doit être fluctuant (e.g., un horodateur intelligent peut relever les niveaux de pollutions qui sont des variants)
FDIA-4	Une unité de décision, centralisée ou distribuée (les objets)est présente.	L'unité de décision traite les données reportées par les objets afin d'établir une estimation de l'état global du système, de publier des rapports, de donner un diagnostic du système ou de prendre des décisions, etc.

TABLE 4.1 – Exigences pour la réalisation d'une FDIA.

satisfassent un même ensemble d'exigences implique des similarités entre-eux, ces dernières soulignent le caractère générique de l'architecture présentée dans ce chapitre, et plus généralement des autres contributions de cette thèse. Néanmoins, cette généricité n'est pas parfaite et l'application de l'approche à un domaine éligible doit suivre un processus de spécialisation.

4.1.4/ PROCESSUS DE SPÉCIALISATION

Pour adapter l'approche à un domaine respectant les exigences définies en section 4.1.3 p. 62, il est nécessaire de suivre un processus de spécialisation que nous avons décomposé en quatre étapes :

— **Identification des objets**. La première étape concerne l'identification des objets spécifiques au domaine dont le rôle est la prise et la transmission d'informations.

— **Identification des propriétés**. L'objectif de cette étape est d'identifier toutes les propriétés altérables, qu'elles soient dynamiques ou statiques transmises par les objets et de déterminer quels sont leurs types ainsi que leurs valeurs limites.

— **Identification des parties spécifiques**. Chaque domaine d'application possède ses spécificités. Elles peuvent par exemple concerner les bases dont les comportements changent d'un domaine à un autre et qu'il faut être en mesure de

TABLE 4.2 – Exigence pour la faisabilité de l'approche.

ID	Exigence	Description
DATA-1	Un historique des données transmises est accessible.	Toute la conception reposant sur des enregistrements de messages émis par les objets il est impératif d'avoir accès à un historique des données pour servir de base de travail.

simuler. Les spécificités peuvent également concerner des schémas d'altération spécifiques au domaine (e.g., modification de trajectoire dans le domaine aérien et le domaine maritime) qui peuvent nécessiter des connaissances métier.

— **Création d'un module d'encodage/décodage pour le protocole cible**. C'est une tâche technique qui est obligatoire si les messages constituant un enregistrement sont encodés. La nécessité de cette étape dépend des sources de données et des données acceptées en entrée par le SUT. Généralement les messages récupérés par des antennes sont bruts et doivent être décodés. À l'inverse, les fournisseurs de données en ligne proposent souvent des messages déjà décodés et lissés par des filtres de suppression du bruit.

4.2/ SPÉCIALISATION AUX DOMAINES D'APPLICATION

Cette section décrit l'instanciation de la démarche aux domaines du contrôle aérien et de la surveillance maritime. Nous étudions en premier lieu comment chacun de ces deux domaines satisfait les exigences données dans la section 4.1.3 p. 62, puis nous montrons l'application du processus de spécialisation en quatre étapes décrit dans la section 4.1.4 p. 63. La première sous-section présente de manière détaillée la spécialisation de la démarche pour le domaine aérien, qui constitue le domaine d'étude principale de cette thèse. La deuxième sous-section détaille le cas du domaine maritime, qui constitue le domaine d'application secondaire de cette thèse.

4.2.1/ SPÉCIALISATION AU LE DOMAINE AÉRIEN

La surveillance de l'espace aérien, notamment avec l'utilisation du protocole ADS-B, est le principale domaine d'application des travaux présentés dans cette thèse. Ayant déjà été abordé en profondeur dans la section 2.1 p. 21, cette sous-section se concentre sur la satisfaction des exigences de mise en œuvre de l'approche puis déroule les différentes étapes du processus de spécialisation.

Dans le tableau 4.3, les exigences données en section 4.1.3 p. 62 sont listées et discutées

vis à vis du domaine aérien.

TABLE 4.3 – Exigences pour le domaine aérien.

ID	Justification
FDIA-1	Les avions sont des objets, transmettant des informations via le protocole ADS-B à des stations au sol (base de premier niveau). Les stations relayant elles-mêmes des données d'autres systèmes (e.g., SDPS - *Surveillance Data Processing System* - ou le système de contrôle en fin de chaîne).
FDIA-2	Les avions possèdent des propriétés intrinsèques dynamiques (e.g., position, vitesse, altitude, code de statut, etc.), ainsi que des propriétés statiques liées à son identité (e.g., code ICAO, callsign).
FDIA-3	Comme précisé dans la section 2.1.3 p. 24, le protocole ADS-B consiste en des envois périodiques de messages par les avions dont le transpondeur est équipé de la technologie.
FDIA-4	Dans le domaine aérien, il existe une multitude d'unités de décision pouvant chacune faire office de SUT selon le niveau dans lequel on se place. Dans les avions, les systèmes d'alerte de trafic et d'évitement de collision traitent les messages ADS-B pour produire des diagnostics sur une situation donnée. Chaque radar peut être amené à produire des rapports sur les données ADS-B reçues, les valider ou encore les fusionner. Enfin les systèmes de contrôle en fin de chaîne ont pour but de produire une estimation de l'état global du trafic aérien.
DATA-1	il est possible de se procurer des récepteurs ADS-B, pour quelques centaines d'euros[3], permettant de se constituer son propre historique de données. Il existe également des fournisseurs de données en ligne comme le réseau OpenSky[4].

Ces exigences étant respectées, il est donc possible d'appliquer la démarche au domaine aérien en suivant les quatre étapes présentées dans la section 4.1.4 p. 63.

L'identification des objets : dans le domaine aérien, les objets peuvent être les avions ou les stations au sol : cela dépend du niveau de lecture de la situation. À un premier niveau, on peut considérer que les avions sont des objets qui envoient leurs propriétés aux stations au sol (bases), pouvant être considérées comme des unités de décisions. À un deuxième niveau, on peut considérer les stations au sol comme des objets envoyant leur données à un SDPS (base) pouvant, lui-même, être considéré comme l'unité de décision. Selon le niveau, on peut donc considérer une station au sol ou un SDPS comme un système à tester.

Identification des propriétés : comme dit précédemment, les avions possèdent des propriétés statiques, constantes tout le long d'un enregistrement :

- **ICAO** : adresse unique de 24 bits attribuée à chaque avion.
- **Callsign** : identifiant du transpondeur
- **Altitudes** minimale et maximale : plus petite et plus grande valeurs atteintes dans l'enregistrement, en pieds.

— **Nombre de positions connues** : nombre de messages contenant l'altitude, la longitude et la latitude de l'avion envoyés durant l'enregistrement.

Ils possèdent également des propriétés dynamiques, évoluant en fonction du temps :

— **Altitude** de l'avion, en pieds.

— **Position** de l'avion : latitude et longitude.

— **Vitesse au sol**, en nœuds.

— **Vitesse ascensionnelle**, en pieds par minute.

— La **route**, en degré, correspond à l'angle entre le nord et le vecteur de déplacement de l'avion.

— La **direction**, en degré, correspond à l'angle entre le nord et le nez de l'avion. La direction est différente de la route bien que proche. En cas de vent par exemple, le nez de l'avion peut pointer à 15° mais son vecteur de déplacement sera de 14°.

— **Squawk** : un code à quatre chiffres donnant le statut de l'appareil (en vol, en approche, en alerte générale, etc.).

— **Alerte** : un booléen indiquant un changement de squawk.

— **Urgence** : un booléen indiquant un changement de squawk vers un code d'urgence (7500, 7600 ou 7700).

— **SPI** (ou Identité) : un booléen indiquant si le transpondeur est en mode *identification*, un mode permettant aux contrôleurs de vérifier le callsign du transpondeur.

— **Sol** : un booléen indiquant la présence au sol d'un avion.

Identification des parties spécifiques : la spécificité majeure du domaine aérien est l'évolution des objets dans un environnement en trois dimensions. Sur la base de cette spécificité, des besoins particuliers ont été identifiés :

— La possibilité de créer des zones en 3D, définies par un ensemble de coordonnées formant un polygone en x et y et par une plage d'altitude. Les zones sont utilisées afin de permettre la sélection géographique des avions ciblés par une altération.

— La possibilité de créer des trajectoires en 3D en définissant des points de passage (coordonnées, altitude et temps de passage). Ces trajectoires sont utilisées pour permettre la création de nouveaux avions dans un enregistrement, ou la modification de trajectoires existantes. Ces mécanismes sont détaillés dans le chapitre 5.

Pour une simulation complète de la circulation de l'information au sein d'un système ATC, partant des avions (objets), puis passant par les stations au sol (bases) pour finir dans les unités de décisions de fin de chaîne, il est nécessaire de simuler le fonctionnement de certaines stations au sol qui émettent des rapports (habituellement de type ATX21 [5]) selon les messages ADS-B reçu.

Création d'un module d'encodage/décodage pour le protocole cible : il n'existe pas de format officiel pour le stockage des messages ADS-B décodés. Ils sont généralement disponibles dans des formats de type CSV, comme par exemple le Socket Data Format (SDF) [6]. En revanche, les messages bruts se trouvent généralement sous forme de données hexadécimales associées à une date de réception, comme par exemple le format Mode-S Beast [7]. Il est donc nécessaire de disposer d'un module permettant de décoder les messages bruts pour en extraire l'information, ce module peut être créer à partir de zéro mais il existe également des solutions de décodage open-source (comme la librairie à disposition par la communauté OpenSky [8]).

4.2.2/ SPÉCIALISATION AU DOMAINE MARITIME

Le second domaine d'application des travaux concerne la surveillance du trafic maritime, avec l'utilisation du protocole AIS, déjà présenté dans la section 2.2.2 p. 30. À l'instar de la sous-section précédente, celle-ci démontre la satisfaction des exigences de mise en œuvre puis le déroulé des étapes du processus de spécialisation. Comme pour la section précédente, les exigences données en section 4.1.3 p. 62 sont listées dans le tableau 4.4. Pour chacune d'elles, il est montré qu'elle est respectée dans le cadre du domaine maritime.

L'identification des objets : dans le domaine maritime les objets peuvent être les bateaux ou les infrastructures faisant partie du système AIS (e.g., bouées, aides à la navigation, satellites ou stations de base). Ainsi, on peut considérer que les bateaux sont des objets qui envoient leurs propriétés aux infrastructures AIS (bases), pouvant être considérées comme des unités de décisions, donc comme des systèmes à tester. À un deuxième niveau, on peut considérer ces infrastructures comme des objets envoyant leur données à des systèmes de gestion de trafic maritime, ou *Vessel Traffic Services* (VTS) qui font offices d'unité de décision. Selon le niveau, on peut donc considérer n'importe quel élément de l'architecture comme le système à tester.

Identification des propriétés : comme les avions, les bateaux possèdent des propriétés

5. L'ASTERIX est la spécification créée par l'organisme Eurocontrol. Elle définie un ensemble de protocoles et de standard. Ici ATX21 fait référence à la catégorie 21 de la spécification, c'est un protocole fournissant des rapports sur un ensemble de message ADS-B.
6. http://woodair.net/sbs/Article/Barebones42_Socket_Data.htm [Dernière visite : Novembre 2020]
7. https://wiki.jetvision.de/wiki/Mode-S_Beast:Data_Output_Formats [Dernière visite : Novembre 2020]
8. https://opensky-network.org/community/projects/20-java-adsb [Dernière visite : Novembre 2020]

TABLE 4.4 – Exigences pour le domaine maritime.

ID	Justification
FDIA-1	Les bateaux sont des objets, transmettant des informations via le protocole AIS à des stations de base ou à des satellites. Ces bases d'acquisition transmettent elles-mêmes des données à d'autres systèmes via des protocoles de plus haut niveau.
FDIA-2	Les bateaux possèdent des propriétés intrinsèques dynamiques (e.g., position, vitesse, cap, code de statut, etc.), ainsi que des propriétés statiques liées à son identité (MMSI, code IMO, callsign).
FDIA-3	Comme précisé dans la section 2.2.2 p. 30 le protocole AIS fonctionne avec des envois périodiques de messages par les bateaux équipés de ce système.
FDIA-4	Comme pour le domaine aérien il existe des unités de décision à plusieurs niveaux. On a les bateaux qui reçoivent des messages AIS et établissent une carte locale du trafic maritime, ainsi que les systèmes de contrôles en fin de chaîne produisant une estimation de l'état global du trafic maritime.
DATA-1	Dans le cadre de la surveillance maritime il peut être compliqué de récolter un volume de données important via une antenne AIS positionnée loin d'un littoral. Dans un tel cas, il est préférable de se tourner vers les fournisseurs de données en ligne[9]. Dans le cadre de cette thèse, un partenariat a été établi avec l'organisation gouvernemental norvégienne Statsat AS[10] qui a fourni 17 gigaoctets de données AIS provenant de plusieurs de ses satellites.

statiques, constantes dans les enregistrements :

— **MMSI** (*Maritime Mobile Service Identity*) : identifiant à 9 chiffres attribué à chaque bateau.

— **IMO** : identifiant unique à 7 chiffres, attribué par l'organisation éponyme (*International Maritime Organization*).

— **Callsign** : identifiant du transpondeur

— **Nom** : nom du bateau, généralement donné par le propriétaire.

— **Longueur**/**largeur** : dimension du bateau, en mètres.

— **Type de bateau** (ou *cargo*) : numéro indiquant le type de bateau selon une liste prédéfinie [11].

— **Nombre de positions connues** : nombre de messages contenant la localisation du bateau dans l'enregistrement.

Ils possèdent également des propriétés dynamiques, évoluant en fonction du temps :

— **Position du bateau** : latitude et longitude.

— **Vitesse sur le fond**, en nœuds.

11. https://coast.noaa.gov/data/marinecadastre/ais/VesselTypeCodes2018.pdf [Dernière visite : Septembre 2020]

— La **route**, en degrés, correspond à l'angle entre le nord et le vecteur de déplacement de l'avion.

— Le **cap**, en degrés, correspond à l'angle entre le nord et le nez du bateau. Le cap peut différer de la route à cause des influences du vent et du courant.

— **Statut** : code donnant le statut du navire, (amarré, en pêche, en déplacement etc.).

— **Tirant d'eau** : hauteur de la partie immergé du bateau, en mètres. Cette hauteur varie principalement selon la charge transportée.

Identification des parties spécifiques : comme pour le domaine aérien, la spécificité majeure du domaine maritime est l'évolution des objets dans un environnement régi par un système de coordonnées, ce qui implique :

— La possibilité de créer des zones en 2D, définies par un ensemble de coordonnées formant un polygone. Elles sont utilisées pour la sélection géographique des cibles des altérations.

— La possibilité de créer des trajectoires en 2D en définissant des points de passage (coordonnées et temps de passage). Ces trajectoires sont utilisées pour créer de nouveaux bateaux ou modifier des trajectoires existantes.

Enfin, pour une simulation complète de la circulation de l'information au sein du système AIS, il est nécessaire de simuler le fonctionnement des infrastructures AIS utilisées dans un scénario de test (bouées, aides à la navigation, satellites, etc.).

Création d'un module d'encodage/décodage pour le protocole cible : il n'existe pas de format officiel pour le stockage des messages AIS décodés : ils sont généralement disponibles dans des formats de type CSV, comme par exemple le format proposé par le site *Marine Cadastre*[12]. Les messages bruts suivent quant à eux une norme mise en place par la *National Marine Electronics Association* (NMEA), reposant sur l'encodage Sixbit[13].

4.2.3/ DISCUSSIONS CONCERNANT L'APPLICABILITÉ À D'AUTRES DOMAINES

Les deux domaines d'application de la thèse que sont les domaines aérien et maritime sont relativement proches mais présente des spécificités dans la nature des données et dans le type d'altération souhaité. Une des perspectives de la thèse sera d'étudier l'application de l'approche aux autres domaines de la gestion des transports, et en particulier pour le transport terrestre (véhicule connecté, ferroviaire). D'autres domaines

12. https://marinecadastre.gov/data/ [Dernière visite : Septembre 2020]
13. https://fr.wikipedia.org/wiki/Sixbit [Dernière visite : Septembre 2020]

plus éloignés, tels que ceux liés aux objets connectés avec systèmes de contrôle (par exemple dans l'Internet des Objets) nous semblent compatibles avec l'approche présentée dans ce chapitre mais il s'agit la aussi d'une perspective d'étude possible à partir de cette thèse. Nous terminons ce chapitre en discutant deux de ces domaines profondément différents de celui des systèmes de contrôle dans les transports : les systèmes de parking intelligent constituent un cas d'étude du projet ANR GeLeaD (dans le cadre duquel s'effectue une partie des travaux de cette thèse - cf. chapitre 1) et une thèse est en cours au sein de notre laboratoire sur ce sujet, et le domaine des réseaux électriques intelligents - Smart grids - fait l'objet d'une intense littérature quant à leur sensibilité aux FDIA.

Le domaine des systèmes de parking intelligent :

Le problème du stationnement est un problème majeur contribuant à la congestion du trafic dans les agglomérations. Entre l'augmentation du nombre de voitures en circulation, le manque d'espace pour créer de nouveaux parkings et une population de plus en plus urbanisée, cette problématique risque de s'accroître dans de nombreux pays en l'absence de plan visant à retirer les voitures des villes tout en conservant une mobilité efficace. Face à cette situation, des systèmes de parkings intelligents commencent à voir le jour. Des grandes villes comme Barcelone, Montréal ou New-York se développent en se concentrant sur le thème de la ville connectée avec comme hypothèse que l'internet des objets (IoT), le *big data*, l'e-commerce local ou les véhicules connectés sont des outils pouvant aider les villes à mieux comprendre et gérer leur mobilité. Dans cette optique, l'entreprise Flowbird [14] propose des horodateurs permettant, en plus de leurs fonctions usuelles, de collecter de nombreuses données environnementales tout au long de la journée, comme le niveau de bruit, la pollution, la température ou encore le taux d'humidité. Ce système s'apparente à un réseau d'objets (horodateurs), transmettant des propriétés évolutives (données environnementales) périodiquement. À première vue ce domaine semble compatible avec l'approche de cette thèse. Les points à éclaircir restent la question de la présence d'une unité de décision et d'un historique de données accessible.

Le domaine des réseaux électriques intelligents ou *smart grids* :

Les réseaux électriques intelligents permettent de réguler la production et d'optimiser la distribution de l'électricité dans les villes grâce à une remontée de l'information des consommateurs vers les fournisseurs en temps réel. Cette remontée d'information est effectuée à travers des boîtiers de communication qui jouent le rôle des objets (e.g., Linky en France). L'information remontée est fluctuante par nature, la consommation électrique à un instant donné n'étant pas une constante. L'unité de décision est décentralisée et la régulation de la production se fait aussi bien au niveau du parc de production que

14. https://www.flowbird.group/ [Dernière visite : Septembre 2020]

du réseau de distribution ou des consommateurs. Pour la question de l'historique des données, les études sur la prévision de la charge du réseau [89] vont dans le sens de l'existence de données historiques, notamment quand celles-ci sont basées sur des techniques d'intelligence artificielle [91].

Ces domaines ont pour point commun d'être centrés autour de l'acquisition de données dans une optique de contrôle d'un système global et décentralisé. Cette similarité correspond à la définition d'un système de contrôle et d'acquisition de données, ou *Supervisory Control And Data Acquisition* (SCADA) [26]. En effet, les SCADA sont des systèmes à grande échelle traitant en temps réel un grand nombre d'informations, dans le but de contrôler des installations techniques. L'acquisition de données massive permet d'établir un consensus, c'est-à-dire une estimation jugée des plus fiable de l'état global d'un système.

On retrouve souvent les SCADA dans les domaines critiques comme les systèmes de canalisation de gaz et de pétrole, les réseaux électriques ou les transports de produits chimiques. Cette criticité implique une vigilance accrue en matière de cybersécurité, et les SCADA, de par leur aspect multi-système et distribué, présentent de nombreuses vulnérabilités qui ont été identifiées dans plusieurs études [110, 44]. L'approche de cette thèse du test de cyber-sécurité par altération sémantique de la données de contrôle nous semble pouvoir intéresser ce large domaine des SCADA, ce qui reste à étudier dans la suite de cette thèse.

4.3/ SYNTHÈSE

Les domaines aérien et maritime possèdent de fortes similitudes à plusieurs niveaux : la nature des objets connectés qui peuplent le domaine (des véhicules de transports d'humains ou de marchandises), les protocoles analogues utilisés par ces derniers (ADS-B et AIS), ou l'architecture sur plusieurs niveaux avec des bases d'acquisition servant de relais (stations au sol pour l'aérien, stations de base ou satellites pour le maritime) vers des bases de décisions (SDPS pour l'aérien, VTS pour le maritime). Sur la base de ces observations, une architecture générique à ces domaines a été proposée dans ce chapitre, elle permet de créer des scénarios de tests pour les systèmes de contrôle aériens et maritimes. Cette architecture couvre le cycle de test ciblé FDIA de tels systèmes, à partir de l'acquisition de données nominales, elle permet à un expert métier de concevoir les cas de tests. Ces cas de tests formalisent la manière dont les données nominales seront altérées afin de former les données de tests, finalement injectées dans le système sous test.

C'est sur cette architecture sur laquelle s'appuie les deux prochaines contributions

présentées dans les chapitres suivants, traitant chacun d'un aspect du cycle de test. Le sujet du chapitre 5 est la génération des données de tests, cette dernière se base sur une scénarisation des tests via un ensemble de langages dédiés présentés dans le chapitre 6.

5

GÉNÉRATION DE DONNÉES ALTÉRÉES

La génération de données fait référence à la modification (ou l'altération) d'un enregistrement nominal afin d'en obtenir une version dite "altérée". L'injection d'un enregistrement altéré dans un système sous test (SUT) puis l'analyse de la réaction de ce SUT constituent la réalisation d'un test et l'établissement de son verdict. Le présent chapitre détaille la manière dont les données de test - appelées cas de test dans la suite - sont générées.

Les contributions présentées dans ce manuscrit se veulent génériques, comme l'architecture présentée dans le chapitre 4, néanmoins, l'absence d'exemples concrets au sein d'un domaine nuit à la compréhension des notions théoriques. C'est pourquoi à partir de ce chapitre et jusqu'à la fin de la partie II, c'est le domaine aérien (principal domaine d'application de la thèse) qui est utilisé en tant que domaine d'application pour présenter les différentes contributions.

Dans la première section de ce chapitre, le processus d'altération est décrit dans son ensemble. La section suivante présente les détails d'implémentation de la génération de données pour chaque type d'altération (voir section 4.1.1 p. 58). En effet, un moteur d'altération est chargé d'appliquer les modifications à l'enregistrement nominal pour chaque type d'altération. Le fonctionnement de chacun de ces moteurs est détaillé dans cette deuxième section. Enfin, la dernière section aborde la question de la pertinence des données générées à travers une expérimentation basée sur l'évaluation d'un modèle de détection d'anomalie via le *Machine Learning* proposé par Habler et al. [39].

5.1/ LE PROCESSUS D'ALTÉRATION

Le processus d'altération vise à générer une version altérée d'un enregistrement nominal sur la base d'un ensemble de directives d'altération. Une directive d'altération spécifie un type de modification à appliquer à un enregistrement au travers d'une liste d'objets cibles associée à une fenêtre de temps. Il existe plusieurs types de directive d'altération

qui partagent des propriétés de base. De manière générale, une directive d'altération est formalisée de la manière suivante : $dir_x = (w, t, P)$ où :

— x est le type (e.g., altération, suppression, etc.) de la directive d'altération. Chaque type est décrit dans une sous-section de la section 5.2 p. 74 qui lui est dédiée.

— w est la fenêtre de temps de l'altération, en secondes. Elle débute par $w.start$ et se termine par $w.end$.

— t (pour *target*) est la liste des avions ciblés.

— P est une liste de paramètres qui précisent la directive d'altération et dépendent de son type.

Ces directives sont prises en charge séquentiellement par un moteur d'altération qui leur est dédié. Le processus global d'altération, partant d'un enregistrement nominal pour produire sa version altérée, est visible dans la figure 5.1, chaque élément numéroté est décrit dans la suite de cette section.

① **L'enregistrement nominal** est l'enregistrement sur lequel les directives d'altération sont appliquées.

② **La liste de directives d'altération** spécifie toutes les modifications à effectuer sur un enregistrement nominal afin d'en obtenir sa version altérée.

③ **Le sélecteur** itère sur la liste des directives d'altération et, pour chacune d'elles, fait appel au moteur d'altération correspondant au type de la directive.

④ **Les moteurs d'altération** ont pour rôle d'altérer les enregistrements nominaux. Comme dit précédemment, il existe un moteur d'altération pour chaque type de directive d'altération, selon le processus détaillé dans la figure 5.2. Chaque moteur est décrit plus en détail dans la section 5.2 p. 74.

⑤ **L'enregistrement altéré** est obtenu en sortie du processus. Il est plus ou moins proche de sa version originale selon le nombre d'avions ciblés et la durée des fenêtres de temps définis dans les directives d'altérations.

5.2/ LES MOTEURS D'ALTÉRATION

Dans cette section, chaque moteur d'altération est passé en revue dans une sous-section qui lui est dédiée. Dans ces sous-sections, les spécificités de l'implémentation sont mises en avant via la présentation des algorithmes utilisés dans les tâches du processus partagées par les différents moteurs (figure 5.2).

Ce processus se divise en plusieurs étapes. La première consiste à déterminer si un prétraitement est requis par l'algorithme du moteur d'altération courant. En effet, certains

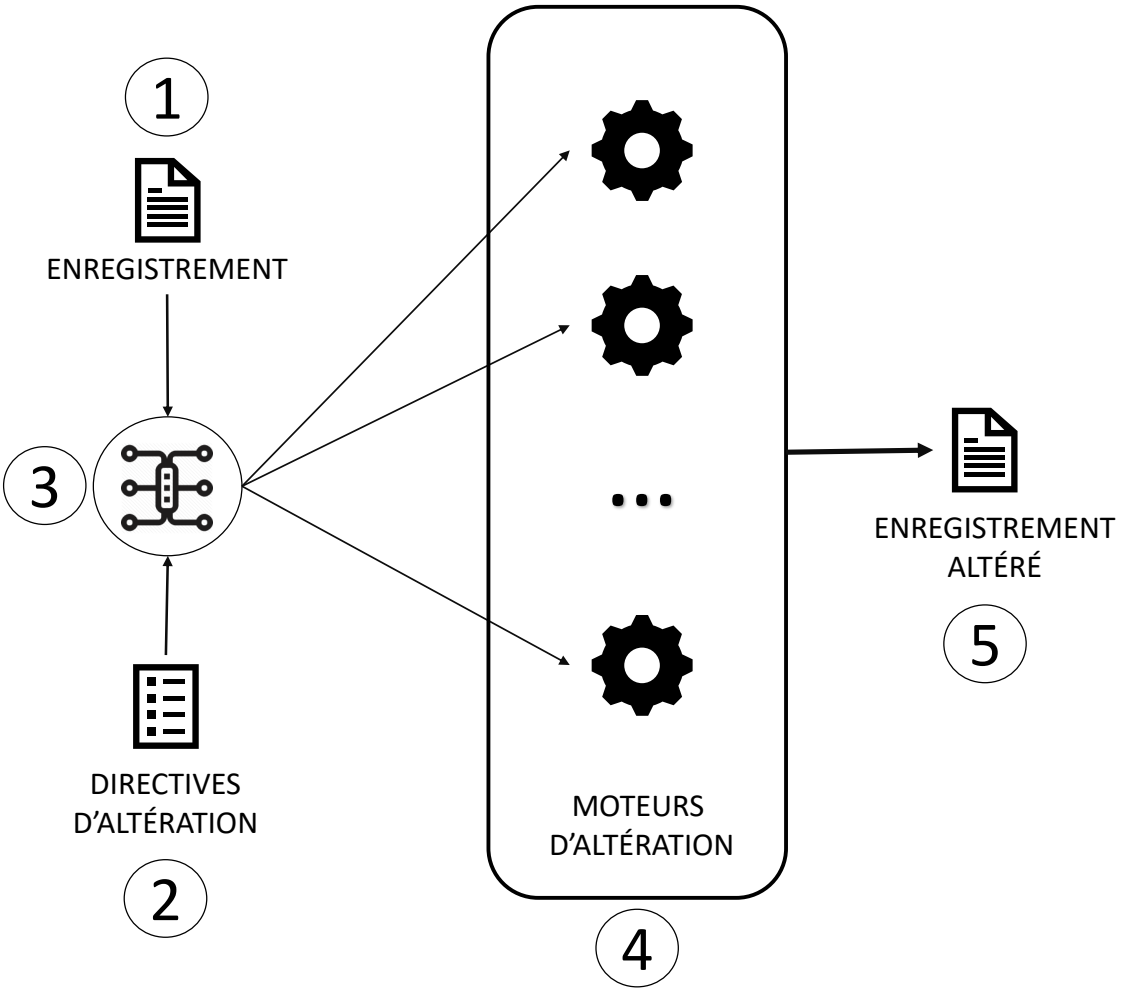


FIGURE 5.1 – Vue d'ensemble du processus d'altération

algorithmes doivent extraire des informations de l'enregistrement avant d'être en mesure de produire des messages altérés. Par exemple, dans le cas de la modification de trajectoire, le moteur dédié a besoin de générer des fonctions d'interpolation à partir des positions 3D successives des avions cibles. Ces fonctions sont ensuite interrogées afin de produire des valeurs altérées de latitude, de longitude et d'altitude, formant ainsi les nouvelles trajectoires des avions ciblés.

Par la suite, ce processus est principalement basé sur les tâches TRAITER ACTION et APPLIQUER ACTION. La première itère sur les messages de l'enregistrement et vérifie pour chaque message s'il satisfait des conditions spécifiques (e.g., si le message est dans la fenêtre de temps de la directive d'altération et si l'avion émetteur en est la cible). Si c'est le cas, alors le message est envoyé vers la tâche APPLIQUER ACTION qui

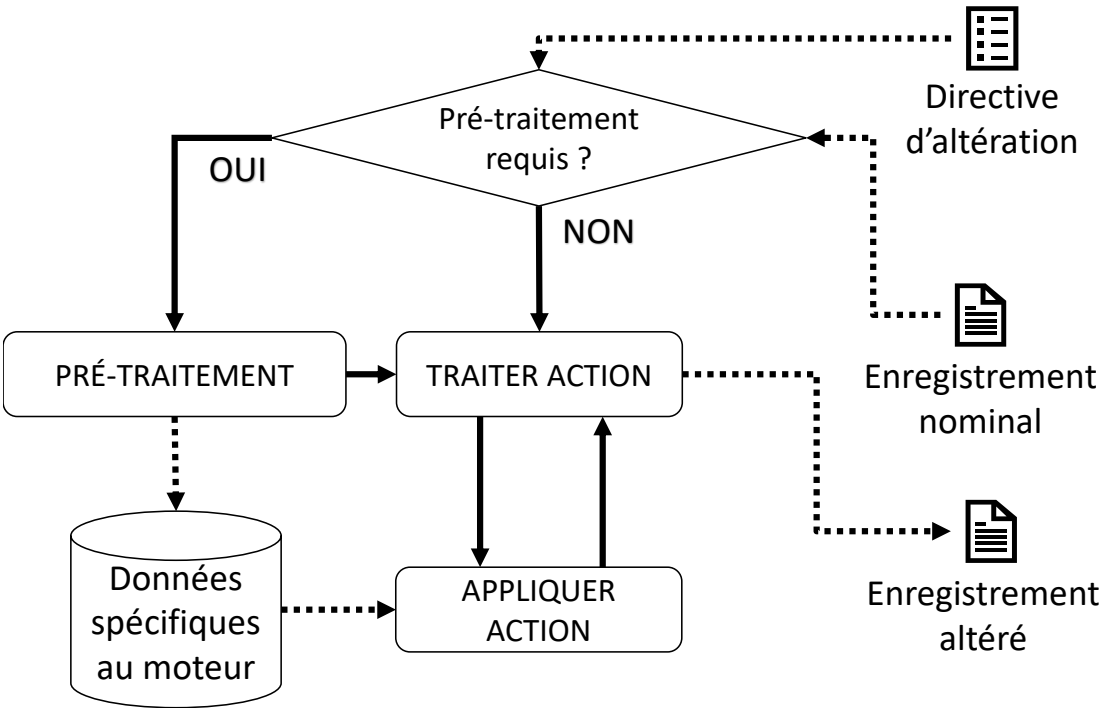


FIGURE 5.2 – Processus partagé par les moteurs d'altération

applique la directive d'altération à celui-ci puis le retourne à la tâche TRAITER ACTION. Cette dernière termine l'itération en écrivant le message (qui a pu subir, ou non, une ou plusieurs altérations) dans l'enregistrement.

Chaque moteur d'altération implémente sa propre version des tâches PRÉ-TRAITEMENT, TRAITER ACTION et APPLIQUER ACTION. Chacune d'elles est détaillée dans les sous-sections suivantes.

5.2.1/ LA MODIFICATION DE PROPRIÉTÉ

La plupart des altérations consistent à changer de manière abstraite du point de vue de l'utilisateur les valeurs des propriétés contenues dans les messages. En effet, l'utilisateur ne spécifie pas toujours directement les valeurs des propriétés. Par exemple, dans le cas d'une modification de trajectoire, la nouvelle vitesse de l'avion n'est pas choisie par l'utilisateur mais dépend de la nouvelle trajectoire de l'appareil. À l'inverse, la modification de propriété permet aux utilisateurs de spécifier de manière concrète les valeurs à attribuer aux propriétés ciblées. La modification de propriété rend possible, par exemple, les fausses alertes ou les usurpations d'identité, les deux consistant à changer – au moins – une propriété : respectivement le squawk et l'ICAO de l'avion.

Pour ajouter du réalisme aux altérations, il peut toutefois être nécessaire de modifier des

valeurs de propriétés supplémentaires. Par exemple, dans le cadre d'une fausse alarme, les propriétés *alerte* et *urgence* (voir section 4.2.1 p. 64) doivent être changées à *vrai*. En effet, le protocole ADS-B prévoit qu'un changement de squawk modifie la valeur de la propriété *alerte* à *vrai*, tandis qu'un changement de squawk à un code d'urgence (7500, 7600 ou 7700) modifie la valeur de la propriété *urgence* à *vrai*. Cet exemple illustre les connaissances qu'il est nécessaire d'avoir sur le protocole afin de concevoir des scénarios crédibles pour les systèmes de contrôle.

L'interprétation du schéma d'une altération simple produit des directives d'altérations qui se formalisent de la manière suivante : $dir_{prop} = (w, t, P)$ où :

— P est une liste non vide de modifications de propriété. Une modification est formalisée par $p = (i, v, m)$ où :

  — i est l'identifiant de la propriété (e.g., altitude, vitesse au sol, etc.).

  — v est la nouvelle valeur à attribuer à la propriété.

  — m est le mode de modification. En effet, il existe quatre façons de modifier la valeur d'une propriété : $REPLACE$, $OFFSET$, $NOISE$ et $DRIFT$, représentant respectivement un remplacement, un décalage, du bruit, et une dérivation (ou décalage graduel). Leurs effets sur les propriétés sont décrits avec l'algorithme 2.

Algorithme 1 : Algorithme de la tâche TRAITER ACTION pour la modification de propriété

Input : $recording$: liste de messages nominaux
dir_{prop} : directive d'altération
Result : $al\_recording$: liste de messages nominaux et altérés

```
1  msg_idx ← 1
2  foreach old_msg ∈ recording do
3      if old_msg.date ∈ dir_prop.w ∧ old_msg.icao ∈ dir_prop.t then
4          new_msg ← appliquerAction(old_msg, dir_prop, msg_idx)
5          msg_idx ← msg_idx + 1
6      else
7          new_msg ← old_msg
8      end
9      al_recording ← al_recording ∪ {new_msg}
10 end
11 return al_recording
```

L'algorithme 1 décrit les opérations appliquées aux messages dans la tâche TRAITER ACTION dans le cadre d'une altération simple. L'algorithme itère sur les messages de l'enregistrement (ligne 2). Pour chaque message une vérification est faite : si la date d'envoi d'un message ($old\_msg.date$) émis par un avion cible se trouve dans la fenêtre de temps de la directive (ligne 3) alors le message est passé en paramètre de la fonction *appliquerAction* (tâche APPLIQUER ACTION), cette dernière retourne alors une version altérée du message selon la directive dir. Si la vérification échoue alors

le message initial est préservé (ligne 7). Finalement, le message (altéré ou préservé) est ajouté au nouvel enregistrement $al\_recording$ (ligne 9). À noter que la variable msg_idx (ligne 1) est un compteur utilisé pour le mode $DRIFT$ (quand $dir.m = DRIFT$). Son utilisation est détaillée dans l'algorithme 2 qui décrit l'algorithme de la tâche APPLIQUER ACTION.

Algorithme 2 : Algorithme de la tâche APPLIQUER ACTION pour la modification de propriété

Input : msg : message nominal
dir_{prop} : directive d'altération
msg_idx : indice du message
Result : al_msg : message altéré

```
1  al_msg ← copy(msg)
2  foreach p ∈ dir_prop.P do
3      if p.m = OFFSET then
4          al_msg.getParam(p.i).value ← al_msg.getParam(p.i).value + p.v
5      else if p.m = REPLACE then
6          al_msg.getParam(p.i).value ← p.v
7      else if p.m = NOISE then
8          al_msg.getParam(p.i).value ← al_msg.getParam(p.i).value + rand(0, p.v)
9      else if p.m = DRIFT then
10         al_msg.getParam(p.i).value ← al_msg.getParam(p.i).value + p.v × msg_idx
11     end
12 end
13 return al_msg
```

L'algorithme 2 est utilisé dans la tâche APPLIQUER ACTION dans le cadre de la modification de propriété. Il itère sur les modifications de propriété contenues dans $dir.P$ (ligne 2). Pour chaque modification p, une action est faite en fonction de son mode $p.m$ (ligne 3-11). Ainsi, si $p.m = OFFSET$ (ligne 3), alors $p.v$ est ajouté à la valeur initiale de la propriété récupérée via l'instruction $al\_msg.getParam(p.i).value$. Si $p.m = REPLACE$ (ligne 5), alors $p.v$ remplace la valeur initiale de la propriété. Si $p.m = NOISE$ (ligne 7) une valeur aléatoire entre 0 et $p.v$ est ajouté à la valeur initiale. Enfin si $p.m = DRIFT$ (ligne 9), alors $p.v * msg\_idx$ est ajoutée à la valeur initiale puisqu'il s'agit du compteur du nombre de messages transmis en paramètre par TRAITER ACTION.

5.2.2/ LA DISPARITION D'AVION

L'objectif de cette altération est de simuler un brouillage ou des perturbations visant un ensemble d'avions cibles de l'enregistrement, i.e. supprimer certains messages émis par les avions cibles dans une fenêtre de temps donnée. Elle est formalisée par la direction d'altération suivante : $dir_{del} = (w, t, n)$ où le paramètre P, spécifique à la modification de propriété disparaît. Un nouveau paramètre n indique le nombre de messages consécutifs à supprimer, par exemple, si $n = 0$ alors tous les messages seront supprimés, tandis que si $n = 3$, trois messages sur quatre seront supprimés. La disparition d'avion est

sans doute la forme d'altération la plus simple car elle n'a pour seul effet que de supprimer des messages originaux de certains avions pendant une certaine durée. L'algorithme se cantonne à la tâche TRAITER ACTION, où les messages sont écrits dans l'enregistrement altéré seulement si les avions émetteurs ne sont pas ciblés par l'altération et durant la fenêtre de temps spécifiée, i.e. $\neg(msg.icao \in dir.t \wedge msg.date \in dir.w)$. La tâche APPLIQUER ACTION n'est donc ainsi jamais déclenchée.

5.2.3/ LA MODIFICATION DE TRAJECTOIRE

La modification de la trajectoire d'un avion est un problème plus complexe que la simple modification de la valeur d'une propriété à un moment donné. En effet, cette altération doit être réaliste au regard des caractéristiques physiques des avions, auquel cas elle serait trop facilement détectée par les systèmes de contrôle et perdrait son intérêt. Les propriétés dynamiques d'un avion peuvent être formalisées par des fonctions continues de valeurs évoluant en fonction du temps, de telles fonctions se rapprochant ainsi au plus près du comportement réel d'un avion. C'est pourquoi une méthode d'interpolation est utilisée pour modéliser les fausses trajectoires qui sont générées par l'altération de modification de trajectoire. Cette section se divise en deux sous-section : la première présente la méthode d'interpolation et la seconde détaille l'implémentation de la modification de trajectoire.

5.2.3.1/ LA MÉTHODE D'INTERPOLATION AKIMA

L'interpolation d'Akima [5] a été choisie pour modéliser les propriétés dynamiques des avions pour sa facilité d'implémentation. Elle permet en outre d'obtenir des trajectoires lisses, sans effet d'oscillation car c'est une technique d'interpolation locale. En effet, elle utilise uniquement les points voisins pour les calculs. Au contraire, les techniques d'interpolation globales utilisent tous les points connus pour calculer des approximations. L'interpolation globale est alors sujette au phénomène de Runge (problème d'oscillation qui se produit sur un ensemble de points d'interpolation équidistants [30]), ce qui rend impossible la modélisation correcte d'un avion volant en ligne droite. De plus, l'utilisation des points voisins dans le calcul de l'approximation se rapproche du comportement naturel des avions, la position et la vitesse d'un avion vingt minutes dans le passé ne doit pas avoir d'influence sur la position et la vitesse courantes. Enfin, l'interpolation locale est plus rapide car elle utilise moins de paramètres dans le calcul. Étant donné le volume de données que peuvent atteindre les enregistrements (un enregistrement de trente minutes peut contenir autour de 150000 messages selon les zones), une méthode d'interpolation rapide contribue grandement au passage à l'échelle des approches présentées dans cette thèse.

L'utilisation de cette méthode permet d'éviter d'avoir recourt à des simulateurs telle que la librairie BADA [1] proposée par EUROCONTROL qui nécessite une expertise métier poussée. Encore une fois, l'utilisation de l'interpolation a été jugée acceptable dans ce travail de recherche et fait office de preuve de concept même si cette méthode s'avère certainement insuffisante en terme de réalisme dans un contexte industriel. De plus, comme l'a mis en évidence l'état de l'art des méthodes de test par simulation (voir section 3.2 p. 45), un degré de réalisme élevé à ce niveau nuit à la généricité de l'approche. Ainsi, le réalisme de l'altération reste majoritairement déterminé par le nombre et la qualité des points de passage de la fausse trajectoire fournis par l'utilisateur.

5.2.3.2/ IMPLÉMENTATION DE LA MODIFICATION DE TRAJECTOIRE

La modification de trajectoire est formalisée par la directive d'altération $dir_{vtm} = (w, t, \Omega)$ où le nouveau paramètre Ω est un ensemble non vide de points de passage. Un point de passage se compose de coordonnées (latitude et longitude), d'une altitude et d'un temps de passage et se formalise de la manière suivante : $\omega = (lat, lon, alt, date)$.

Afin de faciliter la compréhension de son implémentation, la modification de trajectoire peut être expliquée de manière informelle en la décomposant en trois étapes :

1. Pour chaque avion ciblé par la modification de trajectoire, on récolte chacune de ses positions 3D (latitude, longitude et altitude) qui se trouve en dehors de la plage de temps de l'altération. Ces points sont utilisés pour créer trois fonctions d'interpolation qui représentent respectivement la latitude, la longitude et l'altitude de l'avion en fonction du temps.
2. On utilise ensuite les points de passage fournis par l'utilisateur et contenus dans Ω pour remplir les fonctions d'interpolation avec la propriété correspondante (latitude, longitude ou altitude).
3. Pour chaque avion, on a donc trois fonctions d'interpolation qui ensemble, représentent une trajectoire 3D modifiée selon les points de passages fournis par l'utilisateur. Il suffit alors de questionner chacune de ces fonctions à un moment donné afin d'obtenir la nouvelle position de l'avion à cet instant.

L'algorithme de modification de trajectoire requiert un pré-traitement pour être appliqué, ainsi la tâche PRÉ-TRAITEMENT est déclenchée dans le cadre de cette altération. L'étape de pré-traitement consiste à créer les trois fonctions d'interpolation représentant chacune une composante de la trajectoire de l'avion (latitude, longitude et altitude) amputée de la partie qui sera remplacée par les points de passage suite à la modification de trajectoire. Suite au pré-traitement, la tâche TRAITER ACTION est déclenchée. Elle a pour rôle d'itérer sur les messages de l'enregistrement et, pour chaque message cible,

1. https://www.eurocontrol.int/model/bada

de remplacer les valeurs des propriétés concernées par celles obtenues en interrogeant la fonction d'interpolation associée.

Algorithme 3 : Pré-traitement pour la modification de trajectoire

Input : $recording$: liste de messages authentiques
dir : directive d'altération
Result : $trajs$: liste de trajectoires altérées

```
1  updated ← zeros(dir.t.size)
2  foreach msg ∈ recording do
3      if msg.icao ∈ dir.t then
4          if ¬(msg.date ∈ dir.w) then
5              trajs(msg.icao).addPos(msg.lat, msg.lon, msg.alt, msg.ts)
6          else if updated(msg.icao) = 0 then
7              foreach ω ∈ dir.Ω do
8                  trajs(msg.icao).addPos(ω.lat, ω.lon, ω.alt, ω.ts)
9              end
10             updated(msg.icao) ← 1
11         end
12
13     end
14 end
15 return trajs
```

La création des fonctions d'interpolation dans l'étape de pré-traitement est décrite dans l'algorithme 3. Considérons une structure de données nommée $Traj$ contenant trois fonctions d'interpolation (pour la latitude, la longitude et l'altitude) et un identifiant (ICAO). L'algorithme itère sur les messages de l'enregistrement. Pour chaque message émis par un avion cible et durant la fenêtre de temps $dir.w$ (lignes 2-4), la trajectoire correspondant à l'émetteur est récupérée et mise à jour avec les propriétés adéquates (ligne 5). Pour chaque avion cible, à partir du premier message émis dans la fenêtre de temps $dir.w$ (ligne 6) l'algorithme itère sur les points de passage $dir.\Omega$ et ajoute chacun d'eux, alors noté ω, à la trajectoire de l'avion émetteur du message courant (ligne 8). La trajectoire est ensuite marquée comme "altérée" (ligne 10) afin de ne plus rentrer dans la boucle des points de passages pour l'avion courant. Enfin, le reste des messages en dehors de $dir.w$ sont ajoutés aux différentes trajectoires. Le résultat de l'algorithme est une liste de trajectoires (une pour chaque avion cible), chacune de ces trajectoires est amputée des positions émises durant $dir.w$, ces dernières étant remplacées par les points de passages $dir.\Omega$.

Une fois que chaque avion cible possède une trajectoire associée, la tâche TRAITER ACTION est déclenchée. Son rôle est d'itérer sur l'enregistrement. Pour chaque message émis par un avion cible dans la fenêtre de temps $dir.w$, la tâche APPLIQUER ACTION est déclenchée. Cette dernière remplace les valeurs des propriétés de la manière suivante :

— Pour la latitude, la longitude et l'altitude, les valeurs sont remplacées par les valeurs interpolées lors du pré-traitement et contenues dans les structures de données $Traj$.

— Pour la vitesse au sol, la route et la vitesse ascensionnelle, les nouvelles valeurs sont calculées en se basant sur les positions interpolées. Par exemple, pour déterminer la nouvelle valeur de la vitesse au sol contenue dans un message m, l'algorithme calcule premièrement la distance horizontale (l'altitude est ignorée) entre deux points de la trajectoire temporellement proches, le premier point étant une seconde antérieure à m et le deuxième étant une seconde postérieure à m. La distance obtenue divisée par le temps entre les deux points donnant ainsi une estimation de la vitesse au sol de l'avion au moment de l'émission du message m.

Il est à noter qu'avec cette implémentation aucun nouveau message n'est créé, seuls des messages existants sont modifiés. Le fait de ne pas créer de nouveaux messages entraîne des effets de bord non négligeables. Par exemple, si la trajectoire altérée est plus longue que la trajectoire initiale (i.e. plus de distance est parcourue par l'avion), alors la vitesse au sol de l'avion doit être augmentée en conséquence, étant donné que le temps lui n'est pas ajustable (borné aux premier et dernier messages). En effet, les valeurs des propriétés associées au mouvement de l'avion (vitesse au sol, vitesse ascensionnelle, et route) sont modifiées automatiquement en fonction de l'allongement de la trajectoire. Néanmoins cette implémentation possède une limite, en particulier concernant la vitesse au sol, bien que cette implémentation permette de modifier des propriétés pour qu'elles correspondent à la trajectoire altérée de l'avion, cette solution est à double tranchant car un allongement trop important de la trajectoire peut mener à des valeurs erronées de la vitesse (i.e. dépassant les capacités des avions de ligne). Pour résoudre ce problème, la solution serait de créer de nouveaux messages afin d'augmenter le temps mis par les avions cibles pour parcourir leur trajectoire. À noter que cette solution, bien que possible, demande une maîtrise poussée des périodes d'émission des messages ADS-B.

5.2.4/ LA CRÉATION D'AVION FANTÔME

La création d'avion fantôme consiste à créer une fausse piste à partir de zéro, ce qui implique la création et l'insertion de faux messages au sein de l'enregistrement initial. Cette attaque est représentée par la directive d'altération $dir_{gac} = (w, P, \Omega)$, dont tous les paramètres ont été présentés dans les sections précédentes. Comme pour la modification de trajectoire, une étape de pré-traitement est nécessaire pour la création des trajectoires, toutefois ces dernières sont créées seulement à partir des points de passage $dir.\Omega$. Ces trajectoires sont ensuite passées à la tâche TRAITER ACTION dont le fonctionnement est détaillé dans l'algorithme 4. Étant donné que les messages ADS-B sont émis toutes les 400 à 600 millisecondes, la tâche TRAITER ACTION génère des

messages aléatoirement en incrémentant le temps de manière à respecter ce délai (ligne 6), l'itération commence à la date $dir.w.start$ pour finir à la date $dir.w.end$ (lignes 1-2). Pour remplir chaque nouveau message, la tâche APPLIQUER ACTION de la modification de trajectoire est appelée, cette dernière utilisant les trajectoires construites lors du pré-traitement.

Algorithme 4 : Algorithme de la tâche TRAITER ACTION pour la création d'avion fantôme

Input : $recording$: liste de messages authentiques
dir : directive d'altération
Result : $al\_recording$: liste de messages authentiques et altérés

```
1 mts ← dir.w.start
2 while mts <= dir.w.end do
3 |   msg ← newMsg(dir.P)
4 |   msg.date ← mts
5 |   msg ← applyAction(msg)
6 |   mts ← mts + rand(0.4, 0.6)
7 end
8 return al_recording
```

5.2.5/ LA SATURATION DE LA SITUATION AÉRIENNE GÉNÉRALE

Initialement cette altération consiste à créer subitement un nombre important d'avions fantômes dans le but de saturer la SAG. Toutefois, les systèmes de détection ont acquis une certaine résilience face à ce type de déni de service. Ici la définition de l'attaque est légèrement modifiée pour devenir un *flood* de modification de trajectoire. L'objectif est ainsi de générer plusieurs trajectoires différentes pour chaque avion cible comme si ce dernier se séparait en plusieurs clones déviant progressivement de la trajectoire de l'avion original à partir d'un même point. La saturation est formalisée par la directive d'altération $dir_{gaf} = (w, t, z, k, \alpha, v, d)$ avec les nouveaux paramètres suivants :

- z est le nombre de clones à créer.
- k est la valeur maximale de la distance entre l'avion cible et ses clones.
- α est une plage de valeurs dans laquelle est tirée une valeur pour chaque clone afin de déterminer sa divergence de l'avion cible par rapport à k.
- v est la vitesse de divergence, i.e. avec quelle rapidité le clone s'éloigne de l'avion cible ; elle est exprimée par une durée donnant le temps que met le clone à atteindre sa limite de divergence exprimée par $k * \alpha$.
- d est la direction de la divergence ; c'est une plage de valeurs qui représente un cône horizontal de 90° partant du nez de l'avion cible et centrée sur sa direction.

Ainsi, chaque clone est créé de manière à dévier lentement et horizontalement de la trajectoire de l'avion cible dans une direction tirée dans $dir.d$ à une distance de

$dir.k * dir.a$, $dir.a$ étant tirée dans $dir.\alpha$. Une fois la distance maximale atteinte, le clone suit une trajectoire similaire à l'avion cible tout en conservant son décalage de position horizontale.

Algorithme 5 : Algorithme du pré-traitement pour la saturation pour un avion cible

Input : $iniTraj$: trajectoire originale de la cible
dir : directive d'altération
Result : $trajs$: trajectoires des clones

```
1  for i ← 1 to dir.z do
2      α_i ← sample(dir.α) ; ts ← dir.w.start
3      d_i ← sample(dir.d) ; traj_i ← iniTraj
4      removeValues(traj_i, dir.w)
5      while ts ≤ dir.w.end do
6          pCoef ← min((ts − dir.w.start)/dir.v), 1)
7          step ← pCoef * α_i * dir.k
8          lat ← iniTraj.interpolate("lat", ts) + step+
9              cos(fake_ac.d)
10         lon ← iniTraj.interpolate("lon", ts) + step+
11             sin(fake_ac.d)
12         traj_i.addPos(lat, lon,
13             iniTraj.interpolate("alt", ts), msg.ts)
14         ts ← ts + dir.
15     end
16     trajs ← trajs ∪ {traj_i}
17 end
18 return trajs
```

Les trajectoires des clones sont créées durant la tâche de pré-traitement dont le fonctionnement est détaillé dans l'algorithme 5. On suppose dans un premier temps que la trajectoire $iniTraj$, créée préalablement, contient la trajectoire de l'avion cible entre $dir.w.start$ et $dir.w.end$. Pour chaque clone, l'algorithme tire la valeur α_i dans l'intervalle $dir.\alpha$ (ligne 2) afin de déterminer sa divergence finale avec l'avion cible puis la trajectoire $traj_i$ est créée par duplication de $iniTraj$ (ligne 4). Ensuite, des points de passage formant la trajectoire du clone sont créés à intervalles réguliers de $dir.w.start$ à $dir.w.end$. À chaque intervalle ts, l'algorithme calcule le pourcentage de la progression de la divergence, noté $pCoef$ (ligne 6) en divisant le temps écoulé depuis le début de l'altération $ts - dir.w.start$ par la durée de la divergence $dir.v$. La distance euclidienne représentant le décalage entre le clone et la cible est notée $step$. Elle est calculée en multipliant le coefficient de progression $pCoef$ par la divergence finale du clone, obtenue par $a_i * dir.k$ (ligne 7). Finalement, les latitude et longitude courantes sont calculées en utilisant $step$ ainsi que les fonctions trigonométriques $sinus$ et $cosinus$ et une direction d_i tirée dans la plage de données $dir.d$ (lignes 8-11). L'algorithme retourne la liste des trajectoires de tous les clones $trajs$.

La dernière étape consiste finalement à convertir les trajectoires des clones en messages ADS-B à injecter dans l'enregistrement initial. Pour chaque message envoyé par une cible

dans la fenêtre de temps $dir.w$ et pour chaque clone, le message initial est dupliqué et les propriétés dynamiques qu'il contient sont remplacées par les valeurs obtenues à partir des fonctions d'interpolation contenues dans $trajs$.

5.2.6/ LE REJEU D'UN ENREGISTREMENT

Cette attaque se base sur le rejeu de messages réellement émis auparavant par des avions et de ce fait s'abstrait des problèmes de réalisme. Un exemple typique d'une telle attaque consiste pour des individus malveillants à récolter des messages ADS-B durant un vol régulier un jour donné, puis quelques jours plus tard, à détourner un avion effectuant le même vol et à profiter de leur interaction physique avec le transpondeur pour envoyer des messages préalablement enregistrés. Cette manœuvre permet aux attaquants de détourner l'avion toute en gardant une trajectoire habituelle. Dans ce scénario, il n'y a pas besoin de calculer des valeurs réalistes afin de duper les mécanismes de détection étant donné que les faux messages ADS-B proviennent d'une source légitime. En conséquence, l'attaque par rejeu est certainement en mesure de tromper les systèmes de détection de FDIA basés sur une analyse du réalisme des données.

Cette attaque est formalisée par la directive d'altération $dir_{replay} = (w, t, r)$ où le nouveau paramètre $dir.r$ est l'enregistrement source duquel sont extraits les messages à rejouer, ces derniers étant les messages émis par les avions ciblés contenus dans $dir.t$.

La première étape de cette attaque est l'extraction des messages de l'enregistrement source. Ceci est fait durant le pré-traitement en itérant sur les messages de ce dernier puis en vérifiant si leur émetteur est contenu dans $dir.t$ et s'ils ont été émis dans la fenêtre de temps $dir.w$. Si c'est le cas, le message est marqué comme "rejoué". Le résultat du pré-traitement est la liste des messages ainsi marqués comme "à rejouer".

La seconde étape consiste à ajuster l'horodaté des messages extraits précédemment afin de les insérer correctement dans l'enregistrement cible. Pour chaque message msg à rejouer, l'algorithme lui assigne un nouvel horodatage au sein de l'enregistrement cible de la manière suivante : $msg.ts \leftarrow first\_ts + msg.ts - first\_rep\_ts + offset$ avec :

- $first\_ts$ est l'horodatage du premier message de l'enregistrement cible.
- $first\_reo\_ts$ est l'horodatage du premier message extrait de l'enregistrement source.
- $offset$ est une valeur en seconde exprimant à quel moment de l'enregistrement cible les messages extraits doivent être insérés.

La dernière étape est la fusion entre l'enregistrement cible et les messages extraits de l'enregistrement source. Elle est réalisée par la tâche TRAITER ACTION et son

Algorithme 6 : Tâche TRAITER ACTION dans le cadre d'un rejeu d'enregistrement

Input : $reps$: liste de messages extrait de l'enregistrement source
$recording$: enregistrement cible
Result : $al\_recording$: enregistrement altéré

```
1  cur ← rec.first
2  rp ← reps.first
3  while cur ≠ ∅ do
4      while reps ≠ ∅ ∧ cur.date > rp.date do
5          al_recording ← al_recording ∪ {rp}
6          reps ← reps − {rp}
7          rp ← reps.next
8      end
9      al_recording ← al_recording ∪ {cur}
10     cur ← rec.next
11 end
12 while reps ≠ ∅ do
13     al_recording ← al_recording ∪ {rp}
14     reps ← reps − {rp}
15 end
16 return al_recording
```

fonctionnement est détaillé dans l'algorithme 6. L'algorithme itère sur les messages de l'enregistrement cible (lignes 3 et 11). Si la date d'émission du premier message extrait rp est antérieure à la date du message courant cur (ligne 4) alors rp est insérée dans l'enregistrement altéré $al\_recording$ (ligne 5) et supprimé de la liste des messages extraits. Si la date d'émission de rp est postérieure à celle de cur, alors c'est ce dernier message qui est ajouté à l'enregistrement altéré. Si l'enregistrement cible a été intégralement parcouru et que la liste des messages extraits n'est pas vide, alors les derniers sont ajoutés à la fin de $al\_recording$.

5.3/ VALIDATION EXPÉRIMENTALE

Le domaine du contrôle aérien s'appuyant sur une des infrastructures les plus critiques de la planète, il est très difficile de démontrer l'efficacité de la génération de données en situation réelle, e.g., en envoyant les données altérées à travers un outil de détection de FDIA utilisé dans le domaine et de rendre les résultats publiques.

Pour remédier à ce problème et ainsi valider le bon fonctionnement de la génération d'altérations, la démarche expérimentale a consisté à reproduire la phase de validation d'une méthode de détection d'anomalies dans le protocole ADS-B à partir de la technique proposée par Habler et al. [39]. En effet, pour valider cette méthode de détection, l'auteur génère manuellement des anomalies plutôt simples. Le but ici est de montrer que toutes les anomalies utilisées par Habler et al. peuvent être reproduites avec les moteurs d'altération présentés dans ce chapitre et que les limites de leur modèle

de détection peuvent être atteintes grâce au réalisme proposé par notre génération de données altérées. Dans la première sous-section, le modèle de détection et ses caractéristiques sont introduits. La deuxième sous-section présente le jeu de données utilisé pour effectuer les altérations. Enfin la troisième et dernière sous-section présente l'analyse les résultats de cette expérimentation.

A noter que ces expérimentations ont été réalisées dans le cadre du projet ANR GeLeaD [2] (*Generate Learn and Detect*) en collaboration étroite avec Antoine Chevrot, un autre doctorant de l'équipe dont le sujet de thèse porte sur la détection des anomalies ADS-B avec des techniques de Machine Learning. Des extensions de ce travail d'expérimentation et de validation sont présentées au chapitre 8, en section 8.2, en utilisant notre chaîne outillée FDI-T complète. La présente section est focalisée sur la validation de la partie génération des données altérées de nos travaux.

5.3.1/ CONTEXTE EXPÉRIMENTAL : LE MODÈLE HABLER

Habler et al. propose une méthode de *Machine Learning* afin de distinguer les messages malveillants des messages légitimes au sein d'un enregistrement ADS-B [39]. Pour cela, les auteurs utilisent une architecture de *deep learning* (ou apprentissage profond) qualifiée de LSTM-based encoder-decoder [32] (LSTM pour *Long Short-Term Memory*, ou réseau récurrent à mémoire court et long terme). Ce modèle est composé de deux sous-modèles :

1. **L'encodeur** apprend à créer une représentation de ce qu'il reçoit en entrée. Dans cette expérimentation, cette représentation est basée sur les cellules LSTM, dont l'utilisation est pertinente dans l'analyse de séries temporelles [121, 73], i.e. des suites de messages ADS-B horodatés. L'encodeur est incapable d'apprendre seul car il n'a aucun moyen de vérifier si sa sortie est correcte par lui-même. C'est pourquoi il est couplé à un décodeur.
2. **Le décodeur** prend en entrée la sortie de l'encodeur, i.e. la représentation vectorisée des données ADS-B, et la décode de manière à obtenir une sortie attendue. Dans le cas du modèle d'Habler et al., la sortie attendue du décodeur correspond aux données d'entrée de l'encodeur. Ce type d'encodeur-décodeur est en fait appelé un auto-encodeur. Dans ce type de modèle, le décodeur a une architecture similaire, sinon identique, à celle utilisée dans l'encodeur.

Ce modèle est entrainé avec des vols complets isolés, i.e. du décollage à l'atterrissage, et découpé en fenêtres de 15 messages consécutifs. À partir de ces fenêtres de plusieurs messages, le modèle calcule la similarité entre son entrée et sa sortie, le but étant de maximiser cette similarité. Une fois le modèle entrainé avec des données légitimes,

2. https://projects.femto-st.fr/gelead/fr [Dernière visite : décembre 2020]

c'est-à-dire n'ayant subi aucune altération, un seuil d'anomalie est choisi en se basant sur la similarité, cette dernière représentant le score de reconstruction de l'architecture. S'il est entraîné correctement, le modèle pourra difficilement auto-encodé des données malicieuses. Si la prédiction diffère trop des données d'entrée, le seuil d'anomalie est alors dépassé, entraînant la détection de l'anomalie.

5.3.2/ DONNÉES D'EXPÉRIMENTATION

Afin de reproduire et d'entraîner le modèle et de tester son bon fonctionnement, plusieurs vols ont été collectés dans la base de données OpenSky [97]. Les données contiennent des vols européens avec des morphologies aussi variées que possible, e.g., des vols Madrid-Moscou utilisant différentes routes selon des contraintes météorologiques.

Après avoir entraîné le modèle, des scénarios d'altération sont créés dans le but de reproduire les anomalies définies dans l'article d'Habler et al. et utilisées ici pour valider la génération de donnée. Pour chaque anomalie, le scénario (i.e. l'ensemble de directives) équivalent est formalisé en utilisant la notation introduite dans la section 5.2 p. 74 pour formaliser les directives d'altération. En complément, les éléments suivants sont utilisés :

— $R_{original}$: Un enregistrement contenant un vol entre deux villes européennes.

— R_{source} : Un enregistrement contenant un vol entre deux villes européennes où $R_{original} \neq R_{source}$.

— a_1 : L'avion contenu dans $R_{original}$.

— a_2 : L'avion contenu dans R_{source}.

— t_n : Une durée en seconde telle que $t_n \leq t_n + 1$ et ne pouvant pas excéder la durée $R_{original}$.

À partir de ces paramètres et des directives d'altérations définies dans les sections précédentes, il est possible de formaliser les cinq anomalies utilisées dans les travaux de Habler et al.

Bruit aléatoire (RND) - Cette anomalie est générée en ajoutant du bruit aléatoire aux messages. Pour se faire, les valeurs des propriétés contenues dans le message original sont multipliées par un nombre aléatoire entre 0 et 2. L'altération **RND** nécessite d'effectuer une altération simple (section 5.2.1 p. 76). Le scénario S_{RND} est formalisé de

la manière suivante :

S_{RND}	=	$(R_{original}, \{dir_1\})$
dir_1	instance of	dir_{prop}
$dir_1.w$	=	$[t_0, t_1]$
$dir_1.t$	=	$\{a_1\}$
$dir_1.P$	=	$\{p_1\}$
p_1	=	$(ALTITUDE, 2, NOISE)$

Route différente (Route) - L'anomalie **Route** consiste à remplacer un segment entier de vol dans l'enregistrement initial par un vol issu d'une autre enregistrement. Cette anomalie nécessite la combinaison du rejeu d'un enregistrement (section 5.2.6 p. 85) et de la disparition d'avion (section 5.2.2 p. 78). Ce scénario d'anomalie est formalisé de la manière suivante :

S_{Route}	=	$(R_{original}, \{dir_2, dir_3\})$
dir_2	instance of	dir_{replay}
$dir_2.w$	=	$[t_0, t_1]$
$dir_2.t$	=	$\{a_2\}$
$dir_2.r$	=	R_{source}
dir_3	instance of	dir_{del}
$dir_3.w$	=	$[t_0, t_1]$
$dir_3.t$	=	$\{a_1\}$
$dir_3.n$	=	1

Chute graduelle (DOWN) - Cette anomalie consiste à faire dériver l'altitude. Ainsi, l'altitude des messages ciblés est multipliée par un multiple croissant de -25 pieds. De cette manière, l'altitude du premier message est réduite de 25 pieds, puis l'altitude du deuxième message de 50 pieds, etc. L'anomalie **DRIFT** nécessite un scénario contenant une altération simple. Il se formalise de la manière suivante :

S_{DRIFT}	=	$(R_{original}, \{dir_4\})$
dir_4	instance of	dir_{prop}
$dir_4.w$	=	$[t_0, t_1]$
$dir_4.t$	=	$\{a_1\}$
$dir_4.P$	=	$\{p_2\}$
p_2	=	$(ALTITUDE, -25, DRIFT)$

Dérive de vitesse (VEL) - Semblable à la précédente, cette anomalie est une dérive graduelle appliquée à la vitesse au sol. L'anomalie **VEL** nécessite donc également un

scénario contenant une altération simple qui se formalise de la manière suivante :

S_{VEL}	=	$(R_{original}, \{dir_5\})$
dir_5	instance of	dir_{prop}
$dir_5.w$	=	$[t_0, t_1]$
$dir_5.t$	=	$\{a_1\}$
$dir_5.P$	=	$\{p_3\}$
p_3	=	$(SPEED, 1.0, DRIFT)$

Blocage de message (BLK) - Dans cette anomalie, seul le premier message de chaque suite de cinq messages consécutifs est conservé afin de simuler une perte de qualité du signal par exemple. L'anomalie **BLK** est reproduite avec un scénario contenant une disparition d'avion. Il est formalisé de la manière suivante :

S_{BLK}	=	$(R_{original}, \{dir_6\})$
dir_6	instance of	dir_{del}
$dir_6.w$	=	$[t_0, t1]$
$dir_6.t$	=	$\{a_1\}$
$dir_6.n$	=	4

Grâce au directives d'altération, il est possible de reproduire toutes les anomalies utilisées dans l'article [39] associé au modèle Habler. En supplément, une modification de trajectoire, notée **TRJ**, a également été expérimentée sur le modèle de détection. Elle consiste à décaler la trajectoire d'un vol d'environ cent kilomètres durant plus de la moitié du vol (environ 60%).

5.3.3/ RÉSULTATS

Parmi les cinq anomalies utilisées par Habler et reproduites dans cette expérimentation, les résultats obtenus sont similaires à ceux de Habler pour quatre d'entre elles. La figure 5.3 montre par exemple le score d'anomalie pour l'anomalie de chute graduelle (DOWN).

On constate que le seuil d'anomalie (trait rouge) est placé à 0.2. L'altération commence au message 150 et le score d'anomalie augmente progressivement au fur et à mesure que l'altitude diminue. À partir d'une certaine différence avec le vol initial, le score d'anomalie dépasse le seuil fixé. Le résultat est similaire pour les autres anomalies présentées, et les résultats de Habler et al. sont bien reproduits.

Afin d'aller un peu plus loin et tester les limites du modèle ainsi que la qualité de la génération de données, une modification de trajectoire a également été réalisée :

Demi-tour (DT) - Dans cette anomalie, un avion part de Moscou pour aller à Madrid, à la

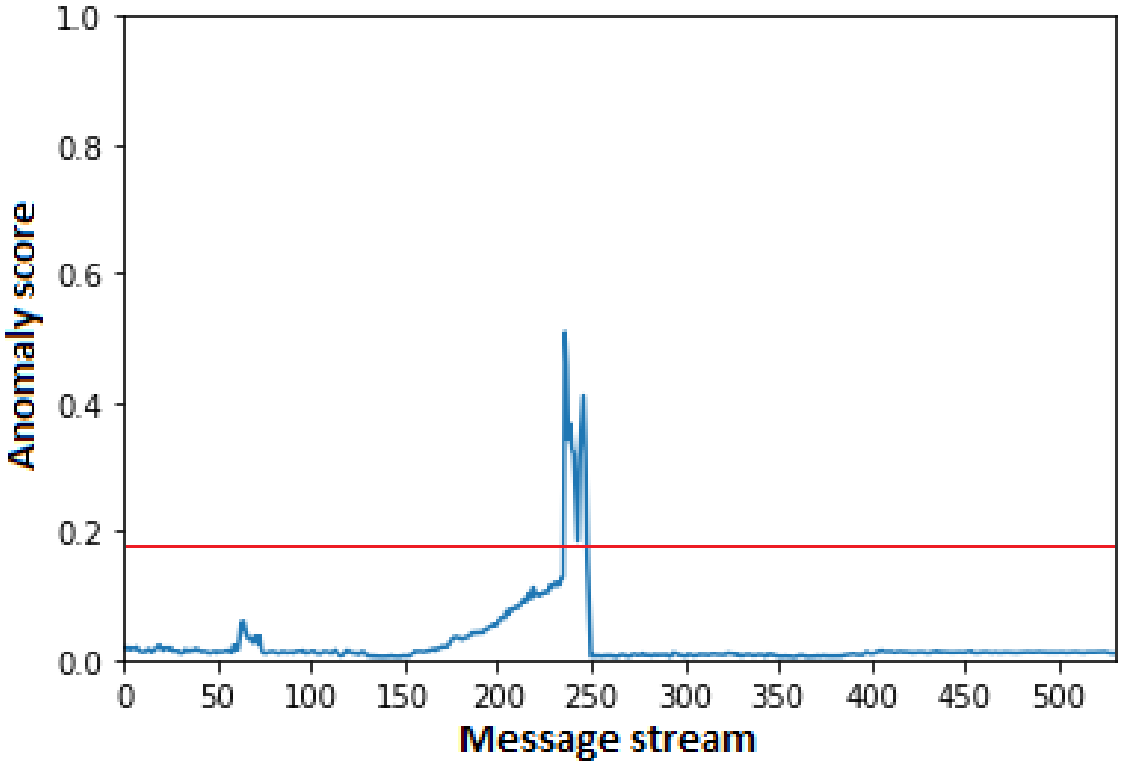


FIGURE 5.3 – Score d'anomalie sur l'anomalie **DRIFT**

moitié du vol, il fait demi-tour pour retourner à proximité de son point de départ :

S_{DT}	=	$(R_{original}, \{dir_7\})$
dir_7	instance of	dir_{vtm}
$dir_7.w$	=	$[t_0, t1]$
$dir_7.t$	=	$\{a_1\}$
$dir_7.\Omega$	=	$w_1 = (55.972, 36.922, 8125, t_{0.5})$

La figure 5.4 montre le demi-tour qui survient à la moitié du vol Moscou-Madrid. On remarque qu'avec un seul point de passage spécifié, la trajectoire du retour est lisse comparée à la trajectoire de l'aller.

Le résultat de la détection est visible dans la figure 5.5. On remarque qu'avec un seuil d'anomalie placé à 0.5, une anomalie est détectée à la moitié de l'enregistrement. Cela correspond au moment où l'avion effectue le demi tour, un mouvement que le modèle n'a jamais constaté sur les vol Moscou-Madrid. En revanche, le modèle ne détecte pas que la trajectoire générée est fausse puisqu'au delà du point de demi-tour, le score d'anomalie ne dépasse jamais le seuil fixé. Toutefois, à l'œil nu, on parvient facilement à distinguer la fausse trajectoire à cause de son aspect trop "lisse" par rapport à la trajectoire réelle. L'ajout de bruit aux trajectoires générées pourrait néanmoins corriger cet aspect.

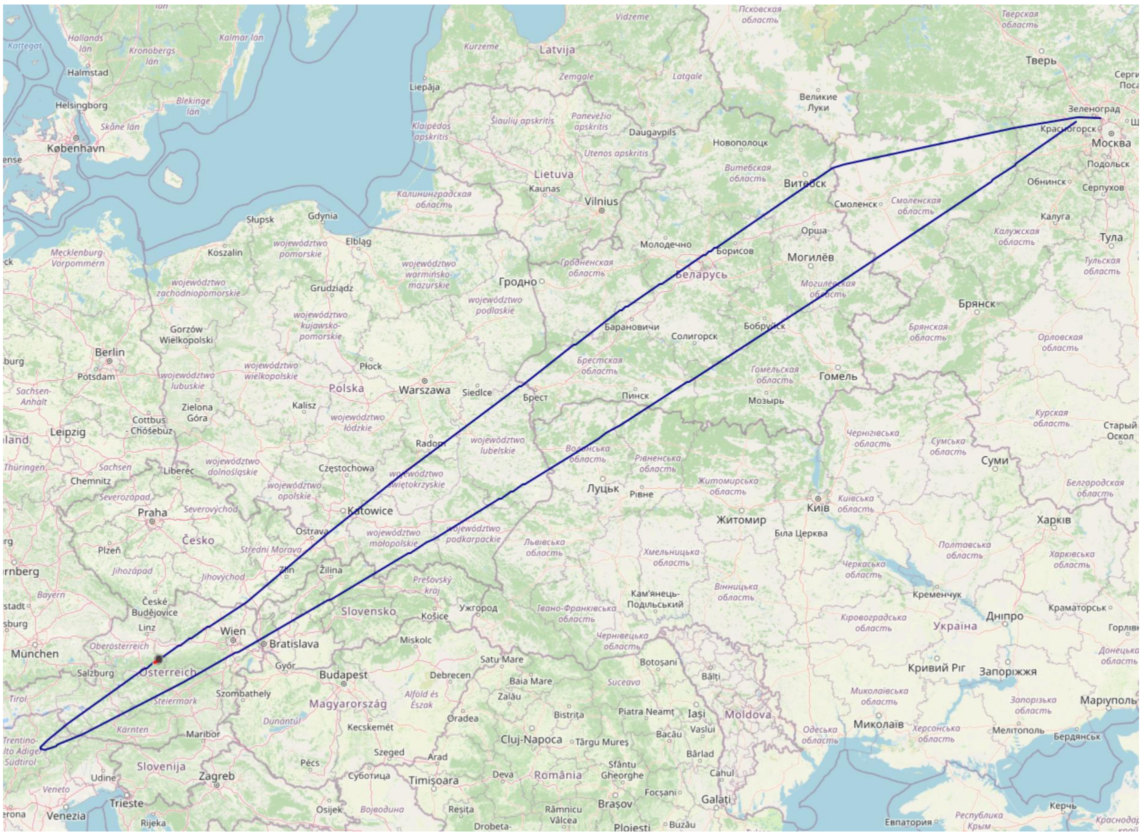


FIGURE 5.4 – Exemple de l'anomalie **DT** appliquée à un vol Moscou-Madrid

On constate donc que notre méthode de génération de données répond à un besoin dans les travaux sur la détection d'anomalie. Avec une telle méthode, des chercheurs comme Habler et al. n'ont pas besoin de développer des scripts peu réutilisables pour générer leurs anomalies. Toutefois, comme cela a été constaté en section 3.3.2 p. 51, la spécification d'un scénario à faire générer par un simulateur (ou un générateur) est un travail laborieux et les scénarios sont peu réutilisables et notre générateur de données ne fait, en l'état, pas exception. En effet, la spécification des directives d'altération fournies en entrée au générateur pose plusieurs problèmes :

— Elles sont au format XML et sont donc difficilement lisibles par un humain, ce qui a tendance à renforcer la barrière entre développeur et expert du domaine.

— Les fenêtres de temps et les cibles sont spécifiées "en dur" dans les directives, elles sont dépendantes de l'enregistrement sur lequel elles s'appliquent et, par conséquent, sont très peu réutilisables.

— Elles ne permettent pas la massification car elles n'utilisent que des valeurs discrètes.

— L'utilisateur doit lui-même analyser l'enregistrement nominal afin de déterminer les cibles et les fenêtres de temps des altérations. Ce travail peut être très long si

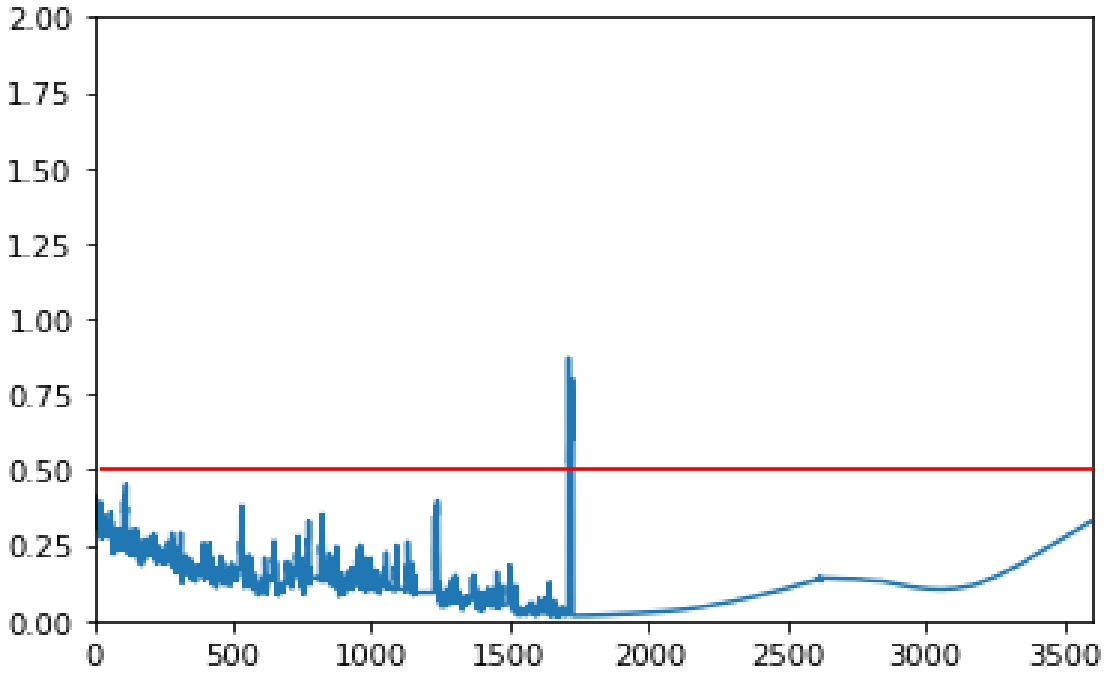


FIGURE 5.5 – Score d'anomalie sur l'anomalie **DT**

l'utilisateur recherche une situation précise sur laquelle réaliser une altération.

Toutes ces raisons soulignent la nécessité d'une méthode et d'un outillage de scénarisation qui soient plus facile d'utilisation et plus générique. La solution choisie a été la mise en place d'un ensemble de langages dédiés. Cet ensemble de langages dédiés constitue une contribution de cette thèse à laquelle nous avons consacré une part très importante de notre travail. C'est le sujet du chapitre suivant.

6

LANGAGES DÉDIÉS À LA SCÉNARISATION

Ce chapitre présente un ensemble de trois langages dédiés, ou *Domain Specific Languages* (DSL), utilisés pour faciliter la conception des scénarios de test par injection de fausses données. Cet ensemble de langages se compose d'un premier DSL de scénarisation qui permet la définition de schémas d'altération, d'un deuxième DSL utilisé pour la sélection des cibles des altérations, et d'un troisième DSL visant à définir les conditions de déclenchement des altérations. Dans ce chapitre, le développement de ces trois DSL est détaillé suivant la méthodologie proposée par Mernik et al. [75]. Cette méthodologie formalise le développement d'un DSL au travers des sept étapes déjà abordées dans la section 3.3 p. 50 : la décision, l'analyse du domaine, la conception, l'implémentation, le test, le déploiement et la maintenance.

Les sections de ce chapitre détaillent respectivement les cinq premières étapes du processus, à savoir, la décision, l'analyse du domaine, la conception, l'implémentation et enfin les activités de test. Les activités de déploiement et de maintenance ne sont volontairement pas abordées dans ce chapitre puisqu'elles sortent du cadre de la thèse.

6.1/ LA DÉCISION : POURQUOI DÉVELOPPER UN LANGAGE DÉDIÉ ?

La décision d'initier la création d'un DSL doit être étudiée de façon approfondie en raison des coûts de développement et de maintenance élevés de ces langages. En effet, il est crucial d'opposer ces coûts aux bénéfices potentiels d'un tel processus afin d'être certain que celui-ci soit fructueux. Mernik et al. ont défini une liste de motivations au développement d'un DSL ayant pour but d'apporter une assistance dans la prise de décision. Ces motivations peuvent être, par exemple, la simplification de la configuration d'un système ou l'élimination de tâches répétitives. Dans cette section se trouve une présentation du problème initial portant sur la scénarisation de cas de test pour les

systèmes de contrôle du trafic aérien. Ensuite, la décision de développer un DSL en réponse au problème initial est justifiée en se basant sur les motivations issues du travail de Mernik et al.

Comme vu précédemment dans la section 4.1.1 p. 58, un scénario est un ensemble de schémas d'altération destinés à être appliqués à un enregistrement, chaque schéma ciblant un sous-ensemble de capteurs. Un scénario simple, comme une attaque par fausse alarme, peut être constitué d'un seul schéma d'altération sans événements déclencheurs particuliers et ne ciblant pas d'avions selon des conditions particulières, et dont les modifications des propriétés se font de manière brute (pas de réalisme). La création des directives d'altérations correspondant à un schéma utilisé dans ce cas peut se faire en modifiant les données de façon manuelle, ou par l'écriture d'un script, en restant acceptable en terme d'effort.

La figure 4.2 et l'explication associée de la section 4.1.1 p. 58 précisent que les directives d'altérations sont utilisées par le générateur d'altérations [24], dont le fonctionnement est décrit dans le chapitre 5, afin de produire les données altérées servant au test. D'un point de vue technique, les directives d'altérations sont stockées dans un fichier au format XML que le générateur parcourt afin d'appliquer chaque directive à l'enregistrement initial.

Toutefois, les scénarios plus élaborés se composent de nombreux schémas d'altération déclenchés par des événements et ciblant les avions avec finesse, en se basant par exemple sur leurs propriétés dynamiques (altitude, vitesse, route, etc.). Cette complexité peut se traduire par la génération de centaines, voir de milliers de modifications de l'enregistrement, chacune méticuleusement définie avec des paramètres qui lui sont propres. De plus les campagnes de test impliquent fréquemment d'exécuter des mêmes cas de test avec des données d'entrées différentes afin de satisfaire certains critères de couverture. Il peut donc être nécessaire de pouvoir automatiquement faire varier les paramètres des directives à partir des paramètres du scénario. Un besoin est identifié pour une approche répondant à ce problème, facile à utiliser pour un non programmeur, tout en offrant un moyen formel de concevoir des scénarios d'altération.

Dans le domaine aérien, les experts raisonnent en terme de Situation Aérienne Générale (SAG), ou *Recognized Air Picture* (RAP), et d'événements liés aux avions. Ils considèrent les avions selon leurs propriétés dynamiques et imaginent les menaces de sécurité potentielles liées aux FDIA en tant qu'altérations des données de surveillance à haut niveau, i.e. les experts imaginent l'impact des FDIA sur les données plutôt que la manière dont sont réalisées ces attaques. Ces scénarios à risques sont définis de manière informelle et décrits en langage naturel.

En se basant sur l'observation précédente, un DSL apparaît être un moyen de conception efficace pour scénariser des FDIA. Les motivations évoquées par Mernik et al. viennent en outre renforcées cette idée :

— **Notation *user-friendly*** : Les experts sécurité décrivent habituellement les scénarios de FDIA dans un langage naturel et les équipes de développement sont chargées de leur traduction en cas de test. L'utilisation d'une syntaxe proche du langage naturel et de la terminologie du domaine d'application dans un DSL de conception de scénario permettra aux experts sécurité (ayant généralement des compétences modestes en programmation) d'exprimer facilement les scénarios qu'ils jugent pertinents.

— **Automatisation des tâches** : La définition manuelle des directives d'altération, en plus d'être une tâche répétitive et chronophage, requiert une analyse profonde de l'enregistrement afin de déterminer quels avions cibler et quand altérer leurs propriétés. Un DSL proposant une assistance dans la sélection des cibles ainsi que la possibilité de définir des événements permettra d'automatiser une part importante de l'analyse de l'enregistrement.

— **Système *front-end*** : L'utilisation d'un DSL permettra d'offrir aux experts une interface permettant de traduire des scénarios informels en scénarios interprétables par une machine afin d'obtenir automatiquement les cas de test correspondants.

6.2/ L'ANALYSE DU DOMAINE

Analyser un domaine d'application consiste à en définir les limites et à rassembler les connaissances nécessaires à sa compréhension, c'est-à-dire les différentes entités, les types de données, ainsi que les relations qui animent l'ensemble. Pour cela, diverses sources de données peuvent être utilisées : consultation d'experts, lecture de documents techniques ou d'études sur le domaine, etc.

De nombreuses méthodes d'analyse du domaine ont été développées lors des trois dernières décennies, chacune se basant sur différentes techniques d'extraction de l'information et différents degrés de formalisme. Voici quelques-unes d'entre-elles : DARE (*Domain Analysis and Reuse Environment*) [34], DSSA (*Domaine Specific Software Architectures*) [108], FODA (*Feature-Oriented Domain Analysis*) [42] ou encore FAST (*Family-Oriented Abstraction, Specification and Translation*) [22]. Ces méthodes, bien qu'ayant apporté de bons résultats dans la conception de langage, ont également montré qu'elles étaient complexes et très chronophages à mettre en place. De plus, il est rare de trouver des lignes directrices claires sur la manière d'exploiter l'information récoltée pour la phase de conception. Il est à noter de surcroît qu'il n'existe pas d'outil au point et maintenu fournissant une assistance dans le processus de conception d'un DSL : les développeurs de DSL se retrouvent forcés d'exploiter manuellement l'information extraite du domaine, augmentant ainsi la probabilité d'avoir une analyse incomplète ou erronée en raison du grand nombre d'interactions et d'entités interdépendantes [63]. L'utilisation de

ces méthodes reste donc assez limitée et l'analyse du domaine est souvent réalisée de manière informelle. En effet, pour la majorité des DSL trouvés dans la littérature et dont l'analyse du domaine est explicitement détaillée, il apparaît que celle-ci a été réalisée de manière informelle [55]. Et bien que le bénéfice d'une analyse informelle est un gain conséquent de temps par rapport au respect strict du processus d'une des méthodes susmentionnées, le risque d'une analyse du domaine erronée n'en reste pas moins plus élevé.

L'utilisation d'ontologies a pris de l'ampleur depuis quelques années en tant que support à l'analyse de domaine [107, 18]. La définition la plus répandue d'une ontologie est la suivante : "une spécification formelle et explicite d'une conceptualisation partagée" [106]. Une conceptualisation est la vue globale d'un domaine, bien que pouvant être abstraite et simplifiée. Le terme de partage renvoie ici à des connaissances consensuelles, comprises et admises par l'ensemble des experts du domaine. Une ontologie est dite formelle en référence à la possibilité d'être interprétée par une machine, et explicite car les propriétés, les concepts et les relations du domaine sont définis explicitement. Les ontologies sont utilisées dans des domaines variés, comme l'intelligence artificielle ou dans le domaine du web sémantique [14]. D'un point de vue technique, une ontologie est une hiérarchie de classes, un ensemble de relations entre ces classes, et un ensemble de relations entre les classes et les types de données [9]. Des travaux ont montré que les analyses de domaine basées sur une ontologie étaient au moins autant efficaces que les analyses de domaine plus traditionnelles [18]. Le bénéfice majeur des analyses de domaine basées sur une ontologie est que la conception ontologique est une pratique bien établie, avec des langages standardisés (RDF [58], OWL (Web Ontology Language) [72]) et supportée par de multiples outils commerciaux ou open-source (Protégé [1], Stardog Studio [2], Topbraid Composer [3]) facilitant le processus de création. Grâce aux notations formelles et à ces outils de support, la conception ontologique offre des capacités de raisonnement (e.g., avec un raisonneur comme FaCT++ [113]) et d'interrogation (*querying*). Il est ainsi possible de valider les ontologies, ce qui contribue à prévenir substantiellement les erreurs dans le développement du DSL. À noter également à ce propos, l'existence d'une approche visant à fournir des règles de conversion des axiomes d'une ontologie vers des règles de grammaires d'un DSL ainsi que les outils en mesure d'automatiser cette conversion [86].

L'analyse du domaine pour le développement du DSL de conception de scénarios d'altération présentée dans ce chapitre s'appuie sur une ontologie conçue avec OWL, l'un des langages d'ontologie le plus répandu, avec le support de Protégé, l'outil open-

1. https://protege.stanford.edu/ [Dernière visite : Décembre 2020]
2. https://www.stardog.com/studio/ [Dernière visite : Septembre 2020]
3. https://www.topquadrant.com/products/topbraid-enterprise-data-governance/ [Dernière visite : Septembre 2020]

source référence pour la conception d'ontologie [4]. Comme mentionné en section 6.1 p. 95, le DSL doit être, pour les experts du domaine, un moyen de définir formellement des scénarios d'altération pouvant être transformés automatiquement en un ensemble de directives d'altération. Voici les exigences quant aux fonctionnalités fournies par ce DSL (ces exigences sont issues de la consultation d'experts avioniques dans le cadre du projet SARCoS) :

— **EX-1.** Le DSL doit permettre la conception de scénarios qui couvre la taxonomie des attaques de type FDIA du domaine. Cela comprend aussi bien la possibilité de modifier, supprimer ou créer des messages transmis par les avions, que des altérations spécifiques au domaine (e.g., modification de trajectoire). De plus le DSL doit permettre la combinaison de différentes altérations.

— **EX-2.** Le DSL doit permettre de définir des critères de sélection des avions exprimés selon les propriétés statiques et dynamiques de ces derniers. Le but est de pouvoir cibler précisément un sous-ensemble d'avions affecté par une altération.

— **EX-3.** Le DSL doit permettre de définir des évènements basés sur les propriétés dynamiques d'un ou plusieurs avions. Ces événements étant ensuite utilisés comme déclencheurs de début ou de fin des altérations.

— **EX-4.** Le DSL doit être générique, dans le sens où les scénarios doivent être applicables à n'importe quel enregistrement, contrairement à la conception graphique de scénario, présentée en section 4.1.1 p. 58, où le scénario produit dépend de l'enregistrement à partir duquel il a été conçu.

— **EX-5.** Le DSL doit offrir des possibilités de massification, c'est à dire la capacité à générer massivement des cas de test à partir d'un seul scénario. Cela implique la possibilité de définir des listes ou des plages de valeurs dans les scénarios, plutôt que des valeurs discrètes. L'intérêt d'utiliser des valeurs continues est de pouvoir appliquer des stratégies combinatoires (pairwise, produit cartésien, fuzzing, etc.) afin de générer un grand nombre de variations d'un même scénario, augmentant de fait la couverture de celui-ci.

Les prochaines sous-sections présentent les classes modélisées dans l'ontologie, chaque classe ayant des relations avec d'autres classes. Elles sont traduites plus tard en règles de production et en terminaux pour la grammaire du DSL. L'analyse du domaine est menée en suivant les exigences susmentionnées et des références y sont faites dans les sous-sections suivantes afin de justifier l'existence des classes et de leurs relations. Il y a néanmoins une exception pour l'exigence **EX-4** : il n'existe pas de classe ou de relation spécifiquement liée à cette exigence, cette dernière servant plutôt de ligne directrice à l'analyse du domaine. Chaque sous-section détaille donc une classe

4. Le fichier OWL est disponible sur GitHub à l'adresse suivante : https://github.com/aymeric-cr/dsl-scenario/blob/master/fdit-dsl-ontology.owl

de l'ontologie, certaines de ces classes sont liées aux définitions de la section 4.1.1 p. 58 mais sont spécifiques au DSL. Par exemple, le scénario comme il a été défini en section 4.1.1 p. 58 retrouvera, dans cette section, son pendant formalisé au sein de l'ontologie. Afin d'éviter toute confusion, la notation des classes de l'ontologie se fait en *italique* et commence par une majuscule, e.g., *Scenario*, *Recording*, etc. En plus des classes, il est également question d'individus. Dans sa relation aux classes un individu peut être assimilé à un objet en programmation orienté objet, à la différence que cette dernière utilise l'hypothèse du monde clos : ce qui n'est pas explicitement défini n'est pas établi. À l'inverse, les ontologies utilisent l'hypothèse du monde ouvert : ce n'est pas parce qu'on ne connaît pas une information que cette information est fausse. Cette différence implique l'utilisation d'axiomes afin de restreindre l'appartenance d'un individu à un nombre limité de classes. Durant toute cette section, la définition des différents axiomes de l'ontologie est faite en notation préfixée. Pour une vision plus globale de l'ontologie, son arbre d'héritage est représenté en annexe A.1 à travers trois figures.

6.2.1/ LES SCÉNARIOS

On définit les individus membre de la classe *Scenario* comme une composition d'individus *Declarations* (i.e. de variables contenant des listes ou des plages de valeurs) et d'individus *Schema* avec les propriétés correspondantes respectives *hasDeclarations* et *hasSchemas*. Au sein d'un *Scenario*, la présence d'au moins un *Schema* est requise, contrairement à la présence de *Declaration* qui est optionnelle. Un *Scenario* doit être lié à exactement un *Recording* au travers de la propriété *hasRecording*. Ainsi, un *Scenario* sera exprimé en logique de description selon l'axiome suivant :

$$
\begin{aligned}
\text{Scenario} \sqsubseteq\ & \forall\, hasSchemas.Schemas && (1)\\
& \sqcap \forall\, hasRecording.Recording\\
& \sqcap \forall\, hasDeclarations.Declarations\\
& \sqcap \exists\, hasSchemas.Schemas && (2)\\
& \sqcap\ \geq 1\ hasRecording.Recording\\
& \sqcap\ \leq 1\ hasRecording.Recording
\end{aligned}
$$

Le sous-axiome (1) est un axiome de fermeture faisant office de restriction universelle (appelée *universal restriction* dans OWL) et agissant sur la propriété *hasSchemas*. Une restriction universelle a pour but de spécifier les types d'instances qu'une propriété peut contenir. Ici *hasSchemas* ne peut contenir que des instances de la classe *Schema*. Les mêmes restrictions sont définies pour les propriétés *hasRecording* et *hasDeclarations*. Ce type de restrictions est nécessaire avec OWL pour renforcer le typage à cause de l'hypothèse du monde ouvert. En effet, un individu est potentiellement membre de toutes les classes, sauf celles dont il est explicitement exclu. La partie (1) restreint donc la "portée" des propriétés (le type des individus qu'elle contient) quand leur "domaine" (le

type de l'individu dans laquelle elle est contenue) est un membre de la classe *Scenario*. De manière formelle, il est spécifié ici que l'ensemble des individus de la classe *Scenario* est inclus dans l'ensemble des individus dont la propriété *hasSchemas* vide ou contenant uniquement des individus de type *Schema*.

Le sous-axiome (2) définit une restriction de cardinalité (*cardinality restriction* dans OWL) sur la propriété *hasSchemas*. De manière formelle, cette restriction et les deux suivantes signifient qu'un individu de la classe *Scenario* est inclus dans l'ensemble des individus dont la propriété *hasSchemas* contient au moins un individu *Schema* et dont la propriété *hasRecording* contient exactement un individu *Recording*.

Les axiomes de fermeture similaires à ceux de la partie (1) sont implicites pour le reste de l'analyse du domaine car, bien qu'étant nécessaires à la validité de l'hypothèse du monde ouvert des ontologies, leur présence est évidente de par le nommage des classes et de leurs propriétés (e.g., *hasSchemas* contient des individus de type *Schema*). Ces restrictions universelles sont systématiquement ajoutées à toutes les classes de l'ontologie de sorte à alléger les sous-sections suivantes pour se focaliser davantage sur les restrictions de cardinalité. De plus, il est considéré par la suite que toutes les classes de même profondeur sont disjointes, à moins que le contraire ne soit précisé (e.g., la classe *TimeWindow* et ses sous-classes).

6.2.2/ LES SCHÉMAS D'ALTÉRATION

Les schémas d'altération constituent la colonne vertébrale du DSL de scénarisation. En effet ils font le lien entre objectifs d'altération haut-niveau que sont les scénarios, et les instructions d'altération formelles que sont les directives d'altération. Les schémas diffèrent des directives par leur capacité de sélection des cibles, par leur capacité à se déclencher en fonction d'événements, ainsi que par leur pouvoir de massification. Les schémas constituent un intermédiaire entre la nature abstraite des scénarios et des directives d'altération concrètes.

Un schéma est la spécification d'un type d'altération (création ou suppression d'avions, modification de propriétés, saturation, etc.) issu de la taxonomie des attaques. En accord avec l'exigence **EX-1**, il est possible de combiner plusieurs schémas au sein d'un scénario. Dans le modèle OWL, la classe correspondante est la classe *Schema*. Cette classe possède des sous-classes représentant chacune un type d'altération :

— *AlterationSchema* : modifie les propriétés d'un avion selon des valeurs fournies manuellement.

— *CreationSchema* : créé un faux avion (ou avion fantôme) par insertion de nouveaux messages dans l'enregistrement.

— *DeletionSchema* : supprime les avions ciblés pour une période de temps donnée.

— *SaturationSchema* : clone les avions cibles afin d'inonder le SUT de messages, chaque clone divergeant lentement de sa version originale. Dans le domaine aérien, cette divergence se matérialise par une superposition initiale laissant place à séparation géographique progressive de l'avion et de ses clones.

— *ReplaySchema* : rejoue un sous-ensemble de messages issu d'un enregistrement "source" au sein de l'enregistrement cible.

Les schémas d'altération étant nécessairement liés à un type d'altération, on considère la classe *Schema* comme abstraite. Cela se spécifie en logique descriptive par le biais d'un axiome couvrant (ou *covering axiom*) :

$$\begin{aligned} \text{Schema} \equiv\ & AlterationSchema \sqcup CreationSchema \\ & \sqcup DeletionSchema \sqcup ReplaySchema \\ & \sqcup SaturationSchema \end{aligned}$$

L'axiome ci-dessus met en place la notion d'héritage entre la classe *Schema* et ses sous-classes. Il spécifie que l'ensemble des membres de la classe *Schema* est équivalent à l'ensemble des membres des sous-classes de *Schema*. Cela a pour but d'empêcher un individu d'être membre de la classe *Schema* sans être également membre d'une de ses sous-classes. Par la suite, une classe abstraite est dite "couverte" par ses sous-classes lorsque ces dernières partagent les propriétés de la classe dont elles héritent.

6.2.3/ LES CIBLES

Les cibles des altérations sont représentées au travers de la classe *Target*. Cette classe est également une classe abstraite et possède deux sous-classes : *AnyNodeTarget* et *AllNodesTarget* représentant respectivement un seul avion cible ou un ensemble d'avions cibles.

Certains types de schéma ne ciblent aucun avion en particulier, c'est par exemple le cas du schéma visant à créer un nouvel avion, représenté à travers la classe *CreationSchema*. Il est donc nécessaire de définir la classe abstraite *TargetedSchema* et d'en faire une sous-classe de *Schema*. La classe *TargetedSchema* possède les mêmes sous-classes que *Schema* à l'exception de la classe *CreationSchema*. La classe possède la restriction de cardinalité suivante, qui exprime le fait que la propriété *hasTarget* contient exactement un individu *Target* :

$$\begin{aligned} \text{TargetedSchema} \sqsubseteq\ & Schema \\ & \sqcap\ \geq 1\ hasTarget.Target \\ & \sqcap\ \leq 1\ hasTarget.Target \end{aligned}$$

6.2.4/ LES FENÊTRES DE TEMPS

Les fenêtres de temps précisent le moment où s'exécute un schéma. Elles s'appuient sur un intervalle de temps composé d'une date de début et d'une date de fin. La classe abstraite *TimeWindow* représente les fenêtre de temps au travers deux sous-classes que sont les classes *StartTimeWindow* et *EndTimeWindow* qui possèdent chacune une propriété contenant un unique nombre entier positif, nommé respectivement *hasStartTime* et *hasEndTime*. Ces classes ne sont pas disjointes, cela signifiant qu'il est possible pour un individu d'être membre des deux classes à la fois, auquel cas il représente une fenêtre de temps possédant une date de début et une date de fin. Tous les schémas nécessitent de s'exécuter sur une plage de temps définie, c'est pourquoi l'absence de date de début ou de date de fin implique respectivement qu'un schéma s'applique dès le début de l'enregistrement ou jusqu'à la fin de celui-ci.

$$\text{Schema} \sqsubseteq \neg(> 1\ hasTW.TimeWindow)$$

6.2.5/ LES POINTS DE PASSAGE

Pour les schémas impliquant une création d'avion ou une modification de trajectoire, les utilisateurs doivent être en mesure de fournir une trajectoire composée d'un ensemble de points de passage. Un point de passage se compose de coordonnées (latitude et longitude), d'une altitude et d'un temps de passage. Par conséquent, il est nécessaire de disposer d'une classe *Coordinates* afin de représenter le système de positions géographiques. Cette classe est définie par deux propriétés *hasLatitude* et *hasLongitude*, les axiomes suivants sont appliqués à ses membres de façon à garantir qu'une coordonnée contienne exactement une latitude et une longitude de type décimal ($\mathbb{R}$) :

$$\begin{aligned}\text{Coordinates} \sqsubseteq\ & \geq 1\ hasLatitude.\mathbb{R} \\ \sqcap\ & \leq 1\ hasLatitude.\mathbb{R} \\ \sqcap\ & \geq 1\ hasLongitude.\mathbb{R} \\ \sqcap\ & \leq 1\ hasLongitude.\mathbb{R}\end{aligned}$$

De manière similaire, la classe *Waypoint* (représentant un point de passage) possède deux propriétés primitives de type entier positif ($\mathbb{N}+$), que sont *hasAltitude* et *hasTime*, ainsi qu'une propriété contenant exactement une position géographique. Les contraintes de cardinalité précise que chaque point de passage contient exactement un temps de

passage et une altitude :

$$\begin{aligned} \text{WayPoint} \sqsubseteq\ & \sqcap \geq 1\ hasCoordinates.Coordinates \\ & \sqcap \leq 1\ hasCoordinates.Coordinates \\ & \sqcap \geq 1\ hasAltitude.\mathbb{N}+ \\ & \sqcap \leq 1\ hasAltitude.\mathbb{N}+ \\ & \sqcap \geq 1\ hasTime.\mathbb{N}+ \\ & \sqcap \leq 1\ hasTime.\mathbb{N}+ \end{aligned}$$

Les modifications de trajectoire et les créations d'avions sont les deux types de *Schema* qui requièrent une trajectoire en tant que paramètre. Cette contrainte est exprimée de la manière suivante :

$$\text{TrajModSchema} \sqsubseteq \geq 1\ hasWayPoints.WayPoint$$

Si un seul point de passage est suffisant pour modifier une trajectoire existante, les trajectoires créées de toutes pièces requièrent au moins deux points de passages pour être valides. Cela s'exprime de la façon suivante :

$$\text{CreationSchema} \sqsubseteq \geq 2\ hasWayPoints.WayPoint$$

6.2.6/ LES PROPRIÉTÉS DES AVIONS

Comme vu en section 4.1.1 p. 58, les avions sont représentés par les propriétés qu'ils transmettent tout au long de l'enregistrement et qui peuvent être dynamiques (e.g., altitude, vitesse au sol) ou statiques (e.g., numéro ICAO, callsign). Dans la conception d'un schéma d'altération les propriétés occupent une place centrale. En effet, les valeurs des propriétés sont utilisées en tant que termes au sein de calculs arithmétiques, capturés dans l'ontologie par la classe *Expression* (voir section 6.2.7 p. 106), ou de comparaisons, capturées par la classe *PropertyEvaluation* (voir section 6.2.9 p. 107). De plus, les propriétés apparaissent également dans les paramètres de schéma (voir section 6.2.10 p. 108) en tant que cible d'une modification (e.g., augmenter la propriété $ALTITUDE$ de 1000 ft.).

La classe *AircraftProperty* représente les propriétés des avions. Elle est abstraite et couverte par deux sous-classes ayant la particularité d'être jointes : *AircraftPropertyType* et *AircraftPropertyScope*. Ces deux classes sont jointes car elles représentent deux aspects indépendants des propriétés :

— **Le type** : Statique ou dynamique, il est capturé par la classe *AircraftPropertyType* elle-même abstraite. Elle est couverte par deux sous-classes permettant de différencier les propriétés statiques des propriétés dynamiques, respectivement *StaticAircraftProperty* et *DynamicAircraftProperty*.

— **La portée** : Individuelle, partielle ou globale, elle est capturée par la classe

AircraftPropertyScope, également abstraite et couverte par trois sous-classes permettant de différencier les trois sortes de portée : *SingleAircraftProperty*, *PartialAircraftProperty*, *GlobalAircraftProperty*.

Le but d'une portée est de créer des expressions faisant référence aussi bien à un avion unique qu'à un ensemble d'avions. Cette notion de portée permet de répondre à l'exigence **EX-3** qui stipule que les événements peuvent être basés sur les propriétés d'un ou de plusieurs avions. Les trois portées possibles se distinguent par le nombre d'avions qu'elles englobent :

— Individuelle : Elle fait référence à la valeur d'une propriété pour un seul avion.

— Globale : Elle fait référence à l'ensemble des valeurs d'une propriété pour l'ensemble des avions de la SAG. Ainsi, contrairement à la portée individuelle, la propriété renvoie à une liste de valeurs, et non à une valeur unique.

— Partielle : Elle est similaire à la portée globale dans le fonctionnement mais fait référence à un sous-ensemble de la SAG satisfaisant un filtre (voir section 6.2.11 p. 109).

Selon le contexte d'utilisation des propriétés (e.g., dans une expression arithmétique ou dans un paramètre de schéma), les portées n'ont pas lieu d'être. C'est par exemple le cas avec l'utilisation d'une propriété en tant que paramètre d'un schéma. En effet, dans ce contexte, il est inutile d'associer une portée à une propriété car cette dernière fait référence à la qualité propre de l'avion et à ses potentielles valeurs. Il est nécessaire d'associer une portée à une propriété uniquement quand on fait référence à la valeur de cette dernière.

Les classes *AircraftPropertyType* et *AircraftPropertyScope* étant jointes, un individu peut être un membre des deux classes. Plus précisément un individu peut être membre d'une sous-classes de chacune de ces deux classes. Par exemple, un individu représentant une propriétés statique à portée partielle est membre de *PartialAircraftProperty* et de *StaticAircraftProperty*.

Au regard des propriétés des avions, comme l'altitude, le callsign, le squawk, etc., il a été décidé de les représenter en tant qu'individus plutôt que par un nouvel ensemble de sous-classes dans l'ontologie. Cela permet d'alléger l'ontologie tout en gardant une certaine flexibilité face à un potentiel ajout de nouvelles propriétés. À noter que le nom des propriétés des avions ne fait pas partie de la grammaire résultant de cette ontologie. Les propriétés étant très spécifiques au domaine, ce choix est en accord avec la volonté de généricité de l'approche décrite dans ce manuscrit.

6.2.7/ LES EXPRESSIONS ARITHMÉTIQUES

Les expressions arithmétiques sont utilisées afin de représenter les opérations arithmétiques communes (somme, produit, etc.). On les retrouve aussi bien dans la comparaison des propriétés (voir section 6.2.9 p. 107) que pour spécifier les paramètres d'altération des schémas (voir section 6.2.10 p. 108). Dans le premier cas, la valeur d'une propriété est comparée à une expression, tandis que dans le deuxième cas le résultat d'une expression est assigné à une propriété dans le cadre d'une altération.

Les expressions s'appuient soit sur des valeurs directement fournies par les utilisateurs, soit sur des références à des déclarations de variables (voir section 6.2.13 p. 111), soit sur des valeurs issues des propriétés des avions. Elles sont représentées, dans l'ontologie, par la classe abstraite *Expression* à travers trois sous-classes :

— *TwoOperandsExpression* : classe abstraite représentant les quatre opérateurs arithmétiques (addition, soustraction, multiplication, et division) par quatre sous-classes. Chacune possède deux propriétés obligatoires, *hasLeftOperand* et *hasRightOperand*, contenant chacune une *Expression*. Ainsi, la classe *TwoOperandsExpression* représente une imbrication d'expressions.

— *NegExpression* : classe représentant la négation arithmétique d'une expression (e.g., -(3 * 5) où 3 * 5 est l'expression dont la négation est exprimée). L'unique propriété de cette classe est *hasSubExpression* et doit obligatoirement contenir une *Expression*.

— *AtomicExpression* : classe abstraite représentant toutes les valeurs possibles pouvant être membre d'une expression. Cela comprend des valeurs primitives de différents types comme des nombres entiers, des nombres flottants, des booléens ou des chaînes de caractères, chaque type possible étant représenté à travers une sous-classe possédant la propriété *hasValue* contenant une valeur de type correspondant. En plus des valeurs primitives, un membre d'expression peut également être une *AircraftProperty*, ce type de membre est également représenté au travers d'une sous-classe possédant la propriété *hasAircraftProperties*. Finalement, un membre d'expression peut faire référence à une déclaration, i.e. une variable contenant une liste ou une plage de valeurs. L'inclusion de références à des déclarations dans les expressions permet de satisfaire l'exigence **EX-5** traitant des capacités de massification.

6.2.8/ LES ZONES

Dans certains cas, un filtre (voir section 6.2.11 p. 109) ou un déclencheur (voir section 6.2.12 p. 110) peuvent nécessiter des conditions basées sur des critères

géographiques pour, par exemple, sélectionner les avions entrant ou sortant d'une zone donnée pendant l'enregistrement. De telles conditions seraient fastidieuses à définir par des comparaisons arithmétiques entre les propriétés dynamiques de l'avions (latitude, longitude et altitude) et des valeurs choisies manuellement. Pour rendre plus simple la création de conditions basées sur des critères spatiaux, la notion de zone a été introduite dans le DSL. Une zone est un prisme défini par un ensemble d'au moins trois coordonnées géographiques et par une plage d'altitude. Les zones sont représentées par la classe *Zone*. Comme pour certaines classes précédentes, la classe *Zone* possède des restrictions de cardinalité de manière à garantir qu'une zone se construit à partir d'un au moins trois coordonnées, et est bornée par exactement une altitude minimale et exactement une altitude maximale entières ($\mathbb{N}$). Cela se concrétise dans l'axiome suivant :

$$\begin{aligned} \text{Zone} \sqsubseteq\ & \sqcap \geq 3\ hasCoordinates.Coordinates \\ & \sqcap \geq 1\ haslowerAltitude.\mathbb{N}+ \\ & \sqcap \leq 1\ haslowerAltitude.\mathbb{N}+ \\ & \sqcap \geq 1\ hasupperAltitude.\mathbb{N}+ \\ & \sqcap \leq 1\ hasupperAltitude.\mathbb{N}+ \end{aligned}$$

De manière similaire aux expressions, les zones sont utilisées dans les comparaisons de propriété de manière à vérifier la présence d'un ou plusieurs avions dans une zone. Ce point est précisé dans la section 6.2.9 p. 107.

6.2.9/ LES COMPARAISONS DE PROPRIÉTÉS

Les filtres et les déclencheurs nécessitent de faire des comparaisons entre le résultat d'une expression et la valeur d'une propriété. Ces comparaisons de propriétés peuvent être liées par des disjonctions (OU logique) ou des conjonctions (ET logique). Il est également permis de les nier (négation logique). La classe *PropertyEvaluation*, qui représente ces comparaisons, est abstraite et possède quatre sous-classes :

— *AndOrPropertyEvaluation* représente les conjonctions ou les disjonctions entre des comparaisons de propriétés, ce qui en fait une classe d'imbrication. Les membres de cette classe sont liés à d'autres membres de *PropertyEvaluation* via deux propriétés, *hasLeftSubPropEval* et *hasRightSubPropEval* (exactement une instance par propriété). Afin de différencier les conjonctions des disjonctions, cette classe est abstraite et couverte par les deux sous-classes *AndPropertyEvaluation* et *OrPropertyEvaluation*.

— *NotPropertyEvaluation* représente la négation d'une comparaison. Les membres de cette classe sont liés à exactement un membre de *PropertyEvaluation* au travers de la propriété *hasSubPropEval*.

— *InZonePropertyEvaluation* représente une comparaison visant à déterminer la

présence ou non d'un avion dans une zone. Cette classe possède la propriété *hasZone* contenant obligatoirement un membre de la classe *Zone*.

— *StraightPropertyEvaluation* représente une comparaison atomique entre la valeur d'une propriété et le résultat d'une expression, i.e. sans conjonction, disjonction ou négation. La classe possède deux propriétés *hasProperty* et *hasPropExpression* contenant obligatoirement et respectivement exactement un membre de la classe *AircraftProperty* et un exactement un membre de la classe *Expression*. Afin de représenter tous les opérateurs de comparaison possible (égalité, infériorité et supériorité, éventuellement stricts) la classe *StraightPropertyEvaluation* est abstraite et couverte par une sous-classe pour chaque opérateur.

6.2.10/ LES PARAMÈTRES DE SCHÉMA

Les objectifs d'un schéma d'altération sont spécifiés par un ensemble de paramètres. Ils peuvent concerner par exemple les propriétés des avions à modifier dans le cas d'une altération simple, ou le nombre de faux avions à créer dans le cas d'une saturation. Les paramètres de schéma partagent des similarités avec les comparaisons de propriétés dans le sens où les deux se composent d'une propriété, d'une expression et d'un opérateur. Toutefois, l'objectif d'un paramètre de schéma étant l'attribution d'une valeur à une propriété, il n'y a pas besoin de conjonctions, de disjonctions ou de négations. De plus, l'attribution d'une valeur ne nécessite pas l'utilisation des différents opérateurs de comparaison mais seulement un opérateur d'attribution (=). Chaque type de schéma possède un ensemble de paramètres qui lui est propre.

La classe *SchemaParameter*, représentant les paramètres de schéma, est abstraite et couverte par les deux sous-classes suivantes :

— *AlterationParameter* permet de modifier la valeur des propriétés statiques ou dynamiques des avions, c'est-à-dire attribuer la valeur d'une expression à une propriété d'avion. Cette classe possède deux propriétés : *hasAircraftProperty* qui contient un membre de la classe *AircraftProperty*, et *hasExpression* qui contient un membre de la classe *Expression*. Chacune des propriétés doit contenir exactement un individu.

— *CreationParameter* permet l'ajout de faux avions (ou avions fantômes) à la SAG. Au-delà de fournir une liste de points de passage, la création d'un avion fantôme peut également nécessiter l'attribution de valeurs à certaines de ses propriétés statiques (callsign, ICAO, etc.). La classe *CreationParameter* possède des propriétés similaires à la classe *AlterationParameter* mais le contenu de la propriété *hasAircraftProperty* est restreint à un membre de la classe *StaticAircraftProperty*.

Les spécificités des différents types de schémas (listées dans la section 6.2.2 p. 101)

sont capturées par des propriétés au sein des classes qui représentent ces schémas. Par exemple, en regardant le cas du schéma représenté par la classe *ReplaySchema* (voir section 6.2.2 p. 101), on remarque que son paramètre principal est le choix d'un enregistrement source à partir duquel extraire un ensemble d'avions. Or ce paramètre est déjà représenté au sein de la classe *ReplaySchema* par la propriété *hasRecording* contenant un membre de la classe *Recording*. De manière similaire, une saturation est représentée par la classe *SaturationSchema*, or cette dernière possède la propriété *hasNumberOfCopies* contenant un entier positif et représentant le nombre de clone à créer pour chaque avion cible.

6.2.11/ LES FILTRES

Il est fréquent pour une FDIA de ne cibler qu'un sous-ensemble des avions présents dans l'enregistrement. Ainsi, les schémas ont la possibilité de définir des critères de sélection via l'utilisation de filtres, comme spécifié dans l'exigence **EX-2**. Les avions ne respectant par ces critères ne sont pas affectés par le schéma d'altération. Le filtre peut être trivial : "cibler seulement les avions de telle compagnie". Dans d'autres cas, il peut s'avérer plus précis et plus complexe à évaluer : "cibler les avions qui n'excède jamais telle altitude et qui volent à telle vitesse dans une telle zone."

Les filtres permettent ainsi aux utilisateurs d'exprimer des conditions liées aux propriétés des avions et de combiner ces conditions avec des conjonctions ou des disjonctions logiques. L'objectif d'un filtre est de générer la liste de tous les avions satisfaisant les conditions exprimées. Pour cela, ils intègrent la notion de temps grâce à l'introduction de deux opérateurs issus de la Logique Temporelle Linéaire (LTL) [12] :

— $ALWAYS$: Les expressions soumises à cet opérateur doivent être vraies durant tout l'enregistrement.

— $EVENTUALLY$: Les expressions soumises à cet opérateur doivent être vraies au moins une fois durant l'enregistrement.

Seules les sous-classes de *TargetedSchema* peuvent être liées à des individus membre de la classe *Filter* via la propriété *hasFilters*. À noter qu'il n'existe aucune restriction de cardinalité sur cette propriété.

Un filtre est la combinaison d'un identifiant et d'une expression (*Expression*) associée à un opérateur temporel. Par conséquent, une classe nommée *FilterExpression* capture la partie expression du filtre. La classe *Filter* possède donc les propriétés *hasName* et *hasFilterExpression* représentant respectivement l'identifiant et l'expression du filtre et possédant chacune une restriction de cardinalité de 1.

La classe *FilterExpression* est abstraite et couverte par trois sous-classes qui se distinguent par une fonctionnalité spécifique des filtres :

— *AndOrFilterExpression* est une classe abstraite possédant deux sous-classes pour représenter les conjonctions et les disjonctions des expressions de filtre. Les opérandes droite et gauche sont liées à la classe au travers des deux propriétés *hasLeftSubFilterExpr* et *hasRightSubFilterExpr* contenant chacune exactement un membre de *FilterExpression*.

— *NotFilterExpression* est une classe représentant la négation logique d'une expression de filtre. Elle est liée à exactement une expression de filtre via la propriété *hasSubFilterExpr*.

— *TemporalFilterExpression* est une classe abstraite représentant l'utilisation d'un des deux opérateurs temporels vu précédemment, $ALWAYS$ et $EVENTUALLY$, mais également leur absence, appelée *None*. En effet, les propriétés statiques étant constantes, elles n'évoluent dans aucun contexte temporel. Chacune des trois possibilités est modélisée par une sous-classe. Cette classe possède la propriété *hasPropEval* contenant un membre de *PropertyEvaluation* afin de modéliser sur quelle comparaison de propriété est appliqué l'opérateur temporel.

6.2.12/ LES DÉCLENCHEURS D'ÉVÉNEMENT

Les avions ciblés par un schéma sont sélectionnés au moyen de filtres. L'étape suivante dans la conception d'un schéma d'altération est le choix des instants de l'enregistrement où les altérations sont à mettre en œuvre. En opposition aux fenêtres de temps, les déclencheurs expriment des évènements en fonction desquels les avions cibles doivent être affectés par une altération, comme le veut l'exigence **EX-3** qui stipule que le DSL doit inclure la notion d'événement. Les déclencheurs sont basés sur des événements tels que "à partir du moment où l'événement E survient" ou "jusqu'à ce que l'événement E survienne".

L'existence des portées, expliquées en section 6.2.6 p. 104, se justifie principalement par leur utilisation dans le contexte des événements. En effet, les portées permettent de déclarer des événements basés sur le comportement d'un à plusieurs avions. Le tableau 6.1 donne les effets de chacun des trois types de portée pour l'événement : "X vole au-dessus de 10000 pieds", avec X comme le(s) avion(s) sujet(s) de l'événement.

Si les filtres ont pour but de savoir si un événement est survenu, les déclencheurs visent plutôt à déterminer les moments où cet événement s'est produit. Pour cette raison, les déclencheurs utilisent des opérateurs temporels différents de ceux utilisés dans les filtres et qui répondent mieux à leur besoin. Cette différence entre filtres et déclencheurs se traduit par le résultat de l'interprétation d'un déclencheur. En effet plutôt que de créer une liste d'avion cible, l'interprétation d'un déclencheur produit, pour chaque avion de l'enregistrement, une liste d'intervalles de temps durant lesquels les conditions de

TABLE 6.1 – Effet des portées sur les événements.

Portée	Description	Évènement
Individuelle	Concerne chaque avion cible à titre individuel	L'avion volent au-dessus de 10000 pieds.
Partielle	Concerne un sous-ensemble des avions de l'enregistrement	Les avions *Air France* volent au-dessus de 10000 pieds.
Globale	Concerne tous les avions de l'enregistrement	Tous les avions volent au-dessus de 10000 pieds.

déclenchements sont vérifiées.

Trois opérateurs de déclenchement ont été alors introduits dans le langage afin de déterminer l'action (altérer ou ne pas altérer) associée à une condition de déclenchement évaluée vraie :

— $WHEN$: les altérations sont appliquées quand la condition de déclenchement est vraie.

— AS_SOON_AS : les altérations sont appliquées à partir du moment où la condition de déclenchement est vraie et ce jusqu'à la fin de l'enregistrement.

— $UNTIL$: les altérations sont appliquées dès le début de l'enregistrement, et jusqu'au premier moment où la condition de déclenchement est vraie.

Les déclencheurs sont représentés dans l'ontologie d'une manière similaire aux filtres : un déclencheur est la combinaison d'un identifiant et d'une expression (*Expression*) associée à un opérateur temporel. Une classe *TriggerExpression* capture la partie expression du déclencheur. À la manière de *FilterExpression* (voir section 6.2.11 p. 109 elle est abstraite et couverte par les sous-classes : *AndOrTriggerExpression*, *NotTriggerExpression* et *TemporalTriggerExpression* qui permettent respectivement les conjonctions et disjonctions, les négations, et enfin, le choix de l'opérateur de déclenchement.

6.2.13/ LES DÉCLARATIONS DE VARIABLES

L'exigence **EX-5** concerne la possibilité de massification des scénarios, c'est à dire la génération d'un grand nombre de cas de test à partir d'un seul scénario. Il a été identifié en amont que la capacité à massifier un scénario nécessite l'utilisation d'ensembles de valeurs (e.g., listes ou plages de valeurs) plutôt que de valeurs uniques. C'est un procédé déjà utilisé par Menzel et al. [74] dans le domaine de la voiture autonome afin d'obtenir plusieurs versions d'un même scénario à risques grâce à l'utilisation de valeurs continues plutôt que de valeurs discrètes. Menzel et al. classent les scénarios selon trois niveaux d'abstractions : fonctionnel, logique et concret. Dans notre cas, seuls les niveaux logique et concret nous intéressent : le premier utilise des valeurs continues (plages de données)

tandis que le second utilise des valeurs discrètes. Les scénarios concrets sont issus de la concrétisation de scénarios logiques. Cette étape consiste à générer une multitude de scénarios concrets en choisissant, selon différentes stratégies combinatoires, des valeurs discrètes parmi les valeurs continues utilisées dans un scénario logique.

Afin de satisfaire l'exigence **EX-5**, ce mécanisme de concrétisation est supporté par l'ensemble de langages dédiés que nous proposons, son implémentation est détaillée dans la section 6.4.4 p. 127. Dans notre contexte, on ne parle pas de scénarios logiques mais de scénarios abstraits. Afin de créer ces scénarios abstraits, les utilisateurs doivent être en mesure de définir des variables contenant des listes ou des plages de valeurs pour ensuite faire référence à ces variables au sein d'expressions.

De façon concrète, l'ontologie capture les déclarations de variable grâce à la classe *Declaration*. Elle est abstraite et possède la propriété *hasName* contenant une chaîne de caractères pour représenter son nom. La classe est couverte par les deux sous-classes suivantes :

— *ListDeclaration* est une classe abstraite qui possède une sous-classe pour chaque type primitif (entier, flottant, etc.). Chacune de ses sous-classes possède une propriété avec une restriction de cardinalité minimum de 1 et sans limite maximum. Cette propriété contient une donnée du type correspondant à la sous-classe dans laquelle elle se trouve, e.g., la sous-classe *StringListDeclaration* possède la propriété *hasStringValues* contenant une liste de chaîne de caractères.

— *RangeDeclaration* est également une classe abstraite couverte par deux sous-classes, une pour les nombres entiers et une pour les valeurs flottantes. Les deux sous-classes possèdent deux propriétés représentant les bornes inférieure et supérieure de la plage de valeurs.

6.3/ LA CONCEPTION DU LANGAGE DÉDIÉ

L'activité de conception correspond à la création de la syntaxe et de la sémantique du langage basée sur l'ontologie résultante de l'analyse du domaine. Dépendamment de l'ontologie, plusieurs patrons de conception (*design pattern*) peuvent être utilisés pour construire la grammaire du langage [75]. Il existe une condition préalable à la conception du langage qui consiste à étudier l'existence de langages et de grammaires utilisant déjà des constructions similaires et pouvant être réutilisées. En s'appuyant sur cette question, le patron de conception *piggybacking* (littéralement "ferroutage") propose d'inclure partiellement la grammaire d'un langage existant au langage en conception. Le *piggybacking* peut considérablement réduire les coûts de conception et d'implémentation, en plus de possiblement rendre le langage plus familier pour les utilisateurs. De manière similaire, les patrons *specialization* et *extension* consistent respectivement à restreindre

(e.g., en supprimant des règles de production) et à étendre (e.g., en ajoutant des règles de production ou des terminaux) un langage existant. Enfin, si le DSL n'a aucun rapport avec un langage existant, c'est le patron de conception *invention* qui est utilisé puisqu'il consiste à créer le langage à partir de zéro.

Plusieurs aspects du DSL présenté dans ce chapitre sont des *piggyback*, des spécialisations ou des extensions d'autres langages et constructions, à savoir :

— les expressions arithmétiques avec associativité, commutativité et distributivité

— les comparaisons booléennes

— les disjonctions et conjonctions pour les comparaisons booléennes.

— les opérateurs de la logique LTL

— la déclaration de variable

À l'inverse, la définition des schémas d'altération est entièrement inventée.

Comme mentionnée dans la section précédente, il existe des approches pour transformer automatiquement les ontologies OWL en grammaire sans contexte (ou *context-free grammars*) [85]. Toutefois, pour ce travail, la transformation a été faite manuellement. En effet, il apparaît qu'une importante partie du DSL peut être empruntée à des notations existantes tandis que les aspects restants, i.e. la définition des schémas, sont aisément convertibles en règles de production par leur aspect simple et plat (pas d'imbrication). Cependant, la formalisation de cette ontologie a été d'une aide précieuse, notamment pour l'identification des fonctionnalités réutilisables à partir d'autres langages (piggyback), des sous-langages autonomes (*standalone DSL*) et des composants réutilisables (utilisé à travers plusieurs sous-langages autonomes).

L'ontologie montre que deux classes peuvent potentiellement être converties en sous-langages autonomes, il s'agit des classes *Filter* et *Trigger*. Or, une des exigences du DSL (**EX-4**) porte sur la généricité : la présence de deux sous-langages autonomes pour définir les filtres et les conditions de déclenchement contribue à satisfaire cette exigence. En effet, avec ces deux-sous langages autonomes, les utilisateurs ont la possibilité de créer des filtres et des conditions de déclenchement indépendamment de tout schéma d'altération, permettant ainsi l'utilisation de ces éléments au sein de plusieurs de ces schémas. De plus, les efforts de conception des schémas sont répartis sur plusieurs tâches plus simples au lieu d'une seule tâche globale et de fait, plus complexe, incluant l'intégralité de la conception d'une altération. Le processus de conception se divise ainsi en trois étapes :

— Définition des filtres

— Définition des déclencheurs d'altération

— Définition des schémas d'altération

L'ontologie a également montré que certaines entités doivent être utilisées au sein de plusieurs sous-langages, c'est le cas par exemple des classes *Expression*, *AircraftProperty*, *PropertyEvaluation*. Ainsi, plutôt que dupliquer les règles de production correspondantes dans les grammaires des divers DSL autonomes, des grammaires abstraites ont été créées pour les expressions et les comparaison de propriétés. Une grammaire abstraite ne se traduit pas en sous-langage autonome mais elle vise, dans un effort de factorisation, à être intégrée et étendue au sein d'un sous-langage.

Les sous-sections suivantes décrivent les deux grammaires abstraites pour les expressions et les comparaisons de propriétés, ainsi que le processus de conception pour les trois sous-langages autonomes dédiés respectivement aux filtres, aux déclencheurs et aux schémas d'altération.

6.3.1/ LA GRAMMAIRE DES EXPRESSIONS DE BASE

La Grammaire des Expressions de Base (ou BEG pour *Base Expression Grammar*) est une grammaire dédiée aux opérations arithmétiques. Elle trouve son origine directe dans la classe *Expression* et ses sous-classes dans l'ontologie. La BEG est présentée en figure 6.1 dans la forme étendue de Backus-Naur (ou EBNF pour *Extended Backus-Naur Form*). Les expressions ne sont décrites dans l'ontologie que sous forme de listes. Il serait fastidieux de produire des règles de grammaire à partir des classes et de leurs relations, en particulier pour inclure des propriétés arithmétiques telles que l'associativité, la commutativité et la distributivité. De plus, les opérations arithmétiques étant exprimées à l'aide d'une notation très répendue, la BEG reprend donc largement les grammaires arithmétiques existantes [5].

```
add           : :=   mult (( + | - ) mult)*
mult          : :=   unary (( * | / | mod ) unary)*
unary         : :=   - ? atomic
atomic        : :=   ( add )
                 |   value
                 |   aircraft_prop
aircraft_prop : :=   (ac_scope .) ? ac_charac
value         : :=   <integer> | <float> | <string>
ac_charac     : :=   ( [A − Z] | '_' )+
ac_scope      : :=   RAP | AIRCRAFT " <identifier> "
```

FIGURE 6.1 – EBNF de la Base Expression Grammar (BEG)

La BEG fournit également une extension permettant aux utilisateurs d'inclure les propriétés des avions dans les expressions. L'ontologie spécifie que les propriétés des avions ont une portée et un type. Étant donné que seules trois portées possibles ont été identifiées pour les propriétés et que l'ajout d'une portée supplémentaire n'est pas prévu, les portées se traduisent par une règle terminale facultative dans la grammaire

5. http://users.monash.edu/~lloyd/tildeProgLang/Grammar/Arith-Exp/ [Dernière visite : Octobre 2020]

(*ac_scope*), qui permet de choisir entre le terminal *RAP* (pour *Recognized Air Picture*) (correspondant à la portée globale) ou un identifiant pointant vers un filtre existant (portée partielle). L'absence de portée implique l'utilisation de la portée par défaut, correspondant à la portée individuelle.

À l'inverse, chaque propriété possède un type fixe, et le choix du type n'est pas permis pour les utilisateurs. C'est pourquoi les types des propriétés ne sont pas inclus dans la grammaire et ne sont pas gérés à cette étape du développement du DSL. Ainsi, la question de la gestion des types intervient plus tard lors de l'analyse sémantique.

Ci-après, un exemple d'expression utilisant la BEG :
(**RAP.MEAN_ALTITUDE** – **ALTITUDE**) / 2

L'évaluation de cette expression correspond à la moitié de la différence entre l'altitude moyenne de la RAP et l'altitude de l'avion cible. En effet, la propriété **RAP.MEAN_ALTITUDE** correspond à l'altitude moyenne de tous les avions présents dans l'enregistrement, et la propriété **ALTITUDE** correspond à l'altitude de l'avion cible.

À noter que cette expression ne peut être évaluée que sur un "instantané" (on utilisera le terme anglais *snapshot*) de la RAP. Un *snapshot* correspond à l'état de la RAP à un moment donné. En effet, les propriétés dynamiques des avions évoluant au cours du temps, une expression n'a de sens que si elle est rattachée à un instant connue de l'enregistrement. Le choix de l'instant pour l'évaluation des expressions est abordé en section 6.4.1 p. 122, qui traite la question de la représentation des données du trafic aérien.

6.3.2/ La Grammaire des Évaluations de Propriétés

La Grammaire des Évaluations de Propriétés (ou PEG pour *Property Evaluation Grammar*) est une grammaire courante de comparaisons booléennes. Cette grammaire est un *piggyback* des comparaisons booléennes issues des langages de programmation à usage général classiques comme Java, Python ou C++. Pour les expressions arithmétiques, elle inclut intégralement la BEG présentée dans la section précédent.

Une expression booléenne oppose deux valeurs avec un opérateur de comparaison. Le résultat est *Vrai* si la comparaison est vérifiée, et *Faux* dans le cas inverse. La PEG spécialise légèrement les grammaires existantes en forçant la présence d'une propriété d'avion en tant qu'opérande gauche de la comparaison (appelée "comparaison de propriété"). Comme présentée plus tôt en section 6.2.9 p. 107, une comparaison de propriété compare la valeur d'une propriété (e.g., altitude) avec le résultat d'une expression. La PEG étend également les grammaires existantes afin de permettre des comparaisons basées sur des zones géographiques, en utilisant un minimum de trois

coordonnées associées avec une plage d'altitude, comme définie dans l'ontologie au travers de la classe *Zone*. La PEG obtenue est présentée dans la figure 6.2.

```
prop_eval   : :=  negation ( (and | or ) negation)*
negation    : :=  not ( relation )
               |  relation
relation    : :=  beg.ac_charac (=|! =|<|>|<=|>=) beg.expr
               |  in zone
               |  ( prop_eval )
zone        : :=  prism with_vertices vertices and
                   altitude from beg.expr to beg.expr
vertices    : :=  coords , coords ( , coords )+
coords      : :=  ( beg.expr , beg.expr )
```

FIGURE 6.2 – EBNF de la Property Evaluation Grammar (PEG)

Voici un exemple d'une conjonction de comparaisons booléennes impliquant des propriétés d'avion : **ALTITUDE** < 350 **and GROUNDSPEED** > 400. Cette expression évalue, pour un avion donné, s'il vole à une altitude inférieure à 350 pieds (**ALTITUDE** > 350) tout en ayant une vitesse supérieure à 400 nœuds, une situation étant considérée comme anormale.

Ci-dessous, un autre exemple utilisant les zones géographiques : **not inside prism with_vertices (43.02,51.09), (42.87,47.26), (40.66,53.11) and altitude from 5600 to 8100**. Cette expression évalue qu'un avion se trouve hors d'une zone composée de trois coordonnées : (43.02,51.09), (42.87,47.26) et (40.66,53.11), et d'une plage d'altitude compris entre 5600 et 8100 pieds. Cet exemple illustre l'intérêt des zones introduites en section 6.2.8 p. 106. Il apparaît en effet fastidieux d'exprimer la présence ou non d'un avion dans une zone en utilisant la notation classique des comparaisons de propriétés car cela impliquerait l'utilisation de relation arithmétique entre les propriétés de l'avion (latitude, longitude et altitude) l'avion et les coordonnées de chaque point de la zone, pour un résultat identique.

De manière similaire à la BEG, la comparaison de propriétés n'a de sens que lorsqu'elle est associée à un instant précis de l'enregistrement. Elle doit donc être effectuée sur un *snapshot* de la SAG (*Recognized Air Picture* ou RAP).

6.3.3/ LE LANGAGE DE FILTRE

Le langage de filtre, comme décrit dans l'ontologie (voir section 6.2.11 p. 109), utilise deux opérateurs temporels empruntés à la Logique Temporelle Linéaire (LTL) [12]. Le langage de filtre permet la sélection d'avions basée sur leurs propriétés au cours de l'enregistrement. L'utilisation des opérateurs LTL rend possible la comparaison de propriété sur une fenêtre de temps de l'enregistrement, plutôt que sur un *snapshot* de la RAP. La grammaire du DSL de filtre est visible en figure 6.3, dans laquelle les extensions (les opérateurs temporels) de la PEG apparaissent en rouge gras.

```
filter      : :=   eval prop_eval
prop_eval   : :=   temporal ((and|or) temporal)*
temporal    : :=   G ( relation )
                |  F ( relation )
                |  not ( relation )
                |  relation
relation    : :=   beg.ac_charac (=|! =|<|>|<=|>=) beg.expr
                |  in zone
                |  ( prop_eval )
zone        : :=   .....
```

FIGURE 6.3 – La grammaire PEG est étendue avec des opérateurs temporels.

Le DSL de filtre étend la grammaire PEG avec les opérateurs temporels ALWAYS et EVENTUALLY, correspondant respectivement aux terminaux **G** et **F** dans la grammaire. Pour faire cela, une règle de production nommée *filter* a été ajoutée. Elle appelle la règle de production *prop_eval* qui a été modifiée pour lui permettre d'appeler une nouvelle règle de production nommée *temporal* plutôt que *negation* (voir figure 6.2). La règle *temporal* contient les opérateurs temporels, ainsi que l'opérateur de négation afin de permettre aussi bien la négation des opérateurs temporels que celle des comparaisons de propriétés grâce à l'appel récursif à *prop_eval* dans la règle *relation*. À noter que les opérateurs temporels ne sont pas applicables aux expressions incluant des propriétés statiques, e.g., ICAO d'un avion. Les opérateurs temporels sont donc optionnels dans la règle *temporal*. Tout comme le contrôle du type des propriétés qui sort du cadre de la grammaire, il en va de même pour la restriction selon laquelle les opérateurs temporels englobent seulement des comparaisons de propriétés dynamiques. La grammaire ne restreignant pas cet usage, son interdiction est prise en charge lors de l'analyse sémantique.

Voici un exemple typique de filtre : **eval not** (**G**(**ALTITUDE** > 30000)) **and F**(**ALTITUDE** > 34000 **and LONGITUDE** < 2.30). Cette expression est composée de deux sous-expressions. La première, **not** (**G**(**ALTITUDE** > 30000)) est évaluée négativement pour les avions volant constamment à une altitude strictement supérieure à 30000 pieds. En d'autres mots, l'avion doit voler au moins une fois en-dessous ou exactement à 30000 pieds, ce qui correspond à la forme alternative suivante : **F**(**ALTITUDE** <= 30000). La deuxième sous-expression, **F**(**ALTITUDE** > 34000 **and LONGITUDE** < 2.30) est évaluée positivement si l'avion à reporté au même moment une altitude supérieure à 34000 pieds et une longitude inférieure à 2.3° (sud de Paris).

Ce dernier exemple illustre la façon d'exprimer le caractère simultané ou indépendant de deux expressions booléennes de même temporalité (**F** ou **G**). Dans l'exemple, la deuxième sous-expression contient elle-même deux sous-expressions qui se trouvent à l'intérieur du même opérateur temporel. Grammaticalement parlant, cela indique que ces deux expressions doivent être évaluées de manière simultanée, i.e. les expressions doivent être vraies au moins une fois *au même moment*. En langage naturel, cette

expression peut s'écrire de la manière suivante : *l'avion a volé au moins une fois au-dessus de 34000 pieds au sud de Paris*. Si chaque expression est englobée dans un opérateur $EVENTUALLY$ différent, tel que :
F(**ALTITUDE** > 34000) **and** **F**(**LONGITUDE** < 2.30) cela implique que les deux expressions sont évaluées de manière indépendante, i.e. chaque expression doit être vraie au moins une fois, mais pas nécessairement au même moment. En langage naturel, cela s'écrit de la manière suivante : *l'avion doit voler au moins une fois au sud de Paris, et doit voler au moins une fois au-dessus de 34000 pieds*.

Cet aspect augmente grandement le pouvoir d'expression des filtres. À noter qu'il n'y a que deux cas dans lesquels cette subtilité existe : la combinaison d'opérateurs $EVENTUALLY$ et de la conjonction logique, et la combinaison d'opérateurs ALWAYS et de la disjonction (inclusive) logique.

6.3.4/ LE LANGAGE DE DÉCLENCHEUR

L'objectif des déclencheurs est de définir *quand* l'altération doit être effectuée. Bien qu'il n'existe pas de grammaire pour faire du *piggybacking*, le langage de déclencheur a été inventé en copiant la grammaire du langage de filtre. En effet, les deux langages sont, par essence, assez similaires : ils proposent tous deux des opérateurs englobant des comparaisons de propriétés. La grammaire du langage de déclencheur ainsi obtenue est présentée dans la figure 6.4.

```
trigger     : :=  eval trigg_eval
trigg_eval  : :=  time_window ((and|or) time_window)*
time_window : :=  asap ( prop_eval )
              |   when ( prop_eval )
              |   until ( prop_eval )
              |   ( trigg_eval )
prop_eval   : :=  negation ( (and | or ) negation)*
negation    : :=  ...
```

FIGURE 6.4 – La grammaire PEG étendue pour les déclencheurs

Il existe toutefois une différence clé entre les opérateurs des deux langages, la négation ne peut pas être utilisée sur les opérateurs du langage de déclencheur. Une stratégie différente doit alors être employée pour étendre la grammaire PEG. Tandis que pour le langage de filtre, les extensions consistent à modifier des règles de production existantes et à ajouter une règle intermédiaire, de nouvelles règles de production sont inventées pour le langage de déclencheur. Ces deux nouvelles règles, *trigg_eval* et *time_window*, spécifient la syntaxe des expressions de déclenchement. Elles sont référencées directement dans la règle de production initiale, *trigger*. La règle initiale de la grammaire PEG, *prop_eval*, est ainsi englobée dans les opérateurs de déclenchement.

Plutôt que de retourner un booléen comme le langage de filtre, l'interprétation d'une

condition de déclenchement sur un avion retourne une chronologie indiquant les moments où l'expression booléenne est évaluée positivement et ceux où elle est évaluée négativement. Un exemple de chronologie pour quatre avions est donnée en figure 6.5. Les barres vertes représentent les intervalles de temps durant lesquels la condition de déclenchement est vraie tandis que les barres rouges représentent les intervalles de temps durant lesquels la condition de déclenchement est fausse. Enfin les barres grises indiquent les moments où aucune information de surveillance n'est émise par l'avion.

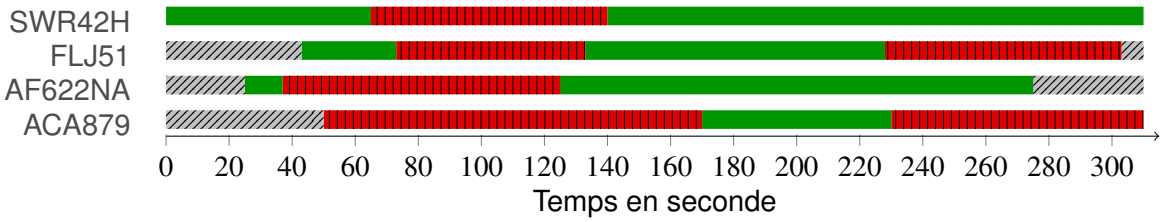


FIGURE 6.5 – Résultat de l'évaluation d'un déclencheur sur quatre avions.

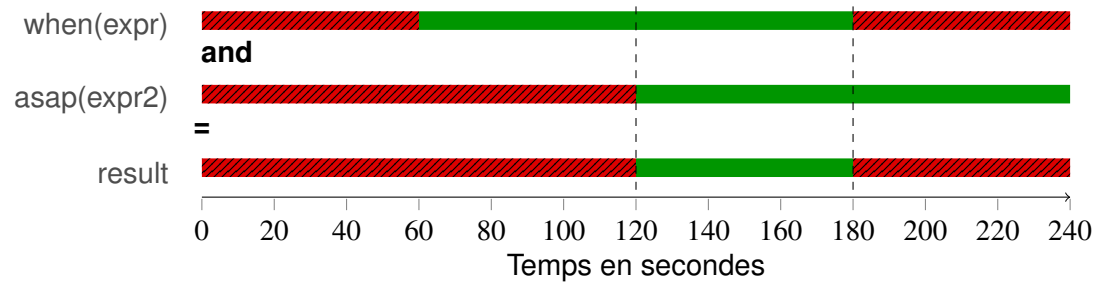
Voici, un exemple simple de déclencheur : **eval when**(**LONGITUDE** > 5.25). Cette expression signifie que l'altération est appliquée à chaque avion cible quand celui-ci se situe au nord de la longitude 5.25°.

Voici un autre exemple utilisant la portée globale : **eval as_soon_as**(**RAP.ALTITUDE** <= 35000). Cette expression signifie que l'altération est appliquée à chaque avion cible à partir du moment où l'entièreté des avions de la RAP volent à une altitude inférieure ou égale à 35000 pieds.

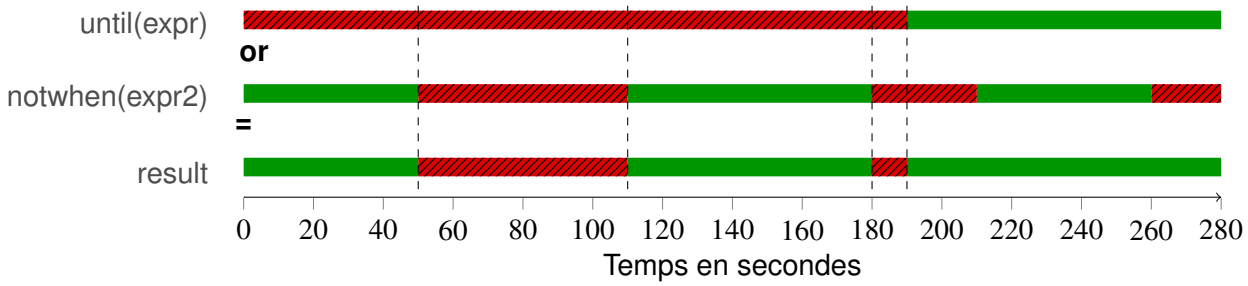
Comme mentionné précédemment, il est possible de combiner des conditions de déclenchement en utilisant des conjonctions et disjonctions inclusives logiques. Par exemple : **eval when**(**AIRCRAFT.LONGITUDE** > 5.25) **and as_soon_as**(**RAP.ALTITUDE** <= 35000). Cette expression signifie que l'altération est appliquée à chaque avion cible quand celui-ci se situe au nord de la longitude 5.25° mais pas avant que tous les avions de la RAP ne volent tous à une altitude inférieure ou égale à 35000 pieds. La manière dont les disjonctions et les conjonctions sont évaluées est présentée dans la figure 6.6 dans laquelle les sections vertes représentent les intervalles positifs, les sections rouges représentent les intervalles négatifs. Pour qu'une conjonction soit vraie au temps t, les deux expressions qui la composent doivent être évaluées positivement à ce moment là, tandis que, pour une disjonction, l'évaluation positive d'une seule expression est requise.

6.3.5/ LE LANGAGE DE SCHÉMA D'ALTÉRATION

Les deux langages de filtre et de déclencheur présentés précédemment sont des extensions ou des spécialisations de langages existants, des notations qui sont déjà bien



(a) Une conjonction des opérateurs *when* et *as soon as*



(b) Une disjonction des opérateurs *not when* et *until*

FIGURE 6.6 – Conjonction et disjonction de déclencheurs.

établies. À l'inverse, le langage de schéma d'altération est nouveau et consiste en une transposition presque directe de l'ontologie. En effet, un sous-ensemble des classes est converti en règles de production, tandis que le reste (généralement les classes abstraites) est converti en règle alternatives. Les propriétés contenant des objets correspondent généralement à des symboles non-terminaux, alors que les propriétés contenant des types de données primitifs sont converties en symboles terminaux. Selon la cardinalité des propriétés, les symboles correspondants peuvent être associés à des opérateurs.

Enfin, le langage de schéma d'altération intègre entièrement la grammaire BEG, au contraire des langages de filtres et de déclencheurs qui n'y sont pas intégrés pour les raisons évoquées en section 6.3 p. 112. L'utilisateur peut néanmoins référencer les filtres et les déclencheurs via leur identifiant. La grammaire du langage de schéma d'altération est visible dans la figure 6.7.

Dans cette sous-section, le langage est illustré par deux exemples. À noter qu'il sera également illustré plus loin dans la sous-section 6.5 p. 129 où les capacités du langage à couvrir la taxonomie des attaques sont présentées.

Tout d'abord, une altération très basique est définie ci-après : **alter all_planes at 126 seconds with_values GROUNDSPEED** = 229.6 Le schéma applique l'altération à tous les avions de l'enregistrement (pas de filtre) 126 secondes après le premier message de l'enregistrement (pas de déclencheur). L'altération consiste à changer la vitesse de chaque avion cible à une valeur constante de 229.6 nœuds.

```
scenario  : :=  decl* schema+
decl      : :=  let var = (list | range)
schema    : :=  ( hide | saturate ) target filter
                   t_scope trigger
              |  create t_scope trigger params
                      waypoints ?
              |  alter target filter t_scope trigger
                      (params|waypoints)
target    : :=  any_plane | all_planes
filter    : :=  satisfying <string>
trigger   : :=  within <string>
t_scope   : :=  at time
              |  at time for time
              |  from time until time
time      : :=  number seconds
              |  <float> * duration
time      : :=  seconds
params    : :=  with_values param (and param)*
param     : :=  param_type = value
waypoint  : :=  coord with_altitude number at
coords    : :=  ( value , value )
list      : :=  ( value (, value)* )
range     : :=  [ <integer> , <integer> ]
value     : :=  <string> | number | var | offset
offset    : :=  (<< | >>) number
number    : :=  <integer> | <float>
```

FIGURE 6.7 – Grammaire du langage de schéma d'altération

Voici un autre exemple d'une altération plus complexe :
create plane from 12 seconds until 251 seconds with_values ICAO = "*39AC47*" **and with_waypoints [** (24.85,53.23) **with_altitude** 4000 **at 12 seconds** , (24.62,53.94) **with_altitude** 4250 **at 251 seconds]**

Ce schéma a pour effet de créer un avion "fantôme" entre la seconde 12 et la seconde 251 de l'enregistrement. L'avion a la valeur *39AC47* pour ICAO et une trajectoire définie par deux points de passage.

Afin de générer des cas de test multiples (i.e. plusieurs variantes d'un enregistrement altéré) à partir d'un seul schéma, il est possible de définir des variables contenant des listes ou des plages de valeurs. Par exemple : **let *$varName*** = **[1,5]** est une définition de variable standard, utilisant le mot-clé **let** comme de nombreux langages de programmation. Cette instruction définit une variable nommée *varName* qui représente une plage de valeur allant de 1 à 5 (inclus). Pour les listes, on utilise la déclaration suivante : **let *$varName*** = **{1,2,3,4,5}** où la variable *varName* représente une liste de cinq entiers allant de 1 à 5. En plus des valeurs numériques, une liste peut également contenir des chaînes de caractères : une liste d'ICAO ou de callsign. Le mécanisme de génération de cas de test multiples à partir de variables est expliqué en section 6.4.4 p. 127.

6.4/ IMPLÉMENTATION DU DSL

Un prototype supportant l'ensemble des DSL présentés précédemment a été développé en Java et est disponible sur Github [6]. Tous les langages ont été développés avec Xtext [7], un framework proposé par Eclipse pour le développement de DSL. Un éditeur graphique a également été développé avec JavaFX [8] pour chaque langage.

Le langage de filtre et celui de déclencheur sont interprétés. En opposition aux langages compilés, cela signifie que le résultat de l'exécution d'un programme n'est pas un langage de plus bas niveau, mais un ensemble de structures de données. (détaillées en section 6.3.3 p. 116 pour le langage de filtre, et en section 6.3.4 p. 118 pour le langage de déclencheur). Le langage de schéma est considéré comme un hybride entre un langage compilé et un langage interprété car son exécution produit un fichier XML qui peut être assimilé à un langage de plus bas niveau. Toutefois, l'approche est similaire à celle utilisée pour l'interprétation des deux autres langages avec le suivi d'un cycle *fetch-decode-execute* [75] (pouvant être traduit par chargement-décodage-exécution).

Avant d'être interprété, chaque langage est analysé par un analyseur sémantique afin de déterminer sa correction sémantique. Cette analyse vérifie la validité des types et des valeurs associées aux propriétés de l'avion (par exemple, une altitude ne peut pas être négative, ni associée à une chaîne), la validité des valeurs spécifiées par rapport à l'enregistrement associé (par exemple, une fenêtre temporelle ne peut pas dépasser la durée d'un enregistrement), ou encore la validité de la portée des propriétés en fonction du contexte (par exemple, une propriété globale, e.g., l'altitude moyenne, ne peut pas se voir attribuer de nouvelle valeur).

Premièrement, cette section détaille la manière dont les données de surveillance sont représentées pour permettre l'interprétation des DSL. Ensuite, deux sous-sections sont dédiées respectivement à la mise en œuvre du langage de filtre et du langage de déclencheur en présentant leurs algorithmes d'interprétation. Pour finir, une dernière sous-section détaille le processus permettant de passer d'un scénario d'altération à des ensembles de directives à appliquer à l'enregistrement.

6.4.1/ LA REPRÉSENTATION DES DONNÉES DE SURVEILLANCE

Le protocole ADS-B est conçu pour transmettre une moyenne maximale de cinq messages par seconde, (deux messages de positions, deux messages de vélocité et un message de statut). Néanmoins, le protocole ne garantit pas que le flux de messages soit ininterrompu. Des pertes de messages peuvent survenir pour de nombreuses raisons,

6. https://github.com/aymeric-cr/dsl-scenario
7. https://www.eclipse.org/Xtext/
8. https://openjfx.io/

comme, par exemple, le relief du terrain ou des perturbations électromagnétiques. Ces pertes peuvent représenter une absence de données de plusieurs dizaines de secondes entre deux messages valides et ainsi compromettre la qualité des résultats de l'interprétation des filtres et des déclencheurs.

En effet, à un instant t, la valeur connue d'une propriété est la dernière valeur transmise par l'avion, cela peut être une limitation pour les filtres et les déclencheurs en cas d'absence de données trop longue (plusieurs dizaines de secondes voir plusieurs minutes). Afin de permettre des systèmes de filtrage et de déclenchement précis (et correct), il est nécessaire d'obtenir, ou du moins d'estimer, les valeurs des propriétés à tout moment de l'enregistrement.

Pour permettre cela, il a été choisi de considérer un enregistrement comme un ensemble d'avions plutôt qu'un ensemble de messages. Cette nuance demande aux experts du domaine aérien de concevoir des scénarios en ciblant des avions avec certaines propriétés, plutôt que de cibler des messages contenant certaines valeurs. Si cette considération rend la conception de scénario plus intuitive et a simplifié le développement des langages, la discontinuité des données est problématique pour la sélection des avions via des filtres et le calcul des intervalles de temps via les déclencheurs.

Les propriétés dynamiques, qui sont un ensemble de valeurs discrètes, doivent alors être formalisées en fonctions continues afin de se rapprocher le plus possible du comportement physique d'un avion. L'interpolation est un bon candidat pour répondre à ce problème. En effet, en opposition à la régression qui calcule une relation entre les valeurs d'un ensemble, l'interpolation permet de "combler les vides" entre chaque paire de valeurs consécutives tout en préservant les données. Une interpolation étant une forme d'approximation, il existe une part d'incertitude, appelée "divergence", dans les valeurs calculées (interpolées entre deux messages). Si il est évident que ce niveau d'incertitude n'est pas acceptable dans des applications critiques comme la prédiction de trajectoire fournissant les traces des avions aux contrôleurs, on fait ici l'hypothèse qu'il l'est dans un contexte orienté recherche. C'est pourquoi la méthode d'interpolation Akima, présentée en section 5.2.3.1 p. 79 est utilisée dans le cadre des filtres et des déclencheurs pour combler les potentielles absences de données.

Plusieurs aspects des contributions de cette thèse s'appuient ainsi sur des fonctions d'interpolation. Dans le cadre du DSL, ces fonctions sont utilisées dans l'interprétation des filtres ainsi que celle des déclencheurs. Elle sont par ailleurs utilisées dans la génération de données comme cela a été vu en section 5.2.3 p. 79 avec la modification de trajectoire.

6.4.2/ LE FILTRAGE DES AVIONS

L'algorithme d'interprétation, visible en figure 7, a été créé pour évaluer les expressions de filtrage des avions. L'algorithme tire parti de la méthode d'interpolation d'Akima présentée dans la sous-section précédente pour déterminer les avions ciblés au sein de l'enregistrement. De complexité $O(n)$, il se comporte comme la plupart des algorithmes d'interprétation en utilisant le patron de conception *visiteur* [43] : il parcourt l'arbre de syntaxe abstraite (aussi appelé AST pour *Abstract Syntax Tree*) généré par l'analyseur syntaxique (*parser*) suite à l'analyse d'un programme correspondant à un filtre. Différentes opérations sont alors effectuées quand les nœuds de l'arbre de syntaxe abstraite sont visités.

Algorithme 7 : Algorithme d'évaluation des filtres

Input : a_id : Identifiant de l'avion
op : Opérateur booléen
prop : Propriété de l'avion (altitude, callsign, etc.)
c_val : Valeur comparée
temp : Opérateur temporel (**G** ou **F**)
step : Pas de temps
t_start : Date du premier message émis par l'avion
t_end : Date du dernier message émis par l'avion

Result : true/false

```
1  t_count ← t_start
2  while t_count ≤ t_end do
3      prop_val ← interpolateProp(a_id, prop, t_count)
4      result ← evaluateExpression(prop_val, op, c_val)
5      if (temp = F) ∧ result then
6          return true
7      else if (temp = G) ∧ ¬result then
8          return false
9      else
10         t_count ← t_count + step
11         if t_count > t_end ∧ t_count < (t_end + step) then
12             t_count ← t_end
13         end
14     end
15 end
16 return temp = G
```

Quand une feuille de l'arbre de syntaxe abstraite (i.e. un opérateur, une propriété d'avion ou une valeur) est atteinte, le contenu du nœud est soumis à une méthode d'évaluation, comme présentée dans l'algorithme 7. Cette méthode prend également en paramètre l'identifiant de l'avion en cours d'évaluation, un intervalle de temps, l'opérateur temporel utilisé, ainsi que les dates des premier et dernier messages de l'enregistrement émis par l'avion courant. L'objectif est d'utiliser la fonction d'interpolation de l'avion pour obtenir les

valeurs de propriétés au cours du temps, étant donné un certain intervalle de temps. En partant de la première date d'émission de l'avion, la première étape consiste à obtenir la valeur interpolée (ligne 3), ensuite l'expression est évaluée (ligne 4). Il existe alors deux cas où la méthode d'évaluation peut retourner un résultat immédiatement : soit l'opérateur temporel est **F**, auquel cas l'expression nécessite seulement d'être évaluée une fois positivement pour retourner *true* (lignes 5-6), soit l'opérateur est **G**, auquel cas il suffit d'une évaluation négative de l'expression pour retourner *false* (lignes 7-8). À un instant t, si aucune des conditions n'est remplie, alors le compteur de temps t_count est incrémenté de la valeur de $step$ (ligne 10) et l'expression est de nouveau évaluée. Si après cet incrément, le compteur pointe une date supérieure à t_end il prend alors la valeur de t_end (lignes 11-13) afin d'éviter une requête en dehors de la fonction d'interpolation. Enfin, si l'algorithme itère jusqu'à ce que t_count atteigne la valeur de t_end, il retourne alors *false* si l'opérateur temporel est **F**(l'expression n'a jamais été évaluée positivement) ou *true* si l'opérateur temporel est **G**(l'expression a toujours été évaluée positivement).

Une légère modification doit être apportée à l'algorithme pour qu'il évalue correctement les conjonctions englobées dans un opérateur **F** ou les disjonctions englobées dans un opérateur **G**. En effet, plutôt que de retourner un booléen, l'algorithme conserve les dates où l'expression a été évaluée positivement. La liste de dates ainsi obtenue est comparée à la liste de dates obtenue de l'autre expression issue de la conjonction/disjonction. Pour une conjonction dans l'opérateur **F**, les expressions doivent retourner une liste de dates identiques afin de garantir que les deux expressions sont vraies *au même moment*. Pour une disjonction dans l'opérateur **G**, l'union des deux listes doit couvrir l'entièreté du temps compris entre t_start et t_end étant donné le pas de temps $step$.

Le choix de la bonne valeur pour $step$ dépend des opérations effectuées à chaque pas, du degré de précision requis ainsi que de temps de calcul disponible. Plus l'intervalle sera grand, plus la précision sera faible, mais plus le calcul sera rapide. Dans l'implémentation de l'interpréteur au sein de la chaîne outillée (présentée en chapitre 7) le pas de temps est fixé à une seconde. Dans le cas de l'interprétation, seules des opérations arithmétiques ou logiques simples sont effectuées, et même en grand nombre le temps d'exécution reste court. En revanche, le parcours des trajectoires via cette méthode se fait avec des pas de temps plus élevés lorsqu'il s'agit de faire de l'affichage, dans le cas de notre chaîne outillée, celui-ci est fixé à cinq secondes.

6.4.3/ LE DÉCLENCHEMENT DES ALTÉRATIONS

Comme pour le langage de filtre, un algorithme d'interprétation, visible en figure 8, est utilisé pour évaluer les conditions de déclenchement des altérations. De complexité $O(n^2)$ l'algorithme d'interprétation des déclencheurs fonctionne d'une manière similaire

à celui des filtres, notamment avec l'utilisation du patron de conception *visiteur* : les valeurs des feuilles sont remontées aux nœuds parents qui sont soumis à différentes méthodes d'évaluation. Le but de l'interprétation des conditions de déclenchement est de déterminer si un ensemble d'avions donnés satisfait une même condition à un instant t. Si cet ensemble ne contient qu'un seul avion, on parle alors de portée individuelle, à l'inverse, si l'ensemble contient plusieurs avions on parle alors de portée partielle ou globale (voir sous-section 6.2.6 p. 104). Le résultat de l'interprétation est un ensemble d'intervalles de temps indiquant les moments de l'enregistrement où l'altération est appliquée. Tandis que pour les portées individuelles chaque avion possède des intervalles de temps qui lui sont propres, pour les portées partielles et globales les intervalles de temps seront communs à tous les avions de l'ensemble ciblé. L'algorithme 8 détaille l'interprétation dans le cadre des portées partielles et globales (donc avec des intervalles de temps communs à tous les avions cibles) :

Algorithme 8 : Algorithme d'évaluation des déclencheurs dans une portée multiple (partielle ou globale)

Input : op : Opérateur booléen
prop : Propriété de l'avion
c_val : Valeur comparé
targets : Liste des avions cibles
step : Pas de temps
t_start : Date de début de l'enregistrement
t_end : Date de fin de l'enregistrement
Result : true/false

```
1  timeline ← list()
2  for t_count = t_start to t_end by step do
3      holds ← true
4      a_count ← 0
5      while (holds ∧ a_count ≤ size(targets)) do
6          ac ← targets(a_count)
7          if exists(ac, t_count) then
8              prop_val ← interpolateProp(a_id, prop, t_count)
9              holds ← evaluateExpression(prop_val, op, c_val)
10         end
11         a_count + +
12     end
13     addToList(timeline, holds)
14     if t_count > t_end ∧ t_count < (t_end + step) then
15         t_count ← t_end
16     end
17 end
18 return timeline
```

Il est considéré ici que les avions dans la portée ont préalablement été identifiés et regroupés dans la liste $targets$. L'objectif de l'algorithme est ainsi de calculer les intervalles

de temps durant lesquels tous les avions de *targets* satisfont une expression booléenne donnée, i.e. comparaison entre la valeur de *prop* et *c_val* selon l'opérateur *op*. Comme pour l'algorithme 7, l'itération sur l'intervalle de temps commençant à *t_start* et finissant à *t_end* se fait par incrémentation d'un compteur *t_count* de la valeur de *step*. Cette dernière valeur déterminant le rapport entre la précision et la vitesse du calcul. À chaque pas de temps, l'expression booléenne est évaluée sur chaque avion de *targets* tant que le résultat de l'évaluation est positif, si ce dernier est négatif, alors on passe au pas de temps suivant (lignes 5 et 9). Pour chaque avion de *targets*, la première étape est de vérifier que l'avion est présent dans l'enregistrement au pas de temps courant (ligne 7), auquel cas la valeur de la propriété est calculée par interpolation (ligne 8) et l'expression est évaluée (ligne 9). Si *targets* a été complètement parcourue la valeur de *holds* est *true* cela signifie que tous les avions satisfont la conditions de déclenchement au pas de temps courant, à l'inverse si un avions n'a pas satisfait la condition pour ce pas de temps, l'itération de *targets* est stoppée et la valeur *holds* est *false*. La variable *holds* est ensuite ajoutée à la liste de booléens *timeline* (ligne 13), cette dernière constituant le résultat de l'algorithme. Pour retrouver les temps de déclenchement réels, il faut multiplier l'indice de chaque booléen de *timeline* par le pas de temps *step* et corréler la liste résultante avec *timeline*.

6.4.4/ GÉNÉRATION DE DIRECTIVES D'ALTÉRATIONS À PARTIR DES SCHÉMAS

Cette sous-section détaille le processus de génération de directive d'altération et introduit la notion de concrétisation d'un schéma d'altération abstrait.

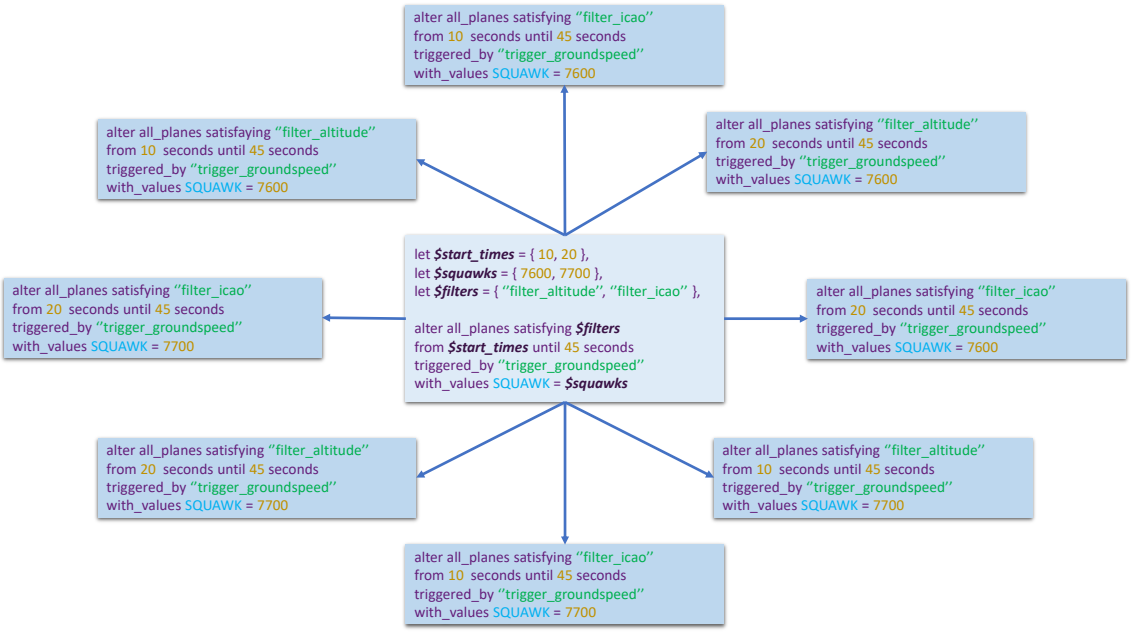


FIGURE 6.8 – Du schéma abstrait aux schémas concrets

Le calcul des directives d'altération peut être divisée en trois activités :

— **La concrétisation du schéma d'altération**. Comme montré dans la section 6.3.5 p. 119, il est possible de déclarer des listes ou des plages de valeurs en tant que variable dans un schéma d'altération. La motivation de cette approche est de permettre la massification d'un schéma d'altération abstrait. Un schéma d'altération est considéré comme abstrait lorsqu'il contient des références vers des variables. Une étape de concrétisation est ensuite nécessaire afin de pouvoir exécuter un tel schéma. Cette étape consiste à créer des instances concrètes du schéma abstrait. Dans un schéma concret, chaque référence à une plage, ou une liste de valeurs, est remplacée par une valeur de cette plage, ou liste. S'il existe plusieurs variables au sein d'un même schéma, cela implique la création d'une instance concrète pour chaque combinaison unique de valeurs. Pour trois listes de taille 2, 4 et 5, référencées dans un schéma abstrait, alors $2 * 4 * 5 = 40$ instances de schéma concret sont créées. Un exemple de combinaison de variables est montré dans la figure 6.8. L'activité de concrétisation permet ainsi la génération d'un large ensemble de schémas d'altération, engendrant un large ensemble de cas de test à partir d'un unique schéma abstrait multivalué.

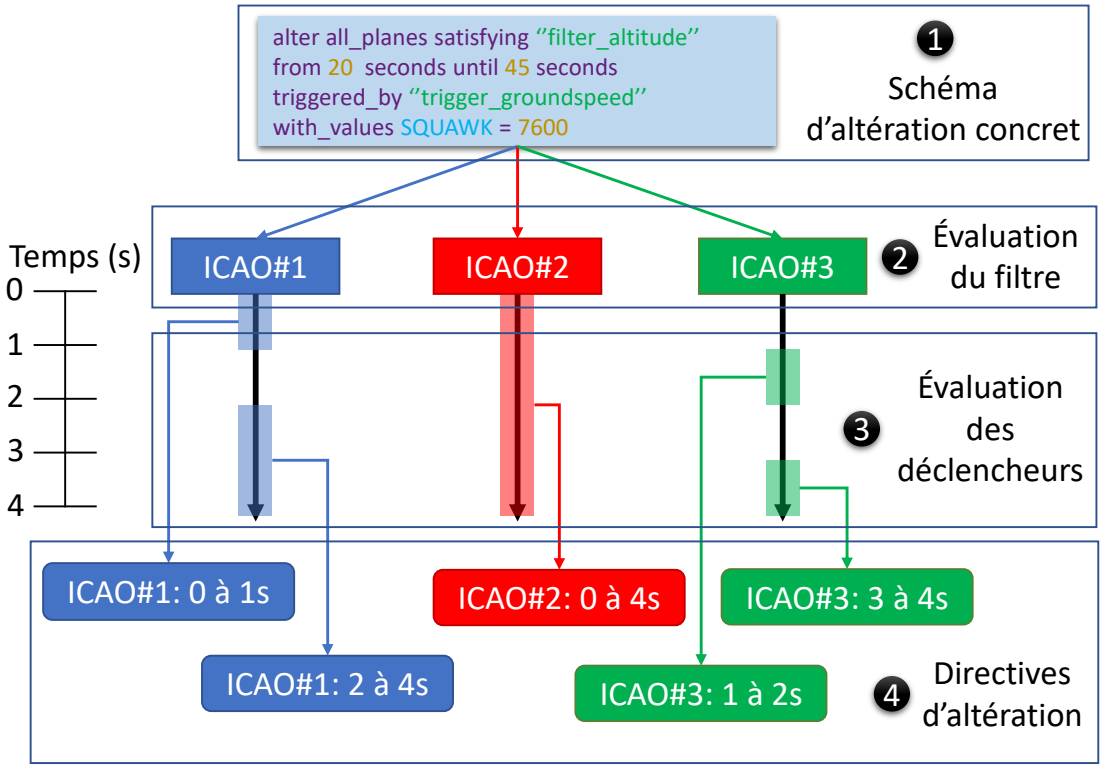


FIGURE 6.9 – Obtentions des directives d'altération à partir d'un schéma concret

— **La sélection des cibles**. Comme représenté en figure 6.9, la sélection des cibles

(i.e. les avions ciblés par le schéma) est effectuée pour chaque schéma concret ①. Une liste d'identifiants ② est obtenue suite à l'interprétation des filtres présentée en sous-section 6.4.2 p. 124. Cette liste est ensuite utilisée pour le calcul des intervalles de temps de l'altération.

— **Le calcul de intervalles de temps**. Comme vu dans la présentation de l'ontologie (voir sous-section 6.2.12 p. 110), en plus d'une liste de cibles, une directive d'altération requiert une fenêtre de temps durant laquelle l'altération doit être appliquée. Cette fenêtre de temps est calculée durant l'interprétation des déclencheurs, décrit dans la sous-section 6.4.3 p. 125. Le résultat de cette interprétation est, pour chaque avion cible, une liste de fenêtres de temps ③. L'association d'une fenêtre de temps et d'un avion cible (et les paramètres liés à l'altération) constitue une directive d'altération ④.

Pour résumer, les variables et les déclencheurs permettent de facilement créer des schémas complexes d'altérations. D'un côté, les variables factorisent la définition d'altérations, tandis que les déclencheurs calculent automatiquement un grand nombre de fenêtres de temps pour les altérations basées sur des conditions spécifiques. Ces deux briques essentielles dans la conception de scénarios permettent de convertir un schéma abstrait vers une liste de directives d'altération. Ces directives d'altération constituent le paramètre d'entrée du moteur d'altération présenté dans le chapitre 5.

6.5/ VALIDATION DES LANGAGES DÉDIÉS

Il a été choisi pour valider les DSL de filtre, de déclencheur et de schéma d'altération, de comparer leur pouvoir d'expression avec la taxonomie des attaques identifiée en section 3.1.3 p. 43, et de mesurer le gain de productivité qu'ils apportent comparé à l'utilisation directe du module de génération (chapitre 5).

Cette section s'organise en deux sous-sections. La première est centrée sur le pouvoir d'expression offert par l'utilisation du langage de schéma couplée à celle des langages de filtre et déclencheur, tandis que la deuxième se focalise sur le gain de productivité constaté.

6.5.1/ EVALUATION DU POUVOIR D'EXPRESSION

Le pouvoir d'expression du DSL de schéma est évalué en le confrontant à la taxonomie des attaques issues de la littérature. Elles sont rappelées ci-dessous :

— *Ghost Aircraft Injection*

— *Ghost Aircraft Flooding*

— *Virtual Trajectory Modification*

— *False Alarm Attack*

— *Aircraft Disappearance*

— *Aircraft Spoofing*

6.5.1.1/ L'ATTAQUE *Aircraft Flooding*

La figure 6.10 représente un exemple de *Aircraft Flooding Attack*. L'instruction **saturate** (ligne 1) constitue la partie *altération* qui, dans cet exemple, est une création multiple de messages.

```
1 saturate all_planes satisfying "filter"
2 triggered_by "trigger"
3 from 20 seconds until 25 seconds
4 with_values random ICAO and AIRCRAFT_NUMBER = 20
```

FIGURE 6.10 – Schéma d'altération de *Aircraft Flooding*

Une saturation est définie par deux paramètres :

— **AIRCRAFT_NUMBER** : Ce paramètre spécifie le nombre d'avions fantômes qui seront générés pour chaque avion cible.

— **ICAO** : Ce paramètre spécifie les ICAO des avions générés. Il peut contenir une valeur simple, une liste de valeurs, ou indiquer que les ICAO doivent être aléatoirement générés (comme c'est le cas dans l'exemple de la figure 6.10).

Le résultat d'une saturation est un certain nombre d'avions fantômes générés autour d'un avion cible réel. Les propriétés des avions générés sont les mêmes que celles de l'avion réel, excepté la latitude et la longitude qui sont calculées de manière à ce que les avions se répandent progressivement autour de l'avion cible.

Les parties *sélection* (ligne 1) et *déclenchement* (ligne 2) sont respectivement des références à un filtre nommé "*filter*" et à un événement nommé "*trigger*"La saturation a lieu à partir de 20 **seconds** jusqu'à 25 **seconds** (ligne 3) à partir du début de l'enregistrement nominal. Finalement, la partie *paramètre* (ligne 4) spécifie que les ICAO des vingt avions fantômes doivent être générés de façon aléatoire.

6.5.1.2/ L'ATTAQUE *False Alarm*

La figure 6.11 montre comment créer un scénario de fausse alerte (*False Alarm Attack*) en utilisant le DSL de schéma.

```
1 alter all_planes satisfying "filter"
2 from 120 seconds until 140 seconds
3 with_values SQUAWK = 7700
```

FIGURE 6.11 – Schéma d'altération de *False Alarm*

Ce schéma s'appuie sur l'utilisation d'une altération simple (représentée par le mot-clé **alter** dans le langage) visant les avions sélectionnés par le filtre "*filter*" (ligne 1), et se produisant entre 120 et 140 secondes après le début de l'enregistrement (ligne 2). Durant cette intervalle de temps, le **SQUAWK** des appareils cibles est modifié à la valeur 7700 (ligne 3), ce qui représente un cas de détresse.

À noter que dans le cas d'un changement de squawk à une valeur se rapportant à une urgence, un moteur d'altération réaliste voudrait que la propriété *alerte* (voir section 4.1.4 p. 63) soit modifiée automatiquement à 1. Ce mécanisme assurant un meilleur réalisme est abordé dans le chapitre suivant.

6.5.1.3/ L'ATTAQUE *Aircraft Disappearance*

Une disparition d'avion (*Aircraft Disappearance*) peut être réalisée avec le schéma d'altération présenté en figure 6.12.

```
1 hide all_planes triggered_by "trigger"
2 from 45 seconds until 60 seconds
```

FIGURE 6.12 – Schéma d'altération de *Aircraft Disappearance*

Dans cet exemple, tous les avions validant les conditions définies dans "*trigger*" sont supprimés de l'enregistrement, via le mot-clé **hide** (ligne 1) pendant une fenêtre de temps de quinze secondes, de 45 à 60 secondes (ligne 2), suivant la condition de déclenchement .

6.5.1.4/ L'ATTAQUE *Ghost Aircraft Injection*

La figure 6.13 montre un exemple de création d'un avion fantôme (*Ghost Aircraft Injection*), permettant de générer un avion qui ne soit pas le clone d'un appareil existant.

L'instruction **create** (ligne 1) constitue la partie *altération* qui est, dans cet exemple, une création de messages. La fenêtre de temps est ici fixée arbitrairement (ligne 2), de 20 à 320 secondes, et ne dépend d'aucune déclencheur. Pour une création d'avion, la partie paramètre est assez conséquente. Dans l'exemple, elle contient six paramètres (lignes 2-7) : l'ICAO, la vitesse au sol, l'indicatif d'appel, la latitude, la longitude et l'altitude. La partie suivante (lignes 8-11) concerne la trajectoire que l'avion fantôme doit adopter. Elle

```
1  create plane from 20 seconds until 320 seconds
2  with_values ICAO = "39AC47" and
3  CALLSIGN = "AFR3984" and
4  GROUNDSPEED = 102.2 and
5  LATITUDE = 2.57684 and
6  LONGITUDE = 47.49875 and
7  ALTITUDE = 26598
8  with_waypoints [
9  (2.476,47.69) with_altitude 29843 at 72 seconds ,
10 (2.39,47.93) with_altitude 33879 at 215 seconds ,
11 (2.319,48.01) with_altitude 38743 at 320 seconds ]
```

FIGURE 6.13 – Schéma d'altération de *Ghost Aircraft Injection*

est spécifiée par trois points de passage donnant les altitudes, latitudes et longitudes à atteindre aux temps 72, 215 et 300 secondes après le début de l'altération.

6.5.1.5/ L'ATTAQUE *Trajectory Modification*

Un exemple de modification de trajectoire (*Trajectory Modification*) est illustré en figure 6.14.

```
1 alter plane satisfying "filter"
2 from 100 seconds until 200 seconds
3 with_waypoints [
4 (29.11,14.21) with_altitude 33902 at 10 seconds ,
5 (29.235,14.3) with_altitude 33979 at 30 seconds ,
6 (29.29,14.35) with_altitude 33954 at 50 seconds,
7 (29.29,14.35) with_altitude 33954 at 70 seconds ]
```

FIGURE 6.14 – Schéma d'altération de *Trajectory Modification*

Comme précédemment, c'est le mot-clé **alter** qui est utilisé pour créer ce schéma d'altération. Le filtre "*filter*" cible un unique avion (ligne 1), entre 100 et 200 secondes (ligne 2). La trajectoire modifiée, i.e. qui vient en remplacement, se compose de quatre points de passage (lignes 4-7) fournissant des coordonnées et des altitudes à respecter aux temps 10, 30, 50 et 70 secondes après le début de l'altération.

6.5.2/ EVALUATION DU GAIN DE PRODUCTIVITÉ

Dans le but de mettre en avant le gain de productivité offert par les DSL, cette section montre combien il peut s'avérer fastidieux de produire directement un fichier XML contenant les directives d'altération prises en entrée par le moteur d'altération.

En d'autres termes, l'objectif de cette expérimentation est d'observer comment l'utilisation des filtres, des événements et des capacités combinatoires du DSL de schéma se traduit

en matières de directives d'altérations. L'objectif revient finalement à évaluer les capacités des DSL :

— À générer des directives dont les fenêtres de temps sont basées sur les propriétés des avions.

— À générer des directives ciblant un nombre restreint d'avions en fonction de leurs caractéristiques.

— À générer une suite de test en une seule exécution en tirant parti des listes et des plages de valeurs dans les schémas d'altérations.

Ci-dessous sont détaillés les composants d'un scénario de FDIA illustrant le gain de productivité apporté par les DSL.

L'enregistrement nominal : L'évaluation repose sur un enregistrement de 17 minutes et 45 secondes daté du 19 avril 2019, entre 15h30 et 15h47. Les messages ont été reçus par une antenne ADS-B située à Luxembourg et l'enregistrement a été obtenu via le réseau OpenSky. Il contient 76353 messages émis par 218 avions différentes. Il est disponible sur GitHub [9].

Les filtres : Trois filtres sont utilisés dans le scénario :

— filt_lowalt : **eval F**(**GROUNDSPEED** > 350) **and G**(**ALTITUDE** < 34000). Ce filtre sélectionne les avions dépassant au moins une fois la vitesse de 350 nœuds tout en volant tout le temps strictement en-dessous de 34000 pieds. Dans l'enregistrement nominal, 64 avions satisfont ce critère.

— filt_lat : **eval G**(**LATITUDE** < 49.5). Ce filtre permet d'identifier les avions qui volent toujours strictement en-dessous de la latitude 49.5. Dans l'enregistrement nominal, 85 avions satisfont ce filtre.

— filt_prism : **eval F**(**inside prism with_vertices (48, 7.7), (50.31,6.35), (49.18,4.22) and altitude from 1000 to 35000**). Ce dernier filtre spécifie que les avions doivent voler au moins une fois à l'intérieur d'un prisme triangulaire situé au-dessus du Luxembourg à une altitude allant de 1000 à 35000 pieds. Dans l'enregistrement nominal, 30 avions satisfont ce filtre.

Ces trois filtres sont utilisés indépendamment ou en conjonction, ce qui diminue alors le nombre d'avions sélectionnés.

Les déclencheurs d'altération : Trois conditions de déclenchement sont utilisées dans le scénario :

9. https://github.com/aymeric-cr/dsl-scenario/blob/master/workspace/journal-data-lux-15min-190419-330pm.bst [Dernière visite : décembre 2020]

— trig_vertrate : **eval when**(**AIRCRAFT.VERT_RATE** >= 0.15). Ce déclencheur spécifie que les propriétés des avions doivent être altérées quand leur vitesse ascensionnelle est égale ou supérieure à 0.15 pieds par minutes. L'évaluation de ce déclencheur sur l'enregistrement nominal produit un total de 820 intervalles de temps, répartis entre 158 avions. À titre d'exemple, l'avion portant l'indicatif d'appel *SWR97X* est concerné par 26 intervalles de temps.

— trig_altgs : **eval when**(**RAP.ALTITUDE** < 41001) **and until** (**AIRCRAFT.GROUNDSPEED** > 350). Ce déclencheur spécifie que les propriétés des avions doivent être altérés quand l'ensemble des avions volent strictement en-dessous de 41001 pieds et ce jusqu'à ce que leur vitesse dépasse les 350 nœuds. L'évaluation de ce déclencheur sur l'enregistrement nominal produit un total de 254 intervalles. Les avions sont concernés par des intervalles quasiment similaires à cause de la portée globale de la condition **RAP.ALTITUDE**. Les différences entre les intervalles de temps d'un avion à l'autre proviennent uniquement de la portée individuelle de la condition **AIRCRAFT.GROUNDSPEED**.

— trig_prism : **eval as_soon_as**(**inside prism with_vertices (48,7.7), (50.31,6.35), (49.18,4.22) and altitude from 1000 to 35000**). Ce déclencheur spécifie que les propriétés des avions doivent être altérés à partir du moment où ils entrent dans le prisme triangulaire défini dans le filtre *filt_prism*. L'évaluation de ce déclencheur sur l'enregistrement nominal résulte en un total de 30 intervalles, chaque avion possédant au plus 1 intervalle.

Les schémas d'altération : Le scénario est constitué de deux schémas d'altération. Le premier schéma présenté en figure 6.15 représente une fausse alerte.

```
1  let $start = {30,200,254,1065}
2  let $squawks = {7500,7600,7700}
3  alter all_planes satisfying "filt_lat"
4  at $start seconds triggered_by"trig_vertrate"
5  with_values SQUAWK = $squawks
```

FIGURE 6.15 – Premier schéma : *False Alarm Attack*

Dans ce premier schéma, deux variables sont déclarées. La première, ***$start***, est une liste de quatre entiers utilisée pour définir le début de la fenêtre de temps de l'altération. La deuxième, ***$squawks***, est une liste de trois entiers contenant différentes valeurs de squawk à utiliser. En combinant les valeurs contenues dans ces deux listes, il en résulte douze combinaisons de valeurs. Ainsi la concrétisation de ce schéma abstrait implique la création de $4 * 3 = 12$ schémas d'altération concrets.

Le deuxième schéma représente une disparition d'avion, comme le montre la figure 6.16.

```
1 let $filters = {"filt_lowalt","filt_prism"}
2 let $triggers = {"trig_altgs","trig_prism"}
3 hide all_planes satisfying $filters
4 at 0 seconds triggered_by $triggers
```

FIGURE 6.16 – Deuxième schéma : *Aircraft Disappearance Attack*

Dans ce deuxième schéma, deux variables sont également déclarées, ***$filters*** et ***$triggers***, contenant respectivement une partie des filtres et des déclencheurs présentés précédemment. Chacune des variables contenant deux valeurs, la concrétisation de ce schéma mène à la production de $2 * 2 = 4$ schémas concrets.

Le scénario : Le scénario est une combinaison des deux schémas abstraits précédents, (figures 6.15 et 6.16). Leur concrétisation menant respectivement à douze et quatre schémas concrets, la suite de test résultant de ce scénario se compose donc de $12*4 = 48$ cas de test.

Résultats : Les cas de test sont produits au travers de fichiers XML contenant les directives d'altérations [10]. Un fichier XML correspondant à un cas de test, on obtient dans le cas présent un ensemble de 48 fichiers XML. Un extrait de fichier de cas de test est montré en figure 6.17, l'ensemble de ces fichiers étant disponibles sur GitHub [11].

Dans ce fichier XML, on remarque deux nœuds actioncorrespondant chacun à une directive d'altération. Le premier, avec l'attribut alterationType=ALTERATION, représente une directive d'altération simple (voir section 5.2.1 p. 76). Il cible l'avion avec l'ICAO *480C1A* entre 135.148 secondes et 1006.640 secondes avec une modification de son squawk à 7700. Le deuxième nœud action, avec l'attribut alterationType=DELETION, représente une disparition d'avion (voir section 5.2.2 p. 78). Il cible l'avion avec l'ICAO *4B1611* entre 3.390 secondes et 1045.121 secondes. Lors de l'application de la directive à l'enregistrement nominal, tous les messages émis par l'avion entre ces deux dates sont supprimés.

Après avoir vu un extrait de fichier XML de cas de test, voici quelques métriques permettant de mieux cerner la volumétrie de l'ensemble de la suite de test générée :

- Chacun des 48 cas de test générés contient entre 9 et 346 directives d'altérations pour une moyenne de 233.25 directives par cas de test. En moyenne, 211.75 directives sont des altérations simples et 54.75 directives sont des disparitions d'avion.
- Les directives sont appliquées durant des fenêtres de temps allant de 160

10. Pour rappel, une directive d'altération spécifie les modifications à appliquer à l'enregistrement nominal, et sont prise en entrée par le moteur d'altération (voir section 5.1 p. 73)
11. https://github.com/aymeric-cr/dsl-scenario/tree/master/workspace/incidents [Dernière visite : décembre 2020]

```
<action alterationType="ALTERATION">
  <scope type="timeWindow">
    <lowerBound>135148</lowerBound>
    <upperBound>1006640</upperBound>
  </scope>
  <parameters>
    <target identifier="hexIdent">480C1A</target>
    <parameter mode="simple">
      <key>squawk</key>
      <value>7600</value>
    </parameter>
  </parameters>
</action>
<action alterationType="DELETION">
  <scope type="timeWindow">
    <lowerBound>3390</lowerBound>
    <upperBound>1045121</upperBound>
  </scope>
  <parameters>
    <target identifier="hexIdent">4B1611</target>
    <parameter mode="simple">
      <frequency>1</frequency>
    </parameter>
  </parameters>
</action>
```

FIGURE 6.17 – Extrait d'un fichier XML de cas de test.

millisecondes à 17 minutes.

- En tout, la suite de test générée contient 11196 directives d'altérations, incluant 9720 altérations simples et 1476 disparitions d'avion.

Au regard de cette volumétrie, il n'apparaît pas envisageable de spécifier manuellement les directives d'altérations prise en paramètre par le moteur de génération. Entre la recherche des cibles selon les conditions définies dans les filtres et le calcul des fenêtres de temps en fonction des déclencheurs, il semble raisonnable de faire l'hypothèse que chaque directive d'altération demande environ cinq secondes à être calculée manuellement, i.e. à l'aide de script créer pour l'occasion. Selon cette hypothèse, le plus petit des cas de test demanderait $9 * 5 = 45$ secondes pour être calculé, tandis que le plus grand demanderait $346 * 5 = 1730$ secondes. Si les temps peuvent paraître acceptables, il faut remettre cela en perspective avec les capacités de massification offertes par les trois DSL. En effet, dans cet exemple, qui reste très modeste vis-à-vis des scénarios communément manipulés, 11196 directives ont été générées.

De surcroît, il est important de prendre en considération l'indépendance des scénarios vis-à-vis d'un enregistrement précis. En effet, le scénario utilisé dans cette expérimentation est applicable à n'importe quel enregistrement car il ne contient aucune donnée spécifique à l'enregistrement nominal utilisé.

6.6/ SYNTHÈSE

Si le moteur d'altération permet d'appliquer les directives d'altérations de manière automatique, il apparaît que la spécification de ces directives est une tâche fastidieuse. Pour répondre à ce problème, un ensemble de trois langages dédiés a été créé : le langage de filtre, le langage de déclencheur et le langage de schéma d'altération. Le développement de ces trois langages a été détaillé dans ce chapitre au travers de cinq activités issues de la méthodologie de développement d'un DSL proposée par Mernik et al. [75].

L'activité de test et de validation des DSL a permis de mettre en exergue les deux points forts de cet ensemble de DSL : la finesse du pouvoir d'expression et la forte capacité de massification. Le premier point est lié directement à la question de recherche suivante : **RQ-1 : Dans quelle mesure un ensemble de langages dédiés assure-t-il la couverture de la taxonomie des FDIA ?**. Dans la section 6.5.1 p. 129, il a été montré que le pouvoir d'expression des DSL permet en effet de couvrir la taxonomie des FDIA identifiées dans le domaine aérien.

Les capacités de massification des langages sont liées à la question **RQ-2 : Dans quelle mesure est-il possible de garantir le réalisme de données simulant des FDIA, tout en les générant de manière automatique et massive ?**. Ici, l'expérimentation de la section 6.5.2 p. 132 donne un début de réponse à la question. En effet, si les DSL offrent des capacités de massification et d'automatisation, mais c'est le module de génération des données qui est le garant du réalisme.

D'une manière générale, les trois chapitres de cette partie donnent des premières réponses aux questions de recherche. La partie suivante vise à compléter ces réponses au travers de trois cas d'utilisation concrets de nos travaux de thèse. Toutefois, avant de présenter ces trois cas d'utilisation, il convient de présenter dans le chapitre suivant comment le moteur d'altération et l'ensemble des trois DSL interagissent au sein de la chaîne outillée nommée FDI-T (*False Data Injection Testing*).

III

IMPLÉMENTATION ET CONTEXTES D'USAGE

7

FDI-T : CHAÎNE OUTILLÉE

L'ensemble des contributions présentées dans ce manuscrit sont implémentées au sein d'une chaîne outillée nommée FDI-T pour *False Data Injection Testing*. FDI-T suit l'architecture présentée dans le chapitre 4. Deux partenaires industriels sont impliqués dans le développement de cette chaîne outillée : l'entreprise bisontine Smartesting [1] a produit le corps de l'application, notamment l'interface graphique, tandis que l'entreprise rennaise Kereval [2] a développé une version du moteur d'exécution qui concerne l'établissement du verdict des tests ainsi que la connexion au SUT. Dans le cadre de la convention de partenariat entre l'Université de Franche-Comté et Smartesting, les développements logiciels réalisés pour la thèse ont été intégrés au logiciel existant notamment avec des éditeurs textuels pour la création de schémas d'altération, de filtres et de déclencheurs, ainsi qu'une version alternative du générateur de données altérées pour répondre à des besoins R&D spécifiques. En parallèle de l'intégration des travaux de thèse à FDI-T, un portage du logiciel vers le domaine maritime a été effectué. Enfin, durant tout au long de la thèse, une part non-négligeable du temps de travail était dédiée au développement logiciel dans le but d'améliorer la qualité globale du logiciel FDI-T. Ces développements concernent l'ajout de nouvelles fonctionnalités, l'augmentation de la couverture de test, la correction d'anomalies, l'amélioration de la qualité du code, ainsi que la mise en place d'une chaîne d'intégration continue. Tous les développements sont réalisés en langage Java.

Ce chapitre présente l'implémentation des travaux de thèse au sein de la chaîne outillée FDI-T et l'apport qu'ils représentent. La première section présente, dans les grandes lignes, le logiciel dans ses versions aérienne et maritime. La deuxième section se focalise sur la mise en place des DSL dans FDI-T. Enfin la troisième et dernière section détaille la mise en place du moteur de génération de données altérées et les options qui lui sont associées.

1. https://www.smartesting.com/ [Dernière visite : décembre 2020]
2. https://www.kereval.com/ [Dernière visite : décembre 2020]

7.1/ FDI-T : *False Data Injection Testing*

Avant l'intégration des travaux de thèse, la chaîne outillée FDI-T se présentait sous la forme d'une interface graphique permettant la conception et l'exécution de tests pour les systèmes de surveillance du trafic aérien à travers différents éditeurs graphiques :

— **L'éditeur graphique de scénario** qui permet la conception de scénarios simples, il propose une visualisation des enregistrements en quatre dimensions (latitude, longitude, altitude et temps) des enregistrements.

— **L'éditeur de radar** qui permet, depuis une carte du monde, de modéliser les radars nécessaires à l'étape de conversion introduite en section 4.1.2 p. 60.

— **L'éditeur d'exécution** qui permet de paramétrer l'exécution des tests et l'injection des données dans le SUT.

— **L'éditeur de zone** qui permet la définition graphique de zones sur une carte du monde, ces zones peuvent ensuite être utilisées dans les scénarios.

Cette section se concentre principalement sur les éditeurs graphiques de scénario et d'exécution. En effet, ces deux éditeurs sont la base de la création et l'exécution de scénarios sur un SUT, l'objectif est de les présenter brièvement et d'en identifier les limitations auxquelles nous avons répondu par l'intégration de nos travaux de thèse à FDI-T.

7.1.1/ L'ÉDITEUR GRAPHIQUE DE SCÉNARIO

La figure 7.1 représente l'éditeur graphique de scénario. Le scénario présenté sur la figure est une modification de propriétés (voir section 5.2.1 p. 76) qui change la valeur du *squawk* à 7700 pour tous les avions de l'enregistrement entre la seconde 79 et la seconde 358. Les paramètres de cette modification sont visibles dans la partie basse de l'éditeur. La carte en haut à droite permet de visualiser les trajectoires de tous les avions présents dans l'enregistrement, tandis que le graphique en haut à gauche permet de visualiser leur altitude en fonction du temps. Sur la carte, on remarque également la présence d'un cercle bleu, il représente un récepteur de données ADS-B associé au scénario.

Si l'éditeur graphique de scénario offre une visualisation conviviale aux utilisateurs, la contrepartie est que les scénarios conçus via cette méthode ne sont applicables qu'à l'enregistrement à partir duquel ils ont été créés. Ainsi, la conception et l'exécution massive d'altérations sur de nombreux enregistrements sont impossibles puisque l'utilisateur est contraint de repasser par la phase de conception graphique pour chaque enregistrement. Cette version graphique ne permet pas de rendre générique un schéma d'altération.

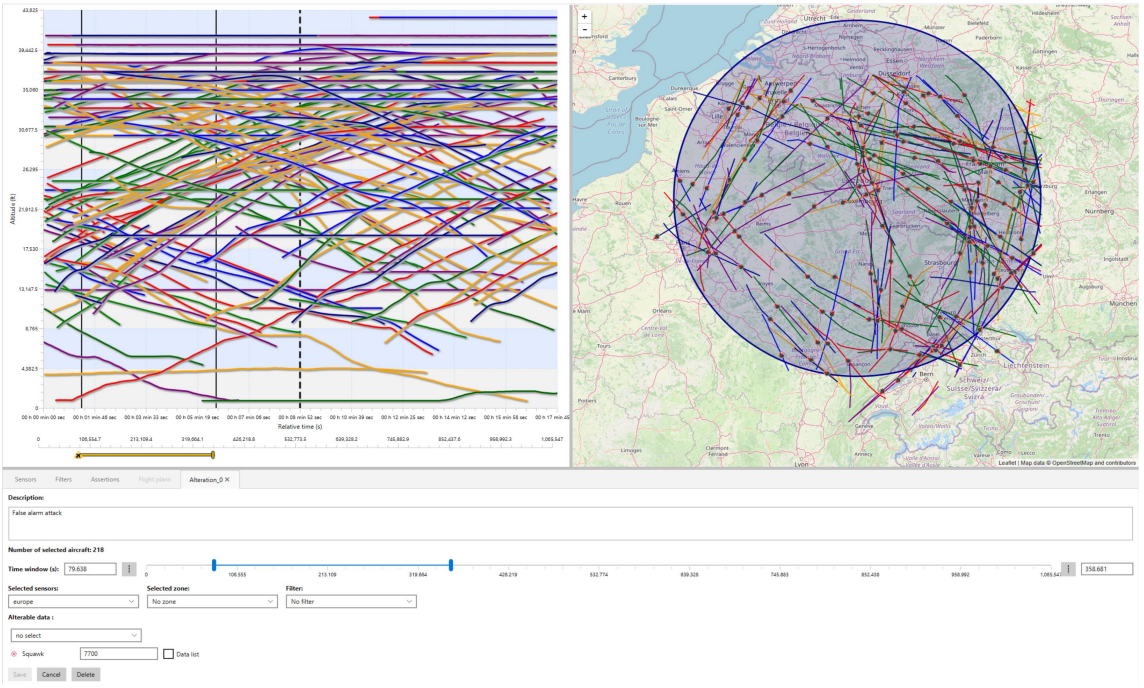


FIGURE 7.1 – Éditeur graphique de scénario de FDI-T pour le domaine aérien.

7.1.2/ L'ÉDITEUR D'EXÉCUTION

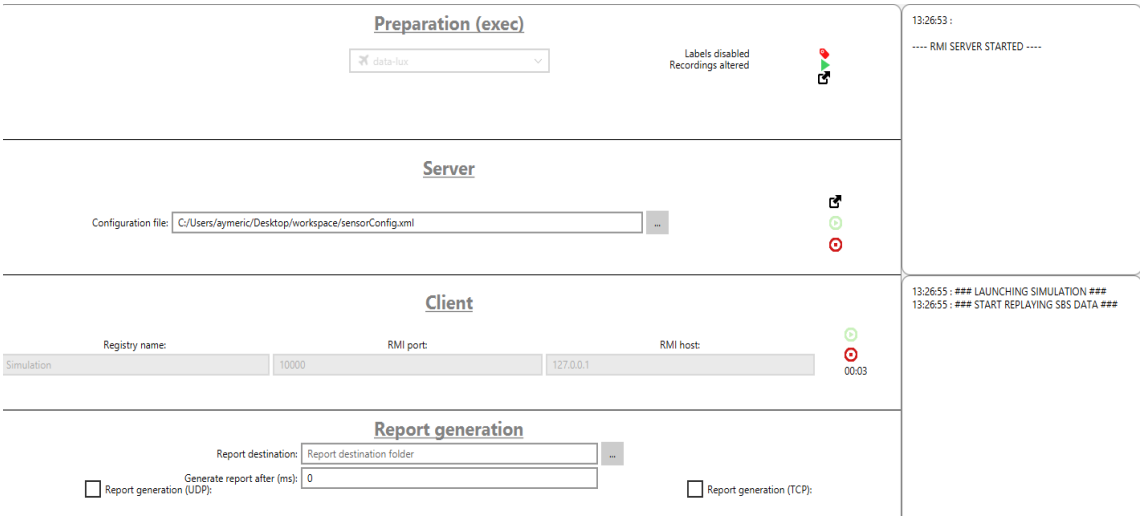


FIGURE 7.2 – Éditeur d'exécution de FDI-T.

L'éditeur d'exécution est visible en figure 7.2, il permet de lancer les tests et de paramétrer la connexion au SUT. Une exécution se divise en plusieurs étapes :

1. L'enregistrement nominal est altéré selon le scénario choisi dans la partie "Preparation" de l'éditeur.
2. La connexion au SUT est effectuée depuis la partie "Server" en suivant des

paramètres définis dans un fichier de configuration au format XML.

3. Le test est exécuté depuis la partie “Client”. Une exécution consiste à envoyer les messages ADS-B contenus dans l’enregistrement altéré en respectant la temporalité des messages. L’injection dans le SUT d’un enregistrement contenant un extrait du trafic aérien de vingt minutes mettra donc autant de temps pour l’exécuter.
4. À l’issue de l’exécution, un rapport de test est généré. Ce rapport est réalisé via des scripts Groovy qui analysent la sortie du SUT. La génération du rapport est configurée dans la partie “Report generation” de l’éditeur d’exécution.

Cet éditeur offre également la possibilité de récupérer directement les données altérées sans passer par une exécution en temps réel, cela permet de visualiser rapidement le résultats d’un scénario sur un enregistrement. Toutefois, la dépendance à l’interface graphique empêche une fois de plus la récupération automatique efficace des données altérées. Pour visualiser l’exécution d’un scénario, le logiciel Virtual Radar[3] est utilisé. Il permet le rejeu des messages ADS-B et affiche le trafic correspondant aux données reçues.

7.1.3/ LA VERSION MARITIME DE FDI-T

L’architecture suivie par FDI-T étant générique, nous avons pu réaliser le portage du code de la version aérienne vers une version maritime avec un effort limité de développement. Du point de vue de l’interface graphique, ce changement de domaine se constate essentiellement dans l’éditeur graphique de scénario.

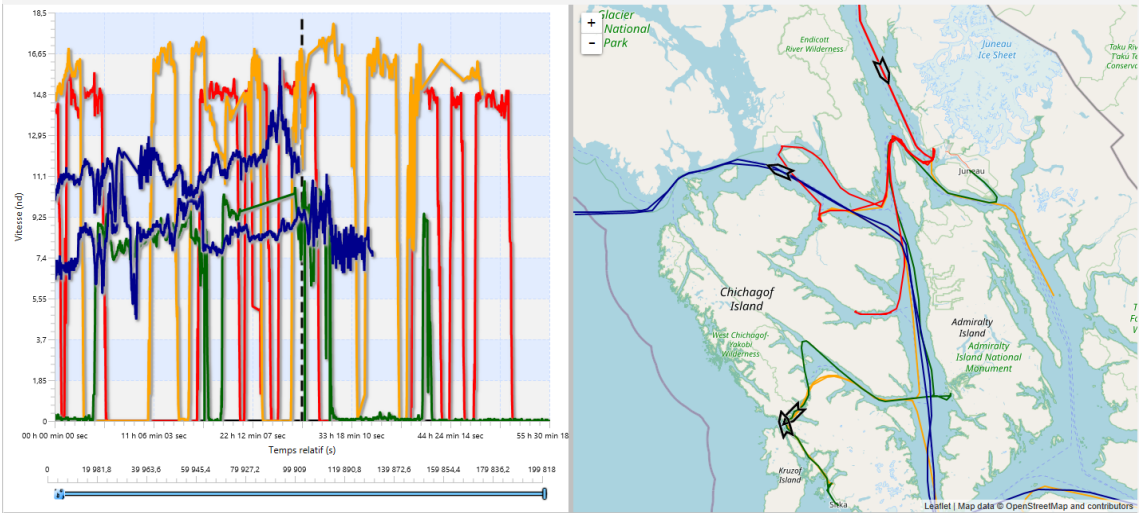


FIGURE 7.3 – Éditeur graphique de scénario de FDI-T pour le domaine maritime.

3. https://www.virtualradarserver.co.uk/ [Dernière visite : décembre 2020]

On remarque en effet dans la figure 7.3 que les trajectoires représentées sur la carte (partie droite) correspondent à des trajets de bateaux. De plus, les bateaux évoluant dans un environnement en deux dimensions, la vitesse des bateaux a été utilisée à la place de l'altitude pour remplir le graphique de l'éditeur de scénario (partie gauche).

Au niveau des données, les adaptations concernent surtout le changement d'environnement (passage de la 3D à la 2D) et les échelles de temps qui sont très différentes d'un domaine à l'autre. En effet, si les enregistrements manipulés dans le domaine aérien durent généralement au plus quelques heures, dans le domaine maritime, il n'est pas rare qu'un utilisateur désire visualiser des enregistrements de plusieurs centaines d'heures. Pour éviter les problèmes de performance pour la visualisation, le pas de temps de l'interpolation pour l'affichage de trajectoire (voir section 6.4.2 p. 124) a été fixé à 10 minutes contre 5 secondes dans le domaine aérien.

D'une manière générale, si l'utilisation d'une interface graphique rend la conception et l'exécution des tests plus conviviale et offre des capacités de visualisation intéressantes, elle reste toutefois un frein à une automatisation efficace. L'intégration des travaux de thèse permet une amélioration de l'automatisation du processus d'altération, sans pour autant perdre les avantages des outils de visualisation intégrés à FDI-T.

7.2/ L'INTÉGRATION DES TROIS LANGAGES DÉDIÉS

Pour chacun des trois DSL présentés dans le chapitre 6, un éditeur dédié a été ajouté à l'interface graphique de FDI-T :

— **L'éditeur de schémas** permet la création de schémas d'altération et leur conversion en scénarios graphiques.

— **L'éditeur de filtres** permet la création de filtres ainsi que leur exécution sur un enregistrement. Cela permet à l'utilisateur d'obtenir la liste des avions qui satisfont les critères du filtre.

— **L'éditeur de déclencheur** permet la création de conditions de déclenchement. Ici aussi, l'utilisateur peut exécuter les déclencheurs sur un enregistrement et ainsi visualiser, pour chaque avion, les intervalles de temps durant lesquels les conditions de déclenchements sont vérifiées.

À noter qu'une version de FDI-T limitée à l'utilisation de ces trois éditeurs est disponible sur Github [4]. Dans cette version, l'exécution de scénarios textuels produit un ensemble de fichier XML qui contiennent les directives d'altération correspondantes.

4. https://github.com/aymeric-cr/dsl-scenario [Dernière visite : décembre 2020]

7.2.1/ L'ÉDITEUR DE SCHÉMA

L'éditeur de schéma permet la définition de schémas d'altération tels qu'ils sont définis en section 6.2.2 p. 101. Cet éditeur, visible en figure 7.4, offre une coloration syntaxique ainsi qu'un retour sur les erreurs (syntaxiques et sémantiques). Il permet également la conversion d'un schéma d'altération vers un scénario graphique.

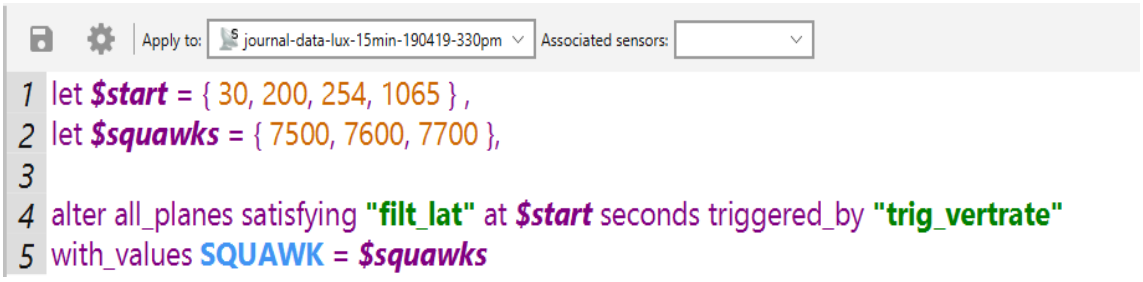


FIGURE 7.4 – Éditeur de schéma de FDI-T.

On remarque dans la figure deux références à un filtre ("*filt_alt*") et à un déclencheur ("*trig_vertrate*"). Ces éléments doivent être présents dans l'espace de travail afin de pouvoir être référencés dans un schéma d'altération. Un scénario étant une combinaison de plusieurs schémas d'altération, l'éditeur d'exécution propose de créer des scénarios textuels en combinant des schémas d'altération. Ce mécanisme est illustré en figure 7.5.

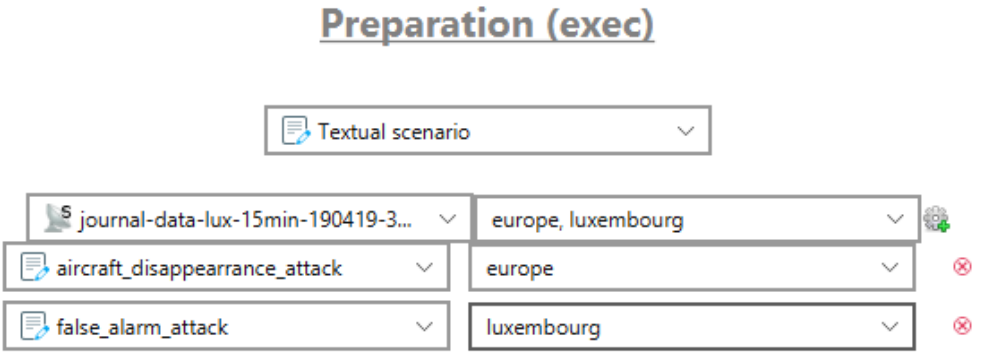


FIGURE 7.5 – Création d'un scénario textuel par la combinaison de deux schémas d'altération.

Dans cet exemple, un scénario textuel est créé sur l'enregistrement "journal-data-lux-15min-...". Les radars "europe" et "luxembourg" sont associés au scénario. Le premier est ciblés par le schéma d'altération "aircraft_disappearrance_attack" tandis que le deuxième est ciblé par le schéma d'altération "false_alarm_attack".

7.2.2/ L'ÉDITEUR DE FILTRES

Les filtres, comme ils sont définis en section 6.2.11 p. 109 peuvent être créés par le biais d'un éditeur dédié. Comme l'éditeur de schéma, il propose une coloration syntaxique ainsi que l'affichage des erreurs syntaxiques et sémantiques du filtre.

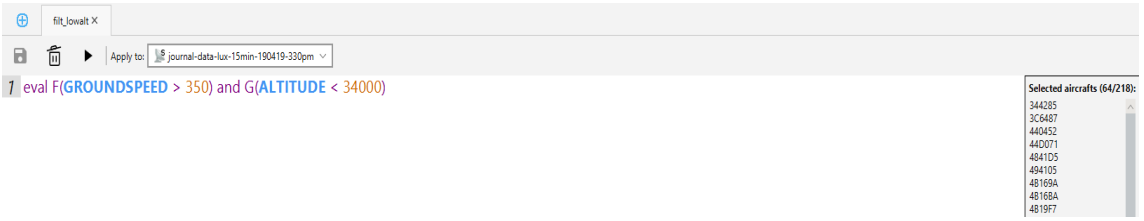


FIGURE 7.6 – Éditeur de filtre de FDI-T.

La figure 7.6 donne un aperçu de l'éditeur de filtre. On remarque dans la partie droite de l'image qu'une liste d'ICAO est affichée. Elle correspond à l'application du filtre à un enregistrement donné : les ICAO listés correspondent aux avions qui satisfont les conditions exprimées dans le filtre, c'est-à-dire les avions dont la vitesse est au moins une fois strictement supérieure à 350 nœuds et dont l'altitude est toujours strictement inférieure à 34000 pieds. Appliquer un filtre en cours de conception à un enregistrement donné constitue une étape clé du processus de conception des filtres, en ceci qu'elle permet de valider que la granularité de filtrage conçue correspond à celle envisagée.

7.2.3/ L'ÉDITEUR DE DÉCLENCHEURS

Enfin, les déclencheurs comme ils sont définis en section 6.2.12 p. 110 peuvent être créés via l'éditeur de déclencheur. Une fois de plus, cet éditeur offre une coloration syntaxique ainsi qu'un aperçu des erreurs syntaxiques et sémantiques.

Dans la figure 7.7, on remarque un déclencheur qui consiste en une condition de déclenchement relative à l'entrée d'un avion dans une zone donnée. Les intervalles de temps produits par l'évaluation de ce déclencheur sur un enregistrement sont visibles dans la partie basse de l'image. Le code couleur utilisé est le suivant : un intervalle rouge signifie que l'avion n'est pas affecté par l'altération à ce moment, un intervalle bleu signifie que l'avion est affecté par l'altération à ce moment, tandis qu'un intervalle gris signifie que l'avion n'émet pas de données durant ce laps de temps.
De même que pour l'éditeur de filtre, la visualisation des intervalles d'altération issus de l'évaluation du déclencheur courant sur un enregistrement donné permet sa validation vis à vis des déclenchements d'altération envisagés. Il est également possible ici de précéder l'application du déclencheur par l'application d'un filtre, en vu de réduire le nombre d'avions sur lesquels sera appliqué le déclencheur. Cette fonctionnalité participe à la fois

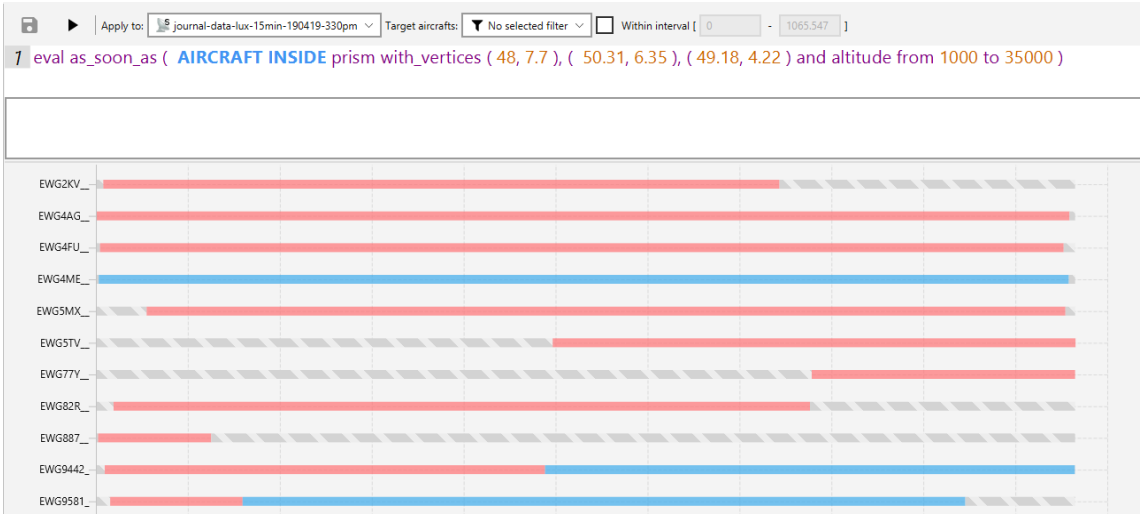


FIGURE 7.7 – Éditeur de déclencheurs de FDI-T.

à valider le déclencheur, par exemple en filtrant un seul avion d'un enregistrement et en observant plus précisément les effets du déclencheur sur cet avion précis, et participe à la fois à la conception du schéma d'altération sous-jacent, puisque les schémas d'altération se conçoivent selon cet ordre filtrage – déclenchement.

7.3/ L'INTÉGRATION DE LA GÉNÉRATION DE DONNÉES ALTÉRÉES

Tandis que l'ensemble des trois DSL est pleinement intégré à FDI-T, le moteur de génération est un élément indépendant de la chaîne outillée. Il se présente comme un module externe et le lien entre les scénarios et la génération de données altérées se fait au travers de fichiers XML contenant les directives d'altération. Une version du moteur de génération est également disponible sur Github [5].

L'implémentation du moteur de génération étant précédemment détaillée dans le chapitre 5, cette section se concentre d'avantage sur sa capacité à labelliser les données pour le test / l'entraînement des composants à base d'apprentissage automatique, ainsi que sur son utilisation en mode console et les différentes options d'utilisation associées.

7.3.1/ LA LABELLISATION DES DONNÉES

Afin d'offrir la possibilité de générer des données pour l'apprentissage supervisé de modèle de *Machine Learning*, un mécanisme de labellisation des données altérées a été mis en place dans le moteur de génération. Pour chacun des deux domaines

5. https://github.com/aymeric-cr/sbs-generation [Dernière visite : décembre 2020]

d'application, ce mécanisme consiste à calculer un masque booléen indiquant, pour chaque message ADS-B ou AIVDM, quelles sont les propriétés qui ont été modifiées.

TABLE 7.1 – Calcul du masque pour la labellisation.

Propriété	Puissance	Valeur
ICAO	13	8192
Callsign	12	4096
Squawk	11	2048
Altitude	10	1024
Vitesse au sol	9	512
Route	8	256
Latitude	7	128
Longitude	6	64
Vitesse ascensionnelle	5	32
Alerte	4	16
Date	3	8
Urgence	2	4
SPI	1	2
Sol	0	1

Le tableau 7.1 liste la puissance ainsi que la valeur associée à chaque propriété. Ainsi, si une altération modifie, par exemple, l'altitude et la route contenues dans un message, le masque booléen sera 00010100000000 et sa valeur décimale sera donc $1024+256 = 1280$. Quand la labellisation est activée, ce masque est ajouté à la suite des messages altérés lors de la génération de données.

Une valeur décimale est également ajoutée à la suite du masque pour spécifier le type d'altération. En effet, une modification de trajectoire et une modification de propriété peuvent potentiellement modifier les mêmes propriétés. Il peut être utile pour les modèles d'être en mesure de différencier ces deux cas, c'est pourquoi l'ajout de cette valeur décimale est nécessaire.

7.3.2/ LE MODE CONSOLE

Afin de contourner les limites liées à l'utilisation d'une interface graphique pour l'exécution des scénarios de bout en bout (de la conception à la génération de données), un mode console a été mis en place. Il permet dans un premier temps l'exécution massive et automatique de scénarios sur un grand nombre d'enregistrements, et dans un second temps l'utilisation de FDI-T sur des systèmes ne disposant pas d'environnement graphique, comme des serveurs de calcul.

L'exécution d'un scénario en mode console se fait via un fichier de configuration ".ini" dont un exemple est visible en figure 7.8. Dans cet exemple, l'utilisateur a spécifié dans la section "recording" un chemin relatif vers un dossier contenant des enregistrements.

```
[recording]
path=recordings/

[sensors]
europe=Europe.sensor

[alterations]
false-alarm='alter all_planes from 0.2 * REC_DURATION seconds until 0.3 * REC_DURATION seconds with_values SQUAWK = 7700'

[scenario]
false-alarm=europe
```

FIGURE 7.8 – Exemple de fichier de configuration pour le mode console.

Lors de l'exécution, tous les enregistrements contenus dans ce dossier seront altérés selon un scénario donné. Dans la section "sensors", l'utilisateur spécifie un chemin vers un fichier représentant un radar et produit via l'éditeur de radar de FDI-T. Les schémas d'altération sont définis dans la section "alterations", dans l'exemple, le schéma en question représente une modification du squawk à la valeur 7700 (statut d'alerte générale). Enfin, la section "scenario" sert à spécifier quels radars sont ciblés par quelles altérations, comme cela est fait dans l'éditeur d'exécution de FDI-T (voir figure 7.2).

À partir de ce fichier de configuration, la génération de données est exécutable via la ligne de commande suivante : `java -jar -f param.ini`, "param.ini" indiquant le chemin (relatif ou absolu) vers le fichier de configuration

Afin de personnaliser la génération de données altérées, trois options peuvent être ajoutées à cette ligne de commande :

— `-l` ou `--label` : permet d'activer la labellisation des données.

— `-r <number>` ou `--random <number>` : permet de ne générer qu'un sous ensemble d'enregistrements altérés parmi toutes les combinaisons possibles entre les listes et les plages de données. La taille de ce sous-ensemble est spécifiée par l'argument `<number>`.

— `-s` ou `--simulation` permet d'activer ou non le mode de réalisme avancé. Il est reservé aux modifications de trajectoires, en effet, dans ce mode les valeurs des propriétés associées au mouvement de l'avion (vitesse au sol, vitesse ascensionnelle, et route) sont modifiées automatiquement en fonction de l'allongement de la trajectoire (voir section 5.2.3.2 p. 80).

7.4/ SYNTHÈSE

L'intégration des travaux de thèse à FDI-T permet d'offrir aux utilisateurs un environnement permettant d'avoir une phase de conception des tests supportée par des outils de visualisation (carte de l'éditeur graphique de scénario et aperçus des résultats

des filtres et des déclencheurs), ainsi que des capacités d'automatisation favorisant la génération massive de données altérées. De plus l'ajout de labels et d'options au moteur de génération, permet de répondre à des besoins plus variés.

A ce stade, l'outil FDI-T étendu avec nos développements de thèse constitue un prototype de recherche. Les chapitres suivants montrent comment la chaîne outillée FDI-T a été utilisée pour répondre à des besoins concrets d'altérations de données pour des travaux de recherche centrés sur la détection d'anomalies via l'utilisation de techniques d'intelligence artificielle.

8

EXPÉRIMENTATIONS DANS LES DOMAINES AÉRIEN ET MARITIME

Ce chapitre rapporte trois contextes de recherche issus des domaines aérien et maritime pour lesquels l'initiation des travaux a été rendue possible grâce à la chaîne outillée FDI-T. Chacun des contextes nécessite en effet des données spécifiques, que ce soit pour reproduire les effets de FDIA ou pour créer des situations absentes des bases de données, seule l'utilisation de FDI-T a permis de satisfaire ces besoins. Les sections qui suivent présentent ces travaux de recherche en cours et, pour chacun d'eux, déterminent le rôle qu'a eu FDI-T en tant que facilitateur d'expérimentation.

Les deux premiers contextes de recherche concernent le domaine aérien, et sont conduits dans le cadre du projet ANR GeLeaD [1] (*Generate Learn and Detect*). Ce projet vise l'étude de mécanismes de détection des FDIA par *Machine Learning* en utilisant des données altérées générées. Un premier axe de recherche concerne la détection des FDIA par apprentissage supervisé, tandis que le second axe explore les possibilités de détection par apprentissage non-supervisé. Ces travaux ont ainsi permis d'explorer plusieurs des enjeux de cette thèse : le pouvoir d'expression des langages DSL pour implémenter les altérations de données nécessaires à GeLeaD, le réalisme des altérations produites, et les capacités de massification.
Le troisième contexte de recherche concerne le domaine maritime, où FDI-T est utilisé pour la construction de modèles de *Machine Learning* dans le cadre de travaux menés au sein du *Simula Research Laboratory* [2] en partenariat avec l'Institut FEMTO-ST pour la thèse de Pierre Bernabé [3] qui est co-encadrée entre les deux centres de recherche.
À noter que dans le cadre du partenariat entre l'Université de Franche-Comté et Smartesting, la chaîne outillée FDI-T est également utilisée dans un contexte industriel lié au domaine aérien. Cette expérimentation a ciblé le test d'un système de contrôle aérien militaire avec comme objectif la production par FDI-T des test du système par altération

1. https://projects.femto-st.fr/gelead/fr [Dernière visite : décembre 2020]
2. https://www.simula.no/ [Dernière visite : décembre 2020]
3. https://www.simula.no/people/pierbernabe [Dernière visite : décembre 2020]

des messages ADS-B. Cette expérimentation a aussi constitué une validation pour nos travaux, mais nous n'avons pas été autorisé à en faire le report dans cette thèse du fait du caractère classifié (confidentiel défense) des expérimentations et des résultats. Cela nous a conduit à restreindre la restitution des expérimentations à celles menées en lien avec les travaux de recherche cités précédemment.

Il est bon de préciser que les travaux liés aux trois contextes de recherche sus-cités sont en cours à l'heure actuelle et ne font pas encore l'objet de publications. L'objectif de ce chapitre est en effet d'illustrer le rôle crucial qu'a FDI-T en tant que facilitateur d'expérimentations, et de montrer ainsi que son utilisation rend possible la conduite de travaux de recherche pour lesquels les expérimentations de validation peuvent être très chronophages, voir impossible, sans l'appui fourni par la chaîne outillée. D'autre part, la qualité potentielle des résultats de recherche issus de ces trois contextes ne constitue pas une forme de validation FDI-T, puisque la chaîne outillée est étrangère à la pertinence des hypothèses de recherche des travaux l'utilisant.

Le chapitre est structuré comme suit. La première section traite le cas de l'apprentissage supervisé dans le domaine aérien. La deuxième section se focalise sur l'apprentissage non-supervisé, également dans le domaine aérien. Le domaine change pour la troisième section qui aborde le cas de l'apprentissage supervisé pour le domaine maritime. Enfin, la quatrième et dernière section dresse le bilan des enseignements à tirer de ces expérimentations. Ces expérimentations constituent un support pour discuter les trois questions de recherche au regard des contributions techniques de la thèse.

8.1/ TECHNIQUE D'APPRENTISSAGE SUPERVISÉ DANS LE DOMAINE AÉRIEN

C'est au sein de l'institut FEMTO-ST [4] et en partenariat avec les entreprises Smartesting et Flowbird qu'est mené le projet GeLeaD. Le projet se focalise sur la détection des attaques altérant la sémantique des données au sein d'un domaine tout en respectant la syntaxe du protocole. La finalité de ce projet est de proposer un ensemble d'outils capables de détecter les FDIA dans deux domaines : celui de la surveillance aérienne avec le protocole ADS-B (Smartesting) et celui des systèmes de parkings intelligents avec la solution *Park&Breathe* (Flowbird). Seul le premier domaine nous intéresse dans le cadre de cette expérimentation. La comparaison entre les approches de détection utilisant une technique d'apprentissage supervisé et celles utilisant une technique d'apprentissage non-supervisé est centrale dans le projet GeLeaD. À ce titre, cette section et la suivante partagent un contexte très similaire, ce sont principalement les

4. https://www.femto-st.fr/fr [Dernière visite : décembre 2020]

techniques d'apprentissage utilisées qui diffèrent. Les travaux autour de l'apprentissage supervisés dans le projet GeLeaD sont menés par Ralph Karam [5], doctorant à l'institut FEMTO-ST avec lequel nous avons collaboré pour ces expérimentations, notre rôle étant la mise à disposition de la chaîne outillée d'altération.

La première sous-section pose le contexte des travaux de Ralph Karam, la deuxième sous-section détaille la mise en œuvre de FDI-T pour l'apprentissage supervisé. Finalement, la troisième et dernière sous-section présente les données d'entraînement qui sont générées par notre approche.

8.1.1/ CONTEXTE DES TRAVAUX DE RECHERCHE

Dans le cadre du projet GeLeaD, Karam et al. ont mené une étude comparative des architectures de *Deep Learning* pour la détection d'anomalies dans les données ADS-B [51]. Les résultats indiquent que les architectures de type *Long Short-Term Memory* [35] (LSTM) semblent être les plus pertinentes pour effectuer ces tâches de détection. Pour les travaux de GeLeaD avec un apprentissage supervisé, c'est donc une architecture LSTM qui est utilisée. Le périmètre des anomalies détectées est réduit à des modifications graduelles de certaines propriétés transmises par les avions. Une attaque typique consiste, par exemple, à une augmentation ou une diminution progressive de l'altitude, de manière plus ou moins fine. Puisque l'apprentissage est supervisé, il est nécessaire que les données soient annotées par une information indiquant, pour chaque message, s'il a été altéré ou non. L'approche de détection mise en place par Ralph Karam est schématisé en figure 8.1, on remarque sur ce schéma que FDI-T génère les données d'entraînement ainsi que les données de test du modèle de détection.

Comme vu en section 1.2.2 p. 13, l'absence de données certifiées anormales en grande quantité dans les historiques du trafic réel requiert la génération de données synthétiques pour l'entraînement du composant de détection. L'objectif est de mesurer l'apport des travaux de thèse au travers de cas d'utilisation concrets dans le cadre de travaux scientifiques. Pour atteindre cet objectif, la chaîne outillée FDI-T est utilisée pour la génération des données d'entraînement. Ce premier cas d'utilisation de nos travaux permet ainsi d'apporter une réponse la question de recherche **RQ-2 : Dans quelle mesure est-il possible de garantir le réalisme de données simulant des FDIA, tout en les générant de manière automatique et massive ?**

En effet, puisqu'il est question d'entraîner un modèle de *Deep Learning*, l'utilisation d'un gros volume de données est primordiale pour garantir un entraînement efficace. De plus, la qualité des données altérées est importante afin de rendre la détection pertinente et d'éviter au mieux les biais, le but n'étant pas de détecter uniquement les anomalies

5. https://www.femto-st.fr/fr/personnel-femto/ralphkaram [Dernière visite : décembre 2020]

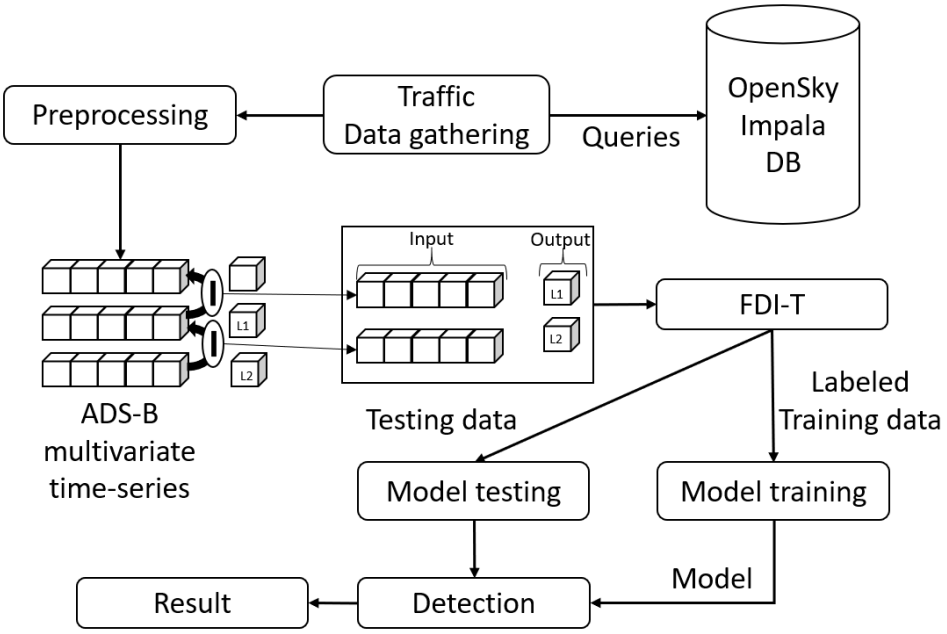


FIGURE 8.1 – Architecture d'apprentissage supervisé pour la détection d'anomalies.

générés par notre approche, mais également celles pouvant survenir en situation réelle. L'idéal serait de confronter les composants de détection à plusieurs générateurs de données anormales, ce qui n'est cependant pas encore réalisable à ce stade des travaux.

8.1.2/ MISE EN ŒUVRE DE FDI-T POUR L'APPRENTISSAGE SUPERVISÉ

Ce cas d'utilisation se focalise sur la détection d'attaques de type "décalage graduel". De telles attaques augmentent progressivement la valeur d'une propriété dynamique au cours du temps. Ralph Karam est en charge de l'implémentation des modèles de détection. À l'heure actuelle (décembre 2020), un modèle de détection a été créé pour chacune des propriétés dynamiques modifiées via un décalage graduel : l'altitude, la vitesse au sol, la route, la latitude, la longitude et la vitesse ascensionnelle. Pour chacune des propriétés, un scénario est utilisé. La figure 8.2 illustre un exemple de décalage graduel de latitude appliqué à un vol entre Londres et Varsovie. La trajectoire jaune représente le vol original, tandis que la trajectoire rouge représente le vol décalé. À l'atterrissage, les deux avions sont séparés par une distance d'environ 80 kilomètres.

Pour le décalage graduel de l'altitude, on remarque, avec l'utilisation du terme **at** 0 **seconds** que l'altération a lieu durant toute la durée de l'enregistrement. Le modèle est entraîné sur une amplitude de 75 pieds. Cette amplitude est un multiple de 25 afin de respecter le pas d'altitude utilisé au sein du protocole ADS-B. De plus cette valeur correspond à la différence moyenne d'altitude constatée entre deux messages originaux

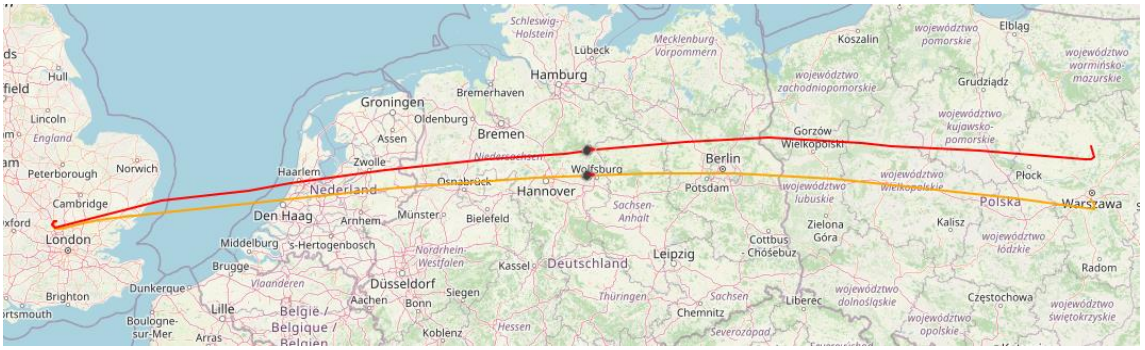


FIGURE 8.2 – Décalage de latitude appliqué à un vol entre Londres et Varsovie.

consécutifs (lorsque ceux-ci attestent d'une variation d'altitude). Pour tous les autres scénarios de ce cas d'utilisation, la différence moyenne de variation est utilisée pour déterminer l'amplitude du décalage. La figure 8.3 représente le scénario utilisé pour le décalage d'altitude.

```
1 alter all_planes at 0 seconds with_values ALTITUDE ++ = 75
```

FIGURE 8.3 – Scénario du décalage graduel de l'altitude.

Pour entraîner le modèle chargé de détecter le décalage de la vitesse au sol, la variation moyenne est de 1.8 nœuds. Ici aussi, l'altération a lieu pendant toute la durée de l'enregistrement. Le scénario utilisé est visible en figure 8.4.

```
1 alter all_planes at 0 seconds with_values GROUNDSPEED ++ = 1.8
```

FIGURE 8.4 – Scénario du décalage graduel de la vitesse au sol.

La route est décalée avec une amplitude de 0.9 degrés. Une fois encore, le décalage est effectué sur toute la durée de l'enregistrement. Le scénario utilisé est visible en figure 8.5.

Enfin, le principe reste le même pour les décalages de latitude et de longitude. Le modèle est entraîné sur des décalages avec une amplitude de respectivement 0.00488 degrés et 0.01278 degrés. Ces décalages s'étendent sur toute la durée de l'enregistrement. Les scénarios utilisés sont visibles en figure 8.6.

8.1.3/ LES DONNÉES D'ENTRAÎNEMENT

Pour le cas de l'apprentissage supervisé, le jeu de données nominal se compose de 100 enregistrements récupérés aléatoirement via la base de données OpenSky. Le volume de l'enregistrement est de 91 Mo. La majorité des enregistrements contient un unique vol qui décolle d'un aéroport pour atterrir à un autre, et quelques enregistrements contiennent un extrait de vol durant quelques dizaines de minutes. Le jeu de données peut-être résumé en quelques métriques :

```
1 alter all_planes at 0 seconds with_values TRACK ++ = 0.9
```

FIGURE 8.5 – Scénario du décalage graduel de la vitesse au sol.

```
1 alter all_planes at 0 seconds with_values LATITUDE ++ = 0.00488
```

```
1 alter all_planes at 0 seconds with_values LONGITUDE ++ = 0.01278
```

FIGURE 8.6 – Scénarios des décalages graduels de la latitude et de la longitude.

— Le vol le plus long du jeu de données dure 3 heures et 19 minutes, tandis que l'extrait le plus court dure 10 minutes et 11 secondes. En moyenne les enregistrements durent 53 minutes et 28 secondes.

— L'enregistrement le plus volumineux contient 30297 messages, tandis que le plus léger contient 1184. En moyenne les enregistrements contiennent environ 8615 messages.

— L'enregistrement le plus dense contient 9.1 messages par seconde, tandis que le moins dense contient 0.7 messages par seconde. La densité moyenne des enregistrements est de 3 messages par seconde.

La figure 8.7 montre les huit enregistrements les plus longs du jeu de données nominal. Ils contiennent tous un vol complet et se situent tous aux États-Unis.

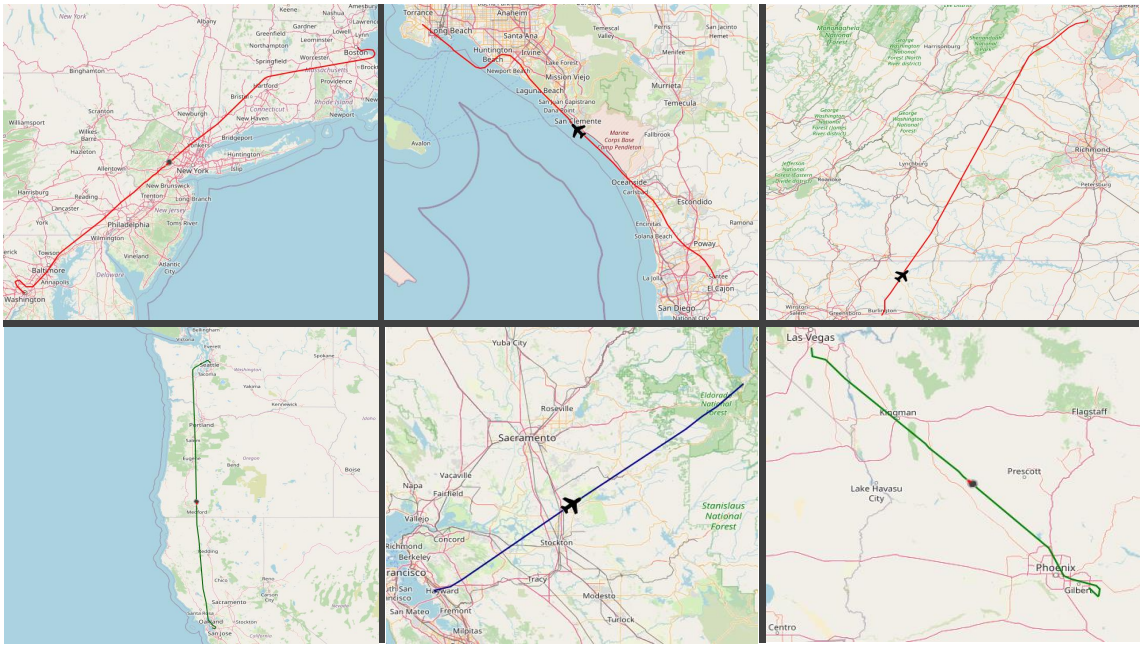


FIGURE 8.7 – Huit vols complets aux États-Unis.

Comme évoqué dans la section précédente, l'entraînement des modèles de détection supervisé se fait grâce à des enregistrements authentiques et d'autre altérés. Ici, le choix a été fait d'avoir un jeu de données équilibré entre enregistrements authentiques et altérés. Un modèle est entraîné pour chaque propriété dynamique altérée (altitude,

vitesse au sol, route, latitude, longitude et vitesse ascensionnelle), le besoin est donc d'un jeu de données par modèle. Ainsi, il y a 6 jeux de données composés des mêmes 100 enregistrements nominaux. Pour chaque jeu de données, 50 enregistrement sont tirés aléatoirement et le décalage de la propriété dynamique correspondante lui est appliqué. Au sein des enregistrements, tous les messages sont labellisés selon la technique présentée en chapitre 7.

Enfin, 20 enregistrements contenant chacun un unique vol complet sont utilisés pour valider la détection. La même stratégie est appliquée que pour l'entraînement : pour chaque modèle, 10 enregistrements sont tirés aléatoirement et un décalage leur est appliqué selon la propriété analysée par le modèle de détection à tester.

À l'heure actuelle, tous les scénarios sont détectés par les modèles proposés par Ralph Karam, et ces résultats feront bientôt l'objet d'une publication.

8.2/ TECHNIQUE D'APPRENTISSAGE NON-SUPERVISÉ DANS LE DOMAINE AÉRIEN

Comme évoqué dans la section précédente, ce deuxième cas d'utilisation partage le même contexte que le cas d'utilisation basé sur une technique d'apprentissage supervisé. Ainsi, les travaux autour de l'apprentissage non-supervisé dans le projet GeLeaD sont menés par Antoine Chevrot [6], doctorant à l'institut FEMTO-ST avec lequel nous avons collaboré pour ces expérimentations, notre rôle étant la mise à disposition de la chaîne outillée FDI-T.

De la même manière que la section précédente, la première sous-section pose le contexte des travaux d'Antoine Chevrot, la deuxième sous-section détaille la mise en œuvre de FDI-T pour l'apprentissage supervisé. Finalement, la troisième et dernière sous-section présente les données de validation qui sont générées par notre approche.

8.2.1/ CONTEXTE DES TRAVAUX DE RECHERCHE

La deuxième approche étudiée dans le projet GeLeaD pour la détection d'anomalies dans les données ADS-B est basée sur l'utilisation d'un modèle d'auto-encodeur variationnel (aussi appelé VAE pour *Variational Autoencoder*). Cette approche se base sur un travail effectué initialement par Habler et al. [39], déjà présenté en section 5.3.1 p. 87, dans lequel les auteurs parviennent à détecter plusieurs types d'anomalie grâce à l'utilisation d'un auto-encodeur LSTM. En partant du constat que les anomalies utilisées par Habler

6. https://www.femto-st.fr/fr/personnel-femto/achevro5 [Dernière visite : décembre 2020]

et al. manquent de finesse, le but est de pallier les limitations de ces travaux en proposant une nouvelle approche qui parvient à détecter des anomalies plus subtiles.

Pour ce faire, une architecture [7] a été mise en place par Antoine Chevrot afin de de disposer d'une chaîne de données qui, à partir de la base de données ADS-B mise à disposition par OpenSky [8], permet d'obtenir des jeux de données d'entraînement ou des jeux de données de validation générés via la chaîne outillée FDI-T. Cette architecture est représentée dans la figure 8.8. Dans un premier temps, des vols sont récupérés de manière isolée depuis OpenSky en utilisant la librairie Traffic [78]. Ensuite, une étape de nettoyage des données est nécessaire en utilisant deux algorithmes de *Local Outlier Factor* [15], avec pour effet de supprimer les messages contenant des positions erronées dues à des erreurs d'encodage. Enfin, chaque vol est découpé en trois phases de vol (ascendante, croisière et descendante), cette étape est nécessaire pour la reproduction du modèle proposé dans les travaux d'Habler et al. Après identification des phases de vol, les données sont fournies en entrée à un auto-encodeur variationnel pour l'apprentissage non-supervisé.

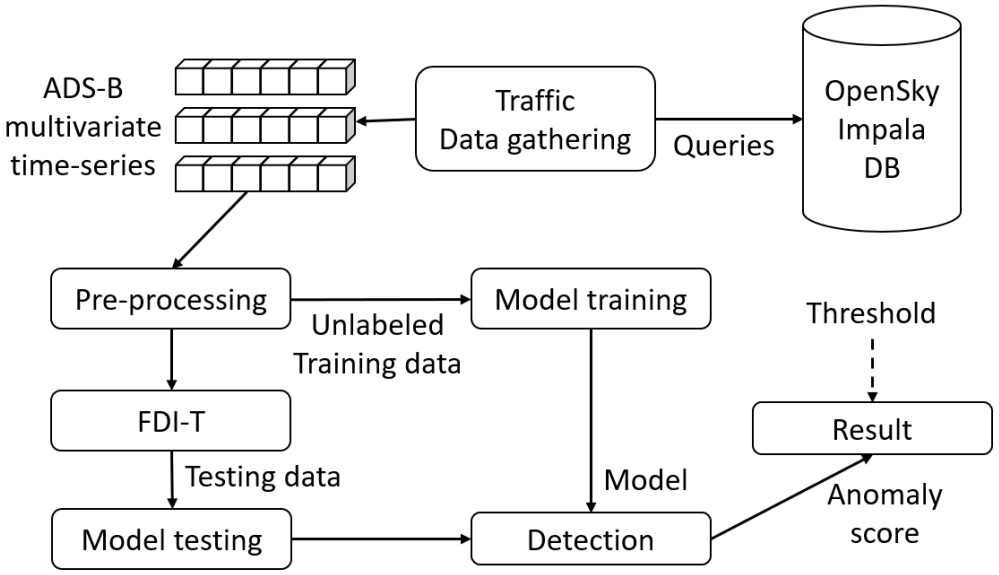


FIGURE 8.8 – Architecture d'apprentissage non-supervisé pour la détection d'anomalies.

Une fois l'apprentissage terminé, l'approche FDI-T est utilisée afin d'évaluer la capacité du modèle entrainé à détecter des altérations dans les données. On identifie une différence majeure avec le cas de l'apprentissage supervisé. En effet, si le réalisme des données reste important dans ce cas d'utilisation, le reliant de fait à **RQ-2**, le pouvoir d'expression des langages dédiés est également central ici. Ainsi, ce cas d'utilisation permet également d'apporter des éléments de réponse à la question **RQ-1 : Dans quelle**

7. https://github.com/Wirden/scifly [Dernière visite : janvier 2020]
8. https://opensky-network.org/, [Dernière visite : décembre 2020]

mesure un ensemble de langages dédiés assure-t-il la couverture de la taxonomie des FDIA ?. Effectivement, si les modèles d'apprentissage supervisé sont entraînés à reconnaître un type d'anomalie grâce à des annotations, les modèles d'apprentissage non-supervisé sont eux entraînés à calculer une représentation de l'espace aérien à un instant donné. Lorsque les données reçues ne permettent pas de calculer une représentation "connue" de l'espace aérien, on peut en déduire que ces dernières sont anormales. C'est sur ce principe que fonctionnent les modèles de détection non-supervisés. Ainsi, on ne connaît pas en amont les anomalies qui pourront être détectées via un modèle supervisé, c'est pourquoi il est important de pouvoir en reproduire un maximum afin de définir les limites du modèle de détection.

8.2.2/ MISE EN ŒUVRE DE FDI-T POUR LA VALIDATION DES MODÈLES D'APPRENTISSAGE NON-SUPERVISÉ

Comme cela est évoqué dans la section précédente, l'objectif ici est d'explorer les limites du modèle de détection non-supervisé par la diversité des anomalies. Toutefois, il convient tout de même de se limiter à certaines classes d'anomalies. Dans un premier temps, et dans le but d'incrémenter les travaux d'Habler, les premières anomalies expérimentées reproduisaient celles utilisées dans les travaux du chercheur. Ces dernières ayant déjà été présentées en section 5.3 p. 86, le focus est mis sur des attaques plus subtiles qui reposent sur la modification de trajectoire. De manière générale, on cherche à détecter les anomalies qui touchent à la trajectoire des avions. À noter que des travaux récents permettront par la suite de s'attarder sur d'autres types d'anomalies comme les fausses alarmes [79].

Le modèle est entraîné à partir des données réelles, issues de la base OpenSky, de plusieurs occurrences d'un même vol effectué à des dates diverses, avec tout ce que cela implique comme différences, notamment au niveau des itinéraires, cela étant dû à plusieurs facteurs (météo, trafic, etc.). Ce sont les vols qui effectuent la liaison entre Londres et Varsovie qui ont été choisis afin de mettre en place l'approche. Ces vols ont l'avantage d'être sur un axe est-ouest, ce qui facilite la mise en place de modifications de trajectoire. En effet, pour faire dévier le vol de sa trajectoire initiale, il est possible de ne modifier que les valeurs de latitude.

```
1  let $waypoints = {»0.2,»0.4,»0.6,»0.8,»1}
2  let $dates = {0.25 * ALT_DURATION, 0.5 * ALT_DURATION, 0.75 * ALT_DURATION}
3  alter all_planes from 0.3 * REC_DURATION until 0.6 * REC_DURATION
4  with_waypoints [
5      ($waypoints,»0) with_altitude »0 at $dates seconds ]
```

FIGURE 8.9 – Scénario abstrait du décalage positif de latitude pour le vol Londres-Varsovie.

On remarque dans la figure 8.9 des déclarations de listes de valeurs, elles sont utilisées pour générer plusieurs scénarios concrets à partir du même scénario abstrait. Le scénario combine la liste ***$waypoints*** de cinq éléments avec la liste ***$dates*** de deux éléments. La concrétisation de ce scénario abstrait mène donc à la génération de $3 * 5 = 15$ scénarios concrets dont voici un exemple :

```
1  alter all_planes from 0.3 * REC_DURATION until 0.6 * REC_DURATION
2  with_waypoints [
3        (»0.2,»0) with_altitude »0 at 0.5 * ALT_DURATION seconds ]
```

FIGURE 8.10 – Exemple d'un scénario concret du décalage positif de latitude pour le vol Londres-Varsovie.

Le scénario concret de la figure 8.10 se lit en langage naturel de la manière suivante : "*Modifier la trajectoire de tous les avions entre* 30% *et* 60% *de la durée de l'enregistrement de sorte à ce qu'ils passent par une latitude de +0.2° par rapport celle prévue à la moitié* (50%) *de cette modification*". Ce sont les termes 0.3 * **REC_DURATION** et 0.6 * **REC_DURATION** qui définissent respectivement la date de début et de fin de l'altération (30% et 60%). Le terme 0.5 * **ALT_DURATION** permet quant à lui de spécifier la date du point de passage utilisé dans la fausse trajectoire. Dans cet exemple, la date de ce point de passage correspond à 50% de la fenêtre de temps de la modification, soit à 45% de la durée de l'enregistrement (au milieu de la fenêtre de temps allant de 30% à 60% de l'enregistrement).

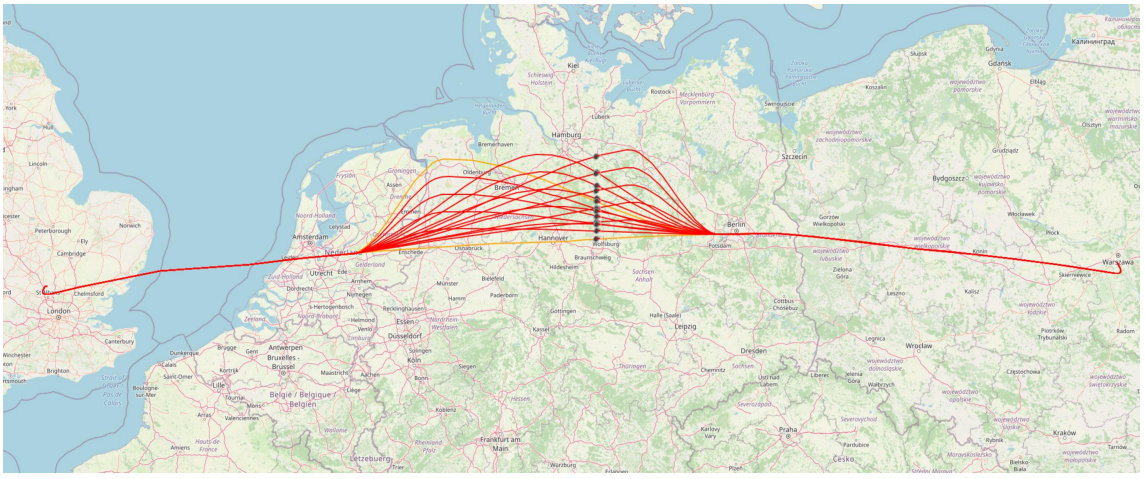


FIGURE 8.11 – Seize exemples de modifications de trajectoires sur le vol Londres-Varsovie.

La figure 8.11 montre le résultat de l'interprétation du scénario abstrait de la figure 8.9 sur le vol Londres-Varsovie. On remarque la présence de seize vols (le vol de base et les quinze vols issus des scénarios concrets), sur lesquels on constate des différentes amplitudes de modification de trajectoire ainsi que des différentes dates du point de passage.

8.2.3/ LES DONNÉES DE VALIDATION

L'apprentissage non-supervisé du modèle de détection se fait à partir d'un jeu de données construit différemment que celui de la section précédente. En effet, tandis que pour l'apprentissage supervisé 100 vols sont tirés aléatoirement dans la base de données OpenSky, dans ce cas d'utilisation, l'apprentissage se fait sur plusieurs occurrences d'un même vol. Le vol en question est la liaison Londres-Varsovie (et non Varsovie-Londres). Pour la construction du jeu de données nominal, tous les vols allant de Londres à Varsovie entre le 1^{er} et le 15 septembre 2020 ont été récupérés dans la base de données OpenSky. Le jeu de données compte 37 enregistrements, chaque enregistrement contenant une occurrence du vol Londres-Varsovie, 8 vols superposés sont visibles en figure 8.12.

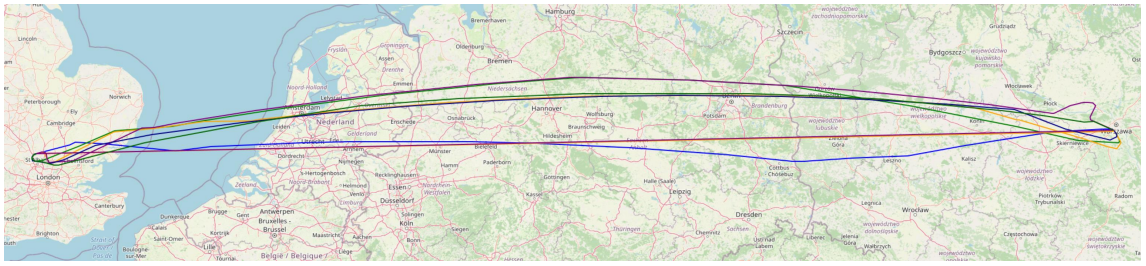


FIGURE 8.12 – Architecture de détection non-supervisée dans les données ADS-B.

On remarque sur la figure que, d'un vol à l'autre, les trajets ne sont pas tout à fait les mêmes à cause des facteurs déjà évoqués dans la section précédente. Un vol dure en moyenne 2 heures et contient 3200 messages. L'altitude et la vitesse au sol moyennes sur cette liaison sont respectivement de 31000 pieds et de 445 nœuds. Le volume total du jeu de données est de 12 Mo., à noter que seulement un message sur deux a été gardé afin de réduire le volume total et ainsi d'accélérer l'entraînement du modèle.

Pour valider les capacités de détection du modèle, 15 scénarios concrets sont générés à partir du scénario abstrait présenté en figure 8.9. Pour chaque enregistrement du jeu de données nominal, on tire aléatoirement de 4 à 6 scénarios concrets parmi les 15 générés, et on les applique à l'enregistrement courant. Ce sont donc en moyenne 5 scénarios concrets qui sont appliqués à 30% des messages de 37 vols, sachant qu'un enregistrement contient en moyenne 3200 messages, ce sont $5 * 37 * 0.3 * 3200 = 177600$ messages qui sont modifiés par une altération de type "modification de trajectoire". Avec les enregistrements altérés, les capacités de détection du modèle sont testées. Pour cela un seuil d'anomalie est fixé et il faut regarder manuellement, pour chaque enregistrement altéré, si ce seuil d'anomalie est dépassé.

8.3/ TECHNIQUE D'APPRENTISSAGE SUPERVISÉ DANS LE DOMAINE MARITIME

Comme précisé dans le chapitre 7, une version de FDI-T a été développée pour le domaine maritime. Cela a permis d'initier une expérimentation de l'approche avec un cas réel d'utilisation dans un second domaine. Cette expérimentation se situe dans le cadre de travaux de recherche menés au sein du *Simula Research Laboratory*[9] en partenariat avec l'Institut FEMTO-ST pour la thèse de Pierre Bernabé déja citée.

Le but de cette section est de montrer l'apport de FDI-T pour la construction d'un modèle de *Machine Learning* développé dans ce cadre pour la détection d'anomalies sur des données AIS. La première sous-section présente le contexte des travaux menés par Simula Research Laboratory, tandis que la deuxième sous-section porte sur l'utilisation de FDI-T au sein de ces travaux. Enfin, la dernière sous-section présente les données générées au travers de différentes métriques.

8.3.1/ CONTEXTE D'UTILISATION DES TRAVAUX

En Norvège, des travaux sont menés au *Simula Research Laboratory* sur l'utilisation de modèles d'IA pour représenter le trafic maritime [13] et la détection d'anomalies (qui constitue le sujet de la thèse de Pierre Bernabé). L'une de ces représentations de *Machine Learning* est basée sur l'apprentissage auto-supervisé [27]. Elle vise à servir de socle pour différentes tâches, majoritairement axées sur la détection de situations spécifiques comme l'arrêt intentionnel du transpondeur AIS, les rendez-vous illégaux ou encore la falsification de signaux AIS. Ces tâches peuvent également s'apparenter à un problème de classification, comme l'identification des différents types de bateau. Toutes ces tâches s'appuient elles-mêmes sur l'utilisation de modèles de *Machine Learning* préalablement entraînés.

La figure 8.13 schématise la relation qui existe entre la représentation du trafic maritime (à gauche de l'image) et les différentes tâches (à droite de l'image). La représentation est en fait un modèle entrainé à partir d'un grand nombre de messages issus des transpondeurs AIS. Le modèle est utilisé pour encoder des données fournies par les différentes tâches en fonction du trafic maritime. Sur la figure on distingue quatre tâches :

- ***AIS Shutdown Detection*** : Elle vise à détecter une coupure volontaire de l'AIS.
- ***Rendez-Vous Prediction*** : Elle vise à prédire une situation de rencontre (moins de 100 mètres) deux navires au moins une heure à l'avance.
- ***Vessel Type Classification*** : Elle vise à déduire le type d'un bateau à partir des informations obtenues avec l'AIS.

9. https://www.simula.no/ [Dernière visite : décembre 2020]

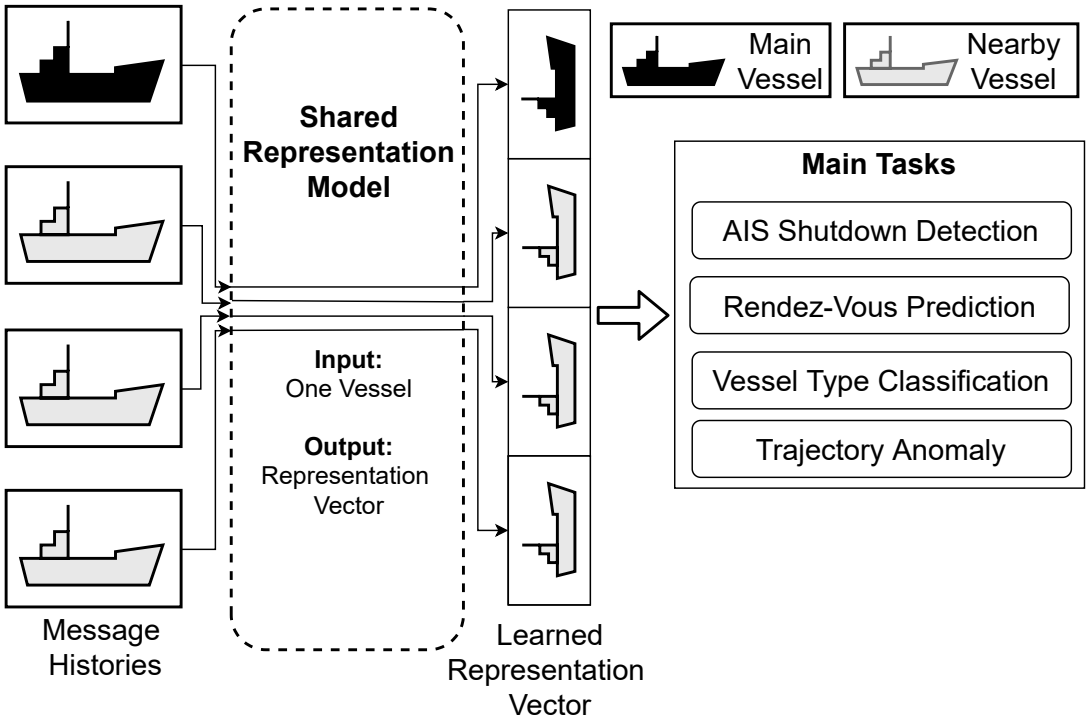


FIGURE 8.13 – Représentation du trafic maritime et son utilisation par différentes tâches.

— ***Trajectory Anomaly*** : Elle vise à détecter si une trajectoire est authentique ou non.

Le point commun entre les trois premières tâches est qu'il est possible de les implémenter grâce à des techniques d'apprentissage supervisé (cette notion est expliquée dans la section 1.2.2 p. 13). En effet, dans le cas des deux premières tâches (*AIS Shutdown Detection* et *Rendez-Vous Prediction*) les situations qu'il s'agit de détecter sont potentiellement présentes dans des données issues du trafic réel (à partir d'enregistrements AIS). Pour la tâche *Vessel Type Classification* il s'agit d'utiliser le type du bateau, qui est une information connue, en tant qu'annotation.

En revanche, pour la tâche *Trajectory Anomaly*, il n'existe pas, dans les données réelles enregistrées (et non modifiées), de cas où il est clairement identifié que la trajectoire transmise par un transpondeur diffère de la trajectoire réelle du bateau. C'est précisément pour l'entraînement supervisé du modèle chargé de la réalisation de cette tâche que les travaux de cette thèse peuvent servir d'appui.

Les cas d'utilisation mis en avant dans les deux premières sections de ce chapitre ont permis de répondre aux aspects relatifs aux questions **RQ-1** (pouvoir d'expression des DSL) et **RQ-2** (automatisation de la génération et réalisme de la données). Dans le cas d'utilisation présenté dans ce chapitre, la question **RQ-2** est couverte au même titre que le cas de détection d'anomalies par apprentissage supervisé introduit en section 8.1.1 p. 155. Néanmoins, le principal objectif ici est de démontrer le caractère générique des

travaux de thèse, avec la mise en place de l'approche au sein du domaine maritime. C'est donc la question de recherche **RQ-3 : Dans quelle mesure l'approche peut elle être générique à plusieurs domaines ?** que cette expérimentation nous permet de discuter. La réponse à cette question se fait par l'utilisation de la chaîne outillée FDI-T dans sa version maritime, utilisée pour générer des données labellisées qui présentent des modifications de trajectoires.

8.3.2/ L'UTILISATION DE L'APPROCHE FDI-T

Afin d'entraîner le modèle chargé de la tâche *Trajectory Anomaly*, il est nécessaire de disposer d'un important volume de données partagées entre trajectoires authentiques et trajectoires altérées. De plus, la mise en place d'un apprentissage supervisé nécessite l'annotation de ces données par une indication qui précise leur nature (authentique ou altérée). Enfin pour que le modèle soit efficace dans la détection, il doit être entraîné avec des trajectoires altérées qui restent suffisamment proches de ce qui est permis par la physique des bateaux. Ainsi l'entraînement s'articule autour de trois classes de données :

— Pas de modification : Les données sont telles qu'elles ont été émises par les bateaux.

— Modification avec vitesse : Les données présentent des modifications de trajectoires et le paramètre de réalisme présenté en section 7 p. 141 est activé. Ainsi, la vitesse des bateaux est recalculée de manière à correspondre à la trajectoire de ces derniers. Au risque d'avoir des vitesses dépassant la vitesse maximale physique du navire.

— Modification sans vitesse : Les données présentent des modifications de trajectoires mais le paramètre de réalisme est désactivé. Ainsi, la vitesse initiale des bateaux est conservée, mais celle-ci peut rentrer en contradiction avec la nouvelle trajectoire (vitesse rapportée face à la distance parcourue).

```
1 let $lat_plus = {»0.5,»1,»1.5,»2}
2 alter all_vessels from 0.2 * RECORDING_TIME until 0.4 * RECORDING_TIME
3 with_waypoints [
4       ($lat_plus,»0) at 0.2 * ALTER_DURATION seconds ]
```

FIGURE 8.14 – Exemple de scénario abstrait de modification de trajectoire avec décalage de latitude positif.

Les figures 8.14 et 8.15 représentent respectivement un décalage de latitude positif et négatif. De la même manière que le scénario présenté dans la section 8.2.2, cette altération a pour effet de modifier la trajectoire des bateaux cibles en les faisant passer par une latitude de plus ou moins 0.5 à 2 degrés à 20% de la durée de l'altération, celle-ci

```
1  let $lat_minus = {«0.5,«1,«1.5,«2}
2  alter all_vessels from 0.2 * RECORDING_TIME until 0.4 * RECORDING_TIME
3  with_waypoints [
4        ($lat_minus,»0) at 0.2 * ALTER_DURATION seconds ]
```

FIGURE 8.15 – Exemple de scénario abstrait de modification de trajectoire avec décalage de latitude positif.

étant effectuée entre 20% (0.2 * RECORDING_TIME) et 40% (0.4 * RECORDING_TIME) de la durée de l'enregistrement.

8.3.3/ LES DONNÉES D'ENTRAÎNEMENT GÉNÉRÉES

Pour l'entraînement supervisé du modèle de détection, des scénarios similaires à l'exemple de la figure 8.14 sont appliqués à un jeu de données extrait de communications S-AIS (AIS par satellite) fournies par Statsat[10]. Statsat est une entreprise publique norvégienne dont le but est de développer et d'exploiter des infrastructures spatiales. L'entreprise a la responsabilité opérationnelle de cinq satellites équipés de récepteurs AIS, assurant donc la couverture mondiale des communications. Statsat a fourni à Simula un mois de données AIS, pour un volume de 17 Go.

Les données fournies par Statsat étant initialement au format NMEA, elles ont été converties dans un format CSV grâce à décodeur[11] développé par la *Danish Maritime Authority*. En vu de la construction d'un jeu de données d'entraînement pour la détection d'anomalie, Pierre Bernabé a organisé ces données dans une base de données, permettant ainsi de récupérer des enregistrements selon des conditions spécifiques.

Le jeu de données initial pour l'entraînement supervisé se compose de 10000 enregistrements. Chaque enregistrement contient 1000 messages émis par un seul bateau. Ces enregistrements sont choisis arbitrairement tant qu'ils respectent la condition que la trajectoire qu'ils contiennent présente au moins 1000 messages consécutifs. Comme précisé en section 8.3.2 p. 166, le modèle est entraîné sur trois classes de données : pas de modification, modification avec vitesse et modification sans vitesse. Le jeu de données se compose de 30000 enregistrements de 1000 messages chacun : 10000 enregistrements ne sont pas modifiés, 10000 présentent une trajectoire (latitude et longitude) modifiée, et 10000 présentent une trajectoire ainsi qu'une vitesse (en fonction de la trajectoire) modifiées. Seules les deux dernières classes présentent des modifications. Ainsi, pour chacune de ces deux classes, tous les trajets sont modifiés suivant une modification de trajectoire, entre 10% et 80% des messages de chaque enregistrement sont concernés, pour une moyenne de 45% de messages modifiés par enregistrement. En tout, ce sont environ 9000000 de messages qui sont modifiés. Tous

10. http://statsat.no/ [Dernière visite : décembre 2020]
11. https://github.com/dma-ais/AisLib [Dernière visite : décembre 2020]

les messages sont labéllisés selon la technique décrite dans la section 7 p. 141.

8.4/ SYNTHÈSE

Ces cas d'utilisation ont permis de valider plusieurs contributions de nos travaux. Tout d'abord, concernant la couverture de la taxonomie des attaques, la diversité des scénarios requise pour l'entraînement et la validation des modèles de détection ne permet pas de répondre totalement à la question de recherche **RQ-2**. Cela montre néanmoins que le pouvoir d'expression des DSL permet de couvrir la totalité des besoins identifiés pour ces cas d'utilisation.

Pour ce qui est de la question **RQ-1**, concernant la massification et le réalisme des données, ces expérimentations ont montré qu'à partir de scénarios ne demandant pas d'effort de conception particulier, il était possible de générer des versions des enregistrements altérés de manière automatisée. Pour ce qui est du réalisme des données, l'implémentation des modèles de détection ne permet pas à l'heure actuelle de donner un verdict définitif. Il a néanmoins déjà été observé en section 5.3.1 p. 87 que le réalisme des données avait ses limites, causées en partie par l'utilisation de l'interpolation Akima. Toutefois, la confrontation à des modèles de détection est, à ce sens, une grande source d'information. En effet, l'observation de la réaction des modèles face aux données altérées offre une synergie qui permet la mise en place d'un processus d'amélioration itératif de la génération de données.

L'expérimentation sur un cas d'utilisation au sein du domaine maritime a permis de montrer la généricité des travaux de thèse, et ainsi de répondre à la question de recherche **RQ-3 : Dans quelle mesure l'approche peut elle être générique à plusieurs domaines ?** De plus, cette expérimentation vient compléter les réponses fournies par la section 6.5 et complète les expérimentations dans le domaine aérien en ce qui concerne les capacités de massification.

CONCLUSION ET PERSPECTIVES

Ce dernier chapitre dresse un bilan des travaux de thèse que nous avons réalisé durant ces 3 dernières années. Il s'agit, dans un premier temps, de discuter les questions de recherche posées dans le chapitre 1 vis à vis de la problématique et du positionnement de la thèse. Ensuite, nous proposons des perspectives au niveau de la recherche mais aussi au niveau industriel, se dégageant de nos travaux.

BILAN DE LA THÈSE

L'objectif de ces travaux de thèse est énoncé à travers la question suivante **RO : Dans quelle mesure une approche générique à plusieurs domaines, peut-elle permettre grâce à l'utilisation d'un ensemble de langages dédiés la génération automatique de données altérées selon des scénarios conçus par les experts du domaine ?**

La réalisation de cet objectif s'appuie sur trois questions de recherche sous-jacentes :

- **RQ-1 : Dans quelle mesure un ensemble de langages dédiés assure-t-il la couverture de la taxonomie des FDIA ?**
- **RQ-2 : Dans quelle mesure est-il possible de garantir le réalisme de données simulant des FDIA, tout en les générant de manière automatique et massive ?**
- **RQ-3 : Dans quelle mesure l'approche peut elle être générique à plusieurs domaines ?**

La réponse à ces questions de recherches a nécessité une étude de l'état de l'art autour de trois concepts : les attaques par injection de fausses données (FDIA), les approches de génération de données de test synthétique, et l'utilisation de langages dédiés (DSL) pour la spécification de scénarios de test. Les analyses bibliographiques menées sur ces trois points ont permis d'identifier des techniques sur lesquelles s'appuyer pour mener à bien nos travaux et de pointer des manques dans l'état de l'art :

- L'étude de l'état de l'art des FDIA retrace l'origine du terme, pour ainsi souligner l'absence d'une définition générique pouvant être utilisée de la même manière à travers différents domaines. En outre, nous avons proposé une définition générique du terme, nous permettant ainsi de produire une taxonomie commune aux domaines aérien et maritime.
- Plusieurs approches permettant la génération de données de test synthétiques ont été identifiées dans la littérature. Toutefois, aucune ne permet de répondre à notre

objectif de recherche car le réalisme qu'elles proposent nuit à leur généricité et à la facilité d'utilisation des outils qui en découlent. Néanmoins, certaines techniques identifiées ont servi de socle pour nos travaux, en ce qui concerne la génération massive de données.

— Enfin, l'analyse de l'état de l'art sur l'utilisation de DSL de scénarisation des tests a mis en évidence la popularité de ce type d'approche pour permettre la génération de données spécifiques. De plus, les recherches bibliographiques ont fait ressortir un ensemble de grands principes pour le développement de DSL sur lequel nous avons pu nous appuyer.

Les réponses aux questions de recherche se matérialisent finalement par les trois contributions techniques de la thèse : la conception de la démarche générale pour la génération de données, la mise en place du moteur de génération de données altérées, et le développement des trois langages dédiés à la scénarisation des test.

La démarche générale pour la génération de données traite directement la question de recherche **RQ-3**. En effet, cette contribution se compose d'une architecture multi-domaine, et du processus de spécialisation à suivre pour mettre en place l'approche sur un domaine donné.

La génération de données altérées traite à la fois du réalisme et de l'automatisation évoqués dans la question de recherche **RQ-2**. En ce qui concerne le réalisme des données générées, une preuve de concept a été mise en place avec l'utilisation de la méthode d'interpolation Akima. Cela a permis de montrer que notre démarche permettait de faire du réalisme tout en restant générique. Ce réalisme n'étant réellement atteint qu'au prix d'un investissement de temps et l'utilisation d'outils appropriés. La dimension automatique de la génération de données est en partie traitée par le moteur de génération. S'il produit bien automatiquement des données altérées à partir de directives au format XML, ces directives sont peu conviviales à manipuler pour les utilisateurs.

L'ensemble des trois DSL résout ce problème en proposant de faire abstraction des directives via un premier DSL de scénarisation qui possède une syntaxe proche du langage naturel et donc facilement assimilable. La question de recherche **RQ-1** interroge le pouvoir d'expression de cet ensemble de DSL, notamment sa couverture de la taxonomie des attaques FDIA identifiées dans l'état de l'art. L'analyse du domaine effectué lors du développement de ces DSL a permis d'intégrer toutes les attaques identifiées dans la taxonomie au DSL de scénarisation. De plus, deux autres langages ont été développés afin de pouvoir spécifier un large spectre de situations : un langage de filtre permettant la sélection des cibles selon leur propriétés, et un langage de déclencheurs permettant de déclencher des schémas d'altération selon les propriétés des cibles. En plus du pouvoir d'expression qu'ils offrent, les DSL permettent de massifier les cas de test grâce à l'utilisation de plages et de listes de valeurs, couplés à des

stratégies combinatoires, complétant ainsi la réponse à la question de recherche **RQ-2**.

Les trois contributions de la thèse ont été directement intégrées à une chaîne outillée existante : FDI-T. Une version de ce logiciel existe pour chacun des deux domaines aérien et maritime. Afin de démontrer l'aboutissement de l'objectif de recherche, trois expérimentations ont été menées. Elles consistent à montrer la capacité de la chaîne outillée FDI-T à faciliter la mise en place de travaux de recherche sur la détection d'anomalies, basée sur le *Machine Learning*, dans les protocoles ADS-B et AIS, notamment grâce à la génération de données labellisées pour l'entraînement des modèles de détection supervisés, et la génération de données de test pour la validation des modèles de détection non-supervisés.

PERSPECTIVES À L'ISSUE DE LA THÈSE

On identifie plusieurs perspectives de poursuite des travaux de thèse, certaines sont orientées sur la recherche scientifique, tandis que d'autres sont plutôt axées vers un contexte industriel.

D'un point de vue recherche, la nécessité d'enrichir les données générées via notre approche, que cela soit pour le domaine aérien ou pour le domaine maritime, avec d'autres sources de données a été évoquée lors de nos échanges avec d'autres chercheurs. En effet, dans le cas des modèles de détection utilisant des techniques d'intelligence artificielle cela peut être limitant de se fonder sur un seul type de données. Il serait bénéfique pour les utilisateurs que FDI-T soit en mesure de générer d'autres types de données, altérées ou non, à mettre en relation avec les données issues des protocoles de base, comme par exemple, les données des plans de vol dans le domaine aérien.

En parallèle, et dans l'optique de mettre à l'épreuve notre processus de spécialisation ainsi que notre architecture multi-domaine, il est nécessaire d'étendre nos travaux à d'autres domaines. Au vu des domaines dans lesquels sont proposés des travaux connexes aux nôtres dans l'état de l'art, le domaine de la voiture autonome semble être un bon candidat pour tester la généricité de notre approche. D'une manière générale, les systèmes de type SCADA apparaissent compatibles avec notre approche de génération de données altérées.

Dans le contexte industriel, la suite immédiate des travaux de thèse concerne la mise en place de techniques de sélection et de priorisation des cas de test. Le but est de combiner la génération massive de cas de test avec une approche de priorisation afin d'exécuter d'abord les tests les plus pertinents pour le système cible. Ces travaux vont se dérouler dans le cadre d'un projet en partenariat avec les entreprises Kereval et Smartesting : le

projet CRIA [12].

Enfin, l'industrialisation de la chaîne outillée FDI-T est un travail permanent qui vise à en augmenter le niveau de maturité technologique selon l'échelle TRL (*Technology readiness level*). L'objectif est de pouvoir proposer FDI-T aux industriels des domaines aérien et maritime pour le test de leurs systèmes de surveillance.

12. https://www.kereval.com/kereval-chef-de-file-dun-projet-visant-a-ameliorer-les-tests-logiciels-grace-a-lintelligence-artificielle/ [Dernière visite : janvier 2021]

BIBLIOGRAPHIE

[1] EUROCAE Working Group 51. Safety, performance and interoperability requirements document for ads-b/nra application. Technical report, The European Organisation for Civil Aviation Equipment, 2005.

[2] Federal Aviation Administration. Automatic dependent surveillance—broadcast (ads–b) out performance requirements to support air traffic control (atc) service. *Final Rule*, Final Rule, CFR Part 91, Federal Register 75 (103), 2010.

[3] Mohiuddin Ahmed and Al-Sakib Khan Pathan. False data injection attack (fdia) : an overview and new metrics for fair evaluation of its countermeasure. *Complex Adaptive Systems Modeling*, 8 :1–14, 2020.

[4] Sefi Akerman, Edan Habler, and Asaf Shabtai. Vizads-b : Analyzing sequences of ads-b images using explainable convolutional lstm encoder-decoder to detect cyber attacks. *arXiv preprint arXiv :1906.07921*, 2019.

[5] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. 17 :589–602, 1970.

[6] Mathias Anneken, Francesca de Rosa, Alexander Kröker, Anne-Laure Jousselme, Sebastian Robert, and Jürgen Beyerer. Detecting illegal diving and other suspicious activities in the north sea : Tale of a successful trial. In *2019 20th International Radar Symposium (IRS)*, pages 1–10. IEEE, 2019.

[7] Regina Asariotis, Hassiba Benamara, Hannes Finkenbrink, Jan Hoffmann, Jennifer Lavelle, Maria Misovicova, Vincent Valentine, and Frida Youssef. Review of maritime transport, 2011. Technical report, 2011.

[8] International Civil Aviation Organization Asia and Pacific Office (ICAO). Guidance material on issues to be considered in atc multi-sensor fusion processing including the integration of ads-b data. Technical report, APANPIRG/19, 2008.

[9] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In *Handbook on ontologies*, pages 3–28. Springer, 2004.

[10] Marco Balduzzi, Alessandro Pasta, and Kyle Wilhoit. A security evaluation of ais automated identification system. In *Proceedings of the 30th annual computer security applications conference*, pages 436–445, 2014.

[11] Alexandre B Barreto, Michael Hieb, and Edgar Yano. Developing a complex simulation environment for evaluating cyber attacks. In *Interservice/Industry*

Training, Simulation, and Education Conference (I/ITSEC), volume 12248, pages 1–9, 2012.

[12] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Temporal Logics and Automata*, pages 27–38. Springer International Publishing, Cham, 2017.

[13] Pierre Bernabé, Helge Spieker, Bruno Legeard, and Arnaud Gotlieb. Encoding temporal and spatial vessel context using self-supervised learning model (student abstract). 2020.

[14] Tim Berners-Lee, James Hendler, Ora Lassila, and T Berners-Lee. The semantic web : Scientific american. *Scientific American (May 2001)*, 2002.

[15] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof : identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[16] Peter Brooker. Sesar and nextgen : investing in new paradigms. *The Journal of Navigation*, 61(2) :195–208, 2008.

[17] Nazly Rocio Santos Buitrago, Loek Tonnaer, Vlado Menkovski, and Dimitrios Mavroeidis. Anomaly detection for imbalanced datasets with deep generative models.

[18] Ines Ceh, Matej Crepinšek, Tomaž Kosar, and Marjan Mernik. Ontology driven development of domain-specific languages. *Computer Science and Information Systems*, 8(2) :317–342, 2011.

[19] Miguel A Cervera, Alberto Ginesi, and Knut Eckstein. Satellite-based vessel automatic identification system : A feasibility and performance analysis. *International Journal of Satellite Communications and Networking*, 29(2) :117–142, 2011.

[20] Yiu-Tong Chan and KC Ho. A simple and efficient estimator for hyperbolic location. *IEEE Transactions on signal processing*, 42(8) :1905–1915, 1994.

[21] Bharvi Chhaya, Shafagh Jafer, William B Coyne, Neal C Thigpen, and Umut Durak. Enhancing scenario-centric air traffic control training. In *2018 AIAA modeling and simulation technologies conference*, page 1399, 2018.

[22] James Coplien, Daniel Hoffman, and David Weiss. Commonality and variability in software engineering. *IEEE software*, 15(6) :37–45, 1998.

[23] Andrei Costin and Aurélien Francillon. Ghost in the air (traffic) : On insecurity of ads-b protocol and practical attacks on ads-b devices. *Black Hat USA*, pages 1–12, 2012.

[24] Aymeric Cretin, Alexandre Vernotte, Antoine Chevrot, Fabien Peureux, and Bruno Legeard. Test data generation for false data injection attack testing in air traffic

surveillance. In *4th International Workshop on Testing Extra-Functional Properties and Quality Characteristics of Software Systems (ITEQS 2020)*, Porto, Portugal, mar 2020.

[25] Gyorgy Dan and Henrik Sandberg. Stealth attacks and protection schemes for state estimators in power systems. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 214–219. IEEE, 2010.

[26] Axel Daneels and Wayne Salter. What is scada ? 1999.

[27] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[28] Paul R Drouilhet Jr, George H Knittel, and Vincent A Orlando. Automatic dependent surveillance air navigation system, October 29 1996. US Patent 5,570,095.

[29] Greg Dunstone. Ads-b technology the experience in australia. https://www.icao.int/SAM/Documents/2017-ADSB/10%20Australia.pdf, 2017.

[30] James F Epperson. On the runge example. *The American Mathematical Monthly*, 94(4) :329–341, 1987.

[31] Guido Ferraro, Björn Baschek, Geraldine de Montpellier, Ove Njoten, Marko Perkovic, and Michele Vespe. On the sar derived alert in the detection of oil spills according to the analysis of the egemp. *Marine pollution bulletin*, 60(1) :91–102, 2010.

[32] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016.

[33] Mélanie Fournier, R Casey Hilliard, Sara Rezaee, and Ronald Pelot. Past, present, and future of the satellite-based automatic identification system : areas of applications (2004–2016). *WMU Journal of Maritime Affairs*, 17(3) :311–345, 2018.

[34] William Frakes, Ruben Prieto, Christopher Fox, et al. Dare : Domain analysis and reuse environment. *Annals of software engineering*, 5(1) :125–141, 1998.

[35] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget : Continual prediction with lstm. 1999.

[36] G Mori Gonzalez, Ivan Petrunin, Rafal Zbikowski, K Voutsis, and R Verdeguer Moreno. Vulnerability analysis of gps receiver software. 2019.

[37] Athanassios Goudossis and Sokratis K Katsikas. Towards a secure automatic identification system (ais). *Journal of Marine Science and Technology*, 24(2) :410–423, 2019.

[38] Yanpeng Guan and Xiaohua Ge. Distributed attack detection and secure estimation of networked cyber-physical systems against false data injection attacks and jamming attacks. *IEEE Transactions on Signal and Information Processing over Networks*, 4(1) :48–59, 2017.

[39] Edan Habler and Asaf Shabtai. Using lstm encoder-decoder algorithm for detecting anomalous ads-b messages. *Computers & Security*, 78 :155–173, 2018.

[40] John Hall, Jordan Lee, Joseph Benin, and Henry Owen. Ieee 1609 influenced automatic identification system (ais). In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2015.

[41] Abbas Harati-Mokhtari, Alan Wall, Philip Brooks, and Jin Wang. Automatic identification system (ais) : data reliability and human error implications. *The Journal of Navigation*, 60(3) :373, 2007.

[42] James A Hess, E William, and A Novak. Feature-oriented domain analysis (foda) feasibility study kyo c. kang, sholom g. cohen. 1990.

[43] Mark Hills, Paul Klint, Tijs van der Storm, and Jurgen Vinju. A case of visitor versus interpreter pattern. In Judith Bishop and Antonio Vallecillo, editors, *Objects, Models, Components, Patterns*, pages 228–243, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[44] Vinay M Igure, Sean A Laughter, and Ronald D Williams. Security issues in scada networks. *computers & security*, 25(7) :498–506, 2006.

[45] IMO. Guidelines for vessel traffic services. resolution a.578. https: //www.directemar.cl/directemar/site/artic/20170302/asocfile/20170302114721/ 578_14.pdf, 1985.

[46] IMO. Revision of imo resolution a.857(20) guidelines for vessel traffic services. https://www.iala-aism.org/product/ seminar-on-the-revision-of-imo-resolution-a-85720-guidelines-for-vts/, 2019.

[47] Didi Istardi, Ardian Budi KA, Nur Sakinah Assad, and Hendra Saputra. Automatic identification system (ais) decode design for ship monitoring using labview software. *Journal of Ocean, Mechanical and Aerospace-science and engineering-*, 54(1) :9–16, 2018.

[48] Shafagh Jafer, Bharvi Chhaya, and Umut Durak. Owl ontology to ecore metamodel transformation for designing a domain specific language to develop aviation scenarios. In *Proceedings of the symposium on model-driven approaches for simulation engineering*, pages 1–11, 2017.

[49] Qiaomu Jiang, Huifang Chen, Lei Xie, and Kuang Wang. Learning-based cooperative false data injection attack and its mitigation techniques in consensus-based distributed estimation. *IEEE Access*, 8 :166852–166869, 2020.

[50] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. A meta language for threat modeling and attack simulations. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–8, 2018.

[51] Ralph Karam, Michel Salomon, and Raphaël Couturier. A comparative study of deep learning architectures for detection of anomalous ads-b messages. In *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*, volume 1, pages 241–246. IEEE, 2020.

[52] Fotios Katsilieris, Paolo Braca, and Stefano Coraluppi. Detection of malicious ais position spoofing by exploiting radar information. In *proceedings of the 16th international conference on information fusion*, pages 1196–1203. IEEE, 2013.

[53] Siddartha Khastgir, Gunwant Dhadyalla, Stewart Birrell, Sean Redmond, Ross Addinall, and Paul Jennings. Test scenario generation for driving simulators using constrained randomization technique. Technical report, SAE Technical Paper, 2017.

[54] X. Kong, X. Song, F. Xia, H. Guo, J. Wang, and A. Tolba. Lotad : long-term traffic anomaly detection based on crowdsourced bus trajectory data. *World Wide Web*, 21(3) :825–847, May 2018.

[55] Tomaž Kosar, Sudev Bohra, and Marjan Mernik. Domain-specific languages : A systematic mapping study. *Information and Software Technology*, 71 :77–91, 2016.

[56] Andrey A Kurekin, Benjamin R Loveday, Oliver Clements, Graham D Quartly, Peter I Miller, George Wiafe, and Kwame Adu Agyekum. Operational monitoring of illegal fishing in ghana through exploitation of satellite earth observation and ais data. *Remote Sensing*, 11(3) :293, 2019.

[57] Richard O Lane, David A Nevell, Steven D Hayward, and Thomas W Beaney. Maritime anomaly detection and threat assessment. In *2010 13th International Conference on Information Fusion*, pages 1–8. IEEE, 2010.

[58] Ora Lassila, Ralph R Swick, et al. Resource description framework (rdf) model and syntax specification. 1998.

[59] Philipp Last, Martin Kroker, and Lars Linsen. Generating real-time objects for a bridge ship-handling simulator based on automatic identification system data. *Simulation Modelling Practice and Theory*, 72 :69–87, 2017.

[60] Antoine Lemay and José M Fernandez. Providing scada network data sets for intrusion detection research. In *9th Workshop on Cyber Security Experimentation and Test ({CSET} 16)*, 2016.

[61] Beibei Li, Rongxing Lu, Wei Wang, and Kim-Kwang Raymond Choo. Distributed host-based collaborative detection for false data injection attacks in smart grid cyber-physical system. *Journal of Parallel and Distributed Computing*, 103 :32–41, 2017.

[62] Yi-Gang Li and Guang-Hong Yang. Optimal stealthy false data injection attacks in cyber-physical systems. *Information Sciences*, 481 :474–490, 2019.

[63] Liana Barachisio Lisboa, Vinicius Cardoso Garcia, Daniel Lucrédio, Eduardo Santana de Almeida, Silvio Romero de Lemos Meira, and Renata Pontin de Mattos Fortes. A systematic review of domain analysis tools. *Information and Software Technology*, 52(1) :1–13, 2010.

[64] Yao Liu, Peng Ning, and Michael K Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1) :13, 2011.

[65] Nicolas Longépé, Guillaume Hajduch, Romy Ardianto, Romain de Joux, Béatrice Nhunfat, Marza I Marzuki, Ronan Fablet, Indra Hermawan, Olivier Germain, Berny A Subki, et al. Completing fishing monitoring with spaceborne vessel detection system (vds) and automatic identification system (ais) to assess illegal fishing in indonesia. *Marine pollution bulletin*, 131 :33–39, 2018.

[66] Miao Ma. Resilience against false data injection attack in wireless sensor networks. In *Handbook of Research on Wireless Security*, pages 628–635. IGI Global, 2008.

[67] Daniel Maciel, Ana CR Paiva, and Alberto Rodrigues da Silva. From requirements to automated acceptance tests of interactive apps : An integrated model-based testing approach. In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 265–272. SCITEPRESS-Science and Technology Publications, Lda, 2019.

[68] Mohsen Riahi Manesh and Naima Kaabouch. Analysis of vulnerabilities, attacks, countermeasures and overall risk of the automatic dependent surveillance-broadcast (ADS-B) system. *International Journal of Critical Infrastructure Protection*, 19 :16 – 31, 2017.

[69] Ivan Martinovic and Martin Strohmeier. Security of ads- b : State of the art and beyond. *DCS*, 2013.

[70] Fabio Mazzarella, Michele Vespe, Dimitrios Damalas, and Giacomo Osio. Discovering vessel activities at sea using ais data : Mapping of fishing footprints. In *17th International conference on information fusion (FUSION)*, pages 1–7. IEEE, 2014.

[71] Donald McCallie, Jonathan Butts, and Robert Mills. Security analysis of the ADS-B implementation in the next generation air transportation system. *International Journal of Critical Infrastructure Protection*, 4(2) :78–87, 2011.

[72] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10) :2004, 2004.

[73] Jefferson Ryan Medel and Andreas Savakis. Anomaly detection in video using predictive convolutional long short-term memory networks. *arXiv preprint arXiv :1612.00390*, 2016.

[74] Till Menzel, Gerrit Bagschik, and Markus Maurer. Scenarios for development, test and validation of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1821–1827. IEEE, 2018.

[75] Marjan Mernik, Jan Heering, and Anthony M Sloane. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4) :316–344, 2005.

[76] Yilin Mo and Bruno Sinopoli. False data injection attacks in control systems.

[77] Anand Mundra, James J Cieplak, David A Domino, Baltazar O Olmos, and Hans P Stassen. Potential ads-b/cdti capabilities for near term deployment. In *First USA/Europe Air Traffic Management R&D Seminar*, 1997.

[78] Xavier Olive. Traffic, a toolbox for processing and analysing air traffic data. 2019.

[79] Xavier Olive, Axel Tanner, Martin Strohmeier, Matthias Schäfer, Metin Feridun, Allan Tart, Ivan Martinovic, and Vincent Lenders. Opensky report 2020 : Analysing in-flight emergencies using big data. In *2020 IEEE/AIAA 39th Digital Avionics Systems Conference (DASC)*, pages 1–9, October 2020.

[80] Suat Ozdemir. Energy-efficient false data detection in wireless sensor networks. In *Wireless Sensing and Processing*, volume 6248, page 62480E. International Society for Optics and Photonics, 2006.

[81] T. Maruthi Padmaja, Narendra Dhulipalla, Raju S. Bapi, and P. Radha Krishna. Unbalanced data classification using extreme outlier elimination and sampling techniques for fraud detection. pages 511–516. IEEE.

[82] Russell A Paielli. Automated generation of air traffic encounters for testing conflict-resolution software. *Journal of Aerospace Information Systems*, 10(5) :209–217, 2013.

[83] Scott Pakin. The design and implementation of a domain-specific language for network performance testing. *IEEE Transactions on Parallel and Distributed Systems*, 18(10) :1436–1449, 2007.

[84] Tom Arne Pedersen, Jon Arne Glomsrud, Else-Line Ruud, Aleksander Simonsen, Jarle Sandrib, and Bjørn-Olav Holtung Eriksen. Towards simulation-based verification of autonomous navigation systems. *Safety Science*, 129 :104799, 2020.

[85] Maria João Varanda Pereira, João Fonseca, and Pedro Rangel Henriques. Ontological approach for dsl development. *Comput. Lang. Syst. Struct.*, 45(C) :35–52, April 2016.

[86] Maria João Varanda Pereira, João Fonseca, and Pedro Rangel Henriques. Ontological approach for dsl development. *Computer Languages, Systems & Structures*, 45 :35–52, 2016.

[87] Leon Purton, Hussein Abbass, and Sameer Alam. Identification of ADS-B system vulnerabilities and threats. In *Australian Transport Research Forum, Canberra*, pages 1–16, 2010.

[88] Rodrigo Queiroz, Thorsten Berger, and Krzysztof Czarnecki. Geoscenario : An open dsl for autonomous driving scenario representation. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 287–294. IEEE, 2019.

[89] F. L. Quilumba, W. Lee, H. Huang, D. Y. Wang, and R. L. Szabados. Using smart meter data to improve the accuracy of intraday load forecasting considering customer behavior similarities. *IEEE Transactions on Smart Grid*, 6(2) :911–918, 2015.

[90] Cyril Ray, Romain Gallen, Clément Iphar, Aldo Napoli, and Alain Bouju. Deais project : detection of ais spoofing and resulting risks. In *OCEANS 2015-Genova*, pages 1–6. IEEE, 2015.

[91] Muhammad Qamar Raza and Abbas Khosravi. A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. *Renewable and Sustainable Energy Reviews*, 50 :1352–1372, 2015.

[92] Branko Ristic, Barbara La Scala, Mark Morelande, and Neil Gordon. Statistical analysis of motion patterns in ais data : Anomaly detection and motion prediction. In *2008 11th International Conference on Information Fusion*, pages 1–7. IEEE, 2008.

[93] H. R. Roth, L. Lu, J. Liu, J. Yao, A. Seff, K. Cherry, L. Kim, and R. M. Summers. Improving computer-aided detection using convolutional neural networks and random view aggregation. *IEEE Transactions on Medical Imaging*, 35(5) :1170–1181, May 2016.

[94] Liyang Rui and KC Ho. Elliptic localization : Performance study and optimum receiver placement. *IEEE Transactions on Signal Processing*, 62(18) :4673–4688, 2014.

[95] Andreas Savvides, Heemin Park, and Mani B Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 112–121. ACM, 2002.

[96] Matthias Schäfer, Vincent Lenders, and Ivan Martinovic. Experimental analysis of attacks on next generation air traffic communication. In *International Conference on Applied Cryptography and Network Security*, pages 253–271. Springer, 2013.

[97] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. Bringing up opensky : A large-scale ads-b sensor network for research. In *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, pages 83–94. IEEE, 2014.

[98] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, 2017.

[99] Peter Schneider and Alexander Giehl. Realistic data generation for anomaly detection in industrial settings using simulations. In *Computer Security*, pages 119–134. Springer, 2018.

[100] Hichem Sedjelmaci and Sidi Mohamed Senouci. Cyber security methods for aerial vehicle networks : taxonomy, challenges and solution. *The Journal of Supercomputing*, 74(10) :4928–4944, 2018.

[101] Leonov Sergey, Oliver Hubbard, Zhen Ding, Hamid Ghadaki, Jian Wang, and Tony Ponsford. Advanced mitigating techniques to remove the effects of wind turbines and wind farms on primary surveillance radars. In *2008 IEEE Radar Conference*, pages 1–6. IEEE, 2008.

[102] Merrill Ivan Skolnik. Radar handbook, 3rd edition. 2008.

[103] A Smith, R Cassell, T Breen, R Hulstrom, and C Evers. Methods to provide system-wide ADS-B back-up, validation and security. In *25th Digital Avionics Systems Conference*, pages 1–7. IEEE, 2006.

[104] Martin Strohmeier and Ivan Martinovic. Poster : Detecting false- data injection attacks on air traffic control protocols. 2014.

[105] Martin Strohmeier, Matthias Schäfer, Rui Pinheiro, Vincent Lenders, and Ivan Martinovic. On perception and reality in wireless air traffic communications security. *IEEE Transactions on Intelligent Transportation Systems*, 18(6) :1338–1357, 2017.

[106] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering : principles and methods. *Data & knowledge engineering*, 25(1-2) :161–197, 1998.

[107] Robert Tairas, Marjan Mernik, and Jeff Gray. Using ontologies in the domain analysis of domain-specific languages. In *International Conference on Model Driven Engineering Languages and Systems*, pages 332–342. Springer, 2008.

[108] Richard N Taylor, Will Tracz, and Lou Coglianese. Software development using domain-specific software architectures : Cdrl a011—a curriculum module in the sei style. *ACM SIGSOFT Software Engineering Notes*, 20(5) :27–38, 1995.

[109] André MH Teixeira and Riccardo MG Ferrari. Detection of sensor data injection attacks with multiplicative watermarking. In *2018 European Control Conference (ECC)*, pages 338–343. IEEE, 2018.

[110] Chee-Wooi Ten, Chen-Ching Liu, and Govindarasu Manimaran. Vulnerability assessment of cybersecurity for scada systems. *IEEE Transactions on Power Systems*, 23(4) :1836–1846, 2008.

[111] Brian J Tetreault. Use of the automatic identification system (ais) for maritime domain awareness (mda). In *Proceedings of Oceans 2005 Mts/Ieee*, pages 1590–1594. IEEE, 2005.

[112] RM Trim. Mode s : an introduction and overview (secondary surveillance radar). *Electronics & Communication Engineering Journal*, 2(2) :53–59, 1990.

[113] Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner : System description. In *International joint conference on automated reasoning*, pages 292–297. Springer, 2006.

[114] Ming-Cheng Tsou and Chao-Kuang Hsueh. The study of ship collision avoidance route planning by ant colony algorithm. *Journal of Marine Science and Technology*, 18(5) :746–756, 2010.

[115] Arie Van Deursen and Paul Klint. Domain-specific language design requires feature descriptions. *Journal of computing and information technology*, 10(1) :1–17, 2002.

[116] Michele Vespe, Ingrid Visentini, Karna Bryan, and Paolo Braca. Unsupervised learning of maritime traffic patterns for anomaly detection. 2012.

[117] Kyle D Wesson, Todd E Humphreys, and Brian L Evans. Can cryptography secure next generation air traffic surveillance ? *IEEE Security and Privacy Magazine*, 2014.

[118] Ferry Wahyu Wibowo and Wahyu Sukestyastama Putra. Sensor array fault detection technique using kalman filter. In *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 124–128. IEEE, 2019.

[119] David N Wiley, Michael Thompson, Richard M Pace III, and Jake Levenson. Modeling speed restrictions to mitigate lethal collisions between ships and whales in the stellwagen bank national marine sanctuary, usa. *Biological Conservation*, 144(9) :2377–2381, 2011.

[120] Le Xie, Yilin Mo, and Bruno Sinopoli. False data injection attacks in electricity markets. In *Smart Grid Communications (SmartGridComm), First International Conference on*, pages 226–231. IEEE, 2010.

[121] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network : A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[122] Xinyu Yang, Jie Lin, Wei Yu, Paul-Marie Moulema, Xinwen Fu, and Wei Zhao. A novel en-route filtering scheme against false data injection attacks in cyber-physical networked systems. *IEEE Transactions on Computers*, 64(1) :4–18, 2013.

[123] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5 :21954–21961, 2017.

[124] W Young. What are vessel traffic services, and what can they really do ? *Navigation*, 41(1) :31–56, 1994.

[125] Tao Zhang, Shuai Zhao, Bo Cheng, and Junliang Chen. Detection of ais closing behavior and mmsi spoofing behavior of ships based on spatiotemporal data. *Remote Sensing*, 12(4) :702, 2020.

[126] Tong Zhou and Krishnendu Chakrabarty. Authentication of sensor network flooding based on neighborhood cooperation. In *IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006.*, volume 2, pages 665–670. IEEE, 2006.

[127] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning. Interleaved hop-by-hop authentication against false data injection attacks in sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 3(3) :14–es, 2007.

A

ANNEXES

A.1/ ARBRE D'HÉRITAGE DE L'ONTOLOGIE

Toutes les entités qui sont identifiées et modélisées dans l'ontologie des trois DSL (voir section 6.2 p. 97) pendant l'activité de l'analyse du domaine sont représentées dans les figure A.1, A.2 et A.3. Ces trois figures montrent les relations d'héritage entre les entités. Il est à noter la présence d'une unique entité *Thing* de laquelle toutes les autres héritent directement ou indirectement (de manière similaire à la classe *Object* dans le langage Java).

FIGURE A.1 – Arbre d'héritage de l'ontologie 1/3.

FIGURE A.2 – Arbre d'héritage de l'ontologie 2/3.

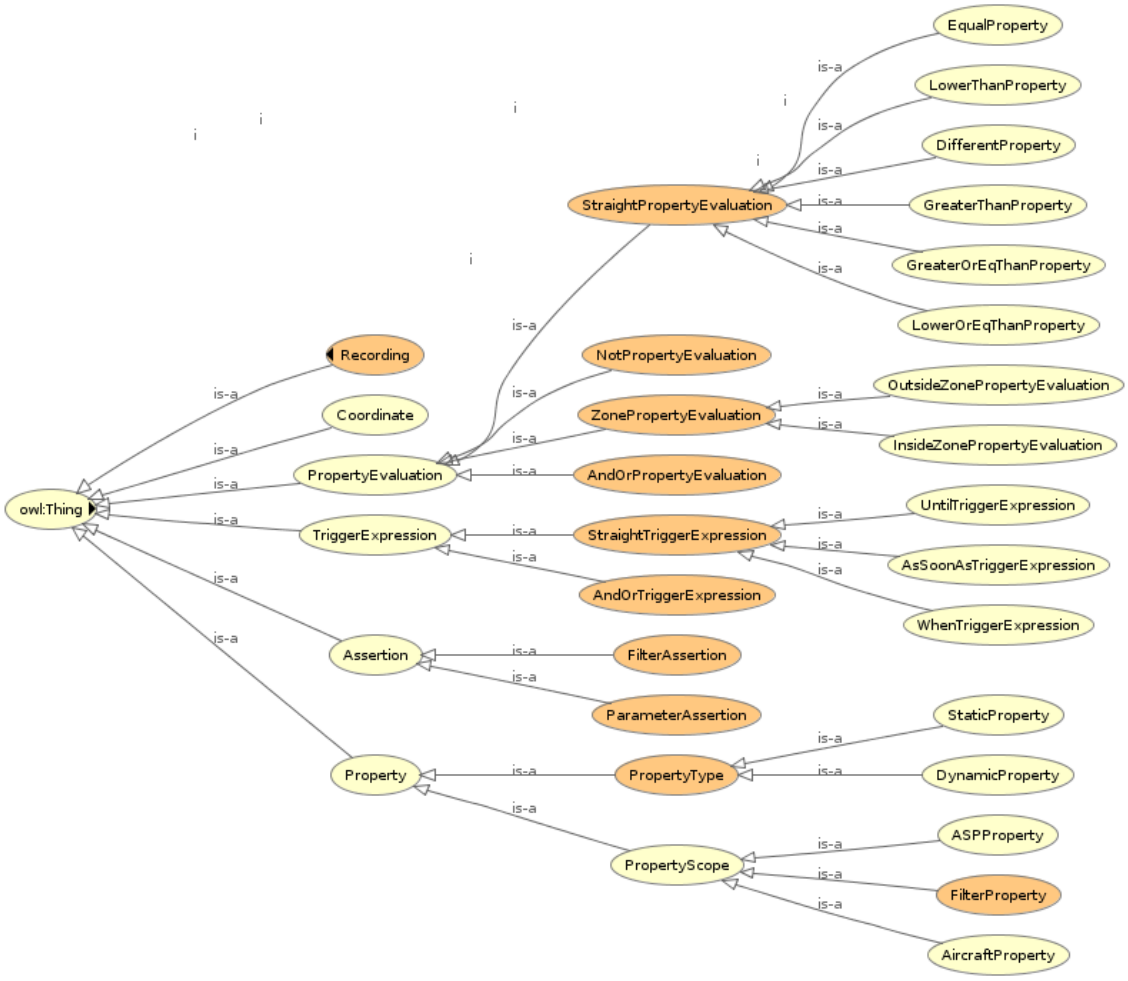


FIGURE A.3 – Arbre d'héritage de l'ontologie 3/3.

Document réalisé avec LaTeX et :
le style LaTeX pour Thèse de Doctorat créé par S. Galland — https://github.com/gallandarakhneorg/tex-templates
la collection de paquets `tex-upmethodology` — http://www.arakhne.org/tex-upmethodology/

école doctorale **sciences pour l'ingénieur et microtechniques**

Titre : Conception et génération de tests par altération de données pour les systèmes de contrôle et de surveillance des transports.

Mots-clés : Vulnérabilités logiques, Protocole ADS-B, Systèmes embarqués connectés, Protocole AIS, Test logiciel, Langage dédié

Résumé :

Face à l'augmentation constante du nombre d'avions en circulation dans l'espace aérien mondial, le domaine de l'ATC (Air Traffic Control) s'adapte et propose des solutions en développant des technologies de surveillance plus précises et moins coûteuses, parfois au détriment des questions de cybersécurité. Le protocole ADS-B (Automatic Dependent Surveillance – Broadcast) illustre parfaitement ces propos : il a pour objectif de réduire les coûts de surveillance en chargeant les aéronefs de déterminer leur position via le système GPS et de la communiquer aux tours de contrôles. La technologie ADS-B vient ainsi compléter un éventail de technologies existantes (radar primaire ou secondaire) en proposant une nouvelle manière de contrôler le trafic aérien. Tandis que les technologies antérieures à l'ADS-B étaient non-coopératives et nécessitaient « d'observer le ciel », le nouveau protocole est coopératif et nécessite « d'écouter le ciel ». Les contrôleurs, qui se basaient sur leurs observations, font désormais confiance à ce que les avions diffusent par messages ADS-B. En plus des questions de sécurité qu'apporte ce nouveau paradigme, le protocole ADS-B a été identifié comme particulièrement vulnérables aux FDIA (False Data Injection Attack). Du fait de son absence de chiffrement et d'authentification, il permet en effet à quiconque d'émettre de fausses informations de surveillance. Les attaques de type FDIA visent donc à altérer l'estimation globale de l'état du trafic par des combinaisons d'opérations basiques (ajout, suppression, modification) sur les messages ADS-B.
Face à ces menaces, des solutions sont proposées afin de renforcer la sécurité autour de la technologie ADS-B : détection d'anomalies, fusion de données, analyse comportementale, etc. Ces solutions sont basées sur la logique des données et doivent idéalement être confrontées directement à des FDIA afin d'évaluer leur efficacité de détection. Toutefois la validation de ces solutions se heurte à un problème récurrent dans les systèmes de détection d'anomalies : l'absence de données de test présentant des anomalies. Pour répondre à ce besoin, cette thèse propose une chaîne outillée pour la conception et la génération de scénarios d'anomalies pour les systèmes de contrôle de l'espace aérien. Cette chaîne outillée se présente sous la forme d'un framework de test nommé (*False Data Injection Testing*). La conception des scénarios de tests s'appuie sur un ensemble de langages dédiés (DSL). La génération des données de test se fait en appliquant des modifications à un extrait de communications ADS-B. À noter finalement que cette chaîne outillée a également été adaptée à la surveillance maritime qui repose sur le protocole AIS dont le fonctionnement et les caractéristiques en matière de sécurité sont similaires au protocole ADS-B. Ce portage a été réalisé dans le but de démontrer la généricité de l'approche globale détaillée dans cette thèse.

Title: Designing and generation of tests by data alteration for transportation surveillance and control systems.

Keywords: Logic vulnerabilities, ADS-B protocol, Embedded connected systems, AIS protocol, Software Testing, Domain Specific Language

Abstract:

Faced with the constant increase in the number of aircraft circulating in global airspace, the field of ATC (Air Traffic Control) adapts and offers solutions by developing more precise and less expensive surveillance technologies, sometimes to the detriment of cybersecurity issues. The ADS-B protocol (Automatic Dependent Surveillance - Broadcast) perfectly illustrates this point: its objective is to reduce surveillance costs by relying on aircraft to determine their position via the GPS system and to communicate it to the control towers. ADS-B technology thus complements a range of existing technologies (primary or secondary radar) by offering a new way of controlling air traffic. While technologies prior to ADS-B were non-cooperative and required "to watch the sky", the new protocol is cooperative and requires "to listen to the sky." Controllers, who relied on their observations, now trust what planes broadcast by ADS-B messages. In addition to the security issues brought by this new paradigm, the ADS-B protocol has been identified as particularly vulnerable to FDIA (False Data Injection Attack). Due to its lack of encryption and authentication, it allows anyone to issue false surveillance information. FDIA-type attacks therefore aim to alter the overall estimate of the traffic state by combinations of basic operations (add, delete, modify) on ADS-B messages.
Faced with these threats, solutions are proposed to strengthen security around ADS-B technology: anomaly detection, data fusion, behavioral analysis, etc. These solutions are based on data logic and should ideally be confronted directly with the FDIA in order to assess their detection effectiveness. However, the validation of these solutions comes up against a recurring problem in anomaly detection systems: the absence of test data showing anomalies. To meet this need, this thesis proposes a tool chain for the design and generation of anomaly scenarios for airspace control systems. This tool chain takes the form of a testing framework called (False Data Injection Testing). The design of the test cases is based on a set of Domain Specific Languages (DSL). The generation of test data is done by applying changes to an ADS-B communications extract. Finally, it should be noted that this tool chain has also been adapted to maritime surveillance, which is based on the AIS protocol, the operation and safety characteristics of which are similar to the ADS-B protocol. This porting was carried out in order to demonstrate the genericity of the global approach detailed in this thesis.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon, France