Architecture multi-agent pour la conception et l'ordonnancement de systèmes multi-senseur embarqués sur plateformes aéroportées

Ludovic Grivault

SORBONNE-UNIVERSITÉ

THÈSE DE DOCTORAT

Architecture multi-agent pour la conception et l'ordonnancement de systèmes multi-senseur embarqués sur plateformes aéroportées

Auteur :
Ludovic GRIVAULT

Directrice de thèse :
Pr. Amal EL FALLAH - SEGHROUCHNI
Professeure à Sorbonne Université - Faculté des Sciences et d'Ingénierie.
Responsable de l'équipe Systèmes Multi-Agent du LIP6

Thèse réalisée pour l'obtention du diplôme de Doctorat

Recherches réalisées au sein du
Laboratoire d'Informatique de Paris 6
Équipe des Systèmes Multi-Agent

et de
Thales Defence Mission Systems
Direction technique algorithmique

Jury de thèse :

Rapporteur :	Pr. Stéphane Galland	Directrice de thèse :	Pr. Amal El Fallah - Seghrouchni
Rapporteur :	Pr. René Mandiau	Examinatrice :	Pr. Safia Kedad-Sidhoum
Examinatrice :	Pr. Salima Hassas	Invité :	Pr. Vincent Corruble
Examinateur :	Pr. Jean-Marc Labat	Invité :	Raphaël Girard-Claudon

6 Décembre 2018

Sorbonne-Université

Résumé

Laboratoire d'Informatique de Paris 6
Équipe des Systèmes Multi-Agent

Architecture multi-agent pour la conception et l'ordonnancement de systèmes multi-senseur embarqués sur plateformes aéroportées

par Ludovic GRIVAULT

Le problème de planification et d'ordonnancement des capteurs d'une plateforme aéroportée peut être assimilé à un problème de perception active avec maximisation de la connaissance de l'environnement et optimisation des actions senseurs qui en découlent.
La problématique de la prise de décision fait naître le besoin d'une architecture au sein de laquelle les algorithmes de décision, les produits des capteurs et les données permettant leur interprétation sont organisés le plus efficacement possible tout en étant conformes avec les contraintes apportées par le contexte.
Les produits de la phase de planification doivent ensuite être exécutés avec un délai minimal par les capteurs et nécessitent ainsi un ordonnancement réactif et efficace adapté à la criticité des missions et à la complexité de l'environnement.
La décision des actions capteurs à exécuter au regard de l'environnement de la plateforme et du commandement demande une capacité de planification propre à l'ensemble des capteurs, afin de prendre des décisions en fonction des perceptions et contraintes de fonctionnement de chaque capteur. Un système multi-capteur, tel que présenté dans ce manuscrit, peut s'apparenter formellement à un atelier du type job-shop dans lequel les machines correspondent aux capteurs du SMS.
Ces dernières années les capteurs embarqués à bord des plateformes aéroportées n'ont cessé de se développer. La majorité de ces instruments permet de recueillir des informations constituant l'environnement de la plateforme qui les accueille. Puisqu'ils permettent de mesurer et *sentir* l'environnement ces instruments, aussi appelés *senseurs*, composent ensemble le système multi-senseur (SMS) de la plateforme. Le nombre de fonctions pouvant être réalisées par ces senseurs est en constante augmentation et leurs fonctions permettent d'accomplir un large éventail d'objectifs en mission.
Les plateformes de type aéronefs de combat ou plateformes pilotés à distance (RPAS pour *Remotely Piloted Aircraft System*) sont aujourd'hui considérées comme les principaux vecteurs d'actions en mission et se voient attribuer des objectifs toujours plus critiques. Les récentes mutations des théâtres d'opérations ainsi que les évolutions des besoins opérationnels amènent aujourd'hui ces systèmes à acquérir une place prépondérante dans des missions au sein d'environnements fortement dynamiques et particulièrement dangereux.
Nous verrons dans ce manuscrit comment le paradigme multi-agent permet de concevoir une architecture répondant au contexte et à ses évolutions moyen terme puis comment un ordonnancement à base d'heuristiques permet d'optimiser globalement les senseurs présents à bord de la plateforme.

Remerciements

Je tiens en premier lieu à remercier ma directrice de thèse, Amal El Fallah - Seghrouchni, pour nos nombreuses discutions et pour m'avoir accompagné et aidé tout au long de cette aventure, sur de nombreux plans.

Il me tient à cœur de remercier mes collègues de Thales, Raphaël Girard-Claudon, Éric Ségura, Cyrille Enderli et Luc Chabod pour la qualité de leurs interventions, pour avoir fait avancer les idées exposées dans ce manuscrit et sans qui cette thèse n'aurait sans doute pas eu la même portée.

Je souhaite remercier le post-doctorant Loïs Vanhée et les stagiaires Davide Guastella et Vitali Roubtsov qui ont contribué aux travaux présentés dans ce manuscrit.

Je tiens ensuite à remercier mes rapporteurs, Stéphane Galland et René Mandiau pour leurs remarques constructives.

Enfin, je remercie Quentin Cazergues et Laurent Savy pour leur relecture minutieuse et leur intérêt pour ce sujet complexe, parfois difficile à suivre.

Table des matières

Liste des acronymes

AAP	Artefact d'Application des Propositions
AESA	*Active Electronically Scanned Array*, antenne active à balayage électronique
ASIC	Application Specific Integrated Circuit, circuit intégré à application spécifique
ATC	*Air Traffic Control*, controle du trafic
BDI	*Belief, Desire, Intention*, Croyance, Désir, Intention
C2	*Command and Control*, centre de commande et de contrôle
CAS	*Close Air Support*, appui aérien rapproché
COMINT	COMmunications INTelligence
DRIL-P	Détection, Reconnaissance, Identification, Localisation - Poursuite
EES	Émetteurs à État Solide
ELINT	ELectronic INTelligence
EM	Électromagnétique
ESM	*Electronic Support Measures*, mesures de soutien électronique
EW	*Electronic Warfare*, guerre électronique
FFT	*Fast Fourier Transform*, Transformée de Fourier Rapide
FIPA	*The Foundation for Intelligent, Physical Agents*, la fondation pour les agents intelligents et physiques
FSM	*Finite State Machine*, Machine à nombre fini d'états
GE	Guerre Électronique
GMTI	*Ground Moving Target Indicator*, indicateur d'objets mobiles au sol
GM	Gestionnaire de Mission
GPR	*Generalized Precedence Relations*
HWIL	*HardWare In the Loop*, matériel dans la boucle
ICMP	Internet Control Message Protocol
IFF	*Identification Friend or Foe*, identification ami ou ennemi
IR	Infrarouges
ISAR	Inverse-SAR
MBSE	*Model Based System Engineering*, ingénierie système basée sur les modèles
MF	MultiFonction
MILP	*Mixed-Integer Linear Programming*, optimisation linéaire en nombres entiers

MITL *Man In The Loop*, homme dans la boucle
OF Objectif Fonctionnel
OO Objectif Opérationnel
PA-H Plateforme Aéroportée Habitée
PA Plateforme Aéroportée
PID Proportionnel, Intégral, Dérivé
PRS *Procedural Reasoning System*, système de raisonnement procédural
RBE2 Radar à Balayage Electronique 2 plans
RCPSP *Resource-Constrained Project Scheduling Problem*, problème de gestion de projet à contraintes de ressources
RF Radiofréquences
ROEM Renseignement d'Origine ElectroMagnétique
RPAS *Remotely Piloted Aircraft System*, plateforme aéroportée pilotée à distance
SAM *Surface-to-air missile*, missile sol-air
SAR Synthetic Apperture Radar, radar à ouverture synthétique
SER Surface Equivalente Radar
SIGINT SIGnal INTelligence
SMA Système multi-agent
SMS Système multi-senseur
SWIL *SoftWare In the Loop*, logiciel dans la boucle
SiTac Situation Tactique
TST *Time Sensitive Target*, Cible de haute importance
UAV *Unmanned Air Vehicle*, véhiculer aérien contrôlé à distance
URL Unité Remplaçable en Ligne

Liste des définitions

- **Théâtre d'opérations**
  Ensemble de la scène dans laquelle la mission prend place.
- **Bibliothèque**
  Base de données embarquée à bord de la plateforme contenant les informations utiles pour l'interprétation des évènements en cours de mission
- **Unité Remplaçable en Ligne**
  Boitier système embarqué sur la plateforme composé des cartes électroniques nécessaires aux fonctions réalisées

Chapitre 1

Introduction

1.1 Problématique générale

L'historique long et complexe des ensembles de senseurs aéroportés mène aujourd'hui à une incapacité à faire évoluer ces systèmes tout en satisfaisant les futures exigences du contexte. L'amélioration de chacun des senseurs aéroportés ne permet plus l'évolution globale du système dans son ensemble. Les interférences inter-senseurs et l'optimisation du fonctionnement global du système appellent alors à la création d'un ensemble fédéré de senseurs au sein duquel les ressources sont partagées, les interférences gérées et l'atteinte des objectifs unifiée. La création d'un tel ensemble de senseurs, appelé un système multi-senseur, demande la création d'une architecture permettant la satisfaction des exigences amenées par les besoins court et moyen terme du contexte tout en respectant les exigences déjà satisfaites par les senseurs actuels. La création d'un système multi-senseur demande une capacité d'unifier non seulement les ressources, données et le réseau du système, mais aussi les senseurs qui le composent.

Les nouvelles architectures de SMS devront permettre la collaboration des différents senseurs tout en garantissant un meilleur traitement des données qu'ils remontent du terrain. En raison de l'aspect réparti et coopératif des senseurs, le paradigme multi-agent a été employé pour la conception de cette architecture : les caractéristiques communicante et pro-active des agents purement communicants en font des candidats idéaux dans ce contexte. Le système multi-senseur ainsi créé permet de positionner les senseurs et les ressources dans une même dimension. La gestion des objectifs et l'adressage des ressources à ces derniers nécessitent une optimisation sur mesure. Ainsi, la collaboration, la coordination et l'optimisation des senseurs reposent essentiellement sur un ordonnancement spécifique, clé de voute de la nouvelle architecture.

Ce manuscrit présente les choix qui ont permis d'aboutir à cette nouvelle architecture ainsi que les mécanismes qui la régissent, en particulier le mécanisme d'ordonnancement.

La première partie de ce document analysera le contexte en profondeur afin de détailler l'ensemble des contraintes opérationnelles, industrielles, systèmes et environnementales qui ont façonné cette architecture et ses mécanismes. Nous définirons le terme senseur et présenterons les systèmes de senseurs dont ce manuscrit traite ainsi que les plateformes aéroportées les embarquant. Les enjeux et contextes opérationnels et industriels seront présentés, afin d'étayer les choix techniques pris tout

au long de la conception de l'architecture. Nous verrons ensuite comment l'historique complexe de la conception des senseurs va à l'encontre des attentes et besoins d'évolutions court/moyen termes des ensembles de senseurs. Dans ce cadre nous poserons ensuite les problématiques que nous essaierons de résoudre et nous proposerons des solutions principalement articulées autour de deux axes : comment une nouvelle architecture orientée agent peut concilier la complexité technico-opérationnelle des systèmes multi-senseur et comment le problème d'optimisation des senseurs du SMS peut être approché par la mise en place d'un ordonnancement adapté au système et son contexte.

La deuxième partie présentera l'état de l'art des recherches concernant les deux grands thèmes abordés dans ce manuscrit : la conception d'architectures orientées agent et l'optimisation des actions senseurs. Une première section s'intéressera aux systèmes multi-agent dans leur ensemble. Les agents et les SMA seront définis et des exemples d'environnements multi-agent (*frameworks*) seront présentés. La section suivante traitera des travaux réalisés en matière d'architectures orientées agents et plus précisément de certains travaux consacrés aux architectures multi-agent pour systèmes de capteurs. La thématique de l'ordonnancement sera abordée dans une seconde section. Nous verrons quelles sont les contraintes opérationnelles et industrielles à prendre en compte pour diriger les recherches d'un ordonnancement adapté à la problématique multi-senseur. Nous classifierons le problème d'ordonnancement selon la classification de Graham sous sa forme complète puis approchée. Nous présenterons ensuite différentes techniques d'ordonnancement pouvant répondre à la classe de problème propre aux systèmes multi-senseur. Nous montrerons que la méthode heuristique est la plus adaptée à la complexité du problème et aux contraintes opérationnelles.

Une troisième partie détaillera les travaux réalisés. Dans une première section nous montrerons les méthodes utilisées pour l'étude des architectures, des contraintes opérationnelles et industrielles, notamment par des méthodes MBSE [1]. Des exemples de scénarios mettant en œuvre des systèmes de senseurs seront présentés à titre d'expérimentation et de mise en situation.

Nous détaillerons dans la section suivante les différentes étapes de conception de l'architecture senseurs. Nous commencerons par présenter le paradigme multi-agent et ferons un rapprochement avec les tentatives actuelles d'unification des différents niveaux de contrôles des systèmes. Nous présenterons ensuite la phase d'agentification du système multi-senseur, étape primordiale de la conception de l'architecture. Nous verrons lors de cette phase que la première approche d'agentifier les senseurs, c'est-à-dire de leur attribuer l'ensemble des capacités cognitives et de planification, ne permet pas de satisfaire les contraintes du contexte. Les senseurs seront alors mis au rang de ressources et les agents correspondront à une virtualisation des objets observés par les senseurs. Ces agents seront appelés les agents tactiques et seront responsables de la planification des actions senseurs. Les structures de l'architecture et des agents seront ensuite explicitées. La boucle d'asservissement des senseurs, basée sur ce principe, ainsi que la dynamique complète de l'architecture seront identifiées.

Dans une troisième section nous montrerons que la problématique d'optimisation multi-senseur peut être assimilée à un problème d'ordonnancement d'atelier (job-shop) au sein duquel les senseurs correspondent aux machines de l'atelier et les

1. *Model Based System Engineering*, ingénierie système basée sur les modèles (MBSE)

fonctions senseurs des plans de tâches ordonnées pondérés d'une priorité fixe calculée par des règles opérationnelles. Nous expliciterons le problème d'ordonnancement comme un RCPSP [2] du type optimisation d'un job-shop à M machines (M étant le nombre de senseurs) avec des contraintes de tâches du type délais inter-tâches début-début/fin-début exactes, possibilité d'exécutions synchrones de tâches sur plusieurs machines, sans retards possibles et des durées d'exécution variables en fonction du temps de début de la tâche. Nous montrerons ensuite que les critères à optimiser sont la minimisation du makespan ainsi que la maximisation de la priorité relative des tâches ordonnancées. Les plans senseurs, issus de la phase de planification réalisée par les agents, seront présentés et détaillés dans cette troisième section. Nous verrons comment les plans senseurs, ses tâches et ses contraintes permettent la définition exacte de l'utilisation des senseurs. Les algorithmes des heuristiques d'ordonnancement prenant en entrée les plans senseurs seront alors détaillés. Nous montrerons que ces algorithmes, basés sur des évènements, garantiront des temps d'exécution convenables au regard du besoin.

La partie quatre mettra en avant les expérimentations réalisées pendant ces travaux et leurs résultats. Les systèmes multi-senseur étant des systèmes complexes encore en phase d'étude, les tests en situations réelles sont impossibles. Pour cette raison l'ensemble des expérimentations ont été réalisées en simulation. Les simulateurs RAMSES I et RAMSES II mettant en œuvre les concepts, l'architecture et l'ordonnancement, détaillés en chapitre 4, seront présentés.

Dans le chapitre 6, nous évaluerons et validerons l'architecture et l'ordonnancement en mettant en phase le contexte et les résultats des simulations. Nous verrons que l'architecture orientée agent permet une modularité et autonomie ne pouvant être atteinte avec une architecture actuelle. Nous verrons que les délais engendrés lors de l'ensemble des calculs de la planification à l'ordonnancement sont compatibles des exigences du contexte.

Nous conclurons ensuite ce manuscrit en synthétisant les avantages de la nouvelle architecture et en ouvrant sur de nouvelles problématiques, permettant d'approfondir ces travaux.

1.2 Présentation succincte du contexte

L'ensemble des instruments embarqués à bord d'une plateforme aéroportée comme les avions ou drones de combats permettent d'accomplir plusieurs objectifs de mission. Un des objectifs essentiels est de recueillir les informations de positions, trajectoires et types des objets présents autour de la plateforme et ainsi contribuer à la construction de la base des connaissances relatives à l'ensemble de l'environnement. Pour cette raison nous les décrirons pour le moment comme des *senseurs* puisqu'ils permettent de *sentir* et mesurer l'environnement. Le nombre d'objets présents autour de la plateforme, les capacités des senseurs, les objectifs donnés à la plateforme sont autant de facteurs pouvant impacter la réussite d'une mission.

2. *Resource-Constrained Project Scheduling Problem*, problème de gestion de projet à contraintes de ressources (RCPSP)

1.3 Évolution du contexte

Les senseurs et autres instruments développés tout au long des dernières décennies ont permis d'obtenir des performances accrues tant au niveau des distances de détection des objets que de la finesse de suivi de ces derniers. Compte tenu des limites de taille des plateformes ainsi que des efforts de baisse de prix de celles-ci, le nombre de senseurs à bord a atteint son apogée. Cependant, grâce aux progrès en traitement du signal, le nombre de fonctions qu'ils peuvent réaliser va continuer d'augmenter.

Plusieurs phénomènes influençant directement l'avenir des systèmes de senseurs ont été constatés : (1) les plateformes aéroportées changent de tailles et les objectifs de missions qui leur sont attribués seront toujours plus contraignant compte tenu de l'intensification des conflits dans les zones d'interventions ; (2) Ces dernières années, une tendance est d'ajouter des instruments et des fonctionnalités aux architectures existantes sans reposer de manière globale les problématiques de cheminement et de traitement de l'information à bord des plateformes, ceci ayant pour effet de mener à des architectures non évolutives dont la complexité n'est pas maitrisée ; (3) Les recherches récentes montrent que la prochaine étape d'amélioration de ces systèmes est de corréler les informations obtenues de plusieurs instruments pour en déduire de nouvelles informations. [Chabod et al., 2014]

Chapitre 2

Contexte applicatif et industriel

2.1 Introduction aux systèmes de senseurs

2.1.1 Présentation des plateformes aéroportées

Des avions aux navires en passant par les hélicoptères, les véhicules et les drones aériens, tout appareil possède des senseurs lui permettant de percevoir l'environnement afin de naviguer, recueillir des informations et agir sur ce dernier. Les recherches dans la suite du document seront concentrées sur deux types de plateformes aéroportées :

- Les plateformes aéroportées avec pilote
- Les plateformes aéroportées pilotées à distance

2.1.1.1 Plateformes aéroportées avec pilote

Aussi appelées avions pilotés, ces plateformes sont des avions mono ou biplaces, utilisés par les forces armées lors des opérations critiques. Les modèles les plus connus, le Mirage 2000 et le Rafale, ont tous deux été construits par Dassault Aviation. Ces deux plateformes sont entrées en service respectivement en 1984 et 2004 et ont été employées régulièrement lors des conflits de ces dernières années.

Ces plateformes aéroportées pèsent une dizaine de tonnes à vide pour une dizaine de mètres d'envergure, peuvent atteindre Mach 2 (2450 km/h) à 50 000 pieds (15 240 mètres) et sont capables d'emporter plus de six tonnes d'équipements et d'armements. Par leurs caractéristiques, ces avions, extrêmement polyvalents, permettent de couvrir un panel de missions très variées : reconnaissance, intervention, recherche d'objets et protection de convois, patrouilles. La figure 2.1 montre deux plateformes aéroportées pilotées.

Ces plateformes avec pilotes sont dotées d'un ensemble de senseurs afin de percevoir l'environnement pendant le vol, permettant ainsi au pilote et commandement de visualiser ces informations et prendre les décisions adaptées en temps réel.

Les senseurs embarqués à bord de telles plateformes sont nombreux et très spécifiques aux missions que ces plateformes doivent accomplir : un radar multifonction, des détecteurs de menaces, des détecteurs de missiles, etc. De plus, certains équipements peuvent être ajoutés sur la plateforme pendant les phases de préparations de vol : des *pods* ou nacelles mettant en œuvre des caméras et lasers, des missiles, etc. Chacun de ces emports peut être considéré comme un senseur supplémentaire à

disposition du pilote pour recueillir des informations sur l'environnement. La figure 2.1a montre une plateforme habitée embarquant un pod.

(A) Dassault Rafale.

(B) Lockheed Martin F-35.

FIGURE 2.1 – Exemples de plateformes aéroportées pilotées.

2.1.1.2 Plateformes aéroportées pilotées à distance : RPAS

Les RPAS (*Remotely Piloted Aircraft System*, plateforme aéroportée pilotée à distance) sont pilotées par une liaison radio permettant au pilote et au commandement, positionnés au sol, de contrôler la plateforme à distance.

À l'instar des plateformes avec pilotes, les RPAS possèdent un ensemble d'équipement permettant de collecter les informations de l'environnement. Les senseurs à disposition sont capables de réaliser les mêmes fonctions que ceux présents sur la plateforme aéroportée avec pilote. Ainsi, le radar, les différents détecteurs et les antennes seront aussi présents à bord d'un RPAS. Les figures 2.2 montrent deux exemples de RPAS.

Plus légères et avec des dimensions proches de celles des plateformes avec pilote, les RPAS étudiés ici pourraient demain les remplacer afin d'accomplir des missions périlleuses sans risquer la vie de pilotes. Une autre possibilité est de constituer des formations avec une plateforme habitée et plusieurs RPAS. Ainsi la plateforme pilotée a à sa disposition l'ensemble de l'équipement et de l'armement de la formation pour accomplir la mission. De plus, les distances entre les senseurs de la formation permettent une plus grande précision de mesures des angles et des distances suivant les algorithmes de traitement du signal utilisés, grâce aux différences de temps d'arrivées des signaux entre les senseurs. Voir [Merrill, 1990], chapitre 25.

(A) Le démonstrateur nEUROn.

(B) Le drone MQ-9 Reaper.

FIGURE 2.2 – Exemples de RPAS.

2.1.2 Définition d'un système multi-senseur multifonction

2.1.2.1 Définition d'un senseur

Un senseur est un instrument permettant de recueillir des données provenant de l'environnement. Le terme *senseur* est le synonyme de *capteur*, mais ce terme inclut des notions particulières non couvertes par la définition d'un capteur. Dans ce document, un senseur est un instrument pouvant être composé de plusieurs émetteurs/-capteurs ainsi qu'une partie « traitement de signal » afin de traiter l'information brute provenant des capteurs.

De manière générale en physique de capteurs, deux groupes sont à distinguer : les capteurs actifs et les capteurs passifs. Les capteurs passifs collectent des données de l'environnement sans le stimuler. C'est le cas des antennes permettant de recevoir les signaux provenant des objets sur le terrain ou d'une caméra. Contrairement aux capteurs passifs, les capteurs actifs nécessitent d'émettre des signaux pour pouvoir ensuite collecter des données. Un radar, un sonar ou un capteur de distance à ultrasons sont des capteurs de type actif.

Avec la généralisation de l'utilisation des systèmes radio et l'avènement de l'électronique numérique et de l'informatique, l'art du combat aérien s'est transformé. Aujourd'hui, la réussite d'une mission relève de la faculté des différents acteurs à collecter et maitriser l'information sur le théâtre d'opérations. Ce type de combat s'appelle la guerre électronique (Guerre Électronique (GE), *Electronic Warfare*, guerre électronique (EW) en anglais). La Guerre Électronique consiste à analyser le spectre électromagnétique afin de détecter, brouiller et localiser des objets situés sur le théâtre d'opérations.

La guerre électronique a aussi pour objectif de recueillir l'ensemble du spectre électromagnétique afin de procéder à un traitement ultérieur et préparer les prochaines missions (SIGnal INTelligence (SIGINT)).

Sur un théâtre d'opérations, la localisation des objets peut se faire de manière passive par localisation de la source émettrice des signaux radio. Pour cette raison, correctement comprendre quand un senseur peut être utilisé est essentiel pour l'aspect stratégique de la mission. Dans le cas où la mission nécessite une certaine discrétion, il faut limiter l'utilisation des senseurs actifs afin de ne pas exposer la position de la plateforme.

L'ensemble des informations concernant les objets du théâtre d'opérations constitue la SiTac (Situation Tactique). La SiTac peut être vue comme une carte du théâtre d'opération renseignée de la position des objets en temps réel. Cette carte peut rassembler les données de vitesse, d'altitude, de type, etc. de chaque objet connu du théâtre et constitue un des principaux centres d'intérêt pour la chaine de commandement opérationnelle. En effet, elle peut être partagée entre plusieurs plateformes de façon à profiter de la répartition géographique de ces dernières sur le terrain. L'ensemble des données remontées par les senseurs permettent de construire et contribuer à la SiTac propre à la plateforme et SiTac collective au fur et à mesure de la mission.

Les senseurs sont caractérisés par une grandeur physique ainsi qu'un sous-ensemble de valeurs de cette grandeur sur laquelle ils sont effectifs. Dans le cas d'un radar aéroporté, cette grandeur physique est de type Électromagnétique-Radiofréquences (RF) et sur des bandes de fréquences qui dépendent des caractéristiques du radar,

Bande X ($\approx$ 10 GHz) pour le radar du Rafale RBE2 (Radar à Balayage Electronique 2 plans), voir figure 2.3. Dans le cas d'une caméra fonctionnant dans le spectre visible, la grandeur physique est aussi électromagnétique dans des longueurs d'onde comprises entre 400nm et 700nm.

FIGURE 2.3 – Le radar à antenne active RBE2.

Un senseur est aussi caractérisé par un type de fonctionnement. Deux types peuvent être énoncés : un type *plage complète* et un type *spécifique*. Les senseurs du type plage complète utilisent le senseur sur tout le sous-ensemble de la grandeur physique qui lui est propre, et ce à chaque instant. C'est par exemple le cas pour un détecteur infrarouge : le détecteur fonctionne sur l'ensemble du spectre à chaque instant. Le pilotage d'un tel senseur revient essentiellement à l'activer ou à le désactiver, sans autre instruction particulière. Un senseur ayant un mode de fonctionnement spécifique requiert plus d'instructions afin de fonctionner. C'est le cas d'un radar aéroporté : le radar fonctionne selon une bande de fréquence spécifique choisie en fonction de l'objectif. À chaque instant, le radar fonctionne sur une bande de fréquence particulière, donc plusieurs bandes de fréquences ne peuvent pas être utilisées en même temps par l'opérateur.

La direction de pointage du faisceau radar est aussi une caractéristique spécifique. Chaque objectif de la mission peut requérir de pointer le faisceau dans des directions particulières. L'instruction à envoyer à un senseur de type spécifique peut donc contenir un grand nombre de paramètres comme l'angle de pointage, la fréquence de fonctionnement. Dans ce cas, il est dit de ce type de senseur qu'il fonctionne dans un *mode* donné. Un mode correspond à un ensemble de caractéristiques qui décrivent le fonctionnement du senseur de façon à abstraire l'ensemble des paramètres pour l'opérateur et ainsi faciliter son pilotage.

Un senseur peut être utilisé de deux façons : manuel ou autonome. Un exemple de pilotage manuel de senseur est l'utilisation d'une caméra de recul sur un véhicule. L'utilisateur souhaite obtenir une information de l'environnement, il active l'équipement et visualise le résultat. Au contraire, un altimètre à ultrasons sur drone est utilisé de manière autonome par la plateforme. L'utilisateur choisit dans ce cas l'attitude de l'appareil, l'algorithme de pilotage asservit les commandes de façon à respecter la consigne en utilisant l'altimètre. L'utilisateur n'utilise pas directement le senseur dans ce cas, mais décide uniquement d'un objectif de *haut-niveau*.

Pour une plateforme aéroportée, avec pilote ou RPAS, le contrôle des senseurs se fait aujourd'hui manuellement. Cela signifie qu'un opérateur est assigné à cette tâche.

Cela peut être le pilote de la plateforme ou un opérateur dédié en fonction du degré de complexité de l'objectif.

2.1.2.2 Définition d'un SMS multifonction

Les senseurs d'une plateforme aéroportée permettent de couvrir un ensemble d'objectifs de missions :

- Objectif spécifique de la mission (patrouiller, imager une zone, etc.) ;
- Construire la SiTac ;
- Protéger la plateforme ;
- Protéger des alliés/civils ;
- Aider au guidage de l'armement ;
- Communiquer ;
- Se repérer.

Ces objectifs ne sont pas de priorités égales et peuvent survenir à différents moments de la mission, en fonction des évènements extérieurs. Aussi, chacun de ces objectifs nécessite l'utilisation de différents senseurs. L'utilisation de senseurs configurés dans des modes particuliers afin de remplir un objectif précis représente une *fonction*.

Une fonction correspond à un ensemble d'objectifs pouvant être rempli et est réalisée par :

- Un ou plusieurs senseurs ;
- Des modes de fonctionnement.

Chaque objectif peut être atteint par une ou plusieurs fonctions, ce qui permet de changer de senseur en cas d'indisponibilité de l'un d'entre eux.

Le choix de la fonction permettant de réaliser un objectif dépend de plusieurs données :

- Le type d'objectif ;
- Les caractéristiques de l'objet concerné par l'objectif ;
- Le statut du senseur utilisé par la fonction (en-ligne, hors-ligne, occupé, libre, etc.) ;
- Le contexte général de la plateforme et du théâtre.

Si certaines de ces données ne correspondent pas au contexte nécessaire à l'accomplissement de l'objectif, une autre fonction doit être envisagée.

Exemple :

Soit le scénario suivant : pendant le vol, un objectif spécifique de mission parvient à la plateforme : *imager une zone du théâtre d'opération*s. Différentes fonctions peuvent être exécutées pour satisfaire cet objectif, comme la fonction d'imagerie optique et la fonction d'imagerie SAR (Synthetic Apperture Radar, radar à ouverture synthétique) (image acquise par le radar).

- **Situation 1 :** L'ensemble des senseurs sont fonctionnels et libres, l'objet est immobile/proche et la situation prend place pendant la nuit. Dans cette situation, l'objectif peut être accompli par les senseurs optiques et radars. Les différents paramètres sont étudiés et le paramètre du contexte général *situation de nuit* impose l'utilisation du senseur optique en mode nuit ou le radar en mode SAR. L'objet étant proche, l'optique en mode nuit sera privilégiée. Ici, la fonction d'imagerie optique en mode nuit sera donc sélectionnée en fonction de la préférence de l'opérateur spécifiée par une politique de choix des fonctions.
- **Situation 2 :** L'ensemble des senseurs sont fonctionnels et libres, l'objet est immobile/loin et le théâtre est chargé de nuages. Dans cette situation, l'objectif peut être accompli par le senseur optique et radar. Les différents paramètres sont étudiés et la météo implique le choix du radar pour accomplir l'objectif. Dans ce cas, la fonction sélectionnée sera la fonction d'imagerie SAR.

Un senseur peut être visualisé comme une ressource utilisée pour accomplir un objectif. Puisque plusieurs senseurs permettent d'accomplir plusieurs objectifs, et ce, de manière dynamique pendant la mission, les senseurs peuvent être considérés comme des ressources à la disposition des fonctions. Ces ressources sont ainsi partagées par l'ensemble des fonctions.

Les senseurs et les fonctions délivrées par ces derniers constituent ensemble un système multi-senseur multifonction (SMSMF). Aujourd'hui, les choix de fonctions/-senseurs sont effectués pendant la mission par l'opérateur au regard des paramètres cités en 2.1.2.2 ainsi que des objectifs de missions.

Le SMS peut être vu comme un fournisseur de service pour l'opérateur. Les services étant ici les fonctions mises à disposition par les senseurs.

2.1.2.3 Exemple d'un senseur

Le RBE2-AESA est le radar embarqué à bord du Rafale (figure 2.3). Développé par Thales, ce radar a une portée de la classe des radars du même type, communément comprise entre 100km et 150km.

Ce radar est constitué d'une antenne AESA (*Active Electronically Scanned Array*, antenne active à balayage électronique), aussi appelée couramment *antenne active*, ce qui signifie que l'antenne est fixe et composée de plusieurs centaines de modules électroniques d'émission et de réception. Les signaux sont émis dans une direction précise grâce au déphasage de chacun des modules électroniques constituant l'antenne. Cette technologie permet au pilote de suivre jusqu'à 40 cibles simultanément et d'effectuer un nombre de fonctions accru. Voir aussi [Merrill, 1990] chapitre 5 :

- Détection d'objets dans l'espace aérien ;
- Poursuite de plusieurs objets ;
- Imagerie SAR d'une zone ;
- Imagerie SAR avec défilement ;
- Guidage de l'armement ;
- Communication ;
- Brouillage ;
- Suivi de terrain ;

- Modes maritimes ;
- Imagerie basse résolution.

Ces fonctions sont toutes utilisables par le pilote pendant la mission.

2.1.2.4 SMS étendu

Les actions comme les actions de protection de la plateforme doivent être effectuées avec un délai extrêmement faible suite à l'observation d'un évènement, pouvant descendre jusqu'à une dizaine de millisecondes .

Certaines fonctions ne consistent pas uniquement à *sentir* l'environnement, mais aussi à agir sur ce dernier. Les fonctions permettant à la plateforme de communiquer, d'aider au guidage de l'armement ou de s'autoprotéger font partie de cette catégorie. Ces instruments s'appellent des effecteurs. À des fins de simplification des architectures et pour des raisons historiques ces effecteurs seront considérés comme des senseurs et des ressources du SMS et devront donc être pris en charge dans les futures architectures.

Par exemple, la détection d'un objet dangereux présent sur le théâtre et localisant la plateforme par le biais du spectre EM doit être contrée par une action de brouillage. Dans ce cas, l'opérateur visualise la donnée remontée par le senseur permettant de détecter la présence de l'objet en question puis déclenche l'action de brouillage appropriée. Cette action de brouillage utilise des instruments particuliers de la plateforme, les brouilleurs (ou *jammers* en anglais).

Dans la situation où l'objet détecté est un missile, le temps de traitement entre la détection de la menace et l'action doit être réduit au minimum, soit de l'ordre de quelques millisecondes. La contre-mesure de ce type de menace est le leurrage par l'envoi de leurres pyrotechniques (contre les missiles à guidage infrarouge) ou paillettes métalliques (contre les missiles à guidage radar). Une autre action de leurrage consiste à brouiller le radar guidant le missile. Aujourd'hui, ce faible délai est assuré par une liaison dédiée entre les senseurs chargés de la détection de la menace et les contre-mesures. Cette action senseur peut être qualifiée d'action réflexe. Les actions telles que les actions de leurrage et de brouillage sont appelées des contre-mesures.

2.1.2.5 Explicitation du SMS étudié

Un ensemble de senseurs et de fonctions a été défini sur le modèle des architectures précédentes ainsi que celle des prochaines plateformes aéroportées. Cet ensemble de senseurs pourra être localement modifié au cours du document afin de l'adapter à certains scénarios/contextes et ainsi montrer une certaine adaptabilité de l'architecture.

Les senseurs faisant partie du SMS étudié sont :

- Radar à antenne active coté droit ;
- Radar à antenne active coté gauche ;
- Optronique (caméra frontale) ;
- Antenne omnidirectionnelle ;
- Récepteur superhétérodyne ;

- Détecteur Laser ;
- Détecteurs Infrarouges (IR).

Ainsi que les effecteurs pouvant être considérés comme faisant partie de la suite de senseurs étendue :

- Brouilleurs ;
- Leurres (thermique et EM).

Cette collection de senseurs permet de mettre à disposition de l'opérationnel les fonctions essentielles à la mission. Un tableau récapitulatif est donné en annexe A [1].

Les **radars à antenne active droit et gauche** permettent de mettre en œuvre les fonctions citées plus haut (2.1.2.3). Ils sont disposés à ≈+60°et ≈-60°relativement à l'avant de la plateforme et couvrent ainsi un volume presque deux fois plus grand qu'avec une seule antenne (chaque antenne peut couvrir un angle d'approximativement 120°).

L'**optronique** permet de réaliser des fonctions d'imagerie et de suivi d'objets sur le terrain. Ces fonctions sont beaucoup plus rapides à exécuter que les fonctions de types SAR (le temps de prise d'images optroniques est de l'ordre de la seconde tandis que le temps d'acquisition SAR est entre 15 et 40 secondes), mais d'une portée plus courte (≈ 50km). Aussi, ces fonctions sont fortement sensibles à la météo, contrairement aux fonctions assurées par un radar.

L'**antenne omnidirectionnelle** et **superhétérodyne** font partie des senseurs essentiels à la GE. Ces deux antennes permettent d'écouter le spectre EM (Électromagnétique) afin de détecter, localiser, identifier les objets présents sur le théâtre d'opérations sur 360°autour de la plateforme.

Les **détecteurs laser** permettent à l'opérationnel d'être averti lors du pointage d'un laser de télémétrie ou de guidage d'arme vers la plateforme. Trois détecteurs sont positionnés sur la plateforme afin de disposer d'une couverture totale.

Les **détecteurs IR** permettent de détecter les menaces de type missile en détectant l'allumage de ces derniers. Lors de la mise à feu d'un missile, un important flash lumineux dans les longueurs d'onde IR, indiquant ainsi une menace potentielle pour la plateforme.

Les **brouilleurs et leurres** permettent de mettre en place des contres-mesures électroniques et thermiques face aux menaces détectées sur le théâtre.

Cet ensemble de senseurs permet de représenter un système de senseurs générique au sein duquel les échelles de temps et les différentes complexités de modes/compatibilités sont représentées.

1. L'ensemble des domaines d'utilisation des senseurs sont des valeurs types provenant de la littérature ouverte et n'ont pas pour objectif de refléter les performances réelles de senseurs précis.

2.2 Contexte industriel et opérationnel

2.2.1 Évolutions du besoin

2.2.1.1 Transformation du contexte stratégique

Les opérations menées ces dernières années affichent une certaine tendance. Les théâtres d'opérations sont la plupart du temps situés sur des territoires éloignés et inaccessibles pour les forces armées. Ceci implique l'utilisation de vecteurs (plateformes aéroportées, terrestres et navales) longue portée pour accomplir les missions. Ces dernières années, la démocratisation de l'électronique entraine une multiplication des dispositifs de communications civiles et militaires.

D'une part, la multiplication des menaces de type missile de croisière/balistiques amène les états à se prémunir de dispositifs de contrôles de l'espace aérien. Ces dispositifs permettent en outre de surveiller une éventuelle pénétration des frontières par des appareils étrangers.

D'autre part, certains théâtres d'opérations peuvent mettre en scène des menaces dissimulées, connectées et coordonnées par le biais de dispositifs de communications radio, parfois bon marché, tels que des talkies-walkies ou de simples téléphones portables. Dans cette situation, étudier les communications permet de déduire une grande quantité d'informations : positions des entités, identités, activités sur le théâtre, etc. Cet aspect n'est pas traité dans ce manuscrit.

L'ensemble de ces dispositifs fonctionnent par le biais d'émissions EM pouvant être détectées, localisées, identifiées par une plateforme aéroportée. Stratégiquement, la surveillance de l'ensemble des signaux radio circulant dans l'espace à des fins d'analyse et d'information correspond à la SIGINT (SIGnal INTelligence). Les opérations de SIGINT (SIGnal INTelligence) peuvent être répertoriées dans deux catégories : l'ELINT (ELectronic INTelligence), centrée sur l'étude des sources émettrices d'ondes radio (comme un radar) et la COMINT (COMmunications INTelligence), l'étude des communications radio entre appareils et instruments.

Aussi, les théâtres concernés par ces missions sont très dynamiques, chargés, montrent parfois une forte prolifération d'armes à bas coût pouvant être utilisées contre la plateforme et les alliés et sont très peu redondants : chaque théâtre d'opération présente des caractéristiques uniques, avec des menaces et des contraintes évoluant rapidement.

2.2.1.2 Transformation des besoins opérationnels

Au regard des évolutions stratégiques, les plateformes aéroportées seront contraintes dans l'avenir d'intervenir sur des théâtres distants, très dynamiques et menaçants. Une des priorités pour l'opérationnel est donc de réduire au minimum les risques liés aux moyens investis dans une mission. Cette minimisation des investissements peut intervenir selon plusieurs axes : (1) une réduction des risques humains ; (2) une baisse des enjeux économiques.

L'aspect du risque humain implique une forte demande de plateformes pilotées à distance de la part de l'opérationnel, afin qu'un minimum de vies soit engagé sur le théâtre d'opérations. Si le pilotage de plateformes à distances est une option pour

réduire ce risque, une autre option est aujourd'hui étudiée : une coopération de plateformes aéroportées pilotées et de RPAS. L'objectif d'une telle formation est multiple : (1) Minimisation de l'investissement humain ; (2) augmentation de la couverture senseurs sur le théâtre ; (3) mise à profit de la vision experte d'un opérationnel sur le terrain. En effet, la prise de certaines décisions comme l'utilisation de l'armement nécessitent de communiquer un grand nombre d'informations avec le centre de commande (images, requêtes spécifiques de vérifications, etc.). Comme vu dans le paragraphe précédent, si le théâtre est distant du centre de commande, ces échanges d'informations peuvent engendrer des délais dans la prise de décision et ainsi impacter la réussite de la mission. Cette contrainte peut être résolue par une formation d'une PA-H/multiples RPAS. L'unique pilote de la formation peut ainsi prendre des décisions en temps-réel en profitant d'un délai de transmission minimal entre les plateformes de la formation avec un risque réduit de rupture de liaisons.

Les RPAS devront néanmoins montrer une autonomie vis-à-vis du pilote ou du centre de commandement. Dans le cas où le RPAS est piloté à distance, les ruptures de liaisons peuvent être fréquentes et plus ou moins intentionnelles : brouillage ennemi, topographie du théâtre, etc. Ces ruptures de liaisons impliquent une forte demande d'autonomie des RPAS de la part des opérationnels, afin que la plateforme puisse prolonger la mission pendant ces périodes de coupures. Dans une situation où le RPAS est en formation avec une Plateforme Aéroportée Habitée (PA-H) l'autonomie est nécessaire afin de décharger le pilote de tâches de pilotages de senseurs du RPAS. Dans ce cas le pilote préfèrera communiquer avec le RPAS par l'intermédiaire d'ordres haut-niveau.

La couverture du théâtre est aussi dans ce cas élargi. Puisque les senseurs de chacune des plateformes représentent des ressources fortement demandées, disposer de plusieurs Plateforme Aéroportée (PA) sur le théâtre permet de recueillir plus d'informations. Aussi, le pilote de la PA-H peut utiliser les RPAS afin d'accomplir certains objectifs périlleux (éliminations de radar, pénétration des lignes de défense ennemies) avec un minimum de risques. Aussi, il est possible de réserver certains RPAS à des objectifs particuliers et de les faire intervenir à un instant précis de la mission (pour l'utilisation d'un emport spécial par exemple).

La perspective économique de la mission résulte en une tendance aujourd'hui vérifiée dans l'ensemble des secteurs économiques : une baisse des prix d'achats du matériel et des frais logistiques. Ceci est d'autant plus vrai dans le cas où les RPAS sont engagés sur des objectifs plus périlleux que des PA-H. Sur des théâtres à prolifération d'armement à bas coût, le prix du RPAS et le système de contrôle de la flotte peuvent être dans ce cas la seule réserve de la mission.

Le dernier point concerne la résilience des systèmes embarqués sur les PA. Les missions étant périlleuses et les menaces nombreuses, une certaine résilience est attendue des prochains SMS. Les opérationnels souhaitent que les systèmes puissent se reconfigurer en cas de panne et de dégâts physiques (armement ennemi) ou numériques (piratage, corruption EM).

2.2.1.3 Nouveaux objectifs des concepteurs de systèmes

À l'heure actuelle, l'électronique embarquée à bord de plateformes aéroportées représente approximativement un tiers du coût total de la plateforme. Pour l'opérationnel, l'objectif de baisse des coûts d'achat ainsi que la nécessité de disposer de

plateformes polyvalentes, facilement adaptable à n'importe quel contexte, amène l'industriel à développer des architectures de SMS modulaires. D'une part, ces architectures modulaires permettraient d'adapter de l'équipement rapidement en phase de préparation de mission, afin de s'adapter au contexte et à ces variations. D'autre part, du point de vue de conception de systèmes, une architecture modulaire et générique permet de limiter le nombre de modifications à effectuer lors de l'ajout d'un équipement et ainsi faciliter l'évolution de la plateforme.

Aussi, la conception de SMS est une tâche conséquente, impliquant des domaines d'études et des savoir-faire techniques très variés. Pour cette raison sa conception est répartie entre plusieurs entités d'une même entreprise, sur des périodes pouvant être réparties sur une dizaine d'années. Par conséquent, il est essentiel pour le systémier de pouvoir découper horizontalement la conception d'un tel système, afin de répartir avec souplesse sa conception entre plusieurs équipes et avec de faibles contraintes de précédences de conception (exemple : nécessité de concevoir un système avant de concevoir les systèmes ascendants). Une conception horizontale de l'architecture implique la spécification de protocoles unifiés au sein de l'architecture.

Une architecture modulaire, unifiée et horizontale permet pour le concepteur de disposer d'une base pérenne et économiquement maîtrisée du point de vue de la conception.

La plupart des objectifs des concepteurs concernent essentiellement les performances du SMS. Les problématiques d'énergie, d'encombrement et de coûts dépendent uniquement des possibilités qu'offrent les évolutions et avancées technologiques :

- Consommation d'énergie : les études se consacrent principalement à un meilleur rendement des composants électroniques ;
- Encombrement : la miniaturisation des composants et l'augmentation de leurs puissances de calculs permettent de réduire l'encombrement des systèmes ;
- Coûts : l'utilisation de senseurs multifonction permet d'économiser l'installation de systèmes supplémentaires. Exemple : antennes moins redondantes/-plus polyvalentes.

2.2.2 Évolutions des SMS de leur création à aujourd'hui

Retracer succinctement un historique des Système multi-senseur permet de constater une accélération du développement de ces systèmes et d'en estimer l'avenir à court et moyen terme.

2.2.2.1 Apparition des systèmes de senseurs

Le radar a été inventé au début du 20ème siècle mais n'était à ce moment là pas aéroporté, il permettait de détecter des navires en mer et d'en déterminer seulement un azimut approximatif. C'est grâce aux recherches et évolutions successives que le radar fut miniaturisé pour ensuite être embarqué à bord de plateformes aéroportées pendant la seconde guerre mondiale. Les radar embarqués étaient alors omnidirectionnels et peu précis.

Par la suite, après le développement d'antennes plus performantes (antennes à fentes de Young, directionnelles) et la découverte du Synthetic Apperture Radar,

radar à ouverture synthétique (SAR), les antennes étaient désormais capables de prendre des images radar du sol avec une bonne définition.

La Guerre Électronique est apparue plus tôt, lors de la 1ère guerre mondiale, et est apparue suite à l'utilisation des radiocommunications entre les plateformes. Il s'agissait alors essentiellement d'écouter le spectre EM et de localiser les plateformes émettrices de signaux radio. Plus tard, pendant la Seconde Guerre mondiale, l'utilisation généralisée des radar sol-air (détection d'avions dans l'espace aérien par un radar situé au sol) motiva le développement de dispositifs de contre-mesures électroniques tels que des paillettes métalliques de type *windows* ou des émetteurs radio permettant de brouiller les signaux des radars au sol ou embarqués.

2.2.2.2 Évolutions et mises-à-jour

L'avènement de l'électronique numérique permit de développer rapidement l'ensemble des instruments fonctionnant sur le spectre EM notamment grâce à la découverte de la *Fast Fourier Transform*, Transformée de Fourier Rapide (FFT), fonction mathématique permettant de discrétiser un signal afin de l'analyser numériquement. Cette fonction accélère le traitement et l'analyse des signaux reçus par les senseurs.

L'invention des semi-conducteurs et des transistors a permis de développer des amplificateurs de puissances supérieures, des vitesses de traitements plus élevés avec une taille réduite. Plus tard, les transistors seront assemblés jusqu'à former des micro-contrôleurs à application spécifique ou Application Specific Integrated Circuit, circuit intégré à application spécifique (ASIC). Ces composants permettent de traiter des informations en logique binaire afin d'automatiser le traitement des signaux recueillis par les senseurs. Ils sont alors conçus spécifiquement pour l'application étudiée, ceci implique des coûts de production élevés. L'ensemble de ses technologies a permis d'augmenter rapidement le nombre de fonctions réalisables par chaque senseur. Ainsi, le radar a été doté d'une multitude de modes, lui donnant la possibilité de couvrir plusieurs objectifs de missions à lui seul.

Les instruments de GE ont aussi été améliorés pendant cette période. Les brouilleurs et émetteurs gagnèrent en puissance et les algorithmes de traitement ont vu leur précision se décupler.

Les systèmes de GE et le radar ont tous deux profité des dernières technologies du traitement et de l'électronique numérique. Cependant, ils ont été développés indépendamment et ont donc par la suite été embarqués à bord des plateformes séparément. Ceci ne permet que peu d'interactions entre les deux systèmes : les uniques interactions possibles était alors les *discrets de compatibilité*, une classe de signaux transmis via des câbles entre les senseurs permettant d'empêcher le fonctionnement simultané de plusieurs fonctions et ainsi d'éviter une gêne mutuelle. Ceci marque l'apparition du premier réseau de senseurs, un réseau primitif au sein duquel peu d'informations se propagent automatiquement et de manière non homogène.

Les données produites par les senseurs des plateformes étaient alors visualisables uniquement par le pilote via des écrans positionnés sur le tableau de bord de l'appareil. Les senseurs sont activés manuellement à la discrétion du pilote.

2.2.2.3 Les SMS aujourd'hui

L'apparition et la diffusion de l'informatique ont permis la conception de nouveaux systèmes permettant de traiter l'information à bord des plateformes aéroportées. Le SMS peut alors profiter des nouveaux algorithmes de traitement embarqués et du stockage pour ainsi traiter et fusionner les signaux reçus par les senseurs et enregistrer les données recueillies. L'informatique embarquée a alors transformé le traitement à bord des plateformes. Il est désormais possible d'emporter les informations du théâtre et d'identification des menaces grâce à l'existence de bases de données embarquées pré-chargées avant la mission. Ces bases de données sont appelées *bibliothèques* et sont propres à chaque type de senseurs. Aussi, l'ère de l'informatique a amené les différentes entités historiques de conception des senseurs à les « informatiser » indépendamment formant aujourd'hui des systèmes complexes dans lesquels l'accès aux informations stratégiques est non homogène dans le SMS.

Les SMS actuels héritent de plusieurs caractéristiques des systèmes préexistants, tels que les discrets de compatibilité. Les données produites par les senseurs sont dorénavant fusionnées par un algorithme embarqué dans une des Unité Remplaçable en Ligne (URL) du SMS. La fusion de données et les signaux de compatibilité révèlent une intensification des communications de données au sein des SMS.

Le pilotage des senseurs se fait désormais par le pilote ou à distance via une liaison radio pour les RPAS. La charge de travail produite par la remontée des produits senseurs au pilote est fonction de la charge du théâtre d'opérations. Un théâtre chargé impliquera de réaliser de nombreuses actions senseurs et pourra induire chez le pilote une surcharge de travail, retarder des prises de décisions critiques et mettre en péril le succès de la mission.

Les SMS sont désormais proches d'un modèle de ressources mises à la disposition de l'opérateur, dans lequel une ressource correspond à un senseur. Cependant certaines actions senseurs sont réalisées en autonomie afin de réaliser les fonctions d'autoprotection de la plateforme. C'est le cas des actions de brouillage et de leurrage lors de la détection de menaces imminentes.

2.2.2.4 Avenir des systèmes

L'électronique et l'informatique continuant d'évoluer, le nombre de fonctions supportées par chaque senseur va augmenter. À court terme, l'amélioration de l'électronique et des traitements mènera à des instruments plus compacts et plus polyvalents.

L'objectif de baisse des coûts des plateformes et donc des SMS implique que les senseurs soient plus polyvalents/dotés de plus de fonctions, donc ceux-ci tendront à se raréfier à bord des plateformes aéroportées.

La limite long terme de cette tendance pourrait mener les SMS à être composés d'un nombre très limité de senseurs (par exemple : trois radars à antennes actives, une antenne omnidirectionnelle, un détecteur IR/laser) fortement polyvalents capables d'assurer l'ensemble des fonctions nécessaires. Cette tendance nécessitera une architecture disposant d'une forte optimisation du partage des ressources et une souplesse permettant d'ajouter facilement de nouvelles fonctions.

La diminution du nombre de senseurs induit une baisse de la consommation énergétique à bord de la plateforme : (1) baisse de la consommation électrique du SMS,

(2) baisse de la consommation en énergie motrice de la plateforme induite par une réduction du poids du SMS.

2.2.2.5 Des senseurs adaptables à différents contextes

Les architectures logicielles, matérielles et systèmes des futures plateformes aéroportées étant encore incertaines, les prochains SMS devront s'adapter à plusieurs plateformes afin de réduire les coûts de conception et le travail d'adaptation. Ainsi l'architecture SMS conçue se doit d'être la plus générique possible, afin de nécessiter un minimum de modifications en cas d'adaptation sur un type de plateforme différent (navale, terrestre ou spatiale). Dans ce contexte, le système de senseurs peut être réparti sur l'ensemble de la plateforme, sur des distances de l'ordre de la centaine de mètres, avec dans ce cas l'apparition de délais de transmission de l'information. L'architecture SMS peut dans ce cas permettre la prise en charge de l'ensemble des senseurs de manière décentralisée. La décentralisation peut intervenir tant au niveau des URL constituant le SMS qu'au niveau des algorithmes de décisions et de traitements intégrés dans ces derniers.

À bord d'un RPAS ou d'une plateforme habitée, de type avion de combat, les distances entre les senseurs sont inférieures à la dizaine de mètres. Pour cette raison, le matériel ne sera pas automatiquement décentralisé. Cependant cette option reste possible et doit être envisagée pour conserver le caractère générique de l'architecture.

Il est envisageable à moyen termes que les plateformes aéroportées soient en forte collaboration afin d'accomplir des objectifs de missions qui leur sont attribués en mettant à profit le caractère réparti des senseurs ainsi disponibles. L'optimisation du fonctionnement des SMS en formations multi-plateforme est alors essentielle, notamment par le développement d'architectures génériques et unifiées entre les plateformes.

L'architecture proposée doit donc présenter un certain nombre de caractéristiques : (A : Agents, O : Ordonnancement) :

- Haut et bas-niveau : permettre à l'opérateur de commander le SMS par ordres haut-niveau et SMS capable de générer l'ensemble des instructions senseurs bas-niveau (A+O) ;
- Temps réel : permet d'ordonnancer l'utilisation des senseurs en temps-réel souple (O) ;
- Adaptabilité : adaptable à différentes configurations de SMS avec un minimum de re-travail (A) ;
- Modularité : permettre l'ajout de matériel en préparation de mission (A) ;
- Autonomie : pro-active et en cohérence avec le contexte (A) ;
- Intelligence : doit intégrer une part de l'intelligence décisionnelle de l'opérateur (A+O) ;
- Robustesse : doit présenter une certaine fiabilité de fonctionnement et résilience (A+O) ;
- Générique : générique à plusieurs types de plateformes (A) ;
- Décentralisation : pouvant être décentralisée (A) ;
- Coopération inter-plateforme : permettre une future coopération inter-plateformes (A).

Ces caractéristiques peuvent être satisfaites au moyen de deux solutions allant de pair : une architecture pour les futurs Système multi-senseur et un mécanisme d'ordonnancement permettant d'optimiser la coordination des senseurs au sein de cette architecture.

2.3 Problématiques

2.3.1 Architecture des systèmes de senseurs

Les contraintes environnementales, le contexte technico-opérationnel et les technologies évoluent. Afin de suivre les évolutions court et moyen-terme les senseurs de plateformes aéroportées devront coopérer afin d'augmenter leur potentiel. D'autre part, le besoin en modularité, les contraintes historiques ainsi que les contraintes inhérentes à la physique des capteurs imposent d'envisager les senseurs comme des entités distinctes, mais devant coopérer pour accomplir la mission. Les senseurs d'un SMS apparaissent alors comme des entités indépendantes, intelligentes et agissant selon leurs propres règles de fonctionnement. Une capacité *plug-and-play* des senseurs peut être atteinte uniquement si ces derniers sont dans un même environnement, dans lequel il leur est possible d'échanger des données et ont des canaux de communications et protocoles communs.

Les besoins exprimés en section 2.2.2.5 peuvent se traduire pour un senseur en un comportement de haut niveau de façon à ce qu'il puisse :

- Adapter ses actions à l'environnement (terrain, météo, etc.) perçu par les autres senseurs ;
- Décider de la meilleure action au regard de l'objectif donné par le gestionnaire de mission ;
- Prendre connaissance des ressources et les réserver ;
- Suivre l'avancement des actions à exécuter et réagir en cas d'imprévu ;
- Être ajouté à un réseau de senseur existant, être intégré et mettre à disposition toutes ses capacités.

L'ensemble de ces capacités ne sont atteignables par une même entité logicielle que si son comportement est développé et ses processus sont de haut niveau. Les agents sont particulièrement adaptés à la réalisation de tâches en autonomie durant lesquelles les traitements et actions à réaliser sont variés. Plus généralement, les caractéristiques d'autonomie, d'intelligence et de décentralisation sont des propriétés intrinsèques aux agents logiciels, par nature intelligents, communicants et pro-actif. Pour cette raison, l'architecture du SMS sera bâtie autour d'un système multi-agent afin de placer les atouts de l'agent logiciel au centre de celui-ci. L'environnement des agents correspondra au réseau de communication du SMS.

Les caractéristiques de niveau d'action de l'architecture, d'adaptabilité, de généricité et de coopération sont déterminées par l'agentification, étape majeure de la construction de l'architecture, ainsi que par les algorithmes et mécanismes implémentés en son sein.

Les évolutions des besoins opérationnels, industriels et stratégiques amènent les futures plateformes aéroportées et plus précisément les SMS à repousser les contraintes des systèmes. Pour cette raison, il est devenu essentiel de renouveler les architectures

SMS existantes afin de couvrir les nouveaux et futurs objectifs auxquels les plateformes seront confrontées prochainement. Aussi, les nouvelles architectures système devront être plus permissives vis-à-vis des futures évolutions système, pour ainsi disposer d'une base pérenne de développement tout au long des prochaines décennies.

Le chapitre 4.2 expose les différentes étapes de la création de cette architecture.

2.3.2 Contraintes de conception d'une architecture multi-senseur

2.3.2.1 Satisfaire les besoins opérationnels

Le premier objectif de la future architecture multi-senseur sera d'allier l'autonomie du SMS tout en respectant l'ensemble des autres contraintes opérationnelles du système.

Une augmentation de l'autonomie des SMS permettra de répondre à deux besoins opérationnels distincts. D'une part, comme décrit en section 2.2.1.2, si ledit SMS est embarqué sur RPAS, une rupture de liaison de communication entre le système sol de pilotage de la plateforme (le gestionnaire de mission) et le RPAS rend impossible l'envoi d'ordres au SMS. Si les algorithmes de commandes de vol s'adaptent à une perte de liaison et peuvent contrôler la plateforme sur une trajectoire de retour, voire de maintenir le plan de vol donné par l'opérateur, la situation est différente pour le SMS. Les actions senseurs à réaliser dépendent du contexte terrain ainsi que d'un ensemble de paramètres qui seront spécifiés par la suite. Avec une architecture de SMS classique, l'ensemble des actions senseurs ordonné par l'opérateur avant une rupture de liaison devient obsolète suite à un évènement imprévu sur le théâtre. La fréquence d'arrivée des évènements est proportionnelle à la dynamique du théâtre. D'autre part, l'augmentation de l'autonomie du SMS permet de répondre à la problématique de surcharge des opérateurs et pilotes. En effet, dans un théâtre chargé, un certain nombre d'actions senseurs sont redondantes et nécessitent l'intervention permanente d'un opérateur afin de les programmer.

Dans la chaine de commandement des actions senseurs, l'opérateur occupe un rôle de décision. L'opérateur prend en charge la gestion de la trajectoire en fonction des objectifs et du théâtre d'opérations. Ensuite il observe l'ensemble des produits senseurs obtenus précédemment, le contexte et les objectifs de missions pour déterminer les actions senseurs compatibles avec la trajectoire puis les soumet au SMS. La problématique d'autonomie implique d'embarquer une partie de l'intelligence décisionnelle de l'opérateur à bord de la plateforme aéroportée.

La SiTac recueillie par le SMS représente un atout pour la chaîne de commandement et la survie de la plateforme. Pour cette raison, cette dernière doit prolonger sa construction si l'opérateur perd contact. De la même façon, pour une Plateforme Aéroportée Habitée, si le pilote est trop chargé ou dans l'impossibilité de lancer les ordres au SMS, un SMS autonome permettrait d'aider le pilote à accomplir l'objectif ou simplement de le décharger.

Une fois la SiTac recueillie, elle doit être transmise au centre de commande. L'analyse de la SiTac en temps réel étant lourde, une synthèse doit être réalisée avant l'envoi à la chaîne de commandement afin de décharger le plus possible l'opérateur et raccourcir par la suite les délais de décisions.

Malgré une forte autonomie de la plateforme, celle-ci doit rester sous autorité complète du gestionnaire de mission. En effet, les objectifs réalisés par ses plateformes sont fortement critiques et nécessitent donc le déclenchement manuel d'actions pendant les phases les plus sensibles (comme la neutralisation de menaces). Même si le SMS ne gère pas le système d'arme de la plateforme, il peut être nécessaire d'utiliser des senseurs pendant les phases d'utilisation des armes.

L'opérateur peut à tout moment décider d'utiliser un senseur manuellement ou de monopoliser une ressource du système sans avertissement. Face à cette situation il est requis que le SMS se plie à la requête de l'opérateur sans gêner sa demande. De la même façon, la trajectoire de la plateforme peut à tout moment être modifiée par l'opérateur ou le pilote et ainsi modifier l'ensemble des actions senseurs déjà planifiées ou en cours. La trajectoire est dans ce cas considérée comme une variable d'entrée connue du SMS, mais non modifiable par ce dernier. Le SMS doit s'adapter et planifier les actions senseurs du mieux possible en fonction de cette donnée.

Enfin, l'ensemble des séquences d'actions senseurs réalisées par le SMS doivent être reproductibles dans des situations similaires. Ceci est une contrainte apportée par la criticité des missions dans lesquelles le SMS intervient. Les scénarios de tests doivent être réalisés plusieurs fois et doivent illustrer le caractère prévisible du comportement du SMS.

2.3.2.2 Répondre aux exigences industrielles

Du point de vue industriel, l'architecture devra être évolutive et modulaire tout en satisfaisant les exigences techniques propres au pilotage de senseurs. Ceci implique plusieurs points de travail au sein de l'architecture : (1) une quantité de re-travail moindre à la modification/l'ajout de senseurs et de fonctions ; (2) Une standardisation des exigences de développement pour l'ensemble des interfaces des senseurs du SMS ; (3) la possibilité d'instauration d'un standard de communication au sein de l'architecture.

La caractéristique d'évolutivité du SMS demande une flexibilité d'adaptation aux futures évolutions de type matérielles et logicielles. Les potentielles évolutions matérielles sont nombreuses :

- Évolution de la connectivité au sein de l'architecture (topologie du réseau, câble cuivre/fibre-optique, modification des temps de propagation des signaux) ;
- Modification de la répartition logiciel/matériel ;
- Décentralisation matérielle ;
- Changement du matériel en interaction avec le SMS (par exemple les URL de la plateforme destinées au pilotage ou à l'armement)

De même pour les évolutions logicielles :

- Nouveaux algorithmes de traitement ;
- Modification des fonctions réalisables par les senseurs ;
- Changement des procédures de gestion des menaces (exemple : ajout d'un nouveau type de menace dans les bibliothèques, les bases de données de stockage des menaces) ;
- Modification des logiciels en interaction avec le SMS (logiciel du gestionnaire de mission au sol, logiciel embarqué de contrôle de l'attitude de la plateforme)

Aussi, les senseurs peuvent être sujets à de nombreuses innovations non seulement concernant leur nombre, leur taille, leur puissance, mais aussi concernant leur façon de fonctionner de manière générale. Concernant les senseurs, la marge d'évolution est étendue :

- Évolution des dimensions des senseurs existants
- Modification de leur fonctionnement et interactions
- Ajout de senseurs « jumeau » (exemple : ajout d'un deuxième radar à antenne active disposé différemment du premier)
- Création d'un senseur fonctionnant sur une dimension physique jusqu'alors non exploitée
- Répartition des senseurs sur plusieurs plateformes

Par exemple, à moyen termes il sera nécessaire de mettre en commun les antennes disposées sur un ensemble de plateformes en formation. Ceci suppose que les architectures SMS des plateformes permettent la création de ce type de coopérations et donc *a minima* d'être conçues sur le même paradigme.

La modularité système implique que le SMS doit être capable d'accepter plusieurs configurations de senseurs et qu'il soit possible de les modifier sans reprogrammer l'ensemble du système. Ceci décrit la capacité « plug & play » du SMS. Cette aptitude suppose que l'ensemble des senseurs et ressources du SMS soit interfacé grâce au même standard de communication et que chacun d'entre eux ait un comportement similaire du point de vue système (protocole de réception des ordres, récupération des données bibliothèques, etc.). Sous l'angle de la conception des systèmes et senseurs, disposer d'un ensemble de standards de communication et de comportement permet de répartir plus facilement la tâche de conception en équipes et simplifie les étapes de vérification système. De plus cette capacité apporte une flexibilité optimale au regard de la préparation, de la maintenance et de la personnalisation du système.

Les senseurs sont des instruments demandant une forte rigueur de pilotage. Ils fonctionnent sur une échelle temporelle pouvant être très réduite (de l'ordre de la dizaine de millisecondes) les instructions de fonctionnement doivent ainsi être à cette échelle afin de ne pas bloquer le flux de commandes leur parvenant. Certains senseurs comme le radar à antenne active disposent en leur sein d'un ordonnanceur temps-réel lui permettant de segmenter des ordres de plusieurs secondes en des instructions de la classe de la milliseconde [Winter, 2008]. Ceci permet d'étaler l'exécution d'un ordre sur une durée plus longue, donc sur de plus longues distances grâce au défilement de la plateforme sur sa trajectoire.

2.3.2.3 Possibilité d'enrichir l'architecture avec d'autres technologies

Ces dernières années, les processus de décision occupent une place de plus en plus importante au sein des systèmes embarqués autonomes. Les algorithmes de décision peuvent être répartis en plusieurs points d'une architecture SMS (récupération et utilisation des données senseurs, transmission, envoi à une URL, traitement décentralisé, etc.). Certaines techniques d'apprentissage, comme les réseaux de neurones par exemple, peuvent nécessiter l'emprunt de canaux de communication spécifiques au sein d'un SMS afin d'avoir à disposition l'ensemble des données nécessaires pour le traitement. Ces données seront d'autant plus accessibles si l'architecture est conçue pour accepter de telles technologies en son sein. Les agents permettent la mise en place de limites et d'interfaces précises au sein des systèmes. Ainsi, les interfaces

entre un agent augmenté d'une technologie et un autre agent ne diffèrent pas tant que la nature de leurs échanges reste inchangée. De la même façon il reste possible lors de l'évolution des systèmes d'insérer un certain nombre d'entités autres que des agents (artefacts, voir section 3.1.2). Ceci permet l'ajout de traitements non encapsulés dans des agents, de réduire la complexité de la modification des systèmes sans toutefois remettre en question le fonctionnement du SMA déjà en place. Les résultats de ces traitements peuvent, par exemple, être accessibles au sein de l'architecture en tant que service aux agents. Des exemples seront donnés en section 6.1.2.2.

2.3.3 Problématique de partage des ressources du SMS

2.3.3.1 Coordination et optimisation des senseurs par ordonnancement

Un ensemble de senseurs peut être assimilé à un atelier de type job-shop, c'est-à-dire un atelier dans lequel plusieurs machines sont à disposition pour réaliser la fabrication d'une pièce. La pièce défile entre chaque machine pour être transformée étape par étape. Les pièces à fabriquer correspondent ici aux objectifs que la suite de senseurs doit atteindre et les machines du job-shop correspondent aux senseurs. De l'apparition de l'objectif dans le système jusqu'à son accomplissement, plusieurs senseurs et ressources doivent être activés et réservés l'un après l'autre afin d'accomplir l'objectif.

Le fait que les machines du job-shop soient des senseurs amène des contraintes supplémentaires en comparaison à un problème de job-shop classique (utilisation synchrone de plusieurs senseurs pour un même objectif par exemple). Le problème de coordination et d'optimisation de la charge de chaque senseur, de la même façon que pour un job-shop, peut être résolu par un ordonnancement adapté aux contraintes du système. Les contraintes temps-réel, la prise en charge bas-niveau des senseurs et la forte concurrence pour l'utilisation des différents senseurs du système nécessitent un choix d'ordonnancement spécifique. La solution d'ordonnancement présentée dans ce manuscrit comporte, d'une part, un algorithme à heuristiques permettant de respecter l'ensemble des contraintes temporelles ainsi que des mécanismes de gestion des priorités des ordres senseurs fonctionnant à base de règles opérationnelles.

Le chapitre 4.3 montrera comment l'ordonnancement élaboré donne une fluidité et une cohérence à l'ensemble de l'architecture. Cet ordonnancement permettra le respect des contraintes de compatibilités et des différents temps de mesures/de préparation des senseurs grâce à la définition exacte de plans senseurs. Nous montrerons comment l'ensemble des décisions agent se traduisent en plans senseurs puis sont transformés par l'ordonnanceur en un ordonnancement global décrivant l'ensemble des actions senseurs à réaliser.

2.3.3.2 Contraintes de partage des ressources

Comme décrit en 2.2.2.4, le nombre de senseurs constituant le SMS va diminuer et le nombre de fonctions réalisables par ce dernier va augmenter. Ceci implique une forte augmentation des contraintes et des exigences pour le SMS lors des prochaines années. Il est alors essentiel d'ordonnancer les actions des senseurs afin d'optimiser la charge et le partage des différents instruments du SMS.

Les caractéristiques d'intelligence, de temps-réel et de prise en charge des instructions bas-niveau dépendent essentiellement de la mise en place d'un ordonnancement efficace au sein de l'architecture. Le théâtre étant en permanente modification et la plateforme constamment en mouvement sur ce dernier, l'ordonnancement doit permettre une réactivité et une cohérence face au contexte.

L'intelligence globale du SMS n'est en effet pas seulement fondée sur les décisions prises par les algorithmes embarqués, mais aussi par les mécanismes d'ordonnancement, capable de privilégier certains objectifs de missions avant d'autres en fonction de la charge du système et la configuration du théâtre.

Conserver les caractéristiques et performances des systèmes actuels

Les systèmes passés et actuels ne possédaient pas d'intelligence décisionnelle embarquée plus complexe que les actions réflexes décrites en 2.1.2.4 et sont incapables d'effectuer des actions senseurs automatiquement pour décharger l'opérateur ou le pilote d'une part de travail. Les actions réflexes de type leurrage et brouillages sont exécutées avec très peu de latence (de l'ordre de la milliseconde) et peuvent venir perturber l'ensemble de l'ordonnancement construit avant la détection de la menace. Ce faible délai, assuré par une liaison directe entre les senseurs impliqués, doit être assuré par un mécanisme d'ordonnancement temps-réel ou temps-réel proche au sein d'une architecture modulaire dans laquelle des liaisons dédiées ne sont pas envisageables.

De la même façon, les décisions sortant de l'architecture doivent être prises dans un délai raisonnable, de l'ordre de quelques dizaines de millisecondes.

L'ensemble de l'architecture doit accepter et pouvoir recevoir les règles opérationnelles (règles spécifiant l'ensemble du comportement du système au regard du contexte opérationnel) de l'opérateur et de la chaîne de commandement. Ces règles sont nombreuses et permettent au système de ne pas déborder de son rayon d'action et de respecter les contraintes apportées par la plateforme. Selon le contexte ces règles peuvent impliquer l'interdiction d'utilisation de certains senseurs, par exemple si l'utilisation du senseur trahit directement la position de la plateforme alors que la mission demande une discrétion totale.

Le SMS étant embarqué sur la plateforme, sa position et ses capacités (distances de détection, angles de vision, etc.) sont directement liées à la position de la plateforme. Cette dépendance implique que le SMS doit être capable de prendre en considération la trajectoire de la plateforme, ordonnée par l'opérateur, et de moduler les décisions prises en fonction de cette dernière.

Conclusion

Ce chapitre présente les raisons, étayées du contexte technique et opérationnel de la nécessité de concevoir une nouvelle architecture de système de senseurs. Une analyse en profondeur a été réalisée afin de détailler l'ensemble des contraintes opérationnelles, industrielles, systèmes et environnementales qui ont façonné l'architecture et ses mécanismes. Le terme senseur a été défini et les systèmes de senseurs et plateformes aéroportées ont été introduits. Les enjeux et contextes opérationnels et industriels ont été présentés, afin d'étayer les choix techniques pris tout au long de la conception de l'architecture. Les besoins de la nouvelle génération de système multi-senseur ont été brièvement introduits et seront détaillés au chapitre 4. Nous

avons montré comment l'historique complexe de la conception des senseurs va à l'encontre des attentes et besoins d'évolutions court/moyen termes des ensembles de senseurs. La problématique a été posée et les réponses ont été proposées en deux axes principaux : comment une nouvelle architecture orientée agent peut concilier la complexité technico-opérationnelle des systèmes multi-senseur et comment le problème d'optimisation des senseurs du SMS peut être approché par la mise en place d'un ordonnancement adapté au système et son contexte.

Chapitre 3

État de l'Art

3.1 Architectures multi-agent pour systèmes multi-senseur et multifonction

3.1.1 SMA

Le terme *agent* a été défini plusieurs fois depuis son apparition dans les années 1980. Dans [Wooldridge and Jennings, 1995], M. Wooldridge et N. Jennings présentent une étude sur les différentes définitions d'agents. De manière générale, les agents sont décrits comme des entités autonomes et proactives interagissant dans un environnement. Un rapprochement entre l'agent et l'individu humain est établi afin de mieux comprendre les caractéristiques et la portée de l'agent. Comme décrit par J. McCarthy dans [McCarthy, 1979], les termes habituellement employés pour décrire les différents comportements des individus humains peuvent être attribués aux caractéristiques et méthodes propres aux machines sous la condition que ces termes doivent décrire des comportements et fonctionnements semblables. L'utilisation de tels termes permet de mieux comprendre le fonctionnement des systèmes et de ses entités.

L'environnement et les agents du système constituent ensemble un SMA dans lequel les agents peuvent exister, interagir et communiquer. Les SMA ont été décrits pour la première fois dans les années 1980 et constituaient alors un nouveau paradigme de modélisation des systèmes. Les SMA sont employés dans des contextes très variés, de la simulation et modélisation pour l'étude des systèmes complexes et sociétés humaines au développement de SMA en ligne permettant de contrôler des systèmes réels. Ils ont été principalement utilisés dans la recherche pour modéliser les systèmes complexes [Jennings, 2000] et étudier les comportements des entités. Plusieurs catégories d'agents et d'environnements existent et peuvent ensemble couvrir un grand nombre de domaines de recherches : modélisation des comportements et étude des phénomènes sociaux dans les sociétés humaines, études des problématiques de transports, modélisation et étude des systèmes d'information, aide à la conception d'architectures logicielles et des systèmes. Le domaine des SMA s'intéresse à toutes les situations dans lesquelles les systèmes étudiés sont composés d'entités autonomes, qui interagissent avec leur environnement.

Chaque SMA peut présenter des caractéristiques très différentes et fait appel à des agents de types spécifiques. Ainsi, un SMA orienté vers l'étude des transports fera appel à un type d'agent totalement différent de ceux d'un SMA visant à modéliser un réseau de communication. Les grandeurs mesurables de ces deux systèmes étant

très différentes. Plusieurs types de SMA peuvent être identifiés [Ferber, 1995, p.16-17] :

- Les SMA dits « situés » mettent en scène des agents ayant une réalité physique dans leur environnement, ce dernier étant muni d'une métrique. Les agents de ce type disposent de fonctions permettant d'agir sur cet environnement, de le percevoir et de se déplacer au sein de celui-ci. Ce type de SMA correspond aux SMA robotiques, l'environnement est physique et les agents des robots évoluent dans cet environnement.
- Les SMA dits « communicants » constitués d'agents disposant uniquement de fonctions de communication au sein d'un environnement numérique. Ces agents ne perçoivent pas les objets du SMA. Leur seule façon d'interagir avec les objets de l'environnement est d'utiliser leurs moyens de communication.

Un SMA hybride situé/communicant peut aussi exister. Dans ce cas les deux types d'agents coexistent au sein d'un même système. C'est par exemple une modélisation possible SMA d'un environnement dans lequel des individus et des systèmes informatiques cohabitent. Dans ce cas, deux environnements coexistent en parallèle : un environnement physique et un autre numérique pouvant être modélisés par deux SMA [Längle and Wörn, 2001].

Le SMS pouvant être assimilé à un système d'information en réseau, le système multi-agent correspondant est du type purement communicant. Ainsi les agents seront du type logiciel, disposés sur le matériel d'une plateforme informatique et interagiront avec leur environnement via leurs fonctions de communication [Ferber, 1995].

3.1.1.1 Agent logiciel

Les agents logiciels sont une des principales entités constituant un système multi-agent purement communicant. Ces agents possèdent une architecture composée d'un ensemble d'éléments leur permettant d'exister et d'interagir au sein d'un SMA : un noyau, une mémoire et des fonctions de communication [Wooldridge, 1992].

Un agent logiciel hérite de l'ensemble des caractéristiques d'un agent notamment de l'autonomie. Cette caractéristique implique que l'agent développe un certain comportement vis-à-vis de son environnement. Ce comportement se reflète dans les décisions prises et dans les interactions avec son milieu [Ferber, 1995]. Un des principaux traits de l'agent logiciel est sa capacité à dire « Non » aux requêtes émanant des autres agents, avec les règles régissant ce mécanisme définies par le concepteur du SMA.

Architecture BDI

Certains agents sont dits BDI (*Belief, Desire, Intention*, Croyance, Désir, Intention) [Rao and Georgeff, 1995, Georgeff and Rao, 1998]. Ceci signifie que leur noyau comporte une implémentation construite autour de l'architecture BDI : *Belief* (Croyances) : Ensemble des données perçues par l'agent , *Desire* (Désir) : l'objectif haut niveau de l'agent, *Intention* (Intention) : les actions qu'il veut réaliser pour compléter ses objectifs. Cette architecture permet un traitement des informations d'une façon proche de celle d'un humain et permet ainsi d'isoler les entrées, les éléments qui

découlent du raisonnement et les actions de l'agent afin de mieux délimiter la portée et la cohérence de la prise de décision. La caractéristique BDI de ces agents leur procure un comportement rationnel en fonction des évènements extérieurs à l'agent.

Connaissances et croyances

Les connaissances et croyances sont des données relatives à l'environnement de l'agent et propres à ce dernier [Wooldridge, 1997]. À chaque évènement provenant de son environnement, l'agent traite l'information et la place dans sa mémoire sous forme d'une ou plusieurs structures de données symbolisant cet évènement. Ces évènements peuvent être distingués en deux catégories : les connaissances et les croyances. Une connaissance est une information de probabilité maximale constatée sur l'environnement de l'agent [Fagin et al., 2004]. Une croyance est une connaissance de fiabilité moindre, formée suite à une perception de l'environnement qui perd en probabilité après application d'un mécanisme d'inférence [Konolige, 1986]. Les croyances permettent à l'agent de se représenter l'environnement et de compléter la vision partielle de ce dernier en extrapolant les connaissances dont il dispose.

Par exemple, la perception d'un objet $\mathcal{O}$ sur un plan peut être traduite par l'observation d'une position et d'une vitesse. Ces deux perceptions par un agent peuvent être constatées à un instant t_0 et donc se traduire par deux connaissances dans la mémoire de l'agent. À l'instant t_1, la position et la vitesse peuvent être déduite par une extrapolation temporelle probabiliste des deux faits constatés à t_0. La vitesse et la position de l'objet à t_1 est donc une croyance, de probabilité plus faible que la connaissance acquise à t_0.

Noyau

L'ensemble des algorithmes régissant le comportement de l'agent sont contenus dans son noyau [Ferber, 1995]. Le noyau correspond à la partie active de l'agent et est l'endroit où sont prises l'ensemble des décisions et où sont réalisés les traitements. À la façon du noyau d'un système d'exploitation, le noyau d'un agent peut être vu comme un gestionnaire de l'ensemble des composants de l'agent. Cette partie de l'agent récupère et traite les données contenues dans sa mémoire. C'est aussi au sein de ce noyau que sont situés les algorithmes implémentant l'architecture BDI de l'agent, dans le cas où l'agent est un agent BDI. L'agent peut exister sous différents états au sein de son SMA. Le nombre d'agents d'un SMA pouvant être variable, ceux-ci peuvent être créés et arrêtés à chaud pendant l'exécution et doivent donc être initialisés afin de préparer et démarrer les fonctions nécessaires à ses activités au sein du SMA. Les états décrivant l'avancement de sa création, initialisation, exécution, arrêt et destruction sont tous décrits dans son «cycle de vie» [Ferber and Gutknecht, 1998]. Le cycle de vie de l'agent est une machine à état régissant l'exécution de l'agent. Ainsi, le SMA est averti de l'état d'avancement de la mise en ligne de l'agent, contrôler le déroulement de son cycle de vie ou le tuer et le redémarrer si une anomalie de fonctionnement est constatée.

Mémoire

La mémoire est un composant nécessaire à l'ensemble des entités informatiques. La totalité des algorithmes situés au sein de ces entités peut faire appel à la mémoire

à différents moments de l'exécution. Ainsi, une instruction d'un programme se résume à un emplacement mémoire qui est lu et exécuté par un processeur. L'agent logiciel étant une entité informatique, il nécessite aussi une telle mémoire afin que son code soit exécuté et que l'agent prenne vie. Les données contenues dans une telle mémoire vont de simples variables d'exécution à l'ensemble des algorithmes de l'agent. Cette mémoire bas niveau sera appelée mémoire d'exécution dans le reste du document.

Une autre mémoire est employée : la mémoire de stockage. De manière générale en informatique, la mémoire de stockage permet de stocker l'ensemble des données utiles aux programmes et à leurs utilisateurs. Au même titre que les entités informatiques, les agents intelligents présents au sein des SMA doivent aussi disposer d'une telle mémoire. Une mémoire de stockage permet d'accéder à des informations pouvant être contenues sous forme de bases de données, afin que l'agent puisse traiter de l'information ou stocker des données, et cela à différents instants de son exécution. Deux sous-types de mémoire de stockage peuvent être distingués : la mémoire interne de l'agent et la mémoire externe. La première, la mémoire interne, est propre à l'agent et convient pour le stockage des données légères (relativement à la mémoire externe). Les données contenues dans cette mémoire sont utilisées par l'agent très régulièrement et doivent donc être accessibles avec une faible latence. Le deuxième type de mémoire, la mémoire externe, correspond aux bases de données accessibles par l'agent. Ces bases de données contiennent l'ensemble des données pouvant être chargées pendant l'exécution du système afin d'accomplir des tâches de traitement, de décision, etc. Le temps d'accès à ces bases peuvent être plus long pour trois raisons : la liaison physique entre la machine où est situé l'agent et la mémoire, la concurrence d'accès aux données entre les différents agents et la taille des données, les informations contenues dans ces bases étant plus volumineuses.

Une troisième mémoire de type externe existe, commune à l'ensemble des agents d'un SMA. Cette mémoire est appelée *tableau noir* et permet aux agents d'échanger des informations en les publiant dans un espace accessible aux autres agents [Ferber, 1995]. Ce type de mémoire n'est plus ou peu utilisée aujourd'hui au sein des SMA puisqu'il affaiblit les capacités de décentralisation et d'autonomie des systèmes.

Communications

Les agents d'un SMA purement communicant sont placés dans un environnement leur permettant de communiquer et d'échanger des informations. La communication est l'unique moyen d'action pour ce type d'agent. Il est donc primordial d'identifier et de spécifier les méthodes de communication afin de comprendre les mécanismes qui régissent le fonctionnement des SMA purement communicants.

Du point de vue de l'architecture d'un SMA, un agent possède un ensemble de fonctions lui permettant d'exister grâce aux algorithmes implémentés dans son noyau. Le langage utilisé pour écrire ces algorithmes importe peu du point de vue interactions, seul le respect des conventions de l'environnement est fondamental. L'agent est perçu par les objets de son entourage uniquement par le biais de ses capacités de communication et de son comportement. Si un agent logiciel est implémenté dans

un langage différent d'un groupe d'agents et est situé sur une autre machine distante, mais qu'il dispose des mêmes normes de communication et le même comportement alors il ne sera pas identifié comme étant différent par le groupe, pouvant jusqu'à poser des problèmes de sécurité pour certaines applications [Cavalcante et al., 2012, Poslad, 2009].

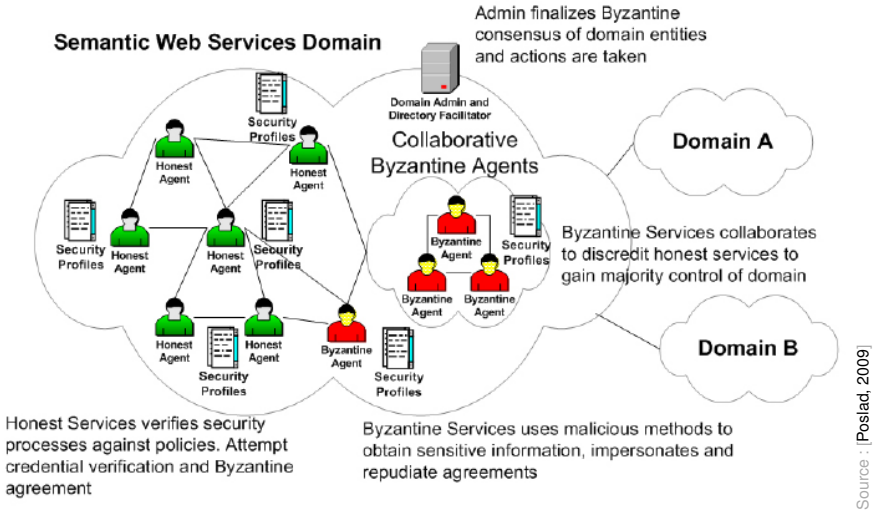


FIGURE 3.1 – Schéma d'agents byzantins malicieux dans un réseau de communication internet.

La capacité de communiquer d'un agent fait intervenir plusieurs composants permettant de mettre en œuvre un type d'échanges propre au SMA. En effet, l'agent doit posséder l'ensemble des composants et conventions nécessaires afin d'assurer les échanges inter-agents. Pour dialoguer avec un agent en particulier, un canal de communication en commun avec cet agent doit être disponible, impliquant que les deux agents doivent être physiquement connectés et que des mécanismes de l'agent permettent de communiquer via cette liaison physique. Aussi, un protocole de communication doit être connu des deux agents, afin qu'ils puissent échanger dans un langage commun.

Les agents logiciels étant intégrés dans des SMA informatiques, les liaisons inter-agents peuvent être virtuelles, donc simulées par une plateforme logicielle, ou encore utiliser des connexions de type TCP/IP (Transmission Control Protocol/Internet Protocol) [Pipattanasomporn et al., 2009]. Les agents font abstraction de toute la complexité des communications réseau grâce à l'ensemble des mécanismes mis en place par la plateforme soutenant le SMA, aussi appelée *plateforme SMA* ou *framework SMA* [Bellifemine et al., 2007].

Les agents communiquent par l'envoi de paquets appelés «messages». Un message est une structure de données complétée par un agent afin d'envoyer une information à un autre ou à un ensemble d'agents (exemple : envoi de type *broadcast*). La structure d'un message est propre à chaque SMA et plusieurs types de messages peuvent être employés au sein d'un même SMA. Le contenu du message est régi par les protocoles de communication adoptés durant la conception du SMA [Bellifemine et al., 2007]. Par exemple, le protocole ACL défini par l'ensemble des standards FIPA (*The Foundation for Intelligent, Physical Agents*, la fondation pour les agents intelligents et physiques) permet de définir un standard de communication inter-agents [Labrou et al., 1999]. L'envoi et la réception de messages sont assurés par la mailbox

de chaque agent [Potiron et al., 2009]. La mailbox est un composant faisant office de mémoire tampon entre la réception et le traitement du message par l'agent. Ceci permet un traitement asynchrone des messages afin que l'agent ne voie pas son cycle de vie impacté lors de chaque échange.

La chaîne de communication fait donc intervenir les trois composants liaison, protocole et mailbox, chacun essentiel dans le processus d'interaction des agents.

SMA purement communicant

Les SMA purement communicants fournissent un ensemble de services aux agents permettant d'assurer leur fonctionnement et leurs interactions. Les services annuaires et distribution de messages sont deux services essentiels aux agents [Ferber, 1995]. Le premier consiste à maintenir une liste des agents actifs du SMA à jour afin de renseigner aux agents l'adresse du destinataire de leurs messages. Ce service peut être questionné par l'agent à la demande. Le deuxième service, la transmission de message, est transparent pour l'agent, qui utilise simplement ses fonctions embarquées pour accéder aux transmissions. La transmission de message est assurée par la plateforme multi-agent en faisant transiter le message entre les mailbox. D'autres services peuvent être ajoutés, tels qu'un watchdog (chien de garde, mécanisme permettant de contrôler la bonne exécution d'un dispositif électronique) pour surveiller le cycle de vie des agents ou un service de réplication d'agent pour la robustesse du SMA.

Certains de ces services peuvent être incarnés par des agents, comme le service d'annuaire ou un watchdog.

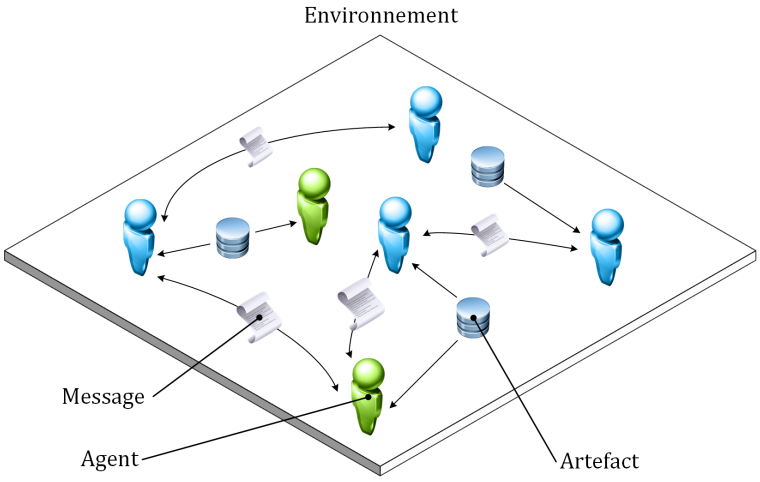


FIGURE 3.2 – Schéma d'un système multi-agent purement communicant.

Ping-Pong : le *Hello-World* des SMA purement communicants

Un SMA minimaliste est constitué d'au moins 2 agents logiciels en capacité d'échanger des messages. À des fins de tests, les agents peuvent envoyer des « ping » et répondre par des « pong » à l'image d'un jeu de balle. Le *ping* est une commande utilisée en informatique pour tester la liaison de communication entre la machine source et une machine cible. La commande *ping* consiste à envoyer un paquet de

données minimal basé sur le protocole Internet Control Message Protocol (ICMP) et chargé des informations telles que l'adresse de l'émetteur, le destinataire et la date d'envoi. À l'instar de cette commande, le ping d'un agent à un autre permet de vérifier l'état de la liaison de communication et le *pong* permet de contrôler l'interactivité de l'agent cible. Ce SMA minimaliste agrémenté de ce mécanisme de ping-pong permet de tester l'ensemble des composants nécessaires au fonctionnement du SMA : le transport de messages, la mailbox, le cycle de vie, les protocoles, le service d'annuaire de la plateforme SMA et les mécanismes élémentaires des agents.

3.1.2 Artefacts

Les agents ne sont pas les seules entités présentes dans un SMA. Le terme *artefact* est employé pour représenter l'ensemble des objets passifs d'un SMA [Omicini et al., 2008] dépourvus d'objectifs. Un artefact est une entité mettant à disposition des ressources aux agents telles que des bases de données ou des accès vers l'extérieur du SMA.

Si un parallèle peut être fait entre un agent et un individu, il en va de même avec l'artefact. L'humain est à l'agent ce que l'outil est à l'artefact. Un artefact peut en effet être vu comme un automate répondant aux ordres et aux requêtes des agents. Un artefact peut correspondre à un senseur, une base de données ou un composant d'échange avec le monde réel. Une métaphore de l'artefact dans le monde réel est un distributeur de billets : l'individu utilisant le distributeur de billets est gouverné par des buts et une volonté alors que le distributeur répond simplement à l'agent en réalisant sa demande. Si le distributeur de billets peut refuser la demande de l'agent, dans le cas où le distributeur est vide par exemple, la logique amenant au refus est des plus simples et ne fait pas intervenir la notion d'objectif.

3.1.2.1 Portée des SMA et applications

Le caractère autonome et communicant de l'agent logiciel permet de placer les SMA au sein d'applications mettant en scène un ensemble d'entités autonomes et indépendantes. Ainsi les systèmes tels que les sociétés humaines, les ensembles de systèmes connectés ou les flottilles de véhicules sont très étudiés au moyen de Système multi-agent (SMA). Une société humaine est composée d'une multitude d'individus, chacun possédant ses propres objectifs, agissant sur les objets de son environnement et interagissant avec les autres individus. Une solution de modélisation des sociétés humaines par un SMA est de positionner un agent par individu.

L'étude par la simulation des sociétés humaines permet de déduire de nouvelles façons de communiquer, de se déplacer et d'interagir afin d'améliorer l'organisation et d'étudier l'impact du changement sur l'ensemble de la société. Par exemple, les travaux de O. Goudet, JD. Kant et G. Ballot, [Ballot et al., 2013], concernant le marché du travail français ont été effectués par l'intermédiaire d'un SMA au sein duquel les individus étaient incarnés par des agents. Cette modélisation permit d'étudier l'impact des récentes réformes du code du travail sur le marché de l'emploi.

Le projet Terra Dynamica, [Reynaud and Corruble, 2013], met en œuvre un système multi-agent pour la simulation de comportements humains à grande échelle. Le projet Terra Dynamica consiste en la simulation de comportements réalistes d'entités autonomes se déplaçant au sein d'environnement urbains (jumeaux numériques de villes existantes). Cette simulation a pour objectifs de simuler la dynamique de

scènes urbaines courantes, mettre des opérateurs dans un contexte réaliste et valider des hypothèses.

Dans [Navarro et al., 2011], L. Navarro, F. Flacher et V. Corruble proposent une architecture multi-agent au niveau de détails variable permettant d'adapter le niveau de détails des interactions entre agents dynamiquement lors de la simulation, principalement pour d'optimiser les temps de calcul croissants dus à la complexité des interactions entre agents et leur nombre lorsque de larges SMA sont simulés (simulation de mouvements de foules par exemple).

Lorsque les décisions sont étudiées au sein d'un groupe d'entités autonomes - comme une flottille de drones - une modélisation par SMA, du type un drone = un agent, permet de mieux comprendre l'impact de chaque décision au sein du groupe et de voir apparaître un comportement intelligent de la part de la flottille. Étudier par la simulation ce système facilite l'analyse des interactions et des décisions prises par chaque drone. Le caractère décentralisable et autonome par nature des agents permet un positionnement direct de ces derniers sur les drones de la flottille. L'avantage majeur de cette correspondance entre les agents et les drones est de pouvoir naturellement et physiquement déployer le système multi-agent sur la flottille afin de bénéficier des données de vol et des observations de l'environnement à chaque instant. L'agent représentant auparavant le drone sera donc embarqué et permettra de prendre les décisions en temps-réel et par conséquent accordera aux drones la possibilité de coopérer afin de satisfaire un objectif commun de manière collaborative. De plus, ils pourront profiter de l'ensemble des spécifications apportées par les standards de communication pour optimiser et fluidifier les échanges dans l'ensemble de la flottille. Voir [Bürkle et al., 2011].

3.1.2.2 Agentification

L'agentification est l'étape principale de la conception d'un SMA. Comme vu dans 3.1.2.1, la répartition des agents sur les différentes entités du système étudié est capital pour le fonctionnement global du système et la conception des mécanismes qui le régissent [Demange et al., 2011]. Puisque l'agent est la partie intelligente et autonome du système, lui attribuer une mauvaise place dans le système pourrait mener à des incohérences de décisions et des dysfonctionnements. L'analyse systémique consiste à déterminer les entrées/sorties du système, comprendre les flux d'informations, identifier les centres décisionnels et situer les objets non intelligents. Suite à cette analyse, une agentification peut être créée si le système est propice à un tel modèle : un système dans lequel un unique agent peut exister ne justifie pas l'adoption du modèle multi-agent. *A contrario*, réaliser une agentification correctement mettra à profit les atouts de l'agent dans l'ensemble du système. Chaque système peut être modélisé par plusieurs agentifications différentes.

3.1.3 SMA en ligne

Si certains agents ou artefacts du SMA sont interfacés à une réalité physique et que les processus agents impactent directement l'état de cette réalité alors le SMA est qualifié de SMA en ligne ou *online*. Comme introduit en 3.1.2.1, un SMA en ligne consiste à utiliser les agents pour prendre des décisions en temps-réel en fonction d'un contexte physique pouvant être perçu par l'intermédiaire des artefacts du SMA.

Les situations propices à la mise en place d'un SMA en ligne sont nombreuses, pouvant aller de la gestion du trafic aérien ou le contrôle de flottille de drones jusqu'à la gestion des équipements connectés d'une maison intelligente.

Une flottille de drones ou les équipements d'une maison intelligente présentent des architectures matérielles décentralisées et permettent de répartir naturellement les agents sur les composants des systèmes. Créant ainsi naïvement une hiérarchie bijective agent/composants du système, agentification pouvant présenter des problèmes de lourdeur si les agents sont trop nombreux [Tumer and Agogino, 2007].

Plusieurs études ont été réalisées sur le thème du contrôle de trafic aérien ou *Air Traffic Control*, controle du trafic (ATC), donnant naissance à plusieurs architectures différentes. La tâche de suivi, de coordination et de contrôle des appareils dans une zone est une tâche lourde et fortement critique aujourd'hui réalisée par un opérateur humain. L'opérateur a plusieurs objectifs afin de mener à bien cette mission : surveiller les détections des aéronefs grâce au radar ou au transpondeur, interagir avec l'appareil par radio pour l'identifier, analyser son insertion dans le contexte du terrain pendant son évolution, élaborer un plan de vol pour l'appareil compatible avec le contexte et enfin renseigner à la plateforme son plan de vol et suivre son évolution sur ce dernier. L'ensemble de ces tâches peuvent être en partie automatisées afin de décharger le contrôleur d'une part de travail.

Plusieurs architectures ont été créées suite aux travaux dédiés à ce domaine [Ljungberg and Lucas, 1992], [Callantine, 2002], [Nguyen-Duc et al., 2008] et [Tumer and Agogino, 2007], menant à plusieurs agentifications possibles du même problème. Ainsi, les travaux de Todd J. Callantine *et al.*, supporté par le FAA/NASA Aviation Safety Program, et ceux de K. Tumer et A. Agogino ont donné naissance à deux architectures dans lesquelles chaque agent a pour objectif de contrôler l'évolution des avions dans une zone qui lui est attribuée. Les dimensions de la zone à contrôler par chaque agent dépendent de la fréquentation et de la dynamique de la zone.

Les travaux de Nguyen-Duc *et al.* s'intéressent aux problèmes de la sécurité des interactions entre les logiciels de gestion des vols et les opérateurs des centres de contrôles aériens. Deux types d'agents ont été créés : le premier, un assistant de l'opérateur, permet de suggérer certaines actions à l'opérateur. Le second, un agent observateur, surveille les échanges et les informations transitant dans le système et alerte en cas d'anomalie.

Les travaux de M. Ljungberg et A. Lucas, qui donnèrent naissance au système OASIS [Ljungberg and Lucas, 1992], abordent le problème de la gestion du trafic aérien sous l'angle le plus intuitif, en positionnant un agent par avion présent dans l'espace aérien. Les algorithmes de raisonnement procédural *Procedural Reasoning System*, système de raisonnement procédural (PRS) [Georgeff and Lansky, 1987], développés par le Stanford Research Center, ont ensuite été intégrés dans le système du contrôle de trafic aérien OASIS de l'aéroport de Sydney, en Australie.

3.1.4 Architectures multi-senseur

3.1.4.1 Transformations des systèmes de senseurs

Comme décrit en 2.3, il est essentiel d'optimiser les systèmes multi-senseur actuels afin de couvrir les futurs besoins industriels et opérationnels. Plusieurs travaux vont dans ce sens, notamment ceux portant sur les transformations architecturales et sur

l'optimisation de l'utilisation des senseurs. Ainsi, les travaux de Chabod *et al.* présentés dans [Chabod and Galaup, 2014] et ceux de S. Kemkemian et M. Nouvel-Fiani dans [Kemkemian and Nouvel-Fiani, 2010] laissent entrevoir une forte polyvalence des futurs systèmes de senseurs, autorisant la réalisation d'un maximum d'objectifs de missions par des senseurs multifonction partagés.

Les informations obtenues grâce aux senseurs situés à bord des plateformes aéroportées peuvent gagner en exactitude dans les cas où les données sont croisées entre les senseurs. Cette démarche est appelée fusion de données (merging en anglais). La fusion de données consiste à compléter et raffiner les produits de senseurs afin d'en déduire de nouvelles informations ou d'augmenter le poids des mesures [Luo et al., 2002].

3.1.4.2 Assistance au pilote

Les recherches axées sur l'assistance des opérateurs et pilotes pendant le vol sont nombreuses. Allant de l'étude de l'état mental du pilote en situation de stress [Hoogeboom et al., 2004, Ibrahim et al., 2013, Schulte et al., 2015] au développement d'un système permettant d'épauler le pilote en vol [Joubert et al., 1995], appelé dans ce cas un « copilote électronique ».

La fiabilité des systèmes est aussi un point d'étude majeur dans ce contexte. Comme vu en 2, la criticité des missions requiert une stabilité et une efficacité prouvées de la part de ces systèmes [Taylor and Reising, 1995].

3.1.4.3 Systèmes multi-agent pour multi-senseur

Plusieurs études se concentrent sur les systèmes multi-capteurs et élaborent des architectures à base d'agents pour satisfaire les exigences opérationnelles. Les travaux de [Pavón et al., 2007] se concentrent sur la gestion multi-capteurs par une architecture décentralisée basée sur le paradigme multi-agent. Les capteurs évoqués dans ces travaux sont des capteurs mono-fonction au fonctionnement simple (capteurs de température, détecteurs de mouvements, caméras de sécurités), en réseau ouvert répartis dans des infrastructures afin de détecter des intrusions dans une zone donnée. Les activités de ces capteurs mono-fonction demandent moins de traitements et engendrent moins de flux de données que les senseurs d'un SMS MultiFonction (MF) aéroporté. Les travaux de [Soh and Tsatsoulis, 2001] présentent une architecture multi-agent dans laquelle chaque agent est responsable d'un senseur de type radar au sol mono-fonction. Dans ce système, les agents négocient pour partager des ressources comme du temps radar, de l'énergie, des fréquences et doivent optimiser l'utilisation du réseau de senseurs pour détecter et poursuivre les objets détectés dans l'espace aérien. Puisque le réseau de radars est extensible et physiquement réparti, les problématiques de répartition et déplacement des objets sont étudiées et traitées par les agents du système. La contrainte temps-réel de l'application et la présence de plusieurs senseurs dans le système font de cet article une base intéressante pour l'étude des systèmes multi-senseur aéroportés. Cependant, la caractéristique mono-fonction sol-air ,l'absence de reconstruction de la planification en cas d'évènement et l'aspect physique réparti de l'ensemble des senseurs empêchent l'utilisation de cette architecture comme une base pour un SMSMF aéroporté, avec un temps réel dur et plus dynamique.

Les travaux de [Lesser et al., 2012] se basent sur la négociation entre agents pour partager et optimiser la charge des ressources système. Cette solution demande cependant des délais trop longs pour une application aéroportée temps-réel.

3.1.5 Frameworks SMA

Le paradigme multi-agent peut intervenir lors de plusieurs phases du développement d'un système. Ce paradigme comme décrit en 3.1.2.1 permet en premier lieu de modéliser et étudier les systèmes sous un nouvel angle. Une fois le système modélisé, des plateformes logicielles permettent de le mettre en œuvre plus rapidement. Ces plateformes sont aussi appelées framework multi-agent, puisqu'elles se résument souvent à un ensemble de paquets d'algorithmes développés dans un langage informatique et mis à disposition des développeurs de SMA. L'héritage de classes intégré dans Java permet aussi de faciliter grandement le prototypage et la mise en place des agents et de ses mécanismes.

3.1.5.1 Vue d'ensemble

Un framework est un ensemble de méthodes, algorithmes, logiciels mis à disposition des développeurs afin de réduire le temps de développement de leurs logiciels. Un framework multi-agent suit cette définition et permet aux développeurs désirant travailler avec des systèmes multi-agent de raccourcir les temps de développement en fournissant un ensemble de composants logiciels déjà implémentés. Ces composants incluent :

- Un système de messagerie, potentiellement conforme aux standards FIPA
- Un système d'annuaire pour indexer les adresses des agents
- Une architecture complète d'agent avec :
  - Un cycle de vie
  - Une mailbox
  - Des méthodes d'accès aux communications
  - Un interpréteur de langages symboliques de façon à créer des agents BDI
- Des méthodes de mises en réseaux en vue de répartir des SMA sur plusieurs machines
- Des logiciels de visualisation du fonctionnement de la plateforme et des échanges entre agents

Ces composants permettent aux développeurs de créer son premier SMA rapidement sans re-spécifier l'ensemble de la plateforme SMA. Un SMA minimaliste tel qu'un ping-pong (voir 3.1.1.1) peut être écrit en quelques dizaines de lignes de codes.

Plusieurs frameworks ont été développés dans différents langages : JADE (Java, FIPA-ACL), Jason (Java, basé sur le langage d'échange entre agents AgentSpeak [Rao, 1996]), SPADE(Python), MaDKit (Java, [Gutknecht and Ferber, 2001]), JACK (Java), Jiac(Java).

La prise en charge native des *threads* (fil d'exécution indépendant, programmation concurrente) fait du langage Java un langage très utilisé pour développer les SMA, les agents pouvant ainsi être implémentés par des *threads* et être exécutés en parallèle. De plus, ce langage est orienté objet, l'implémentation des composants du système est donc plus naturelle (objet mailbox, objet message, etc.).

3.1.5.2 JACK®

JACK® est un framework multi-agent développé par le groupe AOS, spécialisé dans les systèmes autonomes. Cette plateforme multi-agent est développée en Java et implémente une architecture BDI. Le code n'est pas ouvert et ne peut donc pas être modifié par le développeur. Utilisée dans de nombreux projets de recherches notamment dans des projets industriels [Gascueña and Fernández-Caballero, 2011, Merritts, 2013, Evertsz et al., 2004], cette plateforme est reconnue comme fiable et éprouvée [Winikoff, 2005]. L'ajout du paquet FIPA-JACK (plugin pour JACK) permet à la plateforme de respecter les standards FIPA.

3.1.5.3 Jiac V

Jiac V (Jiac 5), développé par le laboratoire DAI-Labor de l'université technique de Berlin, est un framework implémenté en java et destiné à encourager l'industrie à adopter le paradigme multi-agent. Les aspects de standardisation de développements, de sécurité, de gestion et d'extensibilité sont les principaux points d'intérêts des développeurs de Jiac V. Ce framework est bas-niveau et le code est ouvert aux développeurs de SMA. Les agents de ce framework peuvent être dotés d'un interpréteur JADL, un langage permettant d'implémenter les comportements agents, ainsi que d'un module BDI. Le langage BPMN, largement utilisé dans les métiers centrés sur les processus de gestions, permet de spécifier les interactions entre les différents acteurs d'un système lors des diverses phases de fonctionnement. Le BPMN peut ici être utilisé pour spécifier les connexions entre agents ou artefacts et permet de caractériser sous forme graphique l'ensemble des interactions d'un SMA.

3.2 Ordonnancement des ressources au sein d'une architecture SMS

3.2.1 Contraintes du contexte sur l'ordonnancement

La forte dynamique du contexte et la haute criticité des objectifs de missions décrites en 2.2.2.5 impliquent l'utilisation de techniques d'ordonnancement dynamique en temps-réel.

En vol, les besoins en actions senseurs à réaliser peuvent présenter des dynamiques très différentes en fonction de la mission et de la phase de celle-ci. Un exemple simple, si une mission de reconnaissance d'une ville est donnée à un RPAS, le nombre d'actions à réaliser pendant le déplacement et une fois que la plateforme est située au-dessus de la zone à reconnaître sont deux phases de la mission montrant des charges du SMS très différentes : faible pour la phase de déplacement vers le site et très chargé sur le site. Dans le même contexte, les priorités des actions senseurs à réaliser ne sont pas proportionnelles à la phase de la mission. La détection d'un objet dangereux pendant la phase de déplacement demande des actions très prioritaires immédiates alors que la phase de reconnaissance peut être à l'origine d'actions en quantité bien plus nombreuses, mais de priorités proches et moyennes. De façon générale, il peut être retenu que des actions fortement prioritaires et immédiates peuvent être réalisées à n'importe quel instant de la mission et indépendamment de la phase de celle-ci.

3.2.1.1 Actions prioritaires et immédiates

Aujourd'hui, les systèmes multi-senseur n'existant pas sous la forme de réseaux répartis, et dans laquelle l'information peut se propager de façon isotrope, cette nécessité de déclencher des actions senseurs prioritaires et immédiates est mise en œuvre par des connexions dédiées entre les instruments impliqués dans de telles situations. L'action de leurrage après détection d'un objet menaçant est un bon exemple d'action prioritaire et immédiate.

Sur ces SMS le partage des ressources (ici, les senseurs) est alors impacté par des évènements inopinés pouvant ainsi invalider l'ensemble des actions senseurs ordonnancées avant l'évènement. Ceci signifie que l'ensemble de l'ordonnancement déterminé avant l'évènement doit être recalculé. Le délai de calcul d'un nouvel ordonnancement doit être le plus bas possible, afin de prendre en considération le plus rapidement possible les derniers évènements. Pendant ce laps de temps, les évènements observés et l'ordonnancement exécuté par les ressources ne sont plus en phase, mettant ainsi la plateforme dans une situation plus ou moins risquée en fonction de l'évènement encore non traité.

3.2.1.2 Ordonnancement local aux senseurs

Sur un SMS réel, une ressource peut être réservée en même temps par plusieurs actions senseurs. Les senseurs aujourd'hui embarqués à bord de plateformes aéroportées sont capables d'ordonnancer localement chacune des tâches en divisant le temps d'exécution de chaque tâche proportionnellement au taux d'occupation de la ressource [Winter, 2008]. Cet ordonnancement s'appelle le micro-ordonnancement. Par exemple, pour un senseur tel qu'un radar à balayage électronique chargé de réaliser trois tâches telles qu'une imagerie SAR, de la veille radar (*i.e.*recherche d'objets volants en face de la plateforme) et la poursuite d'un objet. Si l'ensemble de ces trois tâches occupent le senseur pendant une période de 1 minute, chaque tâche occupe la ressource pendant un certain temps. Dans ce cas, comme le SAR nécessite 80% de la ressource et la poursuite 15%, la tâche de veille devra se contenter des 5% restants. Le taux d'occupation est déterminé grâce à la compréhension du fonctionnement du système, aux équations de qualités déterminées par les sciences du traitement du signal. Si un senseur comme un radar à balayage électronique reçoit ces trois tâches, les algorithmes embarqués dans le senseur vont découper la minute de fonctionnement en 30 cycles d'exécutions de 2 secondes par exemple. Ensuite, un nombre de millisecondes proportionnel au taux d'occupation seront dédiées à l'exécution de chacune des tâches. Ici - par exemple, par cycles de 2s - 1600ms seront réservées à l'imagerie SAR, 300ms à la poursuite et 100ms à la veille radar. Certaines tâches, comme la capture d'une image SAR, peuvent être exécutées plus rapidement ou lentement en fonction du taux d'occupation de la ressource. Avec un taux d'occupation de 100% de la ressource, la tâche peut être terminée plus rapidement qu'à 80%, ce qui est un facteur majeur lors des choix de priorité pour les actions senseurs à réaliser.

3.2.1.3 Classification du problème d'ordonnancement

La phase d'ordonnancement des actions senseurs consiste à partager le temps de fonctionnement des neuf ressources du SMS afin de placer le plus de plans dans un intervalle de temps donné.

Le problème peut être considéré sous sa forme complète comme un problème d'ordonnancement sur m machines distinctes de N tâches pondérées par un coefficient de priorité w_j avec des liens de précédences entre tâches. L'expression 3.1 montre selon la classification de Graham [Graham et al., 1979, Błażewicz, 1986], reprise et développée dans [Herroelen et al., 1997], que le problème peut s'écrire dans sa forme **complète** comme un triplet $\alpha|\beta|\gamma$ tel que :

$$m, 1|pmtn{-}rep, gpr, \rho_j, \delta_j, M_j, S_{jk}| \sum w_j C_j \tag{3.1}$$

L'ordonnancement est du type Job-Shop à m machines représentant des ressources renouvelables ($\alpha_1 = m, \alpha_2 = 1$). Les j travaux réalisés par ces machines sont contraints par des dates de débuts et des dates de fins strictes ($\beta_3 = \rho_j$ et $\beta_5 = \delta_j$ correspondant aux *releases date* et *due date* des tâches. Les travaux doivent être exécutés dans un certain ordre, décrit par un ensemble de contraintes de groupes ($\beta_2 = gpr$ pour *Generalized Precedence Relations* ou relations de précédences généralisées) et doivent être réordonnancés si préemptés ($\beta_1 = pmtn{-}rep$). Chaque travail peut être réalisé uniquement par un sous-ensemble de machines spécifiques (M_j) et une machine peut-être réservée dans des quantités entières constantes et arbitraires par plusieurs travaux à la fois (β_6 non spécifié). Le critère d'optimisation $\gamma = \sum w_j C_j$ indique que l'ordonnanceur doit minimiser la somme des temps de complétion des travaux pondérés par leur poids w_j (priorité). Des ressources nécessitent un temps de préparation avant le début de chaque tâche (S_{jk}). Ce phénomène est présent pour des ressources comme les antennes radar, qui ont besoin d'un temps de changement de fréquence, d'orientation, ou des deux. En fonction de l'ordre d'exécution de deux tâches à la suite, ce temps de préparation peut varier de manière asymétrique soit $S_{jk} \neq S_{kj}$.

3.2.1.4 Problème d'ordonnancement simplifié

Le problème d'ordonnancement peut être simplifié sans altérer le fonctionnement du système. Dans le cas où nous considérons des ressources entières indivisibles ($\beta_6 = undiv$), la réservation de sous-parties de ressources par plusieurs tâches en même temps est rendue impossible et remplacée par l'exécution des tâches l'une après d'autre dans l'ordre déterminé par l'ordonnanceur et des priorités. Ceci permet de simplifier la problématique d'ordonnancement dans un premier temps en se consacrant uniquement à une réservation binaire des ressources.

Dans cette première version de l'ordonnancement les tâches ne seront pas préemptibles afin de simplifier la gestion de la file d'attente de l'ordonnanceur : β_1 sera non spécifié. Cette caractéristique impacte peu les algorithmes d'ordonnancement, mais influe sur la création de la file d'attente en entrée de l'ordonnanceur. Si un travail est déjà commencé, il ne sera pas réinséré dans la file d'attente et donc ne sera pas déplacé suite à un nouvel ordonnancement.

La troisième caractéristique simplifiée concerne les temps de préparation des ressources S_{jk}. Il est possible d'ignorer ces temps de préparation en le compensant sur la durée des tâches et en considérant que le temps de préparation de chaque tâche est inclus au début de cette dernière. Les temps de préparation étant faibles face aux durées de tâches, ils seront simplement ignorés et considérés fixes au début de chaque tâche.

Le problème d'ordonnancement dans sa forme simplifiée peut ainsi s'écrire comme un triplet $\alpha|\beta|\gamma$ tel que :

$$m, 1|gpr, \rho_j, \delta_j, M_j, undiv| \sum w_j C_j \tag{3.2}$$

3.2.2 Partage de ressources

3.2.2.1 Job-Shop

Un Job-Shop, ou atelier de fabrication, est un environnement dans lequel un certain nombre de machines doivent successivement appliquer des modifications à des pièces afin de les confectionner. Dans un contexte d'ordonnancement, les machines à allouer sont les machines de l'atelier et les tâches de confections à appliquer sur les pièces sont les travaux à ordonnancer. L'ordonnancement des travaux de chaque machine est un exercice NP-Difficile pouvant être plus ou moins complexe en fonction des contraintes de l'atelier. D'un type d'atelier à un autre, les contraintes s'appliquant aux machines, aux pièces et à l'ensemble de l'environnement peuvent fortement varier. Par exemple, un atelier au sein duquel les temps de transports entre chaque machine sont non-négligeables induit une contrainte forte sur l'ensemble de l'ordonnancement des machines. De même, si le temps de travail de chaque machine est négligeable face aux temps de transports des pièces, l'atelier est dans ce cas décrit comme un Flow-Shop. Si certaines machines sont différentes alors elles ne peuvent pas intervenir à n'importe quel moment de la confection des pièces. Un ordre d'exécution devra alors être respecté, induisant des contraintes de précédences entre les tâches à ordonnancer. Aussi, en fonction du type d'atelier, une pièce peut être modifiée une ou plusieurs fois par chaque machine, en cas d'échec de la modification par exemple ou dans le cas où de multiples itérations sont nécessaires. La figure 3.3 montre un exemple de job-job constitué de 6 machines.

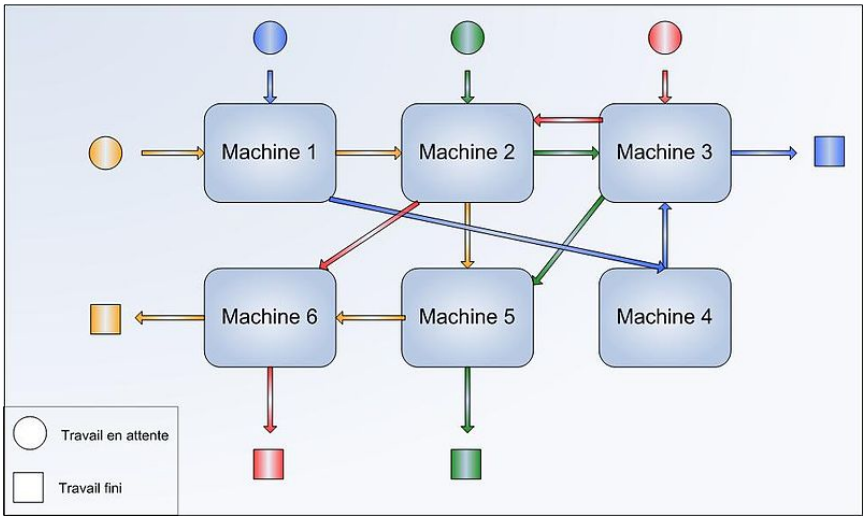


FIGURE 3.3 – Schéma d'un job-shop.

3.2.2.2 RCPSP

L'ordonnancement des actions senseurs peut être modélisé sous la forme d'un *Resource-Constrained Project Scheduling Problem* (RCPSP), problème de gestion de projet à contraintes de ressources, les senseurs sur lesquels les actions sont réalisées

correspondent, dans ce modèle, aux ressources du système.

Les recherches effectuées dans le domaine des RCPSP concernent habituellement des ordonnanceurs permettant d'organiser des personnes et des ressources matérielles utilisées lors de longs projets, au sein des ateliers ou pour des procédés industriels. Dans ce contexte, la contrainte temps-réel est rarement présente et les solutions recherchées sont approximatives - de haute qualité - grâce à des heuristiques d'ordonnancement efficaces ensuite complétées par des recherches locales de solutions, telles que des permutations de tâches.

La connaissance *a priori* du déroulement des tâches et une probabilité de réussite des tâches de 100% par les machines impliquent que l'ordonnancement déterminé au début de l'exécution sera certain puisqu'aucun échec d'exécution ou d'évènements extérieurs n'intervient dans ce cas sur le système. Dans la situation du SMS, les tâches n'ont pas une probabilité de réussite complète, certaines tâches devront donc être planifiées, ordonnancées et exécutées à nouveau en invalidant parfois l'ordonnancement construit jusqu'alors.

Les actions senseurs sont planifiées en cohérence avec les évènements du terrain et de la situation/attitude de la plateforme aéroportée. Ainsi, un retard d'exécution d'une action senseurs peut dans le meilleur des cas rendre l'action inutile et dans le pire des cas mettre en péril la mission. Un exemple classique : si une capture d'image est requise avant un virage de la PA et que celle-ci vient à être retardée par l'ordonnanceur, la capture de l'image ne pourra être terminée, rendant ainsi inexploitable la ressource pendant les nombreuses dernières secondes d'utilisation. Un retard de déclenchement des actions de survie est un exemple classique de mise en péril de la PA.

Un grand nombre d'instances réelles peuvent être décrites sous la forme d'un RCPSP. Ces instances peuvent aller d'une chaîne de confection de véhicules, un cas proche du problème original, à la modélisation du traitement des dossiers par une administration. Dans ce dernier cas, les agents administratifs responsables du traitement des dossiers correspondent aux machines effectuant les tâches.

La problématique d'ordonnancement des senseurs d'un SMS peut aussi être modélisée par un RCPSP. Les senseurs sont dans ce cas les machines et exécutent les différents plans senseurs. La spécificité de ce modèle de Job-Shop est le parallélisme des tâches des plans senseurs. Alors que dans un job-shop classique un objet est confectionné par une unique machine à tout instant, les plans senseurs du SMS peuvent parfois réserver plusieurs ressources en même temps et de manière synchrone. Cela équivaut à ce que plusieurs machines d'un job-shop puissent travailler à certains instants indépendamment et à d'autres instants, ensemble, sur un même objet.

3.2.2.3 Complexité du problème

Au regard de la théorie de la complexité calculatoire, le problème de RCPSP est un des problèmes d'optimisation combinatoire des plus difficiles à résoudre [Garey and Johnson, 1979]. Les RCPSP appartiennent à la classe des problèmes NP-Difficile au sens fort. La théorie de la complexité établit qu'un problème d'optimisation est NP-Difficile si la version décisionnelle lui correspondant est NP-Complet au sens fort. Dans [Garey and Johnson, 1975] Garey et Johnson montrent que la version décisionnelle des RCPSP sur une seule ressource et sans contrainte de précédence, appelé dans ce cas un problème d'ordonnancement limité en ressource, est NP-Complet au

sens fort par réduction du problème de 3-partition. Une observation simple faite par Blazewicz et col. [Blazewicz et al., 1983] laisse entrevoir dans [Uetz, 2001] des résultats bien plus pessimistes quant à la complexité de résolution des problèmes de RCPSP.

Ce niveau de complexité ainsi que la contrainte temps réel du problème amène à utiliser une solution seulement approchée de l'ordonnancement optimal par la mise en place d'heuristiques.

3.2.3 Techniques d'ordonnancement

L'ordonnancement est une étape majeure de nombreux procédés industriels. Différentes techniques d'ordonnancement peuvent prendre place aux cœurs des systèmes de manière répartie ou non en fonction des architectures matérielles de ces derniers.

3.2.3.1 Ordonnancement en ligne ou hors-ligne

Un ordonnancement en ligne permet d'ordonnancer les tâches à leur arrivée dans le système. Ainsi, les tâches à ordonnancer sont maintenues à jour et tout ou partie de l'ordonnancement est réalisé à chaque arrivée de tâche. Un ordonnancement en ligne peut être simulé si un ordonnancement hors-ligne est à nouveau exécuté à chaque arrivée de tâche. Dans ce cas, il est nécessaire que l'ordonnancement soit assez rapide pour maintenir à jour la liste des tâches ordonnancées et en attente.

Une heuristique peut aussi être mise en place pour placer les tâches à exécuter dans l'ensemble des tâches déjà ordonnancées, par exemple en insérant les tâches dans les temps libres ou en reconstruisant une sous-partie de l'ordonnancement en insérant les nouvelles tâches.

3.2.3.2 Ordonnancement centralisé

Une solution d'ordonnancement directe est de réaliser un ordonnancement centralisé au sein d'un système. Un ordonnancement centralisé est réalisé par une seule entité logicielle matériellement non répartie et consiste à rassembler l'ensemble des tâches à ordonnancer pour l'ensemble des ressources du système sur tout ou partie de sa durée de fonctionnement. Les algorithmes d'ordonnancement centralisés permettent de suivre ce principe et implémentent l'ensemble des étapes nécessaires à une allocation efficace des ressources.

Dans le cas des RCPSP et Job-Shop, deux problèmes NP-Difficiles, pour des problèmes de temps de calculs seuls des ordonnancements à base d'heuristiques peuvent être utilisés en pratique.

3.2.3.3 Ordonnancement décentralisé

Par opposition à l'ordonnancement centralisé, l'ordonnancement décentralisé amène à coordonner l'ensemble des ressources d'un système de manière répartie sur l'architecture de ce dernier. En fonction du type d'architecture logicielle et matérielle, l'impact d'un ordonnancement décentralisé est plus ou moins fort. Dans le cas d'une architecture matérielle décentralisée, un ordonnancement lui aussi décentralisé permet une scalabilité et une robustesse du système. Par exemple, dans le cas d'une patrouille de robots mobiles, le fait qu'ils soient ordonnancés de façon décentralisée

permet de se dispenser d'un système d'ordonnancement central et unique (outre une potentielle redondance du système). Or, chaque système centralisé d'une architecture étant un point névralgique pouvant être atteint en cas de faute intentionnelle (ex. : attaque) ou non (ex. : panne), pouvoir remplacer un organe central d'un système par un équivalent décentralisé décharge l'architecture globale d'un point sensible stratégique.

De même, la décentralisation du processus permettant d'ordonnancer l'ensemble des robots d'un système permet aussi d'accroitre sa flexibilité et sa scalabilité.

Par exemple, dans une situation où un robot mobile évolue seul sur un terrain et rencontre un autre robot, le nouveau groupe ainsi formé ne nécessite pas un système leur permettant de se coordonner et peut s'organiser de façon autonome.

Cependant, si l'équivalent décentralisé d'un système amène robustesse, sécurité et flexibilité à l'architecture globale, il peut engendrer plus de besoins en communication et plus de temps de calcul pour aboutir à des solutions de même qualité qu'avec un ordonnancement centralisé.

3.2.3.4 Ordonnancement par négociation

Une architecture à base d'agent permet de bénéficier d'une possibilité d'argumentation afin non seulement de soutenir les décisions tactiques, mais aussi de négocier les ressources à utiliser. Cette approche a été étudiée dans [Lesser et al., 2012], un ouvrage étudiant les mécanismes de coopération entre agents pour partager des ressources au sein de systèmes multi-capteurs, un ensemble de capteurs plus simples que ceux du SMS étudié ici. Deux problèmes majeurs apparaissent par l'utilisation d'une telle méthode d'ordonnancement. Le premier problème est d'ordre temporel : une négociation est basée sur un protocole fondé sur un ensemble d'échanges entre deux ou plusieurs entités et peut impliquer un retard d'une durée fonction des délais de transmissions des échanges entre chaque acteur de la négociation. Il faut aussi considérer que le temps de négociation évolue en fonction du nombre d'acteurs impliqués : l'ajout d'un acteur supplémentaire dans une négociation peut en étendre les délais de façon non linéaire.

Deuxième problème majeur, un ordonnancement par négociation peut engendrer un manque de déterminisme du consensus sortant. Or, la criticité du contexte du SMS étudié nécessite que son fonctionnement soit d'un déterminisme accru. La base de fonctionnement du SMS ainsi que l'ensemble des systèmes le constituant doivent respecter un ensemble de règles opérationnelles définies. Insérer l'ensemble de ces règles dans un processus d'ordonnancement par négociation en ligne engendrerait une complexité et un indéterminisme global pouvant mener à des blocages lors du séquencement des actions senseurs.

Dans [Smith, 1980], R. G. Smith présente le *Contract Net Protocol*, un protocole de négociation permettant d'ordonnancer l'utilisation de capteurs répartis sur un réseau de machines étendu sur une vaste zone géographique (réseau Arpanet [Roberts, 1988]). L'augmentation du nombre d'agents négociants peut engendrer une explosion du nombre de messages circulant dans le réseau. L'agent peut se retrouver à passer plus de temps à générer et traiter la réception et l'envoi de messages que de temps à réellement traiter les données.

En conclusion l'ordonnancement et la planification par négociation engendrent trop de latence pour être placés au sein d'un SMS tel que présenté au chapitre 2.

3.2.3.5 Ordonnancement par MILP

Des méthodes d'ordonnancements optimaux existent et permettent de minimiser ou maximiser des critères d'ordonnancement de manière optimale. Dans cette catégorie, la méthode d'ordonnancement par MILP (*Mixed-Integer Linear Programming*, optimisation linéaire en nombres entiers) permet de trouver les temps de début de chaque tâche tel quel le critère d'optimisation est maximisé (ou minimisé). Un problème de RCPSP est mis sous la forme d'un MILP par la création des variables correspondant aux tâches à ordonnancer comme leurs contraintes de précédences, de dates limites (de début et de fin), la ressource cible ainsi que le taux d'occupation de la ressource. Des algorithmes tels que *branch & bound* permettent de résoudre les problèmes d'ordonnancement [Ignall and Schrage, 1965]. Dans [Koné et al., 2011], Koné et al. présentent une méthode de MILP basée sur les évènements. Les méthodes par MILP sont optimales, mais donne un résultat en un temps exponentiel, de l'ordre de plusieurs heures pour une instance de petite taille. Cette solution peut être utilisée à titre de comparaison dans l'étude des SMS, mais ne pourra ni être embarquée ni utilisée de manière opérationnelle.

3.2.3.6 Ordonnancement par heuristiques

Dans [Kolisch and Hartmann, 2006], Kolisch et Hartmann comparent les résultats expérimentaux d'un grand nombre d'heuristiques basées sur plusieurs techniques : recherches locales, permutations successives, insertion de tâches, algorithmes génétiques, modification des priorités, etc. Les algorithmes à base d'heuristiques permettent de s'approcher d'une solution optimale rapidement et nécessitent moins de puissance de calculs que les méthodes exactes. Les heuristiques permettent de disposer d'une méthode pouvant être embarquée et utilisée de manière opérationnelle. Les heuristiques basées sur les évènements présentent des résultats intéressants sur le plan de la complexité temporelle et de faibles latences [Artigues et al., 2013].

3.2.3.7 Synthèse des méthodes d'ordonnancement

Plusieurs méthodes d'ordonnancement permettent la résolution d'un problème du type *job-shop* auquel l'ordonnancement et l'optimisation des senseurs s'apparentent. Les contraintes du contexte imposent le choix d'un ordonnancement compatible des temps d'exécutions d'un système multi-senseur. L'ordonnancement par heuristique est alors la solution la plus adaptée au type d'architecture envisagée. Même si l'étape de projection des solutions sur architecture physique ne sera pas réalisée dans ce manuscrit, un ordonnancement à base d'heuristique est techniquement compatible avec le matériel aujourd'hui embarqué sur les plateformes aéroportées. Plusieurs mécanismes seront nécessaires afin d'adapter les heuristiques à l'architecture présentée en section 4.2. La section 4.3 détaillera l'ensemble de ces mécanismes et des heuristiques d'ordonnancement.

Conclusion

Ce chapitre a permis de présenter l'état de l'art des recherches concernant les deux grands thèmes abordés dans ce manuscrit : la conception d'architectures orientées

agent et l'optimisation des actions senseurs. Nous nous sommes intéressés dans une première section aux systèmes multi-agent dans leur ensemble. Les agents et les SMA ont été présentés et l'agentification a été présentée comme une étape essentielle de la conception d'une architecture orientée agents. Nous avons ensuite présenté les travaux réalisés en matière d'architectures orientées agents et plus précisément de certains travaux consacrés aux architectures multi-agent pour systèmes de capteurs. Nous avons montré que les agents ont auparavant été exploités dans des architectures (ex. : Oasis *Air Traffic Controler*) afin de construire ce qui peut être assimilé à une situation tactique. La thématique de l'ordonnancement a ensuite été abordée. Nous avons présenté l'ensemble des contraintes opérationnelles et industrielles à prendre en compte pour diriger les recherches d'un ordonnancement adapté à la problématique multi-senseur. Le problème d'ordonnancement a été classé selon la classification de Graham sous sa forme complète et approchée. Nous avons montré que la problématique d'ordonnancement du système de senseurs est NP-Difficile. Nous avons montré que la problématique d'ordonnancement des senseurs pouvait être assimilée à un problème de Job-Shop dans lequel les senseurs sont les machines. Les techniques d'ordonnancement telles que l'ordonnancement par négociations, par méthode du type *branch & bound* et par heuristiques ont été présentées. Nous avons montré que la méthode à base d'heuristiques est la plus adaptée à la complexité du problème et aux contraintes opérationnelles.

Chapitre 4

Contribution

4.1 Analyse des contraintes industrielles

4.1.1 Méthode d'analyse du système multi-senseur

Une architecture système doit correspondre au contexte d'utilisation dans lequel elle sera plongée et mise en œuvre. Ainsi, le premier objectif des travaux de recherche concernant les architectures de SMS a été orienté sur l'étude des missions et des activités durant lesquelles ces derniers sont utilisés. Ensuite, puisque les tests en situation réelle sont onéreux et complexes à mettre en place, un scénario de test a été construit. L'objectif du scénario de test est de rassembler l'ensemble des opérations réalisables par un SMS au cours d'une mission. Ce scénario permet aussi de disposer d'une base fixe pour dérouler les tests de plusieurs architectures existantes afin d'en comparer le comportement.

4.1.1.1 Étude des tâches réalisables par la plateforme

Afin de couvrir l'ensemble des tâches réalisables par un SMS, un travail de mise en situation de la plateforme doit être fait. En effet, afin d'identifier l'ensemble des éléments constituant la chaîne de commandement et de décision, il est nécessaire de correctement situer la plateforme et comprendre son environnement. Ainsi, la problématique étudiée met en situation une plateforme au cours d'une mission. Les termes plateformes et missions font chacun intervenir un grand nombre de notions et de contraintes qui ont un impact sur la composition de l'architecture. La plateforme est composée d'un ensemble de systèmes dont le système de senseurs, ici l'objet de l'étude.

Quant au terme « mission », il se réfère aux objectifs, à un environnement, à une période temporelle ainsi qu'aux acteurs et forces présentes sur le théâtre d'opérations.

De même, chaque système de la plateforme et chaque élément de la mission impacte plusieurs aspects du système étudié et ont donc chacun une influence sur l'architecture de ce dernier. Par exemple, les temps de réponses à des requêtes envoyées au SMS dépendent de la place du système sur la plateforme et dans son environnement. Si d'autres systèmes de la plateforme nécessitent un accès à certaines fonctions, l'architecture doit aussi être adaptée afin de pouvoir prendre en compte ces requêtes ainsi que l'ensemble de l'environnement du système.

Comme décrit plus haut, le terme «mission» implique des objectifs. Les *objectifs de mission* peuvent être divers et sont amenés à évoluer dans le temps. Face à un objectif particulier, la démarche opérationnelle peut varier en fonction de l'expérience

des opérateurs ou des pratiques des opérationnels aux commandes de la plateforme. Les objectifs de mission interviennent à un niveau supérieur des objectifs du SMS et constituent l'ensemble des entrées pour le système de senseurs. De manière simple, nous pouvons noter que les objectifs de mission décident des actions senseurs. Dans le cas d'un RPAS et de son SMS, un objectif de mission O tel que O = « rechercher l'objet A dans la zone Z», cet objectif amène à créer un grand nombre d'actions senseur se succédant dans un ordre précis. Plusieurs stratégies peuvent être mises en place pour trouver l'objet dans la zone et ce choix relève des opérateurs et de leurs pratiques.

En entrée du SMS sont envoyés des ordres correspondant à des objectifs opérationnels. Les objectifs opérationnels correspondent à l'ensemble des tâches que le SMS peut réaliser, par exemple «détecter un objet». Chacun de ces objectifs opérationnels peut être réalisé par au moins un senseur. Le choix du senseur devant réaliser l'objectif opérationnel induit un résultat d'une meilleure précision si le senseur est adapté à l'objectif.

La chaîne d'acquisition des informations concernant un objet du théâtre d'opérations observé par la plateforme est décrite par l'acronyme Détection, Reconnaissance, Identification, Localisation - Poursuite (DRIL-P). Chaque lettre de cet acronyme indique un objectif opérationnel pouvant être accompli pour chacun des objets du théâtre. D'autres objectifs opérationnels existent, permettant de répondre à des situations particulières, par exemple à des fins de communications. Le traitement des données des acteurs terrain est néanmoins d'une importance majeure pour les opérationnels et fait partie des tâches exécutées en grand nombre par le SMS. En effet, cette succession d'actions peut être réalisée pour chaque objet pendant toute la durée de la mission.

Le développement d'une nouvelle architecture senseur implique que cette dernière doit être *a minima* capable de résoudre les problématiques des architectures existantes tout en proposant des améliorations. Comparer les architectures est possible uniquement dans un contexte similaire afin de mesurer les avantages et inconvénients de chacune d'entre elles. Un scénario de test a été construit afin de référencer les objectifs de missions couramment réalisées par des plateformes aéroportées.

4.1.1.2 Définition du scénario de test

Les RPAS peuvent être utilisés lors de diverses missions et permettent d'accomplir plusieurs objectifs. Les objectifs de mission concernent l'ensemble de la plateforme, de la trajectoire aux actions senseurs nécessaires à la récupération des informations recherchées du théâtre. Les objectifs de mission ont été sélectionnés et organisés au sein d'un scénario les regroupant :

Les différents objectifs du scénario sont répartis en deux catégories : les objectifs principaux, établis en préparation de mission et les objectifs d'opportunités, nécessitant d'effectuer certaines actions non planifiées en préparation de mission. Un plan de vol est construit afin de satisfaire les objectifs préparés du mieux possible. Les objectifs d'autoprotection et de veille peuvent être considérés comme parties des objectifs principaux.

4.1.1.2.1 Communications avec la plateforme Les plateformes aéroportées agissent dans un contexte strict, au sein de missions parfois utiles pour en préparer d'autres. Chaque plateforme a un rôle à jouer et une place à occuper sur le terrain. Les décisions sont décidées par la chaîne de commandement et de contrôle (*Command and Control*, centre de commande et de contrôle (C2)) généralement située au sol, impliquant ainsi une nécessité de communiquer avec la plateforme pendant la durée de la mission. Cependant, les opérations pouvant prendre place sur des territoires hostiles, au sein desquels les moyens de communication sont inaccessibles, d'autres canaux doivent être mis en place pendant toute la durée de la mission (exemple : communications satellites, relais au sol, etc.).

Scénario phase I : Détection d'une menace du type radar de veille longue portée et prise de décision en conséquence [1]. Cette première phase prend place au début de la mission, alors que la chaîne de commandement n'a pas connaissance de la présence d'un radar de veille longue portée ennemi sur le théâtre. Les modes de veille des senseurs doivent permettre la détection du signal électromagnétique émis par le radar adversaire. La détection du radar implique une action de brouillage de la part du SMS. En temps normal cette détection implique aussi un changement de trajectoire de vol afin d'éviter la zone. Dans ce scénario, la mission prenant place à portée du radar de veille, la trajectoire de la plateforme ne sera pas modifiée et la plateforme se déplacera jusqu'au site de la phase II.

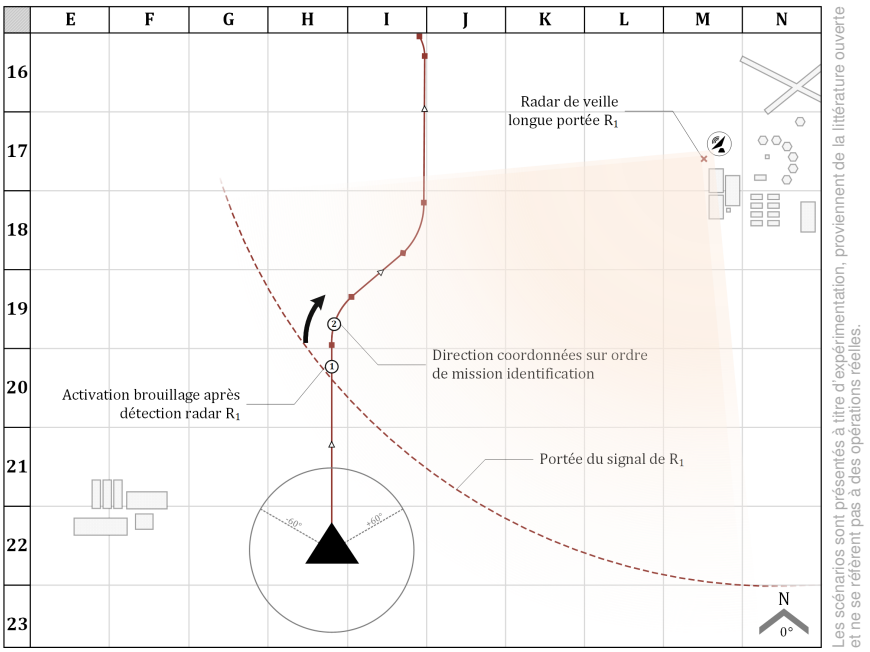


FIGURE 4.1 – Phase de brouillage après détection d'un radar de veille longue portée.

Scénario phase II : Identification d'un objet du type sol sur le théâtre.

Cette phase consiste à s'approcher de la position d'un objet, de récupérer des images et de renseigner la chaîne de commandement du type et de l'état de ce dernier. Cet

1. Les scénarios sont présentés à titre d'expérimentation, proviennent de la littérature ouverte et ne se réfèrent pas à des scénarios réels.

objet est un bâtiment potentiellement occupé par des forces adverses et une identification haute résolution est donc nécessaire afin de confirmer la présence d'éléments confirmant cette information. L'objectif d'identifier ce bâtiment est un objectif prioritaire établi en préparation de mission. Les coordonnées du bâtiment sont connues avec précision.

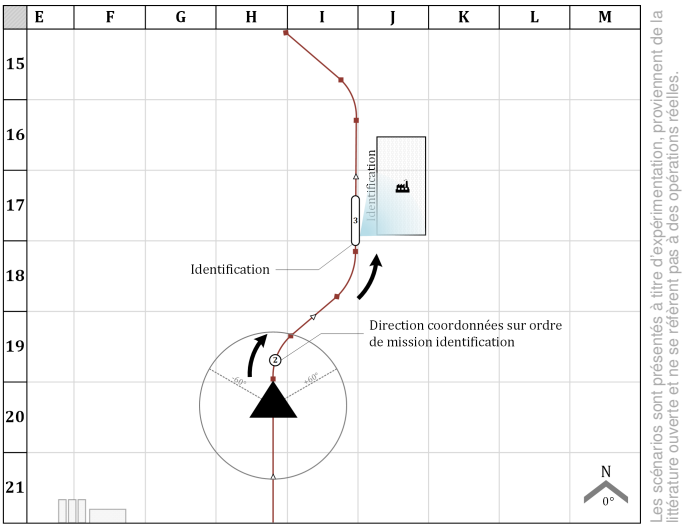


FIGURE 4.2 – Phase d'identification d'un objet au sol sur ordre du Gestionnaire de Mission (GM).

Scénario phase III : Identification d'un objet après détection de son signal radar [2]

Cette phase traite un objectif d'opportunité. Après la phase II, la plateforme se dirigeait vers le site de la phase IV en trajectoire rectiligne uniforme. Pendant le vol les actions de veille senseurs mènent à la détection d'émissions électromagnétiques émises par un bâtiment. La signature radar indique que le bâtiment est probablement adverse. Cette information est envoyée à la chaîne de commandement qui décide de recueillir un maximum d'informations concernant l'objet, la priorité de cet objectif d'opportunité est alors augmentée. Le plan de vol de la plateforme doit alors être modifié par le centre C2 afin de s'approcher à une distance permettant de réaliser une identification du bâtiment. Après identification les données et images relevées sont envoyées au centre C2 qui traitera la donnée comme il le souhaite (préparation mission ultérieure, modification des objectifs de mission, etc.).

Scénario phase IV : Recherche d'un objet mobile du type véhicule terrestre dans une zone.

Une phase de recherche d'objet est préparée en phase pré-mission. L'objet est mobile et évolue dans une zone vaste. Il est discret sur le spectre EM, il ne peut donc être détecté par des senseurs d'écoute passive. Il est désigné par l'acronyme TST, pour *Time Sensitive Target*, Cible de haute importance, l'objectif est le plus prioritaire de la mission. L'objet ayant une vitesse non nulle l'objectif doit être traité dans une fenêtre d'une précision relative à sa vitesse (un objet rapide ayant une fenêtre restreinte et un objet lent une fenêtre étendue).

2. Les scénarios sont présentés à titre d'expérimentation, proviennent de la littérature ouverte et ne se réfèrent pas à des scénarios réels.

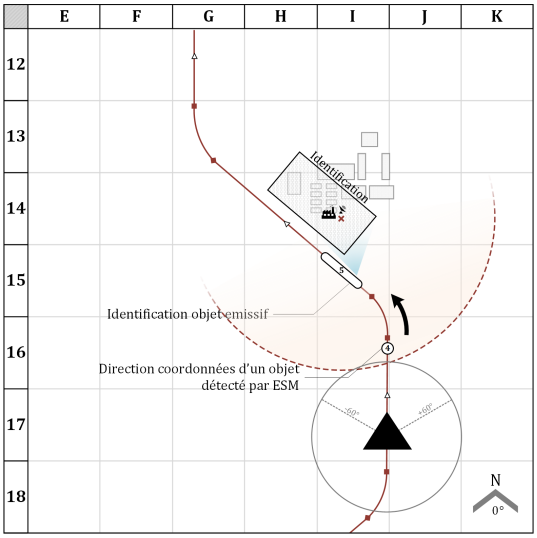


FIGURE 4.3 – Phase d'identification d'un objet émissif après détection.

La recherche d'un objet dans une zone demande une trajectoire adaptée afin que les senseurs embarqués sur la plateforme puissent accéder à chaque partie de la zone. Cette trajectoire est spécifiée dans le plan de vol de la plateforme défini par le centre C2. Le type de trajectoire habituellement suivie pour la recherche d'un objet est une trajectoire en forme de «S» successifs afin de couvrir un maximum de surface le plus rapidement possible donc en consommant aussi moins de carburant. La largeur d'un 'S' correspond à deux fois la portée moyenne des senseurs et la longueur à la longueur de la zone de recherches. Le 'S' est répété jusqu'à couvrir l'ensemble de la zone recherchée. Sur tout le plan de vol les senseurs sont actifs afin de rechercher efficacement l'objet. Les modes des senseurs sont en accord avec le type d'objet recherché. Il est inefficace d'utiliser un mode SAR pour détecter un véhicule en déplacement, le mode SAR étant réservé aux objets statiques.

Plusieurs objets peuvent être détectés pendant la phase de recherche. Les caractéristiques des objets détectés doivent être comparées avec celles de l'objet recherché. Deux procédés sont possibles : 1) l'identification est faite par le C2 après envoi des données, 2) l'identification est réalisée à bord de la plateforme et seul l'évènement de la détection est remonté au C2. Le premier procédé implique que les données permettant l'identification sont à bord de la plateforme et qu'il est possible d'effectuer le traitement des données puis la comparaison à son bord. À environ 60% de la trajectoire de recherche, l'objet est détecté par un des senseurs embarqués puis identifié par une des deux méthodes présentées précédemment. Les données de position et trajectoire de l'objet sont communiquées au centre C2.

Scénario phase V : En fonction de la menace que la TST représente le centre C2 peut décider de neutraliser le radar de conduite de tir ou l'objet mobile [3]. Un plan de vol spécifique ainsi que des actions du système d'arme sont délivrés à la plateforme par le C2. Après neutralisation du radar, des ordres senseurs spécifiques à la vérification sont envoyés au SMS afin de s'assurer de l'élimination de la menace.

3. Les scénarios sont présentés à titre d'expérimentation, proviennent de la littérature ouverte et ne se réfèrent pas à des scénarios réels.

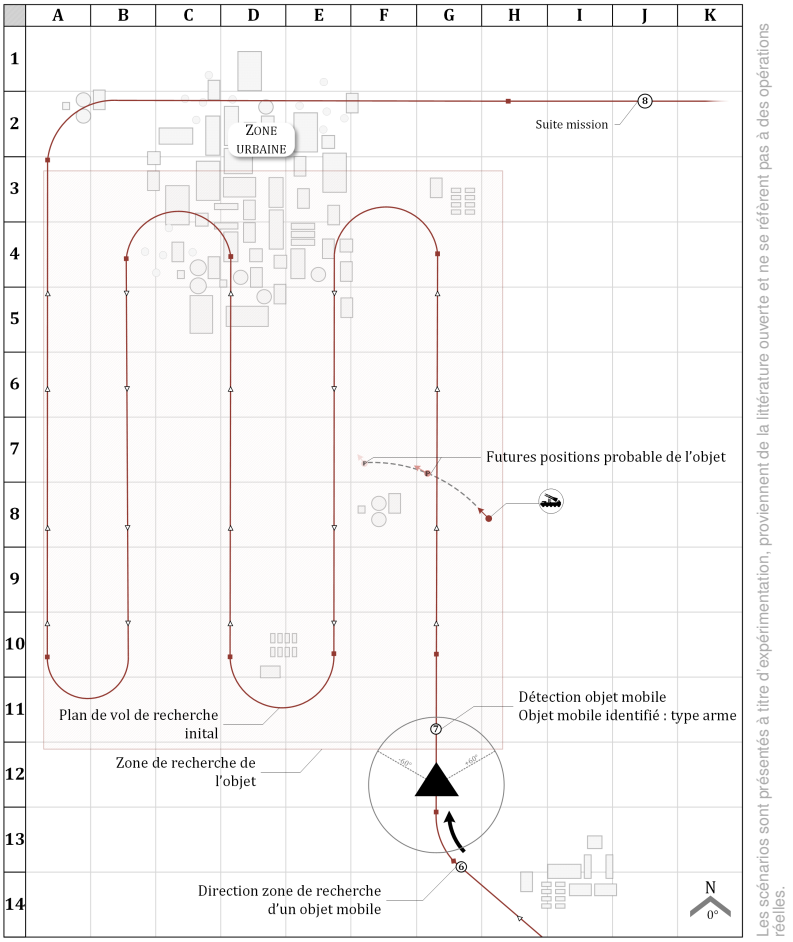


FIGURE 4.4 – Phase de recherche de l'objet mobile dans la zone.

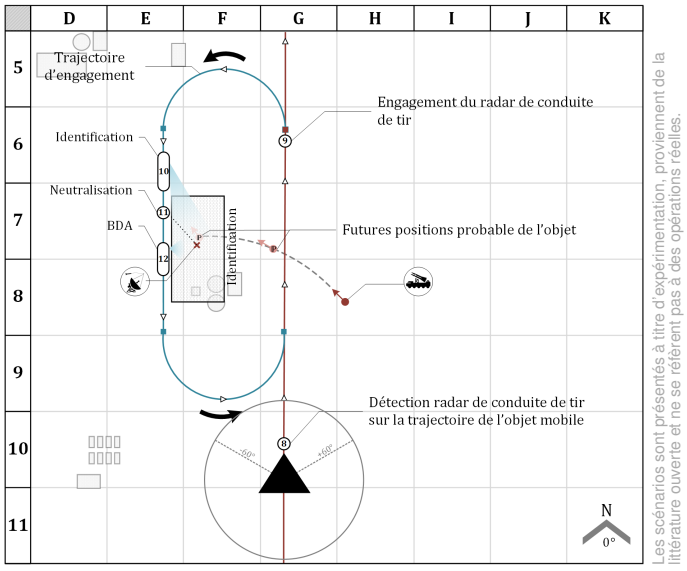


FIGURE 4.5 – Phase de neutralisation de la menace.

Scénario phase VI : Protection d'un convoi allié.

Cette dernière phase du scénario est une phase complexe mettant en œuvre plusieurs changements de trajectoires et un grand nombre d'actions senseurs différentes. L'objectif est de protéger un convoi allié se déplaçant dans une zone avec une trajectoire définie en avance. De la même façon que lors de la phase 4.1.1.2.1 une zone vaste doit être observée de façon à prévenir l'apparition de menaces sur la trajectoire du convoi à protéger, ce qui implique que la même zone doit être observée plusieurs fois pour constater d'éventuels changements. Le plan de vol adopté dans cette situation suit une trajectoire en hippodrome. Les dimensions de l'hippodrome doivent être adaptées à la portée des senseurs et à la redondance d'observations de la même zone (si une zone est particulièrement chargée d'opportunités pour un adversaire, elle devra être observée avec une fréquence élevée). Cette phase du scénario est schématisée sur la figure 4.7. Plus qu'une étape de mission, cette partie du scénario représente un type de mission particulier : une mission de type *Close Air Support*, appui aérien rapproché (CAS).

4.1.1.3 Exploitation du scénario

Le scénario [4] permet de disposer d'une référence en termes d'actions réalisables par la plateforme pendant toute la durée de l'étude d'une nouvelle architecture SMS. Néanmoins, d'autres sections de scénario ont été imaginées en fonction des nouvelles problématiques auxquelles les services d'ingénierie ont été confrontés pendant la durée de l'étude. Ces problématiques proviennent soit de demandes de clients soit des recherches internes visant à améliorer les systèmes.

Chaque phase de ce scénario permet d'évaluer la capacité d'exécution des tâches par la plateforme, qu'elle soit habitée ou non. Plusieurs aspects de l'architecture peuvent être étudiés à travers le scénario décrit plus haut et se résument à l'étude de l'impact de chacun des objectifs de mission sur l'architecture système. Plusieurs axes peuvent être étudiés pour estimer la faculté de traiter les objectifs par les systèmes, décrits en 4.1.2.1.

Plusieurs objectifs opérationnels orientés système de senseurs découlent de chacun de ces objectifs de mission (appelés *objectifs opérationnels* dans le reste du document).

Chaque objectif opérationnel est délivré en vue de recueillir des informations provenant du terrain. Plusieurs objectifs opérationnels fréquemment utilisés peuvent être cités :

- La chaîne d'acquisition DRIL-P, pour *Détecter, Reconnaître, Identifier, Localiser et Poursuivre*, peut être appelée pour tout objet du terrain. Chacun des maillons de la chaîne (ex : *Reconnaître*) peut être exécuté indépendamment. D'autres objectifs propres aux objets peuvent être créés en fonction de la mission.
- Les objectifs opérationnels se rapportant au théâtre. Exemple : la veille. La veille consiste à activer plusieurs senseurs dans un mode de recherche de signaux et permet de renseigner la chaîne de commandement de l'existence ou de l'apparition de menaces sur le théâtre. En rappel, comme décrit en 2 la SiTac est une donnée prioritaire pour le commandement et la veille permet d'alimenter cette dernière en données.

4. Les scénarios sont présentés à titre d'expérimentation, proviennent de la littérature ouverte et ne se réfèrent pas à des scénarios réels.

- Les objectifs propres à la survie de la plateforme tels que l'autoprotection. En effet, bien que la plateforme soit non habitée, la survie de celle-ci est nécessaire à l'aboutissement de la mission.

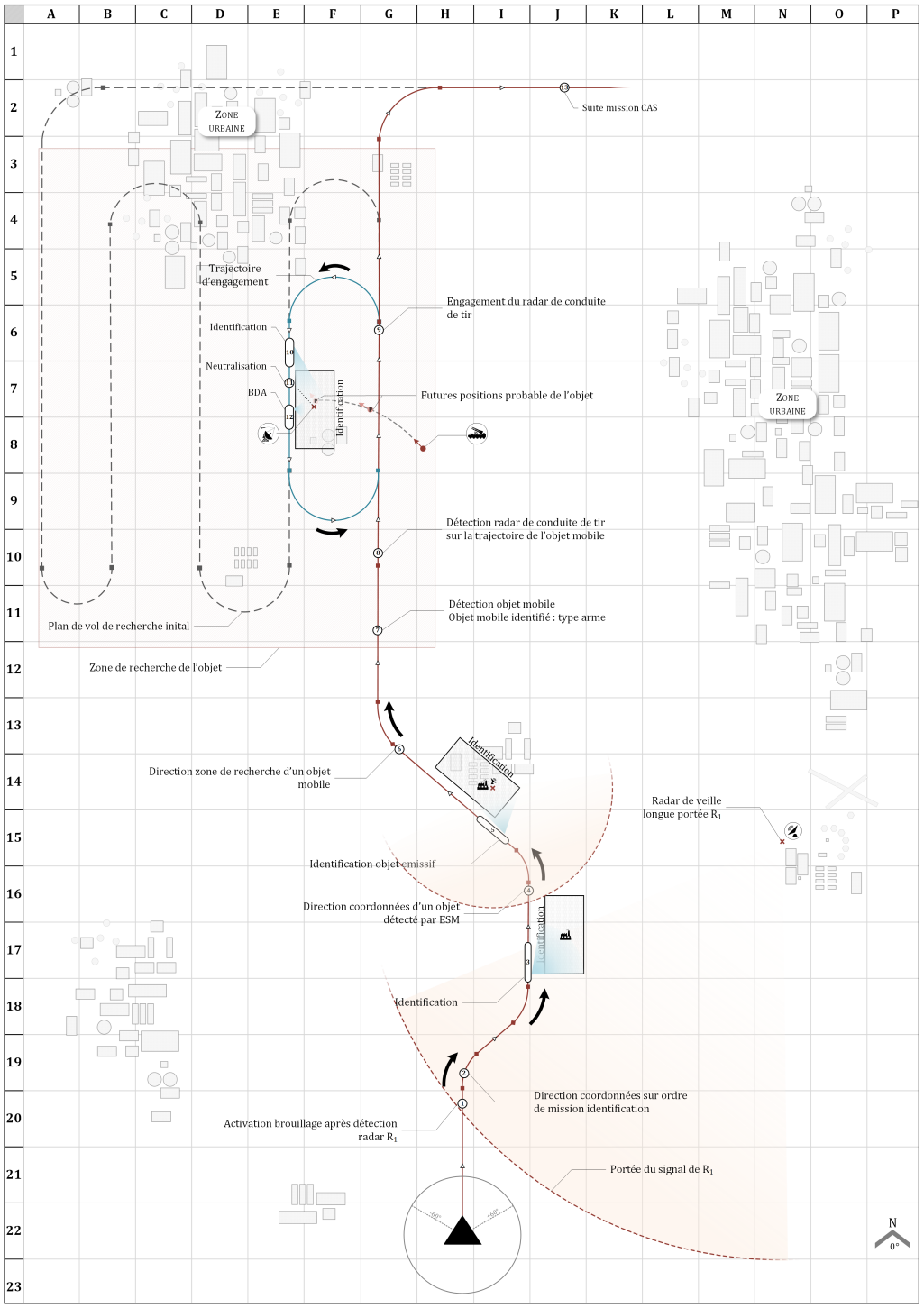


FIGURE 4.6 – Schéma du scénario représentant les 5 premières étapes.

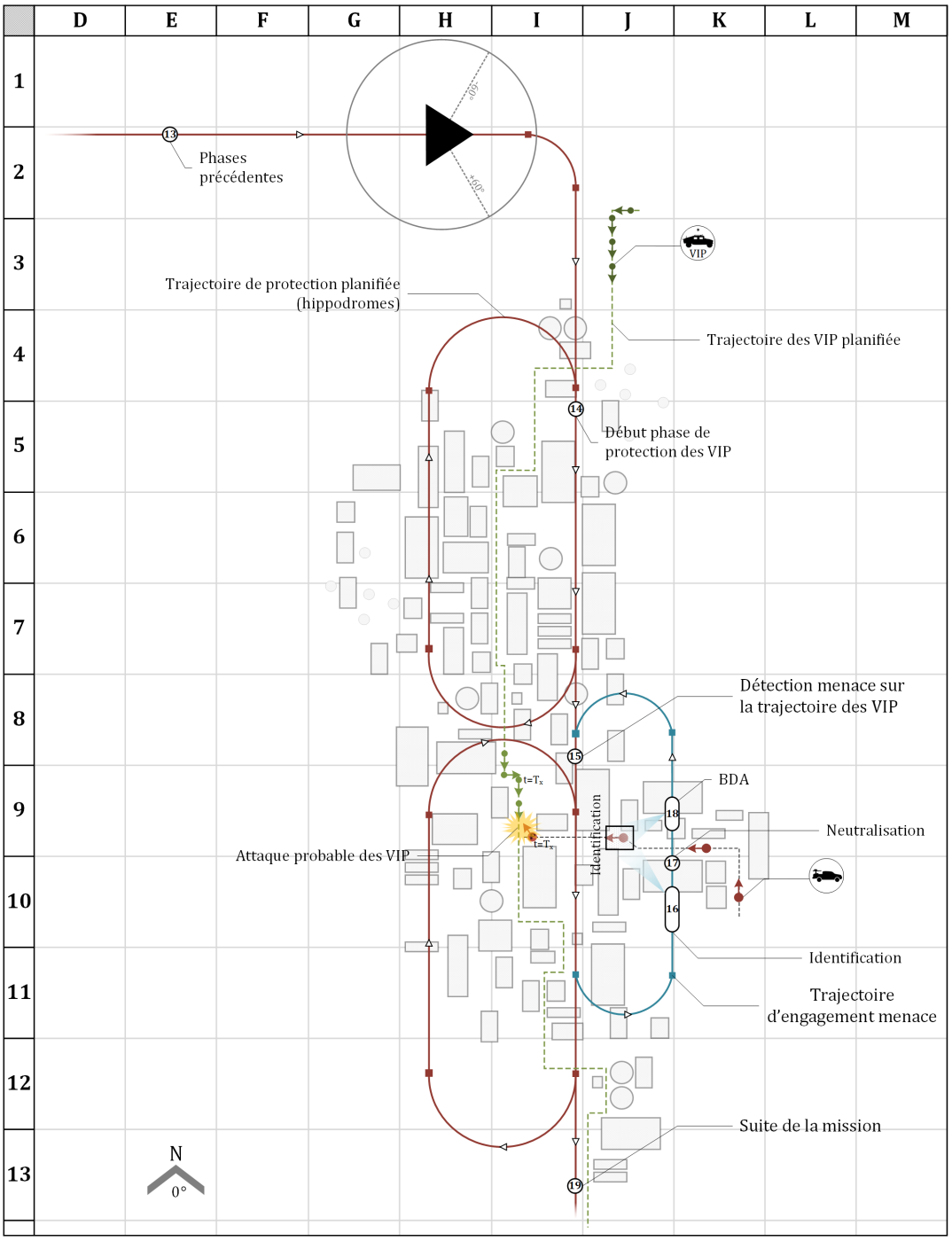


FIGURE 4.7 – Schéma du scénario de la mission CAS.

4.1.2 Étude du besoin : détermination des exigences du système

4.1.2.1 Critères d'appréciation d'une architecture système et méthodologie de conception

Il est nécessaire et inévitable de comprendre quels sont les critères d'appréciation d'une architecture système avant d'en réaliser l'exercice. À la différence d'une science exacte comme la physique ou les mathématiques, la construction d'une architecture

est difficilement quantifiable dans son ensemble. En effet il est absurde de confronter deux architectures construites à différentes époques et dans deux contextes distincts. Le terme *contexte* englobe, entre autres, les notions d'historique des développements : une architecture peut être conçue pour s'adapter à l'existant, la capacité de financement : le coût de réalisation l'architecture conçue, ainsi que l'avancement, la maturité et la maîtrise des technologies : une architecture système dépend fortement des technologies de stockage, de calculs et de communication dont elle peut ou non bénéficier à l'heure de sa création et si les concepteurs ont accès à cette technologie.

Bien sûr, l'élément essentiel permettant de juger l'architecture système est qu'elle doit être en phase avec les besoins. Un besoin non identifié lors de la phase de conception peut rendre l'architecture inefficace une fois implémentée.

La méthodologie de conception d'architectures systèmes Arcadia [Voirin, 2013, Voirin, 2017] a été utilisée pour concevoir l'architecture du SMS décrite en 4.2. Cette méthodologie basée sur le paradigme MBSE est construite en plusieurs niveaux d'analyse :

- L'analyse opérationnelle : identification des besoins de l'utilisateur
- L'analyse fonctionnelle : spécification des fonctionnalités du système en réponse aux besoins préalablement déterminés
- L'analyse logique : conception de l'architecture logique, cette phase décrit comment les fonctionnalités sont réalisées par le système
- L'analyse matérielle : caractérisation de l'architecture matérielle

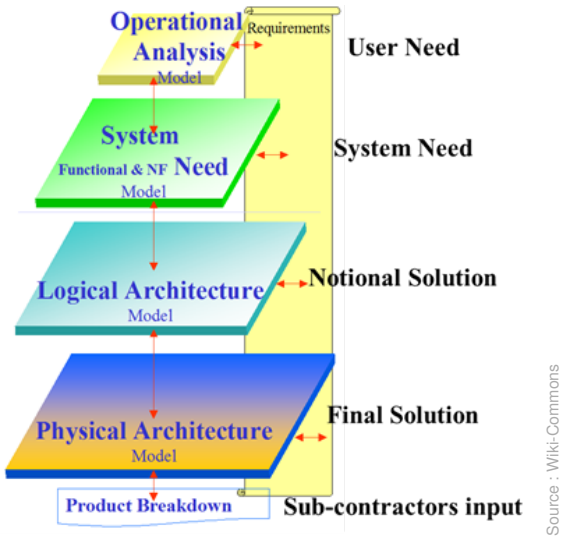


FIGURE 4.8 – Schéma des phases d'études d'une architecture de la méthodologie Arcadia.

Chaque entité engagée sur le développement d'une architecture d'un système complexe voit l'ensemble des spécifications sous un angle différent. Cet angle de vue est appelé un point de vue . Chacun des points de vue permet de vérifier ensuite si l'architecture conçue respecte les attentes spécifiées à chaque niveau des spécifications selon les critères de chaque métier.

La conception d'une architecture débute par la phase d'analyse fonctionnelle, phase essentielle pour comprendre le besoin de l'utilisateur final du système. Ensuite, l'ordre

d'exécution des tâches peut être différent en fonction des projets. En effet, dans un cadre technique très complexe comme celui de l'aéronautique les contraintes matérielles peuvent être fortement limitantes pour la solution conçue. Deux façons de procéder sont donc identifiées pour avancer dans la réflexion qui est de concevoir l'architecture finale : l'approche ascendante (bottom-up en anglais), consistant à identifier les contraintes matérielles et logiques afin de remonter ensuite vers l'étude fonctionnelle, et l'approche descendante (top-down en anglais), consistant à partir du besoin pour aller jusqu'aux spécifications matérielles. Dans un contexte peu contraint (en encombrement, énergie, ressources, puissance de calcul, etc.) la méthode descendante est privilégiée puisqu'elle permet d'éviter un raisonnement procédural complexe pouvant donner un résultat semblable. Cependant, dans un contexte comme celui de la conception d'une architecture SMS amenée à être embarquée à bord d'une plateforme aéroportée, les contraintes en termes d'encombrement, de ressources ou d'énergie ont un impact décisif. Dans ce cadre, l'étude ascendante permet d'identifier quelles sont les solutions possibles avant de concevoir des solutions qui ne seront pas réalistes.

Dans la pratique, une phase de conception ascendante est toujours réalisée, même si courte et implicite, afin de concevoir un système qui respecte des contraintes de ressources, par exemple le prix de la solution.

Un exemple : la tâche de conception d'une solution de services internet est essentiellement descendante, les contraintes d'espaces, de stockages et d'énergies sont ajustables à la demande. Plusieurs solutions seront proposées au client qui choisira en fonction du prix. Dans le cas de la conception d'un SMS aéroporté, ne pas prendre en compte les contraintes bas-niveau reviendrait à développer une solution basée sur des composants trop gros et trop gourmands en énergies, rendant la solution inexploitable.

Aussi, des solutions trop exigeantes au moment de la conception peuvent être imaginées s'il est constaté que certaines ressources évoluent à un rythme constant, par exemple la puissance de calcul ou le stockage de données. La conception d'un système complexe étant une tâche de longue durée, envisager que la puissance disponible sera très supérieure à la fin de ladite conception est possible.

Chacun des multiples points de vue possibles de l'architecture fait naître des critères d'appréciation différents. Une architecture peut donc être appréciée en fonction des différents points de vue que l'on peut lui appliquer. Une architecture conçue sur un fort besoin de robustesse et de résilience sera appréciée selon le point de vue de la sûreté de fonctionnement, mais non selon des points de vue pour laquelle elle n'a pas été conçue, comme la vitesse de traitement des demandes par exemple.

La première phase de conception d'une nouvelle architecture passe donc par le référencement et la hiérarchisation des points de vue pouvant avoir un impact sur la solution conçue.

4.1.2.2 Étude de l'impact des contraintes opérationnelles sur le système

La conception de l'architecture de la future génération de SMS s'inscrit dans un contexte technico-opérationnel complexe. La phase d'identification des besoins et des points de vue architecturaux a été réalisée tout au long de la construction du scénario.

Le contexte opérationnel demande la vérification de plusieurs caractéristiques essentielles à l'acceptation de l'architecture comme solution pouvant servir en situation réelle.

Besoin 1 : Prédictibilité des résultats

L'architecture doit bénéficier d'une forte prédictibilité et reproductibilité des résultats en fonctionnement. Les opérationnels exploitant une telle architecture s'attendent à un comportement connu et bien identifié. Le recours à des algorithmes statistiques pour lesquels les résultats sont incertains est inacceptable dans ce contexte. Ce besoin est notamment soutenu par la règle 2.

Besoin 2 : Fonctionnement par règles

Le comportement global de l'architecture doit être défini par des règles. Les règles métier opérationnelles sont répertoriées sous forme de procédures et très bien connues et identifiées, afin d'être appliquées de manière fiable en situation réelle. Un système fonctionnant dans ce contexte doit aussi être défini par des règles, afin de correspondre du mieux possible à l'utilisation qui sera faite du système final. Un fonctionnement à base de règles permet au système d'être configuré par des opérationnels formés sur des sites non-accessibles aux industriels.

Besoin 3 : Flexibilité de fonctionnement Les règles doivent être modifiables et accessibles après livraison par le client ou l'utilisateur. La modification d'une règle de fonctionnement peut impacter plusieurs composants de l'architecture (logiciels et matériels). Un des objectifs est de réduire cet impact afin de rendre la modification des règles plus rapide et moins coûteuse. La modification d'une règle peut nécessiter l'intervention de l'industriel, concepteur du système.

Besoin 4 : Modularité Les appareils sur lesquels les instances de l'architecture prendront vie n'embarquent pas en permanence les mêmes systèmes à leur bord. En fonction des objectifs et de la mission, des senseurs peuvent y être embarqués et d'autres enlevés. Ceci implique que l'architecture finale doit bénéficier d'une certaine modularité, afin de s'adapter à différentes configurations matérielles. Dans la pratique une architecture ne peut accepter sans pertes de fonctionnalités un composant qui n'a pas été conçu pour être intégrée. L'interfaçage matériel et logiciel du composant est une première barrière, la reconnaissance de ce composant et la compréhension de ses capacités par le reste du système constituent une deuxième barrière, plus contraignante. La modularité d'une architecture est donc rendue possible par le chargement de modules en fonction des composants à y intégrer ainsi qu'une quantité de modifications limitées des systèmes restant en place. Dans notre cas, l'ajout d'un senseur par exemple, demandera le chargement des modules lui permettant d'être reconnu et appréhendé par le reste du système tout en limitant les modifications à apporter aux composants autour du senseur.

Besoin 5 : Faible latence Le contexte fortement contraint et critique de l'utilisation d'un SMS implique que ce dernier réponde dans des délais de l'ordre de quelques millisecondes. Les utilisateurs et l'ensemble des opérationnels en contact avec de tels systèmes connaissent les limites des senseurs et souhaitent principalement que

le système réponde aux règles fixées dans des délais attendus. Cette contrainte expliquera les choix faits concernant l'ordonnancement des actions senseurs, présentés plus loin dans le manuscrit.

Besoin 6 : Exploitation du plan de vol La portée limitée des senseurs signifie que la plateforme doit être déplacée régulièrement pour recueillir les informations présentes sur le théâtre. Cependant, le système multi-senseur n'est pas maître de la trajectoire. Le domaine de la décision des trajectoires et plans de vol est une discipline spécifique et elle aussi, fortement contrainte. L'utilisation de senseurs comme un radar en mode SAR ne contraint pas la trajectoire uniquement en position, mais aussi en vitesse, continuité (une rupture dans la trajectoire corrompt les données recueillies), angle de l'acquisition avec la zone (une image SAR fonctionnant par effet Doppler, elle ne peut être capturée face à la zone, où la vitesse angulaire est nulle) et altitude [Brown and Porcello, 1969]. Ce besoin implique que la trajectoire de la plateforme est une entrée du système, sous la forme d'un plan de vol précis.

Besoin 7 : Robustesse et résilience

Un SMS embarqué sur une plateforme aéroportée nécessite une robustesse lui permettant de résister aux dégâts infligés à la plateforme sur laquelle il est situé. Dans un contexte opérationnel risqué, un tel système peut subir des dysfonctionnements de composants en vol, potentiellement intentionnels. Une façon de pallier ces pannes est de répartir l'ensemble du matériel en une architecture décentralisée. Une redistribution en fonctionnement des ressources supportant les processus permet au SMS de disposer d'une forte résilience et de mener la mission à son terme.

4.1.2.3 Étude de l'impact des besoins industriels sur le système

Besoin 8 : Décentralisation de la conception

De même que pour les opérationnels la décentralisation des systèmes peut avoir plusieurs avantages pour l'industriel. 1) La complexité des systèmes abordés dans ce type d'industries est élevée et nécessite donc un grand nombre de collaborateurs. Être capable de segmenter les architectures en autant de sous-systèmes bien spécifiés permet d'en répartir aussi la conception. 2) La possibilité de décentraliser un système intégré imposant et coûteux en N sous-systèmes de taille et prix moindres mène à un élargissement des volumes de productions des URL et donc aussi une baisse des coûts de production 3) Mise en place d'une maintenance plus simple, avec une capacité à détecter et diagnostiquer des pannes plus rapidement. Le diagnostique et la réparation d'une panne sur un système *intégré* (par opposition à *réparti*) est plus onéreuse en temps et en argent que l'identification d'une panne sur un des N sous-systèmes.

Des composants système décentralisés sont aussi plus simples à redonder, améliorant ainsi la maintenance et la détection d'anomalies ([Cornilleau et al., 2014]).

Besoin 9 : Modularité La modularité apporte aussi une souplesse concernant la gestion de configuration aux industriels. Qu'elle soit logicielle, matérielle ou les deux en même temps, charger des modules, des configurations, remplacer des composants matériels simplement permet une adaptation rapide des différentes architectures

afin de s'adapter à plusieurs types de plateformes. En outre, construire de nouvelles configurations rapidement permet à un industriel d'être capable de commercialiser plusieurs variantes d'une architecture et d'adapter les systèmes à la demande des clients.

Une forte modularité implique un taux de retravail faible (ou rework en anglais), qui ne nécessite donc pas de reprendre le développement des éléments anciens. Un faible taux de retravail implique que le composant embarque des mécanismes permettant un minimum d'adaptations aux protocoles et mécanismes existants.

Le niveau de modularité idéal est atteint lorsque les senseurs possèdent la capacité *plug & play* (connecter et utiliser en français), c'est-à-dire qu'ils peuvent être connectés rapidement à un système sans nécessiter une installation, pouvant aller du chargement de configurations sur les autres composants du système en place à la reprogrammation en profondeur de ceux-ci avec un taux de retravail pouvant être élevé.

Une capacité plug & play est possible lorsque plusieurs mécanismes et protocoles sont présents dans les composants du système concerné. Par exemple, considérons un système tel qu'un réseau d'ordinateurs échangeant des données de manière dynamique et que nous considérons un ordinateur à intégrer à ce réseau. L'ordinateur peut être vu comme un système plug & play si le réseau sur lequel il est intégré est capable de le reconnaître et l'identifier comme une machine capable de converser avec ses homologues. Dans ce cas précis la capacité plug & play est rendue possible par plusieurs mécanismes (attribution dynamique d'adresse, ensemble de services permettant l'échange de données diverses telles que des fichiers) et des protocoles, le langage commun à l'ensemble des machines (TCP, IP, HTTP, etc.). En fonction du système, les protocoles seuls ne permettent pas de rendre un composant plug & play : si le système est composé de systèmes travaillant en coopération il est nécessaire d'implémenter l'ensemble des protocoles de communications, des mécanismes ainsi que des couches applicatives afin que le composant soit capable de comprendre ses homologues et coopérer avec ces derniers.

En conclusion les industriels désirent améliorer la pérennité de la conception et se démarquer de la concurrence en proposant des solutions rapides à implémenter et sur-mesure en fonction de la demande client. Les avancées technologiques successives dirigent les composants des systèmes vers l'obsolescence jusqu'à leur remplacement. Les remplacements successifs de systèmes isolés composant une architecture sont possibles jusqu'à un point au-delà duquel le taux de retravail (repriseà) est trop important pour accueillir de nouveau systèmes et capacités.

Le paradigme multi-agent amène un ensemble de règles architecturales permettant de supporter chacun des points cités jusqu'ici et permettent d'intégrer les mécanismes nécessaires à l'élaboration d'architectures résilientes, décentralisées, modulaires, et fonctionnant à base de règles.

4.1.3 Étude de l'existant

4.1.3.1 Étude des SMS en fonction

Comme décrit en 3 les systèmes de senseurs embarqués aujourd'hui ne sont pas des SMS à proprement parler puisqu'ils ne tentent pas de fédérer l'ensemble des senseurs, de leur fonctionnement et de leurs activités. Ainsi, les senseurs embarqués

actuellement sont constitués de composant synthétisant l'information à plusieurs niveaux/échelles et avec une sémantique différente. Les senseurs ne sont pas interconnectés directement, mais par groupes, c'est-à-dire qu'une information détenue par un senseur ne peut être échangée directement avec tout autre senseur à bord de la plateforme. Les senseurs constituants les systèmes de guerre électronique, par exemple, sont tous connectés au même nœud.

La synchronisation de l'ensemble des senseurs est réalisée par la mise en place de liaisons dédiées entre chacun des senseurs nécessitant d'être synchronisé. Par exemple, des liens de communications entre certaines antennes et le radar existent, afin d'exclure mutuellement le fonctionnement des deux senseurs au même instant. Ceci signifie que l'utilisation d'un senseur en mode manuel par un pilote arrête le fonctionnement d'un autre.

4.1.3.2 Étude des architectures en phase de conception

Des recherches de nouvelles architectures sont aujourd'hui en cours afin d'asservir une partie des besoins exprimés en 4.1.2.2 et 4.1.2.3. Ces recherches visent à orienter les futures architectures vers les services (concept décrit plus loin dans le document). Une des principales tâches réalisée pendant l'étude de ces architectures a été de mettre au même niveau l'ensemble des senseurs et des ressources de calcul. Cette tâche permit de définir avec précision les besoins d'échanges entre chacun de ces composants.

4.2 Architecture pour systèmes multi-senseur multifonction

4.2.1 Étapes de conception

4.2.1.1 Étapes préliminaires

L'étude préliminaire consistait à saisir l'ensemble des composants au contact du SMS. Ces systèmes ont tous un rôle dans le fonctionnement du SMS.

Le gestionnaire de mission :

Le gestionnaire de mission est le système en amont du SMS. Il permet aux opérateurs de suivre la mission, de la coordonner, de contrôler l'ensemble des données provenant du terrain et de donner des ordres aux systèmes de la plateforme aéroportée.

La plateforme aéroportée représente l'ensemble des systèmes fonctionnant en parallèle du SMS. Plusieurs données provenant de la plateforme sont utilisées en mission par le SMS telles que :

- L'attitude en temps réel (3 vecteurs de 3 dimensions pour vitesse, accélération et position)
- La trajectoire future
- L'ensemble des constantes mesurées par la plateforme ayant un impact sur le fonctionnement des senseurs, tel que la température, humidité de l'air, etc.

- **Bases de données :**

Plusieurs bases de données sont présentes et exploitées à bord de la plateforme. Ces bases de données permettent de prendre des décisions en fonction des données recueillies sur l'environnement. Ces bases de données peuvent être divisées à bord de la plateforme en fonction des composants qui ont besoin d'y accéder. Ici de nouveau, les développements architecturaux successifs impliquent qu'il soit possible que les bases de données s'en trouvent segmentées et non accessibles aux autres composants. Ceci peut impliquer un retravail lourd et coûteux si un nouveau composant nécessite une partie ou la totalité des données non accessibles.

- **Les têtes senseurs et autres ressources :**

  Les têtes senseurs et leurs ressources sont à la fois une partie et l'aval du SMS. Le SMS est constitué d'un ensemble de composants dont les senseurs et ressources associées telles que les boîtiers de calculs, l'énergie électrique, les bus de communications. L'ensemble des composants du SMS autre que les senseurs sont présents afin d'exploiter ces derniers. En effet les ressources, les bases de données, les bus de communications, les composants embarquant les mécanismes de planification et d'ordonnancement permettent tous d'accompagner le fonctionnement des senseurs.

4.2.1.2 Paradigme agent et architectures

La génération des ordres senseurs par les objectifs de mission n'est pas à sens unique. Il va de soi que les objectifs de missions ont lieu d'être à la seule condition qu'au moins un senseur peut réaliser cet objectif. Il peut donc exister derrière un ou plusieurs objectifs de mission un unique ordre senseur bas niveau. Historiquement les senseurs étaient directement contrôlés par des experts, qui les manipulaient un à un. Cette tâche implique que l'opérateur connaît les ordres senseurs bas-niveau qui doivent être réalisés pour accomplir les objectifs de mission. Un moyen d'éliminer ce lien fort entre la prise de décision de niveau mission et de décider des actions senseurs est de créer un niveau de traitement intermédiaire. Ce niveau intermédiaire se substituerait à l'opérateur afin de le décharger des multiples actions nécessaires à la manipulation des senseurs. Plusieurs mécanismes peuvent permettre de fournir une telle interface aux senseurs.

4.2.1.3 Des ordres bas-niveau à l'architecture orientée *services*

Un service, terme pouvant aussi correspondre à une fonction délivrée par un senseur, permet d'abstraire les actions senseurs et de rendre leur fonctionnement transparent aux yeux de l'opérateur. La « couche services » fait office de niveau intermédiaire entre l'opérateur et les couches basses du SMS.

Ainsi, le SMS serait considéré comme autonome, acceptant à son entrée les ordres haut niveaux, les « demande de service », provenant de l'opérateur. Le système de senseurs embarqué ne nécessiterait dans ce cas plus d'ordres bas niveau mais des ordres plus haut niveaux, qui induisent un enchaînement d'ordres bas-niveaux. Les ordres bas-niveau fastidieux pour l'opérateur sont ainsi réduits et sa charge s'en trouve fortement diminuée.

4.2.1.4 Niveau de contrôles des systèmes

Plusieurs niveaux de contrôles interviennent dans le pilotage de tout instrument. Du contrôle bas-niveau, impliquant qu'un opérateur doit contrôler en temps-réel tout ou partie du système, au très haut niveau, réduisant les actions de pilotage du système au minimum. Dans cette dernière situation, le système a une plus haute autonomie, et doit avoir un comportement identifié par l'opérateur afin de répondre du mieux possible à ses attentes.

Plusieurs exemples de systèmes montrant cette évolution du niveau de contrôle peuvent être cités : l'éclairage public, les thermostats et les véhicules autonomes.

Un système de contrôle de la température thermostatique permet de remplacer une action directe d'allumage d'un système de chauffage - pouvant être effectuée par un opérateur - par un système électronique de régulation de la température plus ou moins complexe. Le fonctionnement d'un tel régulateur est régi par une ou plusieurs règles connues de l'opérateur, telles que la règle de déclenchement sous une température cible. Malgré la relative simplicité de ce système, certaines règles de fonctionnement peuvent être ignorées par l'opérateur, telles que la température d'hystérésis, pouvant impliquer une oscillation plus ou moins forte de la température régulée. Une règle de fonctionnement peut être omise par le constructeur s'il juge que l'opérateur ne nécessite pas de la connaître pour utiliser le système. Dans le cas de la règle d'hystérésis, il préfèrera annoncer dans la notice que la température est régulée à $\pm 0.2°C$ près pouvant rendre impossible l'utilisation du système pour des applications plus précises. En réalité cette hystérésis est présente pour empêcher d'éteindre et d'allumer le système de chauffage avec une fréquence trop élevée et ainsi créer des effets de résonances et d'instabilités.

Le mécanisme du thermostat conventionnel est simple et relève du domaine de l'automatisme, un circuit électronique constitué d'une dizaine de composants permet de parvenir au résultat attendu. Cependant, des versions beaucoup plus intelligentes pourraient être développées. Ainsi, un dispositif doté d'une caméra, d'algorithmes de reconnaissance d'images et d'un traitement cognitif permettrait d'observer les individus d'une pièce, de déterminer s'ils ont chaud ou froid en fonction de leur température corporelle, de leur façon de s'habiller, de leur attitude et de leur identité. Par apprentissage le système serait capable d'adapter de lui-même la température de la pièce. Ce dispositif pourrait être utile dans des situations où l'individu ne peut donner une consigne. Remplacer un thermostat conventionnel par un tel système serait généralement excessif, mais peut être justifié pour réguler la température d'une pièce pendant le sommeil de l'utilisateur ou encore pour réguler la température d'une chambre d'hôtel où les clients changent souvent et où une rapide adaptation est importante.

Une étape intermédiaire entre un thermostat simple et une version améliorée intelligente est une version de thermostat dans laquelle les coefficients du régulateur Proportionnel, Intégral, Dérivé (PID) régissant son comportement sont déterminés automatiquement en fonction des paramètres de chauffe de la pièce. Par exemple, lors de sa première installation le thermostat pourrait lancer une calibration des coefficients, permettant d'ajuster les temps de chauffe, de coupures anticipées et de la température de l'eau en fonction de la température de la pièce. Ainsi il serait capable de déterminer l'ensemble des paramètres de la pièce, dimensions et inertie thermique de manière agnostique. Cette approche a notamment été étudiée dans [Cong

and Liang, 2009].

Cette expérience de pensée concernant ce possible avenir des thermostats permet de relever plusieurs éléments. Un thermostat conventionnel peut être vu comme un système délivrant un service à l'utilisateur. En d'autres termes, dès que l'utilisateur a pour objectif de réguler la température de la pièce il allume le thermostat et saisi la consigne, il souhaite alors que le thermostat réalise ce service du mieux possible. Son équivalent cognitif n'a pas besoin de consigne mais d'un ensemble de connaissances qu'il doit avoir à sa disposition afin de se représenter la scène où il agit ainsi que les sujets qui y évoluent. Les connaissances peuvent être scindées en deux catégories : les connaissances natives, présentes à la création du système (ex. définitions d'un comportement, objectifs haut-niveau principaux) et les connaissances acquises pendant son fonctionnement (ex. profils des sujets, préférences utilisateurs). Si le premier type de thermostat est orienté service, le deuxième est lui *orienté objets* ou *orienté sujets*. Cette notion peut être étendue à l'ensemble des systèmes autonomes.

De la même façon, au sein du SMS, les têtes senseurs peuvent être contrôlées directement par l'opérateur ou par le biais d'intermédiaires. L'aspect multifonction de chaque senseur amène une complexité très forte lors de la création de l'architecture. Il y a plusieurs années les têtes senseurs étaient contrôlées exclusivement manuellement par un opérateur, cela correspond au plus bas niveau de contrôle des systèmes. L'électronique et l'informatique permirent par la suite de contrôler par des algorithmes les têtes senseurs afin de gérer les ordres bas-niveau. Le niveau supérieur d'autonomie nécessite des fonctions cognitives apportant une autonomie plus haute afin de remplacer une part des décisions fournies jusqu'ici par l'opérateur.

4.2.1.5 Méthode d'évaluation d'algorithmes cognitifs

Une méthode pour évaluer le besoin cognitif d'une tâche exécutée par un système est de le remplacer par une boîte noire dans laquelle nous plaçons un opérateur humain. Cet opérateur, qui remplace les traitements plus tard réalisés par des logiciels et tente de répondre aux entrées en fournissant les sorties que son homologue numérique doit aussi fournir. Chaque acte de pensée doit être consigné lors de l'exercice afin de récupérer la démarche cognitive réalisée par l'opérateur et d'identifier les données qu'il a à sa disposition pour prendre chaque décision. La démarche relevée permet ensuite de définir les règles et les algorithmes que doit embarquer le système.

4.2.1.6 Analogies

Un des points particuliers lors de l'étude de la suite de senseurs est la difficulté d'abstraire l'aspect complexité du fonctionnement de chacun des senseurs. En effet, chaque senseur possède son propre domaine d'étude théorique (électromagnétisme, optique, thermique, etc.). Plusieurs points ralentissent l'étude d'une architecture unifiée pour les SMS :

- La forte exigence des traitements et l'impact des contraintes, pouvant ralentir les recherches dès lors que les pistes envisagées sont gourmandes en puissance de calcul
- La diversité des senseurs nécessite une étude approfondie des différents systèmes de la plateforme et du fonctionnement de chacun des senseurs (principes physiques, traitements, etc.)
- La disparité des domaines physiques de fonctionnement

- Le contexte de leur utilisation est difficile d'accès car protégé par le secret militaire
- La difficulté à obtenir des données ralentit les délais de prises de décision techniques.

Des analogies fidèles permettent d'illustrer les idées et de mieux comprendre les concepts difficiles d'accès. Plusieurs analogies seront utilisées au cours du document afin d'illustrer les décisions prises lors de la construction de l'architecture.

Il est possible d'utiliser tout le savoir apporté ces dernières années par l'utilisation des senseurs à bord des plateformes aéroportées habitées afin d'identifier les différents rôles des opérateurs et systèmes lors de mises en situation réelles.

L'analogie du pilote et du navigateur

Un cas particulier peut être étudié pour mieux identifier comment peuvent être contrôlés les senseurs : une plateforme habitée d'un pilote et d'un navigateur place arrière. Le pilote a pour objectif de contrôler l'ensemble des paramètres de vol, l'attitude de la plateforme et la gestion de la trajectoire. L'opérateur en « place arrière », ou navigateur, est quant à lui chargé de la gestion des senseurs et de la décision des étapes à conduire pour mener la mission. Dans cette configuration il est intéressant de remarquer que l'opérateur des instruments peut être considéré comme faisant l'interface entre les senseurs et les objectifs haut-niveau décidés par la chaîne de commandement. Cette analogie sera appelée l'analogie de « Maverick » et « Goose » : « Maverick » étant le pilote et « Goose » le navigateur, voir figure 4.9.

L'étude de ce cas particulier en prenant en compte le principe d'équivalence homme-système présenté en 4.2.1.5 montre que Goose est dans la même situation que l'humain placé dans la boîte noire de l'expérience. En étudiant avec précision les processus cognitifs engagés dans les décisions de l'opérateur il est ainsi plus simple de définir les algorithmes chargés du contrôle des senseurs.

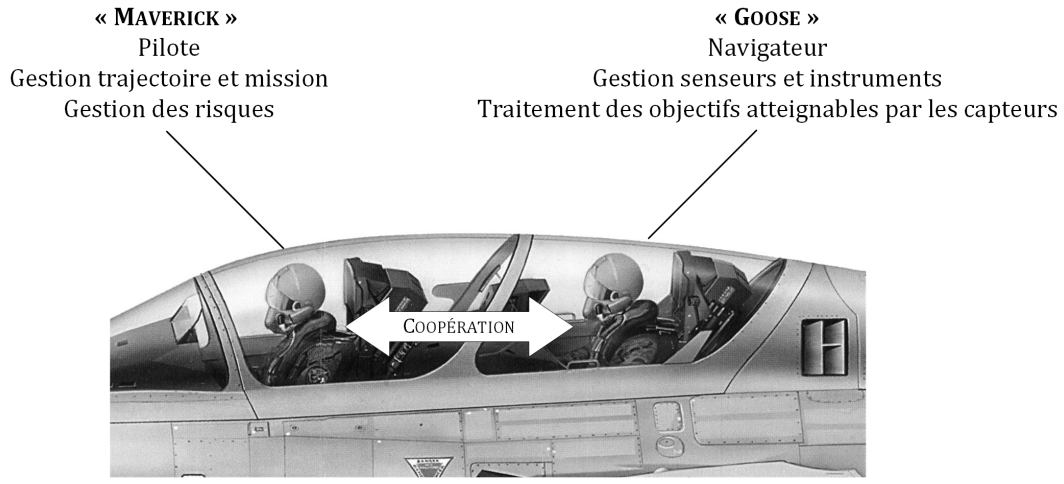


FIGURE 4.9 – Schéma de Maverick et Goose à bord d'une plateforme biplace.

Plusieurs éléments importants découlent de cette approche :

- Le navigateur et le pilote sont deux entités distinctes devant coopérer afin de conduire du mieux possible la mission.

- Ces deux entités disposent de plusieurs moyens de parvenir à un même but commun.
- Les dépendances senseurs/attitude, mission/senseurs et mission/attitudes impliquent que les deux opérateurs peuvent décider des actions à exécuter chacun leur tour avec une priorité dépendant de leur spécialité. (Exemples : manœuvre d'échappement par Maverick, décision de se diriger vers un objet afin de l'identifier par Goose)

L'analogie du sous-marin

Une autre analogie appropriée au système de senseurs dont nous disposons pour une plateforme du type RPAS est l'analogie avec un sous-marin. Malgré des contraintes de places et d'énergie très différentes, nous pouvons trouver à bord d'un sous-marin un ensemble de senseurs d'une complexité proche de ceux présents sur RPAS. La sortie de chaque senseur est exploitée par un opérateur expert du traitement des données de ce dernier. À la manière des senseurs du sous-marins les senseurs du RPAS sont aussi interfacés avec des experts du traitement de données. Ceux-ci se présentent sous la forme de boîtiers de traitements embarqués placés sur les flux de données sortant des senseurs. Le commandant de bord est lui, responsable des décisions opérationnelles, de trajectoire et de mission.

Nous appellerons cette deuxième analogie « l'analogie du sous-marin » tandis que la première sera nommée « analogie du pilote/navigateur ». Ces deux analogies permettront de mieux saisir le rôle de chaque composant et système à bord de la plateforme afin de construire l'agentification la plus efficace possible. La phase d'agentification sera définie et détaillée dans la section 4.2.2.

4.2.2 Agentification

4.2.2.1 Définition

L'agentification est l'étape de la conception dédiée à l'attribution des rôles et fonctions de chacun des systèmes à des entités logicielles intelligentes, ou agents intelligents, pouvant coexister en son sein. Une agentification efficace permet de segmenter un système en un nombre équitable de rôles et se doit d'éviter plusieurs maladresses pouvant porter atteinte au bon fonctionnement du système tout au long de sa vie :

- Un agent a une raison d'être au sein d'un système si et seulement si son rôle comporte une prise de décision (conformément à sa caractéristique proactive [Wooldridge and Jennings, 1995]). Un agent ne prenant pas de décision ne peut être régi que par des processus simples. Cela signifie que son rôle peut être accompli par un artefact et ne nécessite pas la création d'un agent (définition d'un artefact en 3.1.2).
- Un agent peut exister au sein d'un système si les données nécessaires à la réalisation de son rôle lui sont accessibles. C'est-à-dire qu'il doit être capable d'accéder à son environnement, de le percevoir et d'en tirer les informations nécessaires à sa prise de décisions. Cette caractéristique implique qu'un agent isolé de tout évènement et données provenant d'un environnement est inutile et encombrant au sein d'un système.
- L'ensemble des agents et de leurs rôles doivent être répartis équitablement et en cohérence au sein du système multi-agent qu'ils composent. Les décisions

de chacun de ces agents doivent être cohérentes et des mécanismes ou méthodes menant à un consensus commun doivent être présents au sein du SMA. Soit par la mise en place de mécanismes internes à chacun des agents, soit par la création d'agents et d'artefacts spécifiques.

- Les communications au sein du SMA doivent être étudiées et doivent être en phase avec la disposition des agents au sein de l'architecture système. L'impact des communications entre les agents sur l'architecture matérielle doit être judicieusement évalué afin de correspondre du mieux possible aux technologies accessibles à l'heure de la conception de l'architecture.

L'agentification est une étape de la conception pouvant être segmentée en plusieurs sous-phases :

- Identification des entrées-sorties du système et des données accessibles
- Étude des systèmes et sous-systèmes à agentifier et de leur portée
- Découverte des rôles des sous-systèmes
- Étude des capacités de communication au sein de l'architecture matérielle
- Positionnement des agents et attribution des rôles
- Construction des processus agents
- Identification des points bloquants nécessitant des mécanismes d'adaptations pour le consensus commun
- Projection de la solution sur plusieurs scénarios
- Vérification de l'extensibilité de l'architecture SMA choisie à des scénarios non-natifs

Chacune des phases est réalisée au regard des résultats de la phase précédente de manière itérative.

4.2.2.2 Agentification du système multi-senseur

La méthodologie décrite en 4.2.2.1 a été appliquée pour l'agentification du système multi-senseur. Chacune des phases va maintenant être détaillée :

Phase 1 : Identification des entrées-sorties du système et des données accessibles

La première phase de l'agentification, l'identification des entrées et sorties du système, permit d'identifier dans l'ensemble les interactions entre l'architecture et son environnement. Plusieurs des points introduits dans ce document ont été identifiés pendant cette phase :

- L'autorité et la priorité absolue de l'opérateur sur le système
- Les caractéristiques de la gestion de la trajectoire
- La caractéristique multifonction des senseurs
- Le caractère temps-réel du fonctionnement des senseurs
- La haute criticité de l'accès aux ressources nécessaires au fonctionnement des senseurs (bandes de fréquences)

Lors de cette phase, l'environnement de l'architecture a été identifié. Les résultats sont détaillés en 4.2.3.1

Phase 2 : Étude des systèmes et sous-systèmes à agentifier et de leur portée

L'étude des systèmes à bord de la plateforme permit d'établir quel type de senseurs et de matériel pourraient être embarqués à bord de la plateforme aéroportée. Le fait que le SMS pour plateforme aéroportée étudié soit en phase d'étude et présente une architecture système non décisive apporte une grande liberté à la construction de l'architecture. Seul un choix représentatif de senseurs à embarquer était déterminé. Le reste des composants de l'architecture a été déduit à partir des systèmes existants sur d'autres plateformes. La caractéristique centralisée ou décentralisée de l'architecture matérielle finale n'était pas non plus déterminée. Une potentielle architecture matérielle décentralisée était un critère prépondérant dans le reste de l'étude.

Phase 3 : Découverte des rôles des sous-systèmes

Chacun des rôles des systèmes et sous-systèmes a ensuite été identifié. La phase précédente ayant révélé une haute liberté en termes de sous-systèmes, un ensemble de rôles hérités de l'analyse opérationnelle ont été attribués à une ou plusieurs entités non identifiée(s) à ce stade de la conception. Le rôle de chacun des senseurs est facilement identifiable et se résume à créer une passerelle vers le monde physique.

L'autonomie du SMS demandée par les opérationnels implique l'apparition de nouveaux rôles au sein de l'architecture, auparavant attribués à des entités hors du SMS. Ainsi, cette autonomie demande de déporter une part de l'intelligence opérateur au sein des processus décisionnels du SMS. Cette caractéristique impactera l'ensemble de l'architecture système.

Phase 4 : Étude des capacités de communication au sein du système

L'identification des rôles de chacun des systèmes du SMS n'implique aucunement que ces derniers seront tous physiquement situés à bord de la plateforme. Ce sont les capacités de communications sol/RPAS ou PA-H/RPAS qui fixeront le positionnement de certains traitements. La capacité autonome du SMS peut nécessiter des traitements lourds donc des unités de calculs puissantes. En fonction du lien de communication avec le RPAS, il peut être possible et techniquement profitable de positionner ces unités de calculs au sol. Cependant, le contexte d'emploi des RPAS étant critique (potentiellement en discrétion radio) sur des théâtres difficilement accessibles en exploitant des communications faible débit et instables, il est impossible à l'heure actuelle de positionner les traitements lourds à l'extérieur de la plateforme tout en conservant un échange fort avec cette dernière.

Les communications à bord des plateformes ont grandement progressé ces dernières années, notamment avec une possible utilisation de la fibre optique. Les communications par fibre optique présentent plusieurs caractéristiques intéressantes pour les systèmes critiques, comme un débit théorique très élevé ou sa capacité à ne pas être influençable par les champs électromagnétiques externes (bruit ou actions intentionnelles). Le débit théorique accessible de cette technologie apporte un fort potentiel d'échanges entre les différents composants du SMS.

Phase 5 : Positionnement des agents et attribution des rôles

La première solution envisagée était de positionner un agent par senseur. Cette agentification a l'avantage d'être facilement appréhendable aux premiers abords. Cette solution permet d'imaginer chacun des senseurs comme étant une entité intelligente capable de communiquer, d'échanger des données et de coopérer. Cette version de l'agentification correspond très bien à la vision du « senseur intelligent » des industriels. En effet, cette agentification est en phase avec l'historique de développement des senseurs au sein de l'industrie. Il est facile d'imaginer chaque département expert d'un type de senseur développer chaque senseur et chaque agent qui lui correspond. Cependant, que cette vision plaise aux divers services d'une entreprise et qu'elle soit la plus simple conceptuellement ne signifie pas qu'elle est optimale sur le plan de l'autonomie, de la modularité/évolutivité, de la flexibilité et du cycle de vie du SMS.

Afin de comprendre si cette solution est viable sur le long terme dans un contexte dense industriellement et opérationnellement il est nécessaire de se projeter sur les phases 6, 7, 8 et 9 de l'agentification.

L'analogie du sous-marin peut être utilisée pour mieux comprendre l'impact de cette agentification :

Dans une situation traditionnelle, les senseurs du SMS peuvent être représentés par un duo senseur/expert dans le sous-marin. Le rôle décisionnaire est attribué au commandant de bord, voir figure 4.10. Attribuer un rôle décisionnaire à chacun des experts ainsi qu'au commandant de bord n'est pas envisageable puisque le résultat serait soit incertain et sans consensus pour l'ensemble des experts, soit forcé par le commandant, dupliquant ainsi le rôle de décisionnaire sans nécessité. Si les experts sont dépourvus d'un rôle de décision alors ils peuvent être considérés non comme des agents, mais comme des artefacts [Omicini et al., 2008].

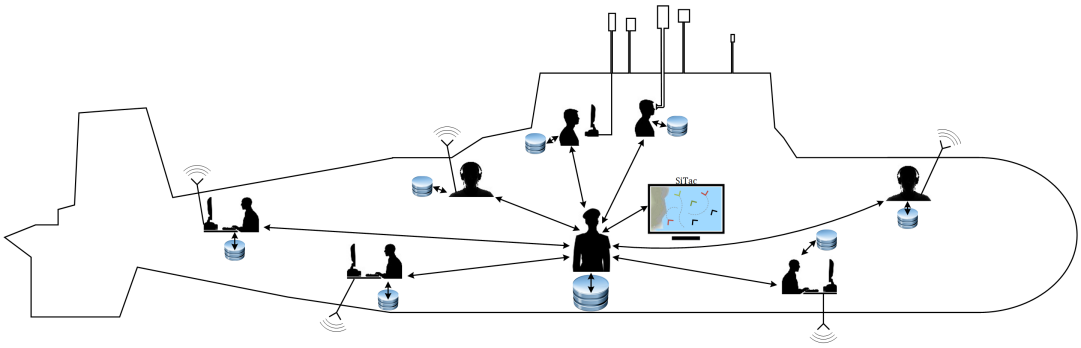


FIGURE 4.10 – Schéma d'un sous-marin et de ces opérateurs senseurs avec commandant de bord.

Dans cette analogie, le duo senseur/agent est représenté par un des opérateurs senseurs du sous-marin. Ce duo signifie que l'opérateur est capable de réfléchir sur la donnée détectée tout en maîtrisant le fonctionnement de son senseur. Il est donc possible d'attribuer à cet agent deux rôles : un rôle de gestionnaire de senseurs et un rôle de décisionnaire opérationnel.

Positionner un agent par senseur revient à diviser le rôle du commandant de bord en autant de décisionnaires que de senseurs. La décision est dans ce cas décentralisée. Les processus agents sont alors facilement identifiables : chaque senseur doit être capable de communiquer avec les autres afin de partager les informations provenant du théâtre, prendre de nouvelles décisions au regard de la SiTac, demander et

négocier des services d'autres senseurs, des accès à des ressources, etc. Les objectifs et buts de chaque agent correspondent aux objectifs opérationnels classiques définis en phase d'analyse opérationnelle : enrichir la SiTac, protéger la plateforme, suivre les objectifs de missions.

Ces nécessités de communiquer, échanger et négocier impliquent de disposer d'une base et d'un langage commun afin de converger vers un consensus avec les autres agents. La base commune de dialogue entre agents correspond aux objectifs de chacun des agents. Par exemple, un senseur ayant détecté un objet dialoguera avec un senseur pouvant compléter ou renforcer les données de cet objet.

Dans cette situation, l'analogie du sous-marin aide de nouveau à comprendre la portée d'une telle agentification. Il est possible de définir la nature des dialogues pouvant avoir lieu entre les experts senseurs présents dans le sous-marin en l'absence d'un commandant de bord. La coopération des experts nécessite un grand nombre d'échanges de manière aléatoire et asynchrone (un expert peut nécessiter un service de n'importe quel autre expert à tout moment). L'ensemble des experts senseurs représentent dans cette situation un commandant de bord réparti. Voir figure 4.11.

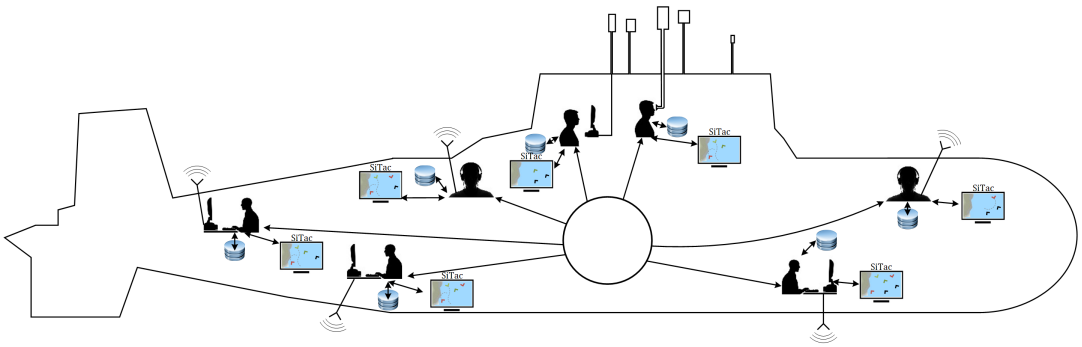


FIGURE 4.11 – Schéma d'un sous-marin et de ces opérateurs senseurs sans commandant de bord.

Étapes de transformation de l'architecture traditionnelle vers l'architecture proposée :

Une architecture SMS traditionnelle peut être vue comme un réseau de senseurs et de composants. Un tel réseau peut être considéré non isotrope (dans lequel l'information se propage différemment en fonction des directions) avec des accès aux données et aux URL de traitement disposées non équitablement, c'est-à-dire que tout composant du système ne peut accéder à toutes les URL. Une telle architecture peut être schématisée par la figure 4.12.

La première étape de transformation consiste à rendre l'ensemble des ressources accessibles les unes aux autres en passant par un point central. Cette transformation mène à une architecture centralisée au sein de laquelle la décision des actions senseurs est prise de manière centrale. Cette architecture peut être schématisée par la figure 4.13

La capacité plug & play escomptée des senseurs peut être atteinte par une modification de l'architecture précédente en architecture décentralisée avec chacun des senseurs responsables de son interface avec le réseau. La suppression du composant décisionnel central est expliquée par une incompatibilité et une redondance des

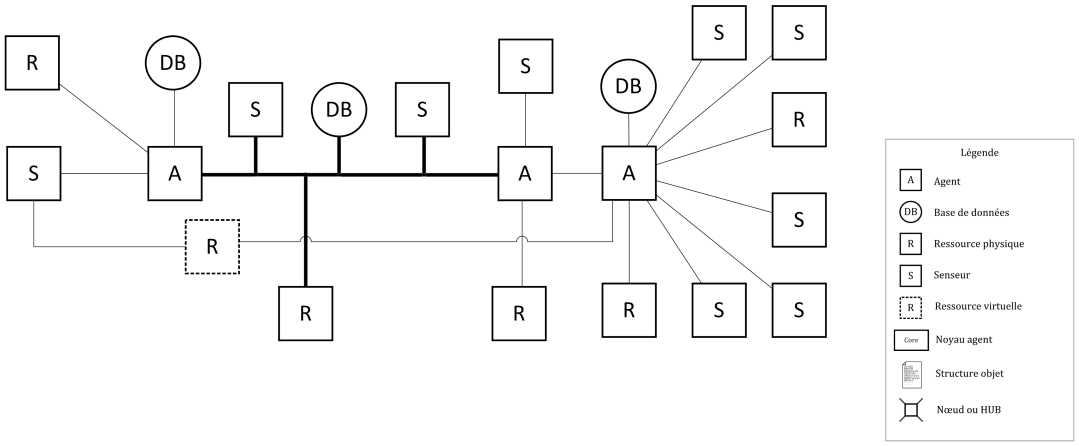


FIGURE 4.12 – Schéma d'une architecture actuelle.

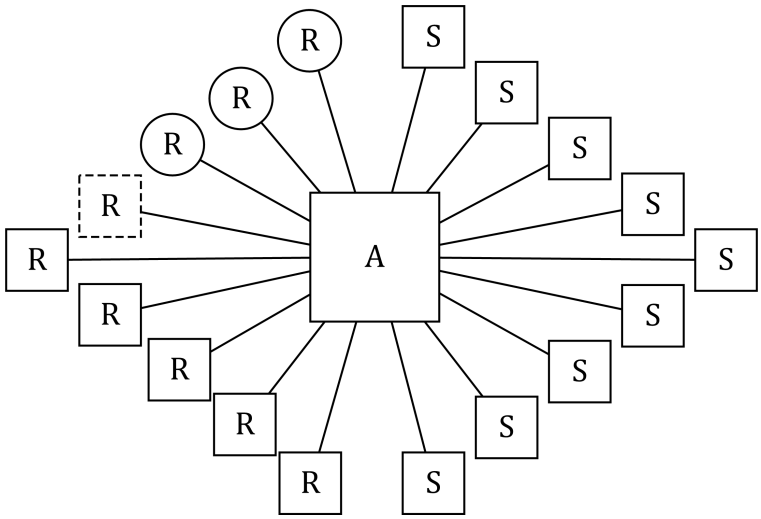


FIGURE 4.13 – Schéma d'une architecture centralisée équitable.

traitements réalisés par les senseurs dans cette configuration. Une représentation de cette architecture est visible sur la figure 4.14.

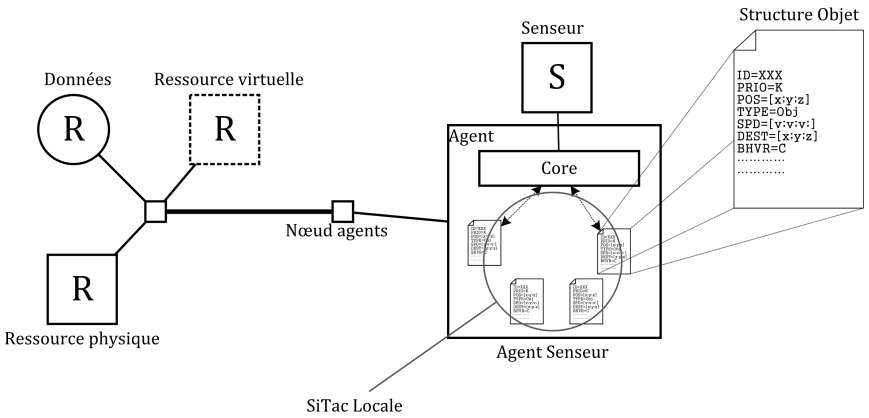


FIGURE 4.14 – Schéma d'une architecture décentralisée avec senseurs plug & play.

Chacun des senseurs maintient une représentation de la situation tactique en local.

Ceci peut se traduire par une représentation de chaque objet de la SiTac localement sous la forme de structures de données que nous appellerons ici une « fiche objet ». Un échange de données entre chacun des senseurs se traduit par un échange de fiches objets. Afin de rendre la coopération des senseurs possible, il est nécessaire de pondérer chacune des fiches objets d'un niveau de priorité, afin de refléter le niveau d'importance du traitement d'un objet par les senseurs. Chacun des senseurs souhaitant traiter un objet [5] tient à jour la fiche de l'objet correspondant. Lorsque le senseur n'est pas capable de traiter l'objet, il transmet la fiche à un senseur homologue afin qu'il la traite à sa place. Ce processus est schématisé sur la figure 4.15.

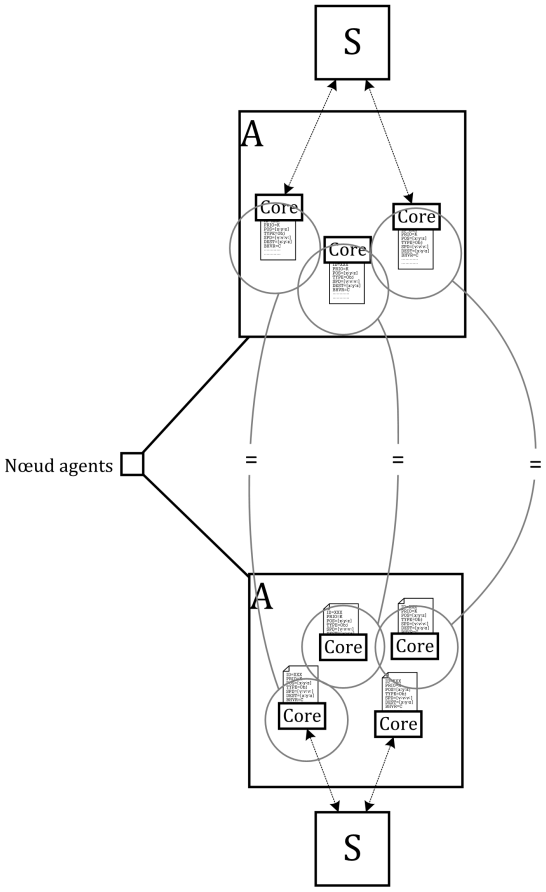


FIGURE 4.15 – Schéma de la coopération multi-senseur par transmission d'informations sur les objets.

Cependant, cette agentification amène un grand nombre de problématiques complexes à résoudre d'un point de vue ingénierie système nous ayant empêché son adoption. Ces problématiques sont engendrées par plusieurs faits :

- Une vision partielle locale à chaque senseur ;
- Un partage des ressources de calculs, de l'énergie et de ressources virtuelles nécessaires au fonctionnement des senseurs (par exemple : bandes de fréquences) ;
- Des échanges denses entre chaque senseur.

5. Traiter un objet signifie générer les actions senseurs nécessaires à l'accomplissement des objectifs opérationnels correspondant à l'objet, par exemple, traiter un objet « véhicule blindé» revient à générer les actions senseurs correspondant à la chaîne d'acquisition DRIL-P tandis que traiter un objet « missile» revient à générer les actions senseurs correspondant à l'objectif opérationnel de survie de la plateforme, donc aux actions de brouillages et de leurrage.

Provoquant les difficultés suivantes :

- Une duplication locale partielle de la SiTac sans vision globale ;
- Une expertise senseur dupliquée ;
- Un consensus entre les senseurs difficile ;
- Un ordonnancement difficile à positionner ;
- Une forte charge de messages sur le réseau ;
- Une probabilité élevée de dead-lock ou interblocage ;
- La priorité de traiter un objet est difficile à représenter si elle est établie par des senseurs distincts (SiTac locale).

Il est donc nécessaire de remodeler l'architecture afin de réduire au maximum les difficultés de conception.

Plusieurs constats sont à l'origine de la prochaine transformation de l'architecture :

- Chacune des décisions prises par un agent (senseur) est prise à partir des données connues d'un unique objet du théâtre ;
- Chaque collaboration de senseurs initiée provient de l'existence d'un unique objet ;
- La priorité de traiter un objet par un senseur doit être la même pour l'ensemble du réseau, donc établie pour la SiTac globale.

Une solution pour alléger les contraintes citées en 5 est d'extraire les fiches objets de chacun des senseurs et d'agentifier la fiche objet. En d'autres termes, la fiche objet détenue par un agent senseur est extraite, les données présentes dans une fiche objets sont attribuées à un agent et les processus des agents sont réalisés à partir de ces données et sur celles-ci. Cette transformation est schématisée sur la figure 4.16.

Cette transformation amène plusieurs changements généraux au sein de l'architecture :

- Chaque agent détient les données de l'objet décrit par la fiche objet précédemment décrite
- L'ensemble des agents constituent l'ensemble de la situation tactique, raison pour laquelle un tel agent sera appelé « Agent Tactique »
- Les senseurs, maintenant au rang de ressources, deviennent des artefacts obéissants aux ordres des agents
- La priorité de l'objet du théâtre est unique pour toute la situation tactique et correspond à une valeur entière qui sera appelée « priorité de l'agent ». Cette priorité peut être établie par un ensemble de règles connues des opérationnels.
- La fiche objet précédemment détenue par un agent senseur est désormais détenue par l'agent tactique. Ceci implique que les actions senseurs à réaliser sont décidées au regard des données détenues par les agents tactiques. Les agents tactiques décident et planifient les actions senseurs.

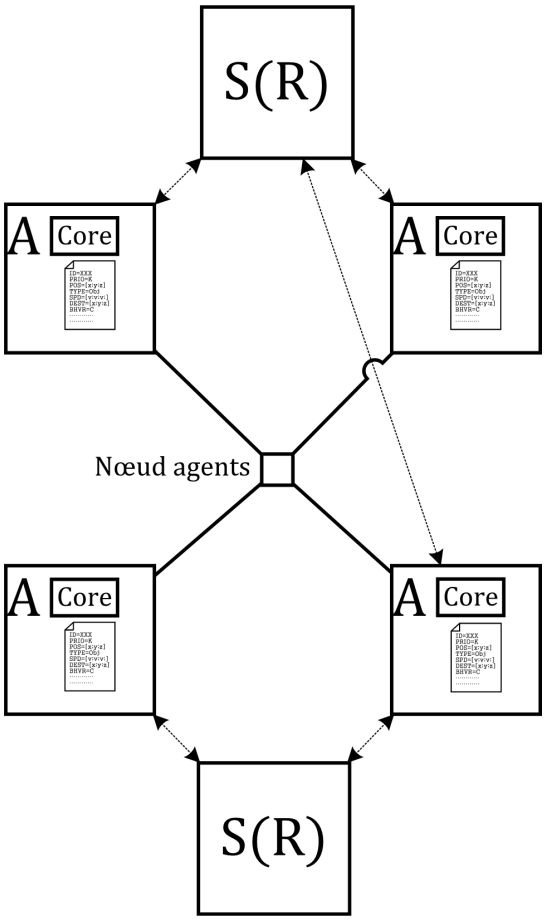


FIGURE 4.16 – Schéma de la transformation de l'architecture : extraction des fiches objets.

Une donnée détenue par un agent signifie qu'elle est accessible uniquement par ce dernier et qu'il est le seul capable de la modifier. Si un agent désire accéder à une donnée détenue par un autre agent il doit lui soumettre une requête et attendre sa réponse. Ceci implique une forte sécurité lors du traitement des données par un agent unique : empêcher des modifications intrusives par une entité extérieure à un agent permet une forte cohérence des données et des traitements définis lors de la programmation des agents. Un schéma de l'agentification finale est visible sur la figure 4.17.

Le concept d'agent tactique à un fort impact sur le SMS. Cette agentification implique plusieurs avantages du point de vue système. Chaque agent tactique reflète un objet du théâtre en temps-réel, l'agent tactique correspond donc au jumeau numérique d'un objet. Cette approche permet d'embarquer un grand nombre de traitements propre aux objets de manière répartie sur l'architecture. Le paradigme agent met en place une encapsulation des données et des processus les exploitant, facilitant grandement la mise en place de mécanismes de chargement de modules au sein des agents afin de leur apporter de nouvelles capacités. Le design et les processus de l'agent tactique seront détaillés en profondeur en section 4.2.2.3.

Phase 6 : Construction des processus agents

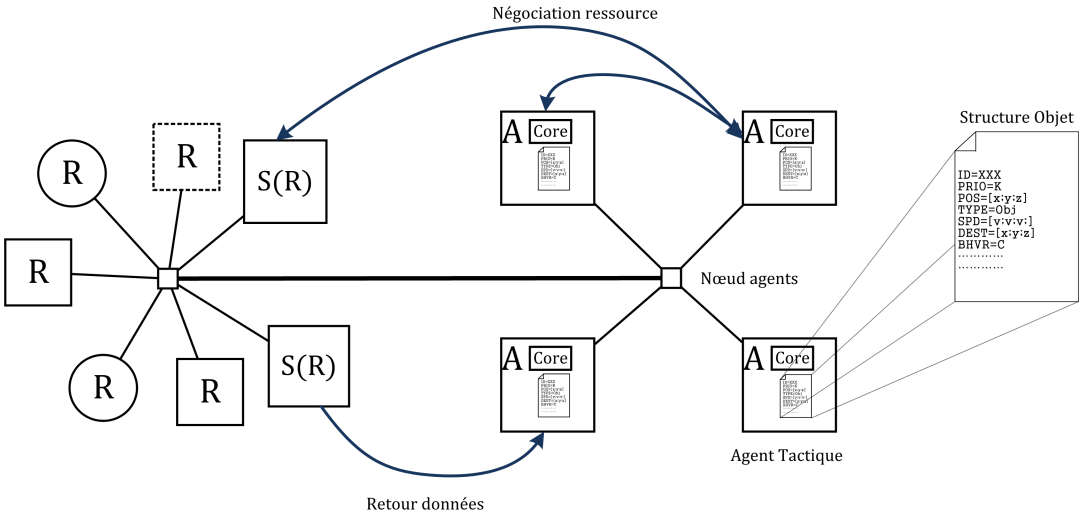


FIGURE 4.17 – Schéma de l'agentification finale.

La construction des processus agent est la deuxième phase la plus importante. Celle-ci détermine quelles tâches sont réalisées par les agents et comment. Ici, l'agent détient l'ensemble des données recueillies sur un unique objet du théâtre. Les processus agent permettront de déterminer les actions senseurs à réaliser.

Les données recueillies du théâtre par les senseurs sont partielles selon plusieurs critères :

- **Temporellement :**

  Une observation réalisée à un instant donné est vraie uniquement à cet instant et pour une durée dépendant de la caractéristique observée ainsi que du type de l'objet. Prenons par exemple l'observation d'un objet par un capteur radar en mode *Ground Moving Target Indicator*, indicateur d'objets mobiles au sol (GMTI). La donnée « Vitesse » de l'objet est vraie à l'instant précis de son observation et devient incertaine rapidement après celle-ci. Les autres caractéristiques de l'objet permettent de remettre cette information dans son contexte et d'établir un profil de probabilité des vitesses à court terme. De la même façon, une incertitude temporelle subsiste lorsqu'un objet est observé à intervalle régulier en fonction de la fréquence des observations. Selon le phénomène expliqué précédemment, un objet doit être observé avec une fréquence fonction de la variabilité de ses caractéristiques. Par exemple, l'observation d'un avion par un radar doit être plus régulière si ce dernier possède de fortes capacités aérodynamiques. La sortie de l'objet des domaines physiques et géographiques

- **Physiquement :**

  chacun des senseurs possède un domaine physique qui lui est propre. Par exemple, une caméra classique peut recevoir des signaux lumineux de longueurs d'ondes comprises entre 390 et 710 nm. Un senseur multifonction optronique pourra quant à lui accéder au domaine physique 390 nm et 710 nm ainsi qu'aux infrarouges moyens et lointains (8-12 et 3-5 μm). L'observation du théâtre sous le point de vue d'un unique senseur créé de la donnée partielle en recueillant des données objets elles aussi partielles. Par exemple, l'observation d'un théâtre par une caméra thermique ne renseignera une situation tactique que partiellement, en esquivant l'ensemble des objets n'émettant pas dans le

spectre du domaine physique de la caméra. Ou encore l'observation d'un volume par un radar ne permettra pas de renseigner sur l'existence d'objets ne réfléchissant pas les ondes électromagnétiques dans ce volume.

Un aspect probabiliste dû à l'imprécision des mesures vient s'ajouter au phénomène précédent. L'écho radar d'un objet, par exemple, est en effet bruité dû au bruit thermique des composants, du bruit provenant de l'environnement et des imprécisions des données prises en compte pour les calculs (infimes variations d'attitude, météo, etc.).

- **Géographiquement :**

  Comme énoncé en 2.1.2.1 aucun senseur n'est parfait. La portée d'un senseur est limitée et la mission ne prend pas place sur un terrain parfaitement plat. Avec un rayon de 6 378km, la courbure du globe terrestre impacte la mesure des senseurs. L'horizon visuel au niveau du sol est différent de l'horizon radioélectrique, impactant différemment la portée de chaque senseur ainsi que leurs modes. La perméabilité des reliefs aux signaux électromagnétiques et optiques crée des zones d'ombres pour les senseurs. Il en va de même avec la météo, pouvant impacter certains modes plus que d'autres.

  Les capacités des senseurs rendent impossible leur utilisation pour des tâches en particulier. Par exemple, un mode GMTI ne détectera pas un véhicule se déplaçant sous une certaine limite de vitesse, créant ainsi des discontinuités de pistage en fonction de la variation de l'attitude du véhicule.

L'ensemble de ces phénomènes créent des lacunes dans les données et une vision partielle du théâtre et des objets qui y évoluent. Aucun senseur ne permet de détecter et renseigner l'ensemble des caractéristiques de tous les objets présents sur le théâtre. C'est à cet instant précis que l'agent tactique intervient. Grâce aux connaissances acquises sur l'objet, l'agent tactique est capable de décider, à partir de règles opérationnelles définies avant la mission, comment compléter les connaissances manquantes. L'objectif commun à chacun des agents tactiques est donc de compléter au maximum les données décrivant l'objet.

En fonction des données acquises concernant l'objet d'autres données peuvent en être déduites. Ceci constitue la première phase de traitement réalisé au sein de l'agent tactique. Un moteur d'inférence permet de construire une part des informations manquantes en fonction des connaissances acquises. Par exemple, une connaissance sur l'attitude de l'objet permet d'établir un certain nombre de paramètres jusqu'ici non mesurés. Un objet ayant une vitesse de 100km.h^{-1} au sol pourra être renseigné comme véhicule. Un objet à 3000 pieds avec une vitesse de 150km.h^{-1} pourra être classé comme plateforme aéroportée. Un grand nombre de données peuvent être déduites à partir de données partielles, comme les paramètres manquants de son attitude, le type de l'objet ou son identité.

Tandis que des paramètres comme la vitesse, l'altitude et le type de l'objet peuvent être directement mesurés par des senseurs, d'autres doivent être calculés. C'est par exemple le cas de la trajectoire. La trajectoire passée d'un objet doit être calculée à partir des différentes acquisitions réalisées par les senseurs alors que la trajectoire estimée future est calculée par une projection probabiliste définie par des règles opérationnelles.

Objets du théâtre et priorités

La priorité de l'objet au sein de la situation tactique est un paramètre aussi calculé et reflète l'importance de traiter l'objet pendant la mission. La priorité de l'objet est définie par un nombre entier au sein de l'agent. L'agent étant le seul à accéder librement aux données de l'objet il est le plus à même de déterminer sa priorité. Cette représentation de l'importance d'un objet et des actions senseurs qui s'en découlent se décline à tous les niveaux de l'architecture, de l'agent aux actions senseurs bas-niveau.

L'ensemble des données détenues par l'objet ainsi que les données déterminées sont contenues dans la mémoire de l'agent sous la forme de connaissances.

Design final de l'agent

À partir des mécanismes précédemment cités, nous pouvons décrire le design final d'un agent tactique.

Un agent tactique est un assemblage de trois composants principaux. Chaque composant correspond à un ensemble d'algorithmes et de mécanismes afin de fournir les fonctions essentielles au fonctionnement de l'agent. Ces composants sont communs à la plupart des agents logiciels pouvant être trouvés dans les plateformes SMA classiques énoncées en 3.1.5.

Ces composants sont :

- Le noyau de l'agent
- Sa mémoire
- Son système de communication

En positionnant ces composants au problème de gestion des données de l'objet, nous pouvons identifier lequel de ces derniers permettra de réaliser les opérations nécessaires à l'accomplissement de ses rôles.

Le noyau contient l'ensemble des algorithmes de décision, de complétion des données, de détermination de la priorité de l'agent. Dans le noyau prend place le processus de gestion de l'agent, permettant à l'agent de vivre selon un cycle de vie. Ce n'est que pendant la phase «exécution» du cycle de vie de l'agent que les processus propres à l'agent tactique seront exécutés. Les autres phases de vie permettent de le préparer à son environnement et de l'initialiser. L'initialisation peut aussi concerner les processus propres.

Les processus propres à l'agent tactique seront ici appelés la *charge-utile* de l'agent.

La mémoire de l'agent permet de stocker toute information nécessaire à la réalisation de ses rôles. L'ensemble des données correspondant à l'objet peuvent y être représentées. La mémoire permet aussi de stocker toute information nécessaire à l'exécution des processus évoluant dans le noyau.

Le système de communication lui permet d'échanger avec son environnement. L'environnement de l'agent est composé d'un ensemble d'agents et d'artefacts munis de systèmes permettant d'échanger. Les agents fonctionnant de manière asynchrone, il est possible de communiquer avec ses derniers uniquement via l'envoi de messages. Pendant l'exécution de ses processus internes, les messages reçus sont examinés.

Phase 7 : Identification des points bloquants nécessitant des mécanismes d'adaptations pour le consensus commun, la synchronisation et l'accès aux données

Une agentification d'architecture transforme sa topologie en profondeur. Modifier sa topologie mène à la création de nouveaux nœuds dans lesquels des données transitent. La tâche de création d'une architecture étant basée en partie sur l'étude de la distribution des données, la propagation des données dans la nouvelle architecture agentifiée doit être étudiée en détail.

Les agents tactiques reflètent les objets du théâtre. Ceci implique qu'à tout instant le nombre d'objets connus du système, c'est-à-dire les objets sur lesquels le système à déjà une donnée (après détection par exemple) est égal au nombre d'agents tactiques (4.1).

$$N_{At} = N_{Od} \tag{4.1}$$

Cette propriété du système implique que sur un théâtre chargé de 500 objets, le nombre d'agents tactiques peut être égal, créant des goulots d'étranglement en plusieurs points de l'architecture. Cette assertion correspond à la première difficulté à surmonter pour que cette architecture agentifiée soit fonctionnelle. Un mécanisme d'ordonnancement adapté sera présenté en 4.3.

La deuxième difficulté correspond à l'opération inverse de la précédente. L'ensemble des données provenant des senseurs doit être attribué à des agents. Un mécanisme doit être responsable de l'attribution des données. Cette opération complexe sera réalisée par un système appelé le « Track Merger » ou « système de fusion de pistes » en français. Ce système sera présenté dans la suite de ce manuscrit, Composant 9.

Ces deux points névralgiques permettent de dessiner une première version simplifiée de l'architecture, voir figure 4.18.

Phase 8 : Projection de la solution sur plusieurs scénarios

Après avoir identifié les principaux composants et rôles au sein de l'architecture des extraits de scénario spécifiques ont été imaginés afin d'évaluer la complexité d'adaptation de l'architecture à d'autres situations. Cette étape donna naissance à plusieurs mécanismes faisant maintenant partie intégrante de l'architecture, comme les politiques de fonctionnement ou l'agent de priorité absolue qui sera décrit plus loin. Les politiques de fonctionnement permettent de régir le comportement du SMS en soumettant l'ensemble des agents et artefacts du SMA à de mêmes règles. Ces politiques permettent à l'opérateur de raffiner le comportement du SMS au-delà des règles de fonctionnement propres à chaque agent.

Phase 9 : Vérification de l'extensibilité de l'architecture SMA choisie à des scénarios non-natifs

De nouveaux scénarios complets ont été construits afin de vérifier la possibilité d'étendre l'architecture. Ces nouveaux scénarios ont montré qu'il était possible de s'adapter à de nouvelles missions en ajoutant de nouveaux modules à l'architecture avec un faible taux de correction de l'existant. Cet aspect sera détaillé en 6.

Une fois ces deux dernières phases passées, cette version de l'architecture peut être considérée comme assez extensible et modulaire pour vivre de futures évolutions. Cet exercice ne montre pas que toutes les futures évolutions pourront être acceptées par l'architecture, mais que la flexibilité est suffisante pour subir des modifications dans le prolongement des développements actuels.

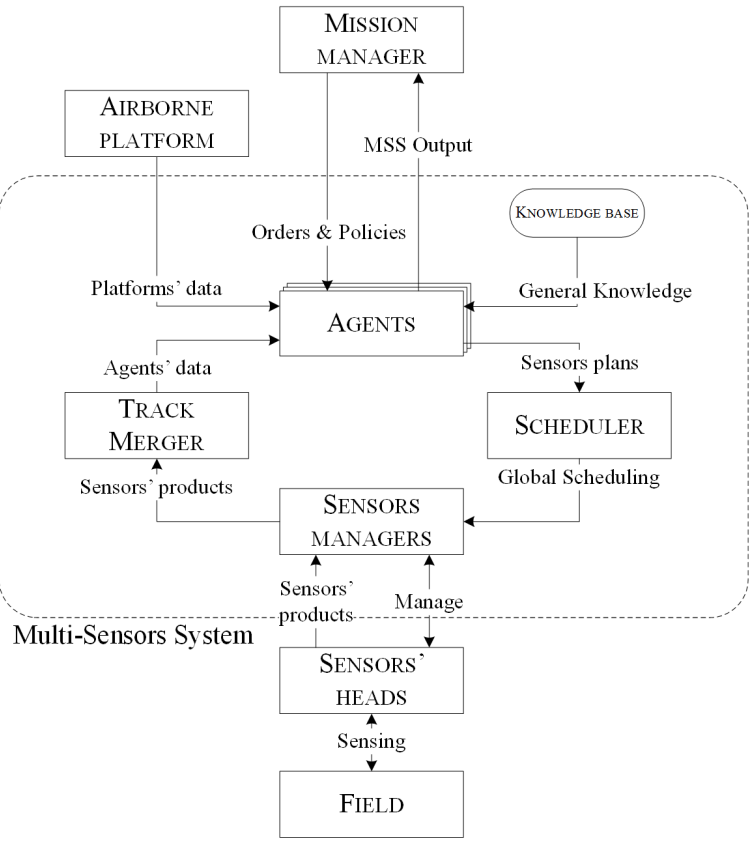


FIGURE 4.18 – Schéma de l'architecture.

4.2.2.3 Agents tactiques

Pour rappel, l'agent tactique représente un objet du terrain et cet objet sera appelé l'objet « virtualisé ». L'agent tactique a plusieurs actions à réaliser :

1. Le maintien des connaissances de l'objet virtualisé ;
2. La génération des ordres opérationnels propres à l'objet virtualisé ;
3. La projection des trajectoires de la plateforme et de l'objet ;
4. Le choix des fonctions senseurs appropriées ;
5. La génération des plans senseurs correspondants.

Un plan senseur est une description précise de l'utilisation des ressources pour réaliser une fonction senseur. Les plans senseurs seront détaillés en 4.3.2

Pour mieux comprendre la portée de l'agent tactique nous pouvons le visualiser comme une entité logicielle responsable, pour le compte de la mission et de l'opérateur, des actions à mener au sein du SMS en fonction des connaissances connues de l'objet, voir figure 4.19.

Un agent tactique est une entité logicielle réalisant les tâches pouvant être faites par un opérateur senseurs. L'agent analyse les données dont il dispose concernant l'objet qu'il virtualise. Il prend ensuite des décisions senseurs adaptées à ces données afin d'accumuler plus d'informations le concernant. Il est en quelques sortes un micro-opérateur senseurs dédié à un objet.

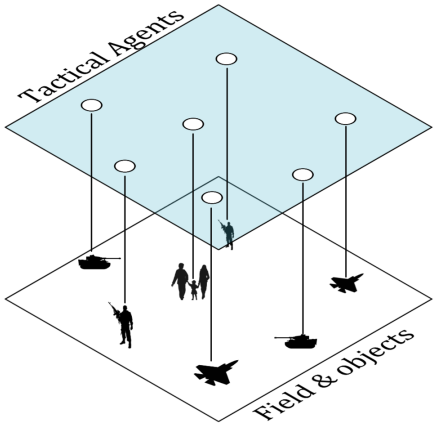


FIGURE 4.19 – Schéma de la virtualisation des objets du théâtre par les agents tactiques.

Les actions réalisées par l'agent tactique nécessitent toutes des données provenant de plusieurs points de l'architecture, à l'intérieur de l'agent ou non. Les rôles vont être détaillés afin de comprendre quelles données sont nécessaires.

Action 1 : Maintien des connaissances de l'objet virtualisé

Ce rôle de l'agent tactique permet de maintenir à jour l'ensemble des connaissances détenues sur l'objet virtualisé. Le maintien des connaissances signifie que les données remontant des senseurs sont analysées par un moteur d'inférence afin de mettre à jour des données déduites de plus haut niveau. Par exemple, si un objet est localisé à une altitude nulle il peut être considéré comme objet sol. Ou par exemple si sa vitesse est dans une certaine classe avec une altitude non nulle, il peut être considéré comme une plateforme aéroportée. La déduction de paramètres par le moteur d'inférence interne à l'agent permet de maintenir à jour les attributs connus de l'objet.

Les données nécessaires au moteur d'inférence peuvent être contenues dans une ontologie regroupant un maximum de caractéristiques observables des objets. L'ontologie permet de relier les caractéristiques observées aux caractéristiques déduites. Une première version simple du moteur d'inférence et d'une ontologie permet d'émettre des affirmations simples concernant l'objet tandis qu'un moteur complexe ainsi qu'une ontologie complète permet de déduire des caractéristiques plus complexes à déterminer. L'aspect probabiliste des observations pourrait aussi être considéré dans une version plus évoluée du moteur d'inférence.

Action 2 : Génération des objectifs opérationnels propres à l'objet virtualisé

Un moteur d'inférence peut aussi intervenir pour la décision haut-niveau des objectifs opérationnels. La décision de cet Objectif Opérationnel (OO) permet d'orienter le choix des fonctions senseurs et des senseurs à utiliser pour compléter la donnée présente dans la mémoire de l'agent. L'agent tactique ayant comme objectif de compléter du mieux possible les données qui concernent l'objet virtualisé, il tentera d'utiliser les senseurs lui permettant d'obtenir un maximum d'informations sur l'objet. Ce processus commence par l'identification de l'objectif opérationnel. Cet objectif appartient ici à la chaîne d'acquisition DRIL-P. L'OO choisi par l'agent doit permettre de continuer la complétion des informations de l'objet virtualisé. Par exemple, si aucun objectif de la chaîne n'est atteint l'OO sera de détecter l'objet. Si l'objet est détecté, reconnu et identifié, il devra être localisé.

Les senseurs exploitent des domaines physiques très différents. Pour cette raison, l'ordre du choix des OO n'est pas direct. Par exemple, avec capteur optique simple l'objet peut être détecté, localisé, reconnu, identifié et poursuivi uniquement dans ce sens, puisque celui-ci provient de la caractéristique optique de l'acquisition de l'image de l'objet, donc par une problématique de taille de l'image et de définition. Sur un SMS complet, l'ordre n'est pas unique. L'objet peut être détecté par une écoute EM omnidirectionnelle, sa signature peut en être déduite, donc il peut être identifié avant d'être localisé.

Action 3 : Projection des trajectoires de la plateforme et de l'objet

De manière à planifier l'utilisation des senseurs et de choisir les plans adaptés au contexte l'agent se doit de projeter les trajectoires de la plateforme et de l'objet virtualisé. La trajectoire de l'objet peut être extrapolée sur la dernière observation faite de ce dernier. La compatibilité d'un plan avec les trajectoires et positions des objets se résume en une somme d'intervalles temporels relatifs au plan de vol de la plateforme, voir figure 4.20.

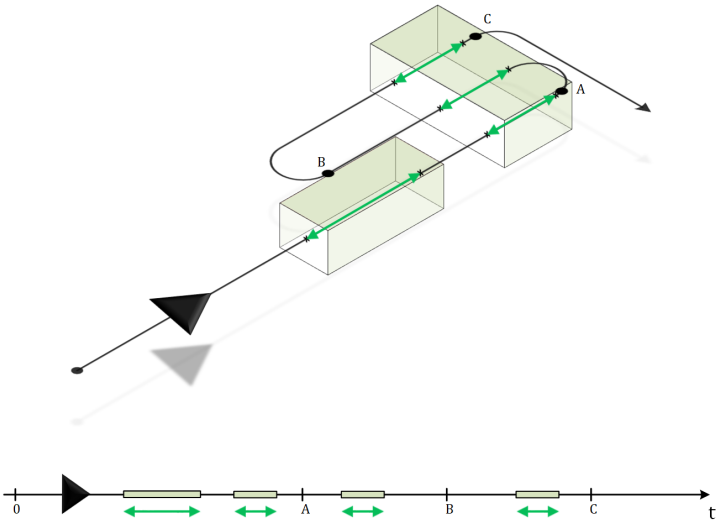


FIGURE 4.20 – Schéma de l'opération de projection des trajectoires.

Action 4 : Choix des fonctions senseurs appropriées

Chaque objectif opérationnel peut être accompli par un ou plusieurs objectifs fonctionnels distincts. Un objectif fonctionnel décrit quelle information doit être récupérée du terrain pour atteindre l'objectif opérationnel. Par exemple, L'OO « Détecter» peut être réalisé par les Objectif Fonctionnel (OF) « Prise d'image» ou «Écoute EM Omnidirectionnelle».

Chaque Objectif Fonctionnel peut être atteint par plusieurs fonctions senseurs. Par exemple l'OF Prise d'image peut être atteint par la fonction « SAR Spot», « SAR Strip » et « Image Optronique ». Dans le cas où plusieurs senseurs permettent de réaliser la même fonction, il est nécessaire de décliner les fonctions. Par exemple, sur une plateforme embarquant deux antennes permettant de réaliser un SAR Spot, il est nécessaire de créer deux fonctions « SAR Spot Droit » et « SAR Spot Gauche » afin par la suite de vérifier la compatibilité de chaque fonction par rapport au contexte.

Le type d'objet permet de déduire toutes les fonctions senseurs réalisables. En effet, toute fonction senseur n'est pas applicable à n'importe quel objet. Par exemple, il est impossible d'appliquer un SAR Spot sur un objet mobile et la fonction Inverse-SAR (ISAR) sur un objet fixe.

Le choix de la fonction à exécuter se fait à partir des données déterminées lors des étapes précédentes : informations sur l'objet, trajectoires, objectif opérationnel, objectifs fonctionnels, fonctions senseurs réalisables pour le type d'objet.

Les données nécessaires aux choix de senseurs sont calculées dans un ordre particulier, de la donnée brute provenant du terrain à la donnée symbolique puis la décision de la fonction senseur à réaliser.

Il est parfois impossible pour l'agent de trouver une fonction senseur qui soit compatible avec les contraintes du terrain. Les contraintes d'utilisation des senseurs (plages de distances, vitesses, etc.) peuvent impliquer qu'aucun plan senseur ne soit réalisable à un instant donné. La projection de la trajectoire de l'objet et de la plateforme peut permettre de lever l'incompatibilité et de trouver un intervalle temporel pendant lequel une fonction est réalisable. La fonction sera donc retenue et planifiée pour cet instant. En règle générale, une fonction sera choisie avec un temps d'avance considéré non nul, c'est-à-dire que la planification de la fonction est réalisée pour le temps suivant, avec un pas de temps le plus réduit possible, en fonction des capacités de calculs et de communication avec le matériel.

Action 5 : Génération des plans senseurs correspondants

Une fois qu'une fonction senseur est choisie, elle doit être traduite en contraintes et en instructions précises pour activer les senseurs. Le fonctionnement d'un unique senseur peut nécessiter plusieurs ressources avec des contraintes temporelles strictes. Par exemple, une fonction prise d'image SAR par une antenne active précise décrit l'utilisation d'un senseur, d'une bande de fréquences, d'un boîtier de traitements d'images SAR et de puissance électrique instantanée. L'antenne permet d'émettre et recevoir un signal réfléchi par les surfaces du sol tandis que le boîtier de traitement SAR en calculera les caractéristiques et en créera une image. La bande de fréquence doit être réservée précisément en même temps que le senseur et que la puissance électrique. Le boîtier de traitement SAR doit quant à lui être réservé après le début de la capture du signal par l'antenne, de manière à ce qu'une quantité de données suffisante soit déjà recueillie par l'antenne pour débuter le calcul de l'image. Les plans et leurs caractéristiques sont présentés en détail en 4.3.2.

Les actions à réaliser détaillées précédemment permettent de définir le design complet d'un agent tactique, visible sur la figure 4.21.

Sur cette figure sont visibles les éléments communs à tout agent purement communicant : les communications, la mémoire, son noyau. L'ensemble des données nécessaires à la réalisation des actions de l'agent est schématisé (ordres et politiques, connaissances opérationnelles et des senseurs, données objets, etc.). Les origines de ces données sont aussi représentées : Gestionnaire de mission, base de connaissances, ordonnanceur, etc. Les sorties de l'agent sont représentées : plans senseurs vers l'ordonnanceur et sortie des données agents vers le gestionnaire de mission.

La vie d'un agent au sein d'un SMA est régie par un cycle de vie commun à l'ensemble des agents. Ce cycle de vie permet de réaliser les étapes importantes de son

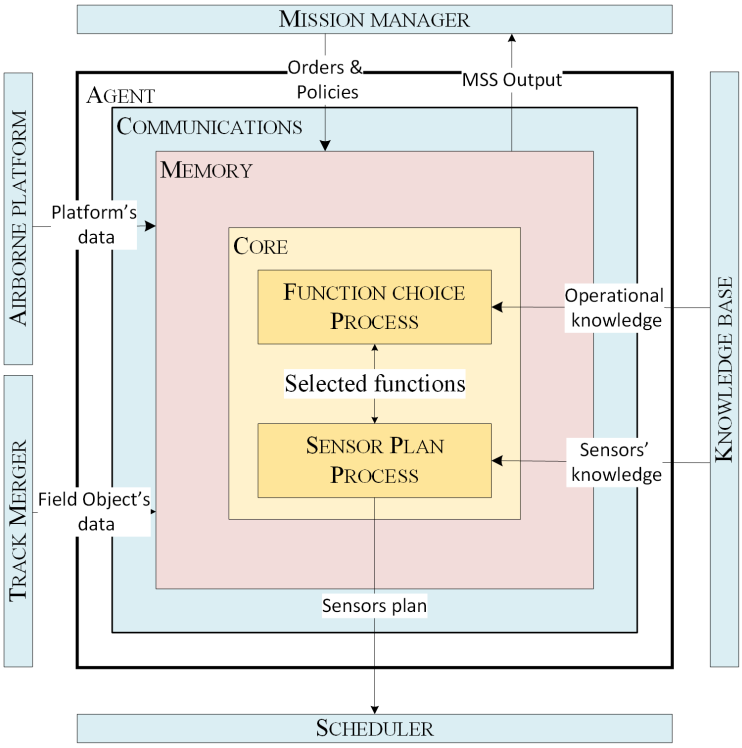


FIGURE 4.21 – Schéma de fonctionnement de l'agent tactique.

insertion au sein du groupe d'agents dans le bon ordre : initialisation des communications, rattachement au nœud d'agents, enregistrement par le service d'annuaire, et autres initialisations propres à la charge utile de l'agent. Ce cycle de vie est détaillé sur la figure 4.22.

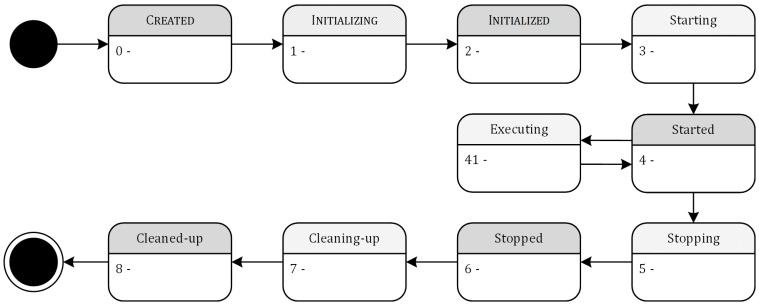


FIGURE 4.22 – Schéma du cycle de vie d'un agent du SMA.

Les agents exécutent les étapes successives de la récupération des signaux perçus à l'élaboration des plans senseurs dans un ordre précis. Cet ordre est mis en place par une machine à états finis appelée « machine à états tactique » permettant de passer d'un état au suivant, en cohérence avec le contexte et les données acquises de l'objet 4.23.

Cette machine à états tactique implique que l'ensemble des données déduites des données de perception d'un objet du terrain est calculé étape par étape tout au long des états. Ce déroulement d'étapes a comme impact qu'il est impossible de prendre en compte de nouvelles données lorsque la phase de calcul a démarré. Il est dans ce cas nécessaire soit d'abandonner les calculs en cours et relancer la machine à état,

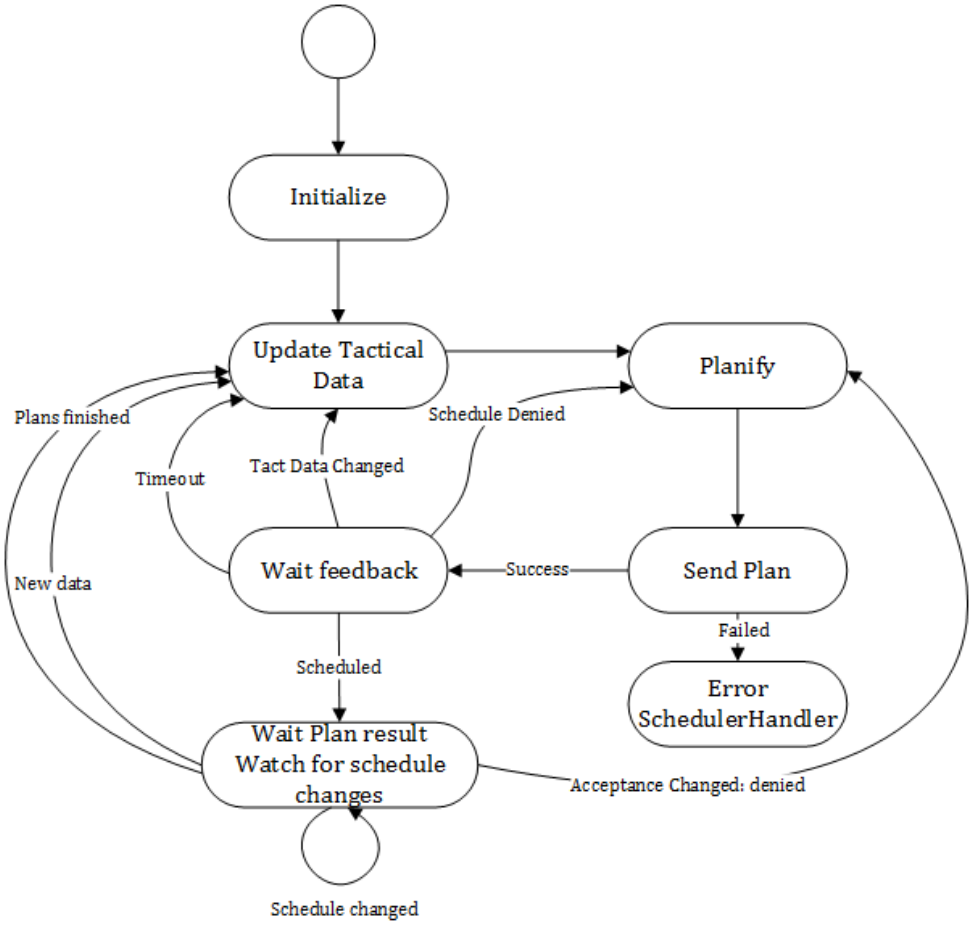


FIGURE 4.23 – Schéma d'une machine à états tactique.

soit d'ignorer les nouvelles données et les prendre en compte lors du prochain lancement de la machine à état. Le choix de redémarrer la FSM ou d'ignorer les nouvelles données peut être fait, au regard des nouvelles données entrantes, si elles invalident complètement le calcul précédent ou si elles sont superflues.

4.2.2.4 Représentation directe de la situation tactique

La situation tactique, donnée importante pour le C2, est dans cette architecture représentée par le groupe d'agents. Puisque chaque agent détient et est responsable d'un objet, l'ensemble des agents représentent la totalité des objets connus du terrain à un instant donné.

Les caractéristiques proactives et autonomes des agents tactiques impliquent que les agents, en cohérence avec la situation réelle, et prendre des décisions senseurs indépendamment les uns des autres en fonction de leurs données propres et de leurs niveaux de priorités. Une représentation du déroulement de ce processus allant de la donnée à la décision puis à l'exécution des instructions par les senseurs est visible sur la figure 4.24.

La phase manquante non représentée dans cette figure est la phase d'ordonnancement. En effet l'ensemble des agents génèrent des plans sans prendre en considération les plans provenant des autres agents. La phase d'ordonnancement permet de construire l'ordonnancement global en prenant en compte toutes les contraintes.

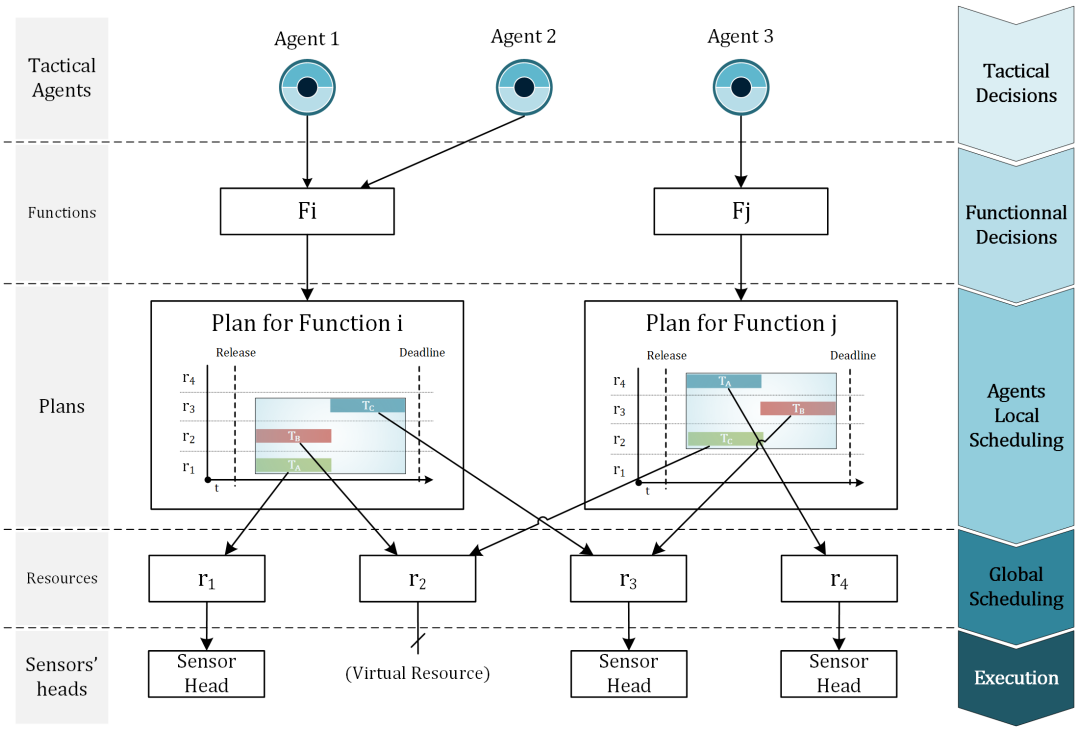


FIGURE 4.24 – Schéma montrant les étapes de la génération des plans à leur exécution par les senseurs.

L'ordonnancement global ou « global scheduling » en anglais est un ensemble d'actions à réaliser pour l'ensemble des senseurs positionnés dans le temps, un exemple d'ordonnancement global est donné en figure 4.25.

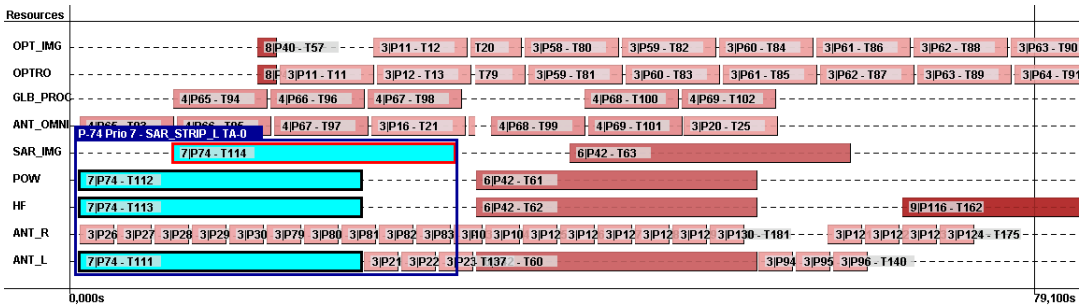


FIGURE 4.25 – Schéma d'un ordonnancement global.

4.2.3 Description de l'architecture

4.2.3.1 Les différentes entités du SMS

À partir de la figure 4.18 ainsi que du modèle d'un agent visible en figure 4.21 il est possible de construire une version détaillée de l'architecture incluant les agents, les ressources réelles et virtuelles, les senseurs et URL, et autres composants de l'architecture comme le gestionnaire de mission, plateforme, et le système de fusion de pistes.

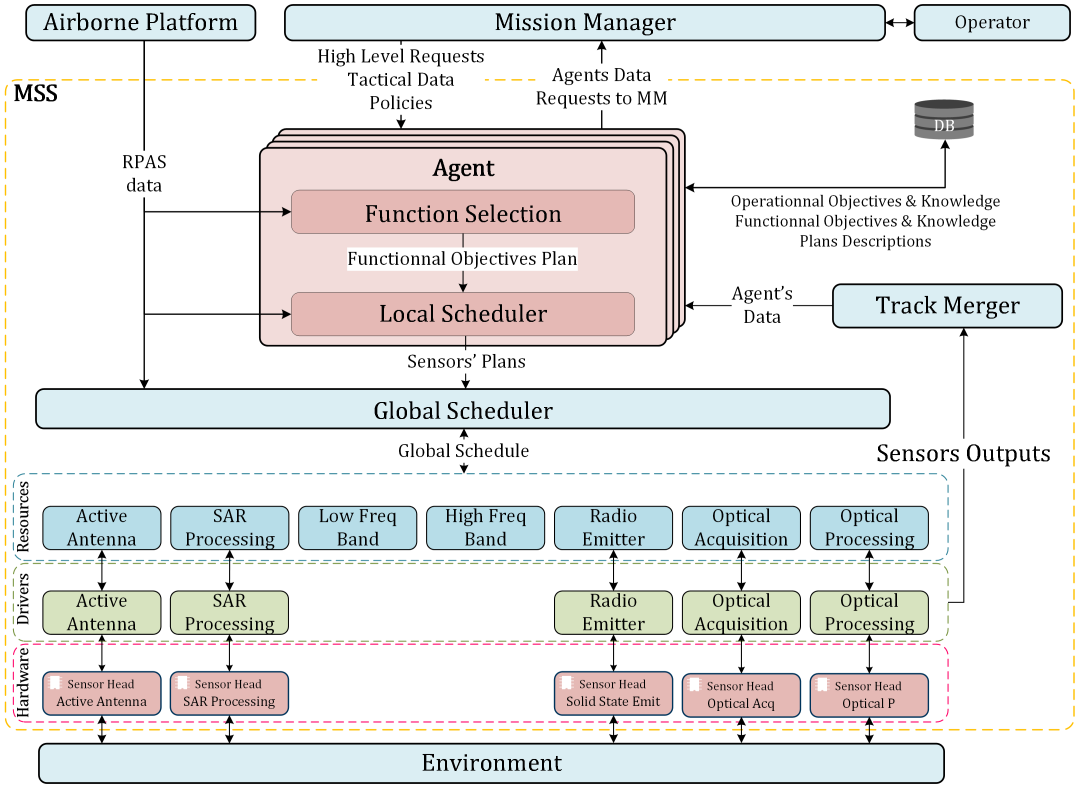


FIGURE 4.26 – Schéma de l'architecture système du SMS multi-agent

Les fonctions de chacun des composants de l'architecture vont maintenant être détaillées.

Composant 1 : Gestionnaire de mission
Le gestionnaire de mission est le point d'entrée du SMS. Ce composant, placé à bord ou en dehors de la plateforme, permet de gérer la plateforme, le système de senseur et permet de suivre la mission. Ce composant communique directement avec le SMS, lui envoie des ordres et gère son fonctionnement. Le gestionnaire de mission régule le fonctionnement du SMA par l'intermédiaire d'ordres haut-niveaux orientés agents, d'ordres senseurs directs et de politiques. Ces dernières permettent d'impacter et réguler le comportement du groupe d'agents.

Composant 2 : Plateforme
Le composant système *plateforme* ici schématisé est le point d'entrée des données nécessaires au fonctionnement du système de senseurs provenant de la plateforme. Les données mises à disposition par ce composant consistent en plusieurs informations instantanées et futures concernant le vol de l'appareil. Les données d'attitude, de trajectoire, et du plan de vol sont toutes nécessaires à la planification et l'exécution des actions senseurs.

Composant 3 : Opérateur
L'opérateur est l'utilisateur du système. Il est la personne donnant les ordres haut-niveau et ayant connaissance des objectifs de mission à atteindre. Il donne s'est ordres par le biais du gestionnaire de mission. Il décide de la trajectoire de la plateforme et choisit les actions senseurs à réaliser tout au long de cette trajectoire.

Composant 4 : Environnement
L'environnement correspond à l'ensemble de ce qui entoure la plateforme. L'atmosphère, le terrain, les objets présents sur ce dernier composent ensemble l'environnement. Il est observé par l'opérateur via les senseurs de la plateforme.

Composant 5 : Senseurs
Les senseurs permettent d'observer et mesurer l'environnement de la plateforme. En nombre variable en fonction du type de plateforme, ils fonctionnent sur des domaines physiques différents, des spectres EM réduits ou vastes avec des portées qui leur sont propres. Les senseurs sont les yeux de la plateforme et de l'opérateur : ils permettent à cette dernière de se repérer, de s'orienter, d'observer et de prendre les meilleures décisions pour la mission. Les senseurs sont pilotés par des instructions précises dans l'espace et dans le temps.

Composant 6 : Ressources
Les ressources constituent l'ensemble des éléments allouable à une fonction à un instant donné pour exécuter une fonction senseur.

Les systèmes présentés jusqu'ici sont communs à l'ensemble des systèmes de senseurs. Les suivants font partie d'une architecture SMS à base d'agents telle que présentée dans ce manuscrit.

Composant 7 : Agents Tactiques
Les agents tactiques accomplissent l'ensemble des actions présentées en 4.2.2.3. Ils constituent le point central de cette architecture. Ils maintiennent les données des objets virtualisés à jour et prennent des décisions senseurs haut et bas-niveaux en conséquence : à partir de règles définies, d'ordres opérateurs haut-niveau et de politiques de fonctionnement. Ils émettent ensuite les plans senseurs compilés correspondant au contexte pondéré de la priorité de l'agent. Ils permettent à l'architecture d'atteindre un niveau d'autonomie suffisant face au contexte exposé en 2.2.1.2.

Composant 8 : Ordonnanceur
L'ordonnanceur reçoit les plans senseurs à ordonnancer dans le temps en fonction de la priorité des plans senseurs et des contraintes temporelles et de ressources qu'ils spécifient. L'ordonnanceur établit l'ordonnancement global des ressources du SMS sous la forme d'une ligne temporelle par senseur spécifiant les dates exactes d'application des instructions.

Composant 9 : Fusion de pistes
Le composant responsable de la fusion de pistes (appelé track-merger) a connaissance de l'ensemble des produits des ressources remontées après l'exécution de leurs instructions. La fusion de pistes nécessite la connaissance des agents présents dans le système afin de délivrer la donnée à l'agent correspondant, cela signifie que les données sortant des ressources doivent être corrélées avec la situation tactique globale afin de la rattacher à l'objet et donc à l'agent correspondant.

Composant 10 : Bases de connaissances

Les ontologies permettent de lier les données observées sur le terrain à des concepts et ainsi à l'agent de raisonner et prendre des décisions en fonction des produits senseurs. Dans le SMS, l'ontologie permet par exemple passer des connaissances objets

aux objectifs opérationnels à adopter. L'ontologie utilisée au sein du SMS sera détaillée en 4.2.3.2. Les bases de données permettent de lier les données recueillies par les senseurs à des valeurs nécessaires à la prise de décision. Dans le SMS, les bases de données permettent par exemple d'identifier la signature d'un signal radio comme étant amie ou ennemie. Ces bases de données sont aujourd'hui appelées bibliothèques dans le contexte des capteurs.

Les bases de connaissances sont des ensembles de connaissances structurées permettant de supporter les agents dans leurs activités. Deux types de connaissances sont essentiels : les bases de données et les ontologies.

Les bases de connaissances sont accessibles à tout agent du SMA à différents instants de leur cycle de vie. Elles sont chargées dans le système en phase de préparation de mission en fonction du contexte précis des opérations.

Au sein de cette architecture SMS, les ressources peuvent soit être matérielles comme un senseur, un bus de données, une URL, soit virtuelles comme une fréquence électromagnétique ou n'importe quelle autre grandeur physique.

Composant 11 : Ressources virtuelles
Les ressources virtuelles ont été introduites en même temps que les travaux de SMS multi-agent afin de pallier le problème de l'inter-compatibilité des senseurs. Sur les systèmes de capteurs actuels, la compatibilité est gérée par des signaux transitant entre les capteurs non-compatibles dans certains modes (exemple : interférences EM). L'introduction de ressources virtuelles permet aux senseurs de réserver cette ressource sans connaître les exigences de ses homologues. Ce mécanisme évite des imbrications complexes entre les fonctionnements de différents senseurs, prévient l'apparition de deadlocks et améliore l'évolutivité du SMS en diminuant le taux de reprise de l'existant en cas d'ajout de capteurs.

Composant 12 : Ressources matérielles
Les ressources matérielles sont réservables au même titre que toute autre ressource. Le fait d'être rattachées à du matériel donne à ces ressources un rôle supplémentaire que de simplement réserver un composant. Dans les premières étapes de conception de l'architecture, ces ressources avaient été envisagées comme des agents autonomes. Cette vision correspondait à la caractéristique autonome des senseurs actuels. Cependant, avec les transformations successives lors de la construction de l'architecture, les senseurs ont été placés au rang d'artefacts, les actions senseurs étant décidées par les agents tactiques. Néanmoins, les ressources matérielles jouent un rôle important dans la conversion des instructions senseurs contenues dans les plans émanant des agents, vers le matériel.

Les ressources matérielles incluent les senseurs, les boîtiers de calculs, bus de données et toute autre URL réservable nécessaire au fonctionnement des senseurs. En fonction de la ressource et donc du matériel interfacé, les instructions doivent être précises afin que la ressource puisse la convertir en micro-actions senseurs. Par exemple, un agent demandant un balayage radar créera une instruction du type « balayage type 1, fréquence A, gisement début, gisement fin, site début, site fin ». Cette instruction est ensuite reçue par la ressource matérielle qui déclenchera le mode approprié dans les plages demandées. Ce processus permet d'échanger entre les agents et les ressources sans se préoccuper des données bas niveau de gestion

du matériel. La modularité s'en trouve augmentée et l'abstraction apportée par les ressources simplifie la gestion des instructions et du développement.

Les données transitant entre les composants système vont maintenant être détaillées.

Donnée 1 : Requêtes haut-niveau
Gestionnaire de mission → Agents tactiques

Les requêtes haut-niveau permettent à l'opérateur de contraindre les décisions des agents en décidant leurs objectifs opérationnels à atteindre. Les requêtes haut niveau s'adressent à un ou plusieurs agents ciblés en fonction des besoins de l'opérateur et permettent d'outrepasser les décisions prises par les agents ou de les compléter. Il peut être nécessaire d'outrepasser les décisions agents dans le cas où l'opérateur voudrait appliquer un raisonnement non saisi dans les bases de connaissances ou pour effectuer une action particulière. En absence de requête opérateur pendant la mission, les agents suivent les règles présentes dans les bases de connaissances.

Donnée 2 : Données tactiques
Gestionnaire de mission → Agent tactique

Les données tactiques sont émises par l'opérateur et proviennent du centre C2. Ce sont des données déterminées lors des phases de renseignements et de préparation de mission. Ces données permettent de renseigner certaines informations qui ne nécessiteront alors pas d'être recueillies de nouveau par le système de senseurs. Ceci permet à nouveau à l'opérateur de spécifier le comportement du SMS avant et pendant le vol de la plateforme. Les données tactiques peuvent concerner n'importe quel objet du terrain déjà reconnu ou identifié et sont passées aux agents tactiques correspondants qui les intègrent dans leurs connaissances. Ceci implique que lors de la préparation de mission si des données concernant des objets sont disponibles il est possible de créer des agents avant le décollage de la plateforme. L'évolution du groupe d'agents tactiques est décrite en 4.2.4.2.

Donnée 3 : Politiques
Gestionnaire de mission → Groupe d'agents tactiques

Des politiques permettent à l'opérateur de contraindre un peu plus le système de senseurs. Il existe 3 types de politiques mises à disposition de l'ensemble du groupe d'agents :

- Les politiques restrictives : Ces politiques permettent d'interdire l'usage de certaines ressources (matériel, bande de fréquences) en fonction du théatre au sein duquel la plateforme évolue. L'opérateur peut mettre en place plusieurs politiques lors de la phase de préparation de mission et choisir entre ces dernières pendant la mission. Un exemple typique de cas où cette autonomie est utile peut être donnée : dans une situation qui requiert une discrétion radio, une politique adaptée permet de contraindre l'ensemble du groupe d'agents à ne pas mettre en place de plans amenant à dévoiler la position de la plateforme sur le terrain. Les ressources du type bandes de fréquences en émission peuvent par exemple être interdites.
- Les politiques d'autonomie : Ces politiques permettent à l'opérateur de limiter le champ d'action des agents. Ainsi l'opérateur peut choisir jusqu'à quel point l'agent peut le supporter dans ses actions. Le niveau d'autonomie peut aller du

niveau 0 : actions uniquement réalisées par l'opérateur jusqu'au niveau maximum : autonomie complète des agents sur les ordres senseurs. Une aide à la décision peut néanmoins être donnée avec un niveau d'autonomie minimal en conservant le raisonnement des agents et en le partageant avec le gestionnaire de mission.

- Les politiques comportementales : Le comportement du point de vue opérationnel des agents doit pouvoir être régi par l'opérateur grâce aux politiques comportementales. L'opérateur peut par exemple faire le choix de rechercher uniquement un type d'objet lors d'une mission ou en ignorer d'autres. C'est à ce niveau que les politiques comportementales peuvent être appliquées.

Les politiques s'appliquent à l'ensemble du groupe d'agents afin qu'aucun ne puisse sortir du périmètre de fonctionnement décidé par l'opérateur.

Donnée 4 : Données plateforme type 1
Plateforme → Groupe d'agents tactiques

Le premier type de données plateforme concerne l'ensemble des constantes émanant de ses différents capteurs nécessaires au fonctionnement du SMS. Par exemple, la température et le taux d'humidité de l'air sont nécessaires à l'exécution de plans impactés par ses données, comme des prises d'images SAR ou optroniques. Les données d'attitude temps réel de la plateforme (position, vitesse) sont aussi nécessaires à l'exécution de plans nécessitant un pointage de faisceau ou d'ouverture.

Donnée 5 : Données plateforme type 2
Plateforme → Groupe d'agents tactiques

Le deuxième type de données plateforme concernent les données à venir nécessaires à la planification des actions senseurs. L'exemple le plus évident est le plan de vol futur, permettant de connaître l'ensemble de la trajectoire de la plateforme et des senseurs sur un horizon temporel fixe.

Donnée 6 : Connaissances opérationnelles
Base de connaissances → Groupe d'agents tactiques

Les connaissances opérationnelles permettent aux agents tactiques de disposer d'un ensemble de règles nécessaires à la prise de décision d'utilisation des senseurs. Ces connaissances sont sous la forme de règles à suivre lors de l'observation des caractéristiques des objets. Chaque agent, lors de l'exécution de son cycle de vie, nécessite l'accès à cette base de connaissances afin de comparer les données des objets avec les règles prédéfinies. Les bases de connaissances opérationnelles permettent à l'agent de choisir l'objectif opérationnel permettant de compléter les connaissances de l'objet. Par exemple, lors d'une détection d'un véhicule terrestre armé, la base de connaissances permet de connaître quels sont les objectifs opérationnels à adopter pour connaître son identité, sa dangerosité, son périmètre d'action, etc. Il peut être décidé après consultation de la base qu'il est nécessaire de l'identifier, de la localiser avec précision et de protéger la plateforme.

Donnée 7 : Connaissances fonctionnelles
Base de connaissances → Groupe d'agents tactiques

De la même façon que les connaissances opérationnelles, les connaissances fonctionnelles ont un rôle à jouer dans la sélection des actions senseurs à mettre en place lors

de l'observation des caractéristiques des objets du terrain. Si les connaissances opérationnelles permettent de répondre à la question du « *Quoi ?* » lors de l'observation d'un objet, les connaissances fonctionnelles permettent de répondre au « *Comment ?* ». Après décision de l'objectif opérationnel à suivre, l'objectif fonctionnel est décidé en fonction des bases de connaissances appropriées. Dans le cas précédent de l'observation d'un véhicule terrestre, si l'objectif opérationnel décidé après consultation des bases de connaissances est « Identifier l'objet » la connaissance fonctionnelle appropriée doit lier le type d'objet et l'objectif opérationnel à la fonction senseur à mettre en place pour satisfaire l'objectif opérationnel.

Donnée 8 : Description des plans
Base de connaissances → Groupe d'agents tactiques

La description des plans senseurs est incluse dans les bases de connaissances. Cette description inclut plusieurs caractéristiques importantes pour la sélection et la planification des plans par les agents :

- La portée physique des plans
- Les ressources à réserver pour réalisation du plan
- Les durées de réservation des ressources
- Les contraintes temporelles de réservation des ressources

Cette description ainsi qu'une implémentation sera reprise en détail en 4.3.2

Donnée 9 : Données agents
Agents tactiques → Gestionnaire de mission

Les données agents constituent la sortie du SMS. Ces données correspondent à l'ensemble des connaissances de chaque agent du groupe d'agents. En d'autres termes, les agents représentant la situation tactique, ces données représentent l'ensemble des données recueillies et déduites par les agents du terrain. Ces données présentent une richesse extrêmement importante pour l'opérateur, qui ne dispose plus seulement des données senseurs brutes mais aussi de l'ensemble des informations déduites et synthétisées sous une forme très proche de la représentation mentale humaine que l'opérateur a du terrain.

Donnée 10 : Plans Senseurs
Agents tactiques → Ordonnanceur

Les plans senseurs sortants des agents correspondent aux descriptions senseurs dits « compilés » c'est-à-dire placés dans le temps après calcul et projection des trajectoires par l'agent. Les durées de chaque tâche est aussi fixée.

Donnée 11 : Instructions ressources matérielles
Agents tactiques → Ordonnanceur → Ressources matérielles

Après planification d'un plan par l'agent et son acceptation, le placement par l'ordonnanceur des instructions spécifiques sont nécessaires pour alimenter les ressources. Ces instructions sont de haut niveau et doivent ensuite être analysées par les ressources pour en déduire les instructions matérielles. Par exemple, si un agent planifie un pointage radar sur un objet à un instant donné, il devra passer en instruction la trajectoire de l'objet en question, afin que la ressource destinataire

puisse déduire quelle sera la position exacte de l'objet à l'instant fixé par l'ordonnanceur.

Les instructions de ressources sont limitées aux ressources matérielles, les ressources virtuelles ne pouvant recevoir d'instruction.

Donnée 12 : Instruction matériel
Ressource matérielle → Senseur/URL

Une fois l'instruction ressource matérielle analysée, l'instruction matériel peut en être déduite. Cette instruction est fixée dans le temps et sera transmise à la ressource (senseur ou URL) afin de la traiter.

Donnée 13 : Sorties senseurs
Ressource Matérielle → Fusion de pistes

Après exécution de l'instruction par la ressource matérielle les données sortantes, appelées produits senseurs sont transmises à la fusion de pistes.

Donnée 14 : Données fusionnées
Fusion de pistes → Agents tactiques

La fusion de pistes corrèle les produits senseurs entre eux et avec les données agents déjà recueillies. Après corrélation des données, une mise à jour des données agents est déduite et transmise aux agents concernés.

Donnée 15 : Requêtes au gestionnaire de mission
Agents tactiques → gestionnaire de mission

Les agents, en fonction de leur paramétrage, peuvent proposer à l'opérateur des actions senseurs qui sortent du périmètre d'autonomie de l'agent. Il pourrait aussi dans certains cas déduire de nouvelles trajectoires permettant une meilleure réalisation des actions senseurs. Des alertes de dangerosité et des notifications peuvent aussi être transmises.

Donnée 16 : Messages agents
Agent tactique → Agents tactiques

Les communications entre agents se font uniquement via des messages envoyés et reçus par leur mailbox. Ils permettent aux agents de communiquer des informations nécessaires à l'amélioration de la compréhension de la situation tactique par le C2. Plusieurs situations peuvent nécessiter la collaboration des agents, par exemple la détection et la représentation d'objets faisant partie d'un même groupe sur le terrain.

4.2.3.2 Ontologie et connaissances des objets

L'ontologie permettant l'analyse opérationnelle par les agents contient plusieurs éléments essentiels à la prise de décision :

- La taxomonie des objets du terrain ;
- Les plans senseurs réalisables pour chaque classe d'objets ;
- Les plans réalisables en fonction des caractéristiques des objets ;
- Les objectifs opérationnels et fonctionnels.

La taxonomie des objets est un classement de tous les types d'objets avec comme classe mère le type « objet générique ». La classe générique est déclinée en 3 classes filles, représentées en figure 4.27 :

- Classe objets air ;
- Classe objets terre ;
- Classe objets mer.

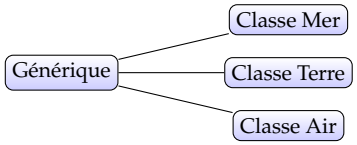


FIGURE 4.27 – Les trois classes principales d'objets

Chacune des classes est ensuite déclinée et nous pouvons y retrouver l'ensemble des classes d'objets mobiles et bâtiments pouvant être rencontrés pendant une mission.

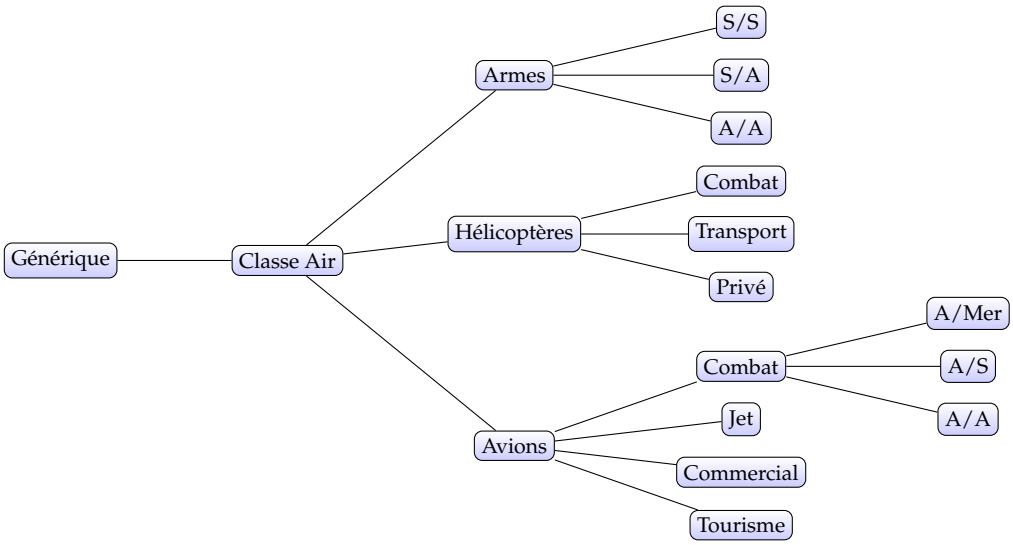


FIGURE 4.28 – La classe objet Air

L'ontologie est construite du point de vue système de senseurs, c'est-à-dire de façon à ce que, pour un objet, ses propriétés importantes pour le SMS lui soient propres et héritent de l'ensemble des caractéristiques des classes mères.

Par exemple dans la classe *air* présentée sur la figure 4.28, les caractéristiques d'un avion de combat dédié air-air héritera de l'ensemble des caractéristiques d'un avion de combat, d'un avion, des objets du type air et des objets génériques.

Il est nécessaire de garder à l'esprit qu'il doit être impossible pour un objet de cumuler plusieurs classes. Par exemple, un véhicule ne peut être de la classe *armé* et de la classe *radar* même s'il est en réalité armé et dispose d'un radar.

Pour cette raison, des clés de connaissances supplémentaires ont été introduites afin d'aider à compléter le profil des objets au sein des agents. Ces clés de connaissances sont par exemple :

- Objet en déplacement
- Objet immobile

- Émetteur RF
- Menace RF
- ...

Les clés de connaissances peuvent être renseignées d'une valeur en fonction de la classe de l'objet. Ces clés permettent d'affiner les traitements faits en plus de la connaissance de la classe. Par exemple, un objet de la classe véhicule ne pourra être traité par le SMS de la même façon s'il est mobile ou non. Si sa classe au cours d'une mission doit être la plus stable possible afin de conserver une cohérence dans la SiTac, les connaissances de l'objet peuvent être mises à jour et vérifiées régulièrement pour s'adapter au contexte.

En résumé, les clés de connaissances représentent des faits pouvant varier en cours de mission, tandis que les classes des objets représentent des faits valides pour un objet tout au long de la mission. Pour cette raison la classe d'objet « Véhicules mobiles armés » peut être utilisée : la caractéristique armée d'un objet peut difficilement varier en cours de mission. Cependant la classe « Véhicules mobiles armés en déplacement » ne peut exister, la caractéristique de déplacement doit être située au niveau des clés de connaissances.

Il est possible en cours de mission de faire erreur sur un type d'objet. Par exemple, un aéronef se déplaçant au sol sur une base aérienne pourrait être identifié comme de la classe véhicule après l'exécution d'une fonction radar/GMTI. La classe de l'objet changera pendant le décollage lorsque la vitesse ou l'altitude sortira des plages de valeurs possibles de la classe véhicule.

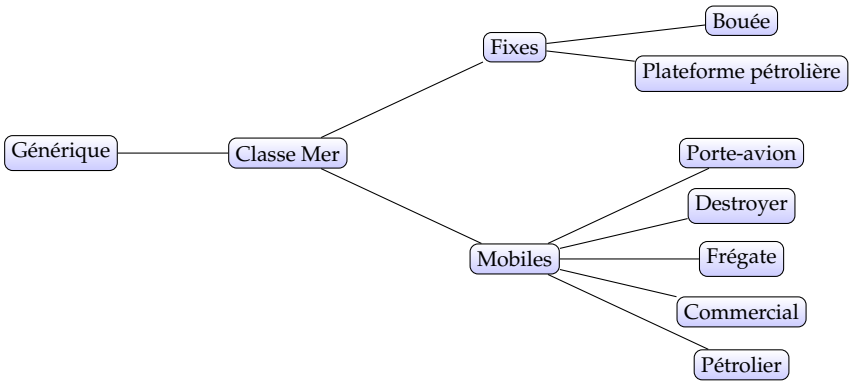


FIGURE 4.29 – La classe Mer

Type de donnée	Variabilité	Impact variation
Classe ontologie	Basse	Modification des plans senseurs réalisables
Clé de Connaissance	Moyenne	Modification des plans senseurs réalisables. Peut changer la classe de l'objet
Mesures variables objet	Haute	Change les clés de connaissances de l'objet

TABLE 4.1 – Tableau résumé des types de connaissances d'un objet et leurs impacts

Les objectifs opérationnels et fonctionnels constituent une partie de l'ontologie. Du recueil de la donnée provenant du terrain à la détermination de la classe de l'objet,

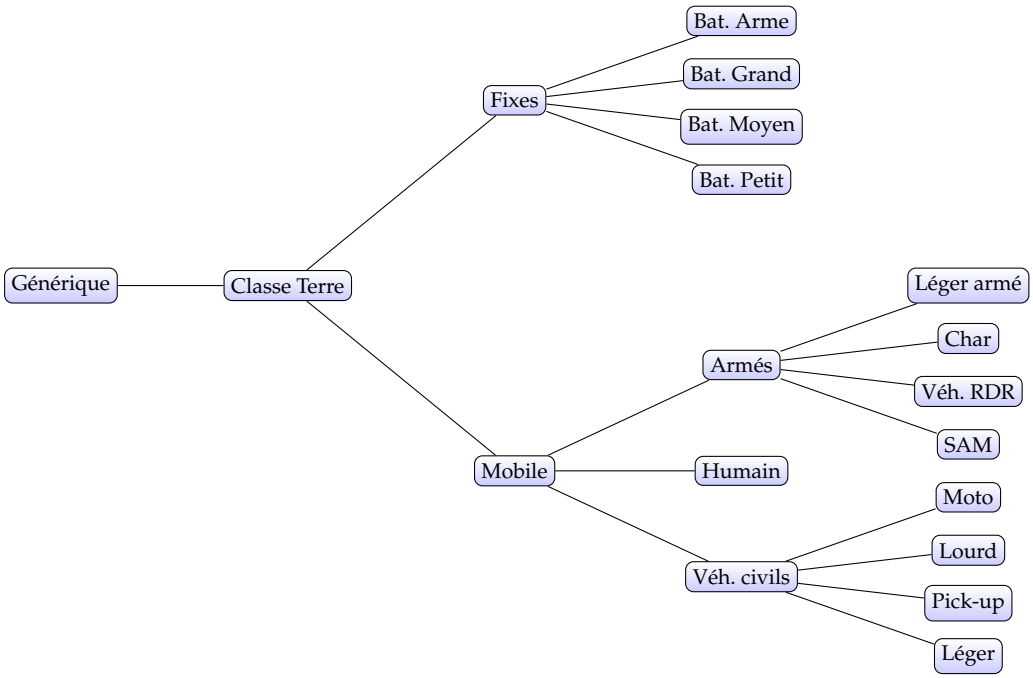


FIGURE 4.30 – La classe Terre

les objectifs opérationnels et fonctionnels constituent une étape intermédiaire dans l'acquisition des connaissances des objets du terrain. Comme présenté en 4, les objectifs opérationnels et fonctionnels sont liés de sorte que chaque objectif fonctionnel puisse être atteint par un ou plusieurs objectifs fonctionnels, voir figure 4.31. Les objectifs fonctionnels sont atteints grâce à l'exécution de types de plans senseurs réalisables par le SMS. Chacune des classes objets représentées au sein de l'ontologie acceptent un ensemble de plans senseurs permettant de recueillir les données nécessaires à l'accomplissement des objectifs fonctionnels, donc, opérationnels. Un type de plan senseur correspond à une fonction senseur réalisable par le SMS. Le « type de plan senseur » est l'étape préliminaire avant établissement des plans senseurs.

Si l'ensemble des objectifs opérationnels et fonctionnels sont atteignables pour une même classe d'objet, il n'en va pas de même avec les types de plans senseurs. Le plan senseur applicable à un objet dépend uniquement de la classe d'objet et des clés de connaissances de ce dernier. Par exemple, il est impossible d'appliquer une fonction Radar/GMTI sur un aéronef ou une fonction Radar/SAR sur un véhicule en déplacement.

Chaque objectif fonctionnel peut être atteint par un ou plusieurs plans senseurs. En conclusion, chaque objectif opérationnel peut être accompli par un ensemble d'objectifs fonctionnels, eux-même atteignables par plusieurs plans senseurs.

Pour aller plus loin il serait possible de détailler pour chaque classe d'objet, l'ensemble de ses caractéristiques et d'inférer sur celles-ci pour déterminer quel plan senseur sélectionner. Ceci permettrait de sélectionner les plans senseurs réalisables sur des objets encore non classés, découverts pour la première fois. Cette description profonde des classes objet n'est pas ce qui a été réalisé dans ces recherches. La contrainte de fonctionnement à base de règles définies sur les connaissances opérateurs (besoin 2 présenté en 4.1.2.2) allant à l'encontre d'un fonctionnement plus profond du moteur d'inférence pour la sélection de plans. Par une telle description

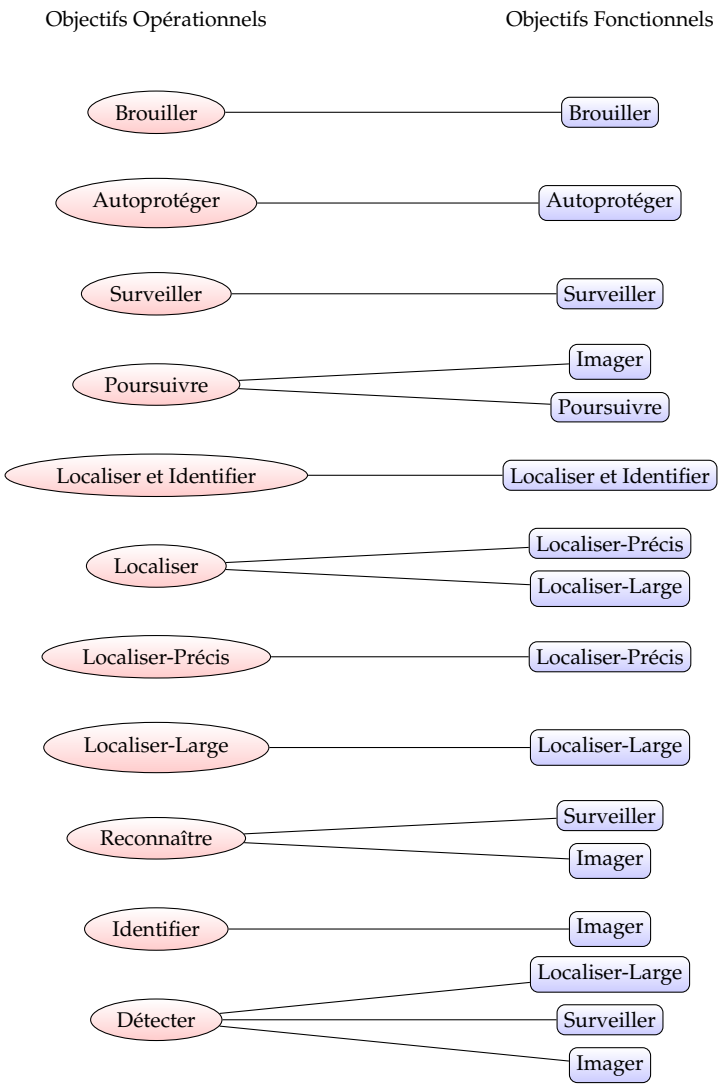


FIGURE 4.31 – Les objectifs opérationnels et fonctionnels

des objets, le fonctionnement du SMS pourrait être imprévisible pour l'opérateur qui attend du SMS des réponses prédéterminées, même si en pratique plus autonome et optimisée.

4.2.3.3 Boucle asservie Agent/Ordonnanceur/Senseurs/Track-Merger

L'architecture et ses composants permettent un contrôle asservi des senseurs grâce au maintien d'une situation tactique proche du contexte, tant du point de vue des informations détenues que sa dynamique.

Les agents constituent le cœur de l'ensemble de l'architecture, mais ne pourraient converger ni être alimentés correctement sans la présence d'un ordonnanceur et d'un track-merger adaptés.

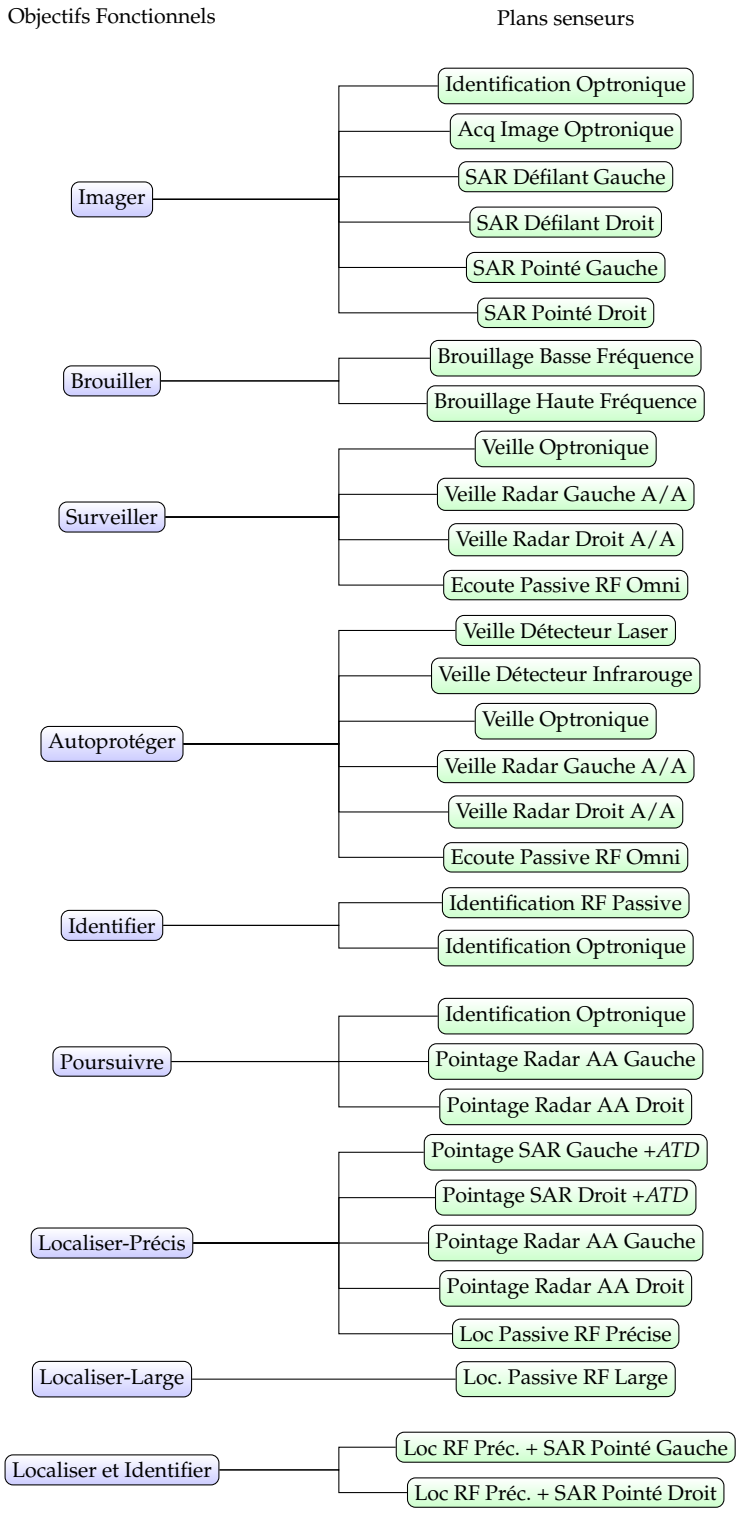


FIGURE 4.32 – Les objectifs fonctionnels et les plans senseurs

La problématique de contrôle des senseurs en fonction des perceptions de ces mêmes senseurs peut être assimilée à une problématique de perception active, voir [Bajcsy, 1988].

Le quatuor formé par les agents, l'ordonnanceur, les senseurs (ou toute ressource matérielle) et le track-merger constitue une boucle d'asservissement des senseurs du SMS. La figure 4.33 schématise cette boucle.

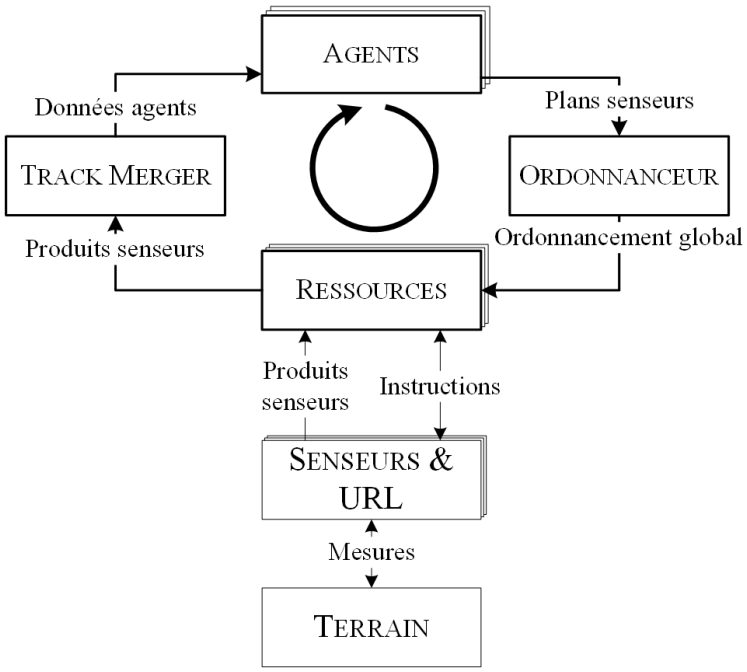


FIGURE 4.33 – Schéma de la boucle d'asservissement des senseurs du SMS

Le besoin 5 est un point essentiel contraignant pour l'ensemble de la conception de l'architecture. Des problèmes peuvent apparaître si un élément de la boucle d'asservissement prend trop de temps. Chacun des composants intervenant dans cette boucle doit conserver des vitesses d'exécution permettant de respecter cette contrainte. La figure 4.34 montre les différents temps de latence des composants constituant la boucle d'asservissement.

Agents :

Le choix et la construction des plans senseurs étant déterminés par un ensemble d'agents, la problématique de complexité et de latence de la prise de décision est répartie sur ces derniers. Le temps de latence entre la prise en compte de nouvelles données par l'agent et la compilation d'un plan permettant de traiter l'objet dépend de l'agent et des algorithmes qu'il est nécessaire d'exécuter jusqu'à la compilation du plan. Les algorithmes tels que la projection des trajectoires, le calcul de compatibilité des plans et la prise en compte des connaissances à disposition expliquent que les temps de latences peuvent différer en fonction des agents tactiques. Sur la figure 4.34 les temps de calcul et la construction des plans correspondent à la durée Δ_1.

Ordonnanceur :

La prise en compte de l'ensemble des plans senseurs est assurée par l'ordonnanceur, chargé de la construction de l'ordonnancement global. À chaque instant, une unique instance de l'ordonnanceur est en exécution, impliquant que l'ordonnanceur soit un maillon critique de la boucle d'asservissement : un retard de l'ordonnanceur implique une augmentation de la latence de toute la boucle. Pour cette raison, le

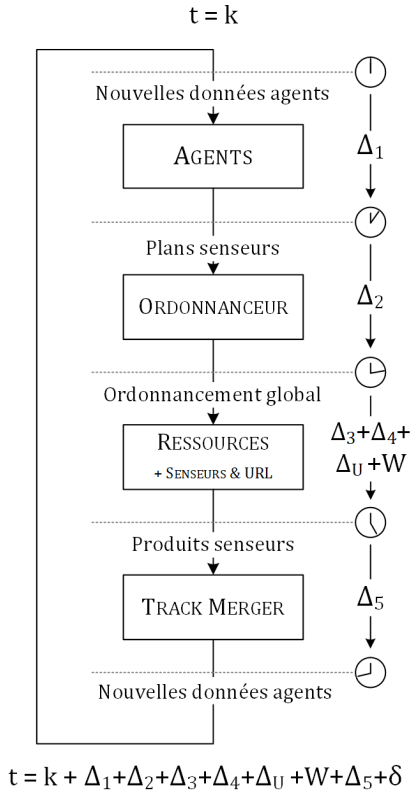


FIGURE 4.34 – Schéma des temps de latence de la boucle d'asservissement

temps de calcul de l'ordonnanceur est réduit à son minimum, empêchant l'utilisation d'algorithmes d'ordonnancement optimaux. Le mécanisme d'ordonnancement est détaillé en 4.3. Sur la figure 4.34, le temps de construction de l'ordonnancement global correspond à la durée Δ_2.

Ressources :

Le temps nécessaire aux ressources pour analyser l'instruction ressource et générer l'instruction matérielle, peut dépendre du traitement à réaliser, et donc, de la ressource. Par exemple, la génération de l'instruction matérielle d'une antenne active nécessite de prendre en compte plusieurs paramètres de trajectoire tandis que l'instruction matérielle d'un boîtier de calcul d'image nécessite la configuration des adresses et flux de données d'entrées à utiliser. Après génération de l'instruction matériel, la ressource matérielle commence son traitement. Le temps utile d'exécution de l'instruction dépend fortement de l'action senseur à réaliser. Une acquisition pour un SAR Pointé peut durer une quinzaine de secondes tandis que la capture d'une image optronique peut être réalisée en 1 seconde. Ce délai, qui sera appelé temps utile, est représenté par Δ_U sur la figure 4.34 tandis que le délai de traitement de l'instruction matérielle par la ressource matérielle est représenté par Δ_4. Le temps utile Δ_U et Δ_4 sont distingués car Δ_U constitue un délai dépendant de la performance du capteur uniquement et indépendant de l'architecture. Le délai W

correspond au temps nécessaire pour attendre l'instruction positionné sur la timeline de la ressource. Ce délai dépend uniquement du placement de l'ordre senseur sur l'ordonnancement global. Après exécution des instructions par le matériel, les ressources récupèrent le résultat et le transmettent au Track-Merger. La somme du temps de génération de l'instruction matérielle et de la récupération/transmission des produits du matériel constitue le temps Δ_3 sur la figure 4.34.

Track-Merger :

Le travail du Track-Merger consiste à regrouper les produits senseurs mis à disposition par les ressources, de les corréler et d'analyser les correspondances avec les agents déjà existants. Cette phase peut prendre plus ou moins de temps en fonction des techniques de fusion de pistes appliquées. Ce délai est représenté par Δ_5 sur la figure 4.34.

Le délai représenté par δ sur la figure 4.34 correspond à la somme des temps de latence provoqués par la propagation des informations au sein de l'architecture (latence du réseau de communication).

En conclusion, le temps nécessaire à la mise à jour des données d'un agent entre la prise en compte des informations qu'il a à sa disposition et le retour après fusion de piste est de la forme $t = k + \Delta_1 + \Delta_2 + \Delta_3 + \Delta_4 + \Delta_U + W + \Delta_5 + \delta$. Le temps de latence $\mathcal{D}$ propre à l'architecture est donc défini par la formule 4.2. Les délais Δ_U et W ne sont pas considérés comme caractéristiques de l'architecture.

$$\mathcal{D} = \sum_{N=1}^{5} \Delta_N + \delta \tag{4.2}$$

4.2.4 SMS en fonctionnement

4.2.4.1 Développement de la SiTac

Déroulement d'une mission :

Comme vu en 4.2.2.4, la situation tactique est représentée par l'ensemble des agents existant au sein du système multi-agent du SMS. En 4.2.3.1 : composant 9, il a été expliqué comment la fusion de pistes permettait de corréler les données acquises par les senseurs aux données courantes de la situation tactique. une étape importante dans la vie de cette architecture SMS correspond à son initialisation, notamment la phase de création du premier agent tactique du groupe d'agent. Le fonctionnement du SMS étant essentiellement basé sur la capacité à des agents de créer et planifier l'usage des senseurs il en va de même lors de sa création et des premiers instants de son fonctionnement. Si aucun agent n'est présent à la création du système, aucun plan ne sera exécuté par les ressources, le groupe d'agents restera donc vide et le SMS inactif.

Un agent a donc spécialement été introduit pour pallier l'absence d'agents au lancement du système. Cet agent est responsable de l'ensemble des ordres de veille du SMS, il permet de lancer les modes spécifiques sur chacun des senseurs afin de surveiller l'environnement de la plateforme et ainsi peupler le groupe d'agents tactiques. Du point de vue opérationnel, si un agent tactique virtualise un objet, cet agent a pour objectif de renseigner au maximum les connaissances sur cet objet. Par

déduction, recueillir un maximum d'informations sur l'environnement de la plateforme peut correspondre à l'objectif propre d'un agent tactique virtualisant la plateforme. Pour cette raison, le premier agent tactique créé lors du lancement du SMS est appelé « Agent RPAS ».

La figure 4.35 montre les différentes phases de fonctionnement du SMS.

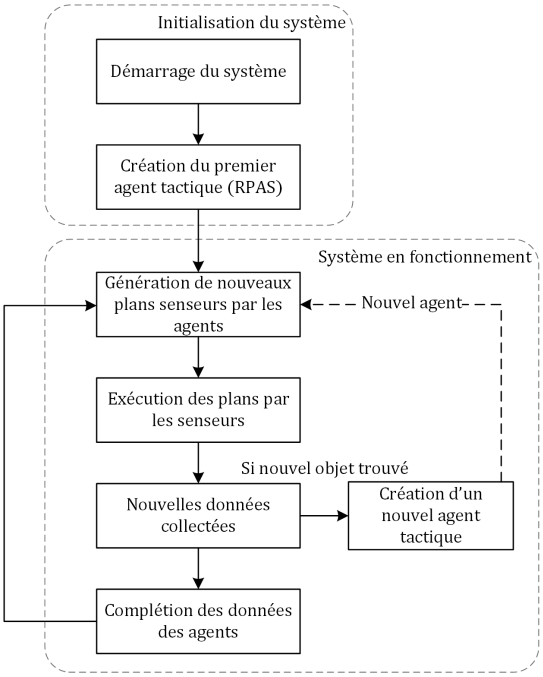


FIGURE 4.35 – Schéma du processus d'ajout des agents tactiques dans le SMS.

Exemple : Scénario simple

Prenons pour exemple un scénario simple dans lequel l'objectif est de surveiller une zone. La trajectoire de la plateforme est définie comme décrivant un hippodrome à une altitude intermédiaire. Sur le terrain est présent un véhicule blindé qui sera uniquement visible par la plateforme par le biais du senseur optronique à la 10ème minute du plan de vol.

Au lancement du système, seul l'agent RPAS existe au sein du SMS. Cet agent reçoit le plan de vol de la plateforme et planifie un ensemble de plans senseurs de veille. L'agent RPAS a alors pour objectif opérationnel et fonctionnel d'autoprotéger la plateforme. L'ensemble des plans qui en découlent, les veilles laser, infrarouge, optronique, radar AA et RF omnidirectionnelles, permettent de couvrir l'ensemble de l'environnement de la plateforme. L'agent RPAS étant à cet instant le seul agent présent dans le système, l'ensemble de ses plans seront acceptés et ordonnancés par l'ordonnanceur global. L'ensemble des plans de veille seront exécutés sur l'ensemble du scénario.

À la 10ème minute de vol, la fonction de veille optronique permettra aux SMS de détecter le véhicule blindé. Après transmission de la piste au track-merger, ce dernier, suite à l'analyse des agents et de l'ensemble des données senseurs décidera de créer un nouvel agent tactique. Cet agent héritera des caractéristiques mesurées lors de la détection de l'objet par le senseur optronique (position, état, type, etc.). Cet agent exécutera sa machine à états tactique, le moteur régissant la pensée de tous les agents

tactiques. Son niveau de priorité sera calculé en fonction de ses caractéristiques recueillies.

Après l'ajout de cet agent dans le système, ce dernier virtualisera l'objet véhicule blindé et sera donc responsable des activités senseurs en lien avec cet objet. Il décidera d'acquérir plus de connaissances le concernant et créera les plans senseurs conformes à ses objectifs opérationnels.

L'ordonnanceur global, destinataire de ses plans senseurs, tentera de placer ses plans sur les timelines des ressources en fonction des priorités des plans, hérités de la priorité de l'agent.

4.2.4.2 Gestion du SMS en mission

En étudiant le scénario simple présenté plus haut, nous allons maintenant détailler comment différents éléments comme les politiques ou les ordres directs impactent le fonctionnement du SMS. Les éléments cités ci-après constituent les mécanismes mis en place afin de supporter les interactions avec l'opérateur, notamment les interactions décrites par les contraintes présentées en 5.

Les politiques :

Comme décrit précédemment en 4.2.3.1 : type de données 3, les politiques sont un ensemble de règles et permettent de contraindre l'ensemble du groupe d'agents. Elles peuvent être modifiées en cours de mission et se distinguent en plusieurs catégories.

Nous pouvons par exemple ajouter les politiques suivantes au scénario précédent :

1. Les renseignements ont annoncé la présence de radars de veilles et d'instruments d'écoutes RF sur le théâtre. Les phases de préparation de mission ont donc montré la nécessité pour la plateforme d'être furtive ou au moins silencieuse (minimum d'émission RF). La politique de restriction a donc été spécifiée en mode « furtive » et rendue accessible à l'ensemble des agents du Système multi-senseur.

   Pendant la phase de sélection des plans par les agents tactiques, les politiques actives sont analysées et les plans impliquant des émissions radio (Pointage SAR par exemple) sont éliminés du choix des plans possibles.
2. La politique d'autonomie a été positionnée sur 3/5, ce qui implique que les agents ont la possibilité de créer/transmettre des plans senseurs, déterminer leur objectif opérationnel et fonctionnel et que de nouveaux agents peuvent être créés et décider de leurs plans après la fusion de pistes sans intervention de l'opérateur.
3. La mission est du type reconnaissance, donc centrée sur la détection et l'identification des objets situés sur une zone géographique donnée. Ainsi, la politique comportementale a été définie sur « reconnaissance degré 1 », politique comportementale impliquant un niveau de priorité des agents élevé lorsqu'ils ont pour objectif opérationnel la détection et l'identification. À contrario, moins élevé sur des objectifs opérationnels de localisation précise et de poursuite.

L'ajout d'objets par le Gestionnaire de mission :

Le gestionnaire de mission a la possibilité d'ajouter des informations à la situation tactique pendant la mission. Puisque la situation tactique est ici représentée par le groupe d'agents tactique, l'ajout d'information se répercute directement sur ce dernier. Plus généralement toute requête du gestionnaire de mission au système de senseurs se répercute sur le groupe d'agent.

Par exemple, le GM veut acquérir une image d'un bâtiment situé à la $5^{ème}$ minute du plan de vol de la plateforme. Lors de la saisie de la requête, spécifiant le type de bâtiment (classe dimensionnelle) et sa position, l'ordre est transmis au SMS qui la traduit en termes d'agent. Un agent est alors créé avec pour connaissances les caractéristiques transmises avec la requête. L'agent va ensuite générer les plans au regard de l'ensemble des connaissances saisies.

Il est aussi possible pour l'opérateur de spécifier avec exactitude quel plan senseur l'agent doit sélectionner, afin d'outrepasser la décision de l'agent si souhaité.

La représentation des zones géographiques :

Lorsque l'opérateur souhaite faire réaliser au SMS une requête du type « rechercher l'ensemble des véhicules blindés dans la zone Z_a » il lui est impossible de préciser la position précise, le nombre et le type des véhicules. Un agent spécial a été introduit afin de réaliser ces opérations plus vaste que les simples pointages d'objets spécifiques. Cet agent spécial est appelé un agent zone. Cet agent choisit les plans senseurs présentant les caractéristiques adaptées à la requête et au contexte :

- Les dimensions de la zone : permet d'établir un découpage intelligent de la zone en fonction des couvertures des plans senseurs ;
- Le plan de vol : permet de choisir les plans senseurs en fonction de leur portée et de leurs scopes ;
- Le type d'objets à détecter : permet de choisir les plans senseurs en fonction de leur résolution et caractéristiques spécifiques (vitesses, etc.) ;
- Les paramètres plateformes : par exemple la vitesse influençant fortement les temps d'exécution des plans senseurs.

Les ordres directs aux senseurs : l'agent de priorité absolue :

Il doit pouvoir être demandé au SMS d'exécuter une fonction senseur en particulier, sans nécessiter la création d'un agent supplémentaire dans la situation tactique. Si l'opérateur désire réaliser une action senseur très spécifique ou simplement piloter un senseur en mode automatique, l'ensemble de la chaîne de décision des agents doit être outrepassée.

Cependant, l'utilisation d'un senseur et des ressources qu'il requiert ne peut se faire en envoyant simplement les instructions matérielles aux ressources matérielles concernées. Le reste du SMS fonctionnant de manière nominale, cette action créerait des interférences avec les plans senseurs provenant des agents tactiques. Pour cette raison un agent tactique de priorité absolue (priorité supérieure à tout autre agent du système) a été introduit. Cet agent tactique permet d'intégrer les plans senseurs stricts demandés par l'opérateur dans le reste du système sans créer d'incompatibilité de fonctionnement.

Les plans senseurs de l'opérateur sont alors les plus prioritaires du système et seront automatiquement placés par l'ordonnanceur 1.

4.3 Ordonnancement au sein du SMS

4.3.1 Mécanismes d'ordonnancement

4.3.1.1 Horizon temporel

L'ordonnancement est une opération coûteuse en énergie de calcul et est exécuté en un temps fonction du nombre de tâches à exécuter. Pour cette raison le nombre de tâches à ordonnancer à chaque instant doit être limité afin de garantir un résultat en un temps raisonnable pour le système ciblé. Dans le cas du SMS comme montré en section 4.2.3.3, chacune des étapes de la boucle d'asservissement est responsable du délai total de prise de décision. Pour cette raison, le délai de l'ordonnanceur doit, au même titre que les autres composants de la boucle décisionnelle, permettre de conserver une réactivité élevée au sein du système (section 4.1.2.2, besoin système 5). Le temps nécessaire à l'ordonnancement croît avec le nombre de plans et de tâches à ordonnancer. Inversement, le délai d'ordonnancement est plus court si le nombre de tâches est réduit. Cette simple observation tend à contraindre le nombre de tâches si l'on souhaite garder le temps d'exécution de l'ensemble de la boucle d'asservissement sous un certain seuil.

Cependant, il est aussi nécessaire pour le système de senseurs de planifier les futures actions sur un intervalle de temps le plus long possible afin de trouver des solutions à l'ordonnancement de meilleure qualité qu'un système sans ordonnancement temporel. Un ordonnancement plus long permet de découvrir des solutions de meilleure qualité sur une localité étendue tandis qu'une solution à un ordonnancement sur une fenêtre restreinte.

Face à ces deux contraintes antagonistes, un compromis doit être établi afin de satisfaire la contrainte de réactivité tout en étant capable de réaliser un ordonnancement temporel efficace.

Plusieurs méthodes permettent de contraindre le nombre de tâches à ordonnancer. Une solution est de fixer un horizon temporel sur lequel l'ordonnanceur devra travailler. Si les temps de libération et d'échéance des plans à traiter sont incompatibles avec la fenêtre temporelle, alors ces plans ne seront pas pris en compte pour le calcul de l'ordonnancement. Si au temps courant t, un ordonnancement ayant une fenêtre d'ordonnancement $[t; t + T_H]$ est lancé et si un plan de la file d'attente P_n avec une date de libération t_r et une durée minimale D_{p_n} et que $t_r + D_{p_n} > t + T_H$, alors il sera rejeté de la file d'attente et sera pris en compte lors d'un futur ordonnancement. Voir figure 4.36. En d'autres termes, le plan est considéré comme trop lointain dans l'avenir pour être ordonnancé à cette itération.

4.3.1.2 Workflow d'ordonnancement

L'impact de la forte dynamique de l'environnement sur les résultats de l'ordonnancement est fort : une modification de l'environnement peut engendrer une invalidation complète de l'ordonnancement calculé avant l'évènement et un ré-ordonnancement prenant en compte les anciens et nouveaux évènements. Voir figure 4.37.

Afin de prendre en compte les nouveaux évènements le plus rapidement possible, l'ordonnancement est déclenché dès lors qu'un plan émis par un agent est reçu par l'ordonnanceur sans synchronisation. La synchronisation de la réception des plans par la mise en place de fenêtres de soumissions fixes des plans impliquerait des

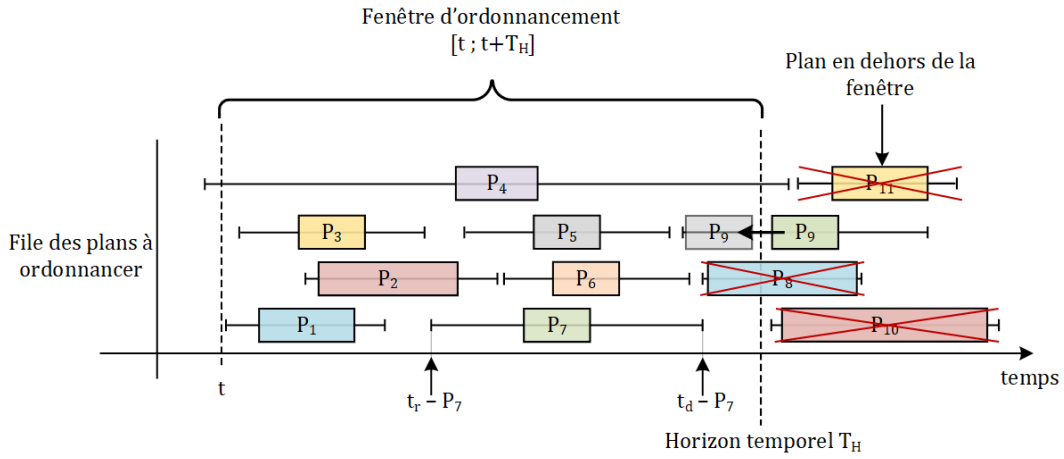


FIGURE 4.36 – Schéma de la file d'attente des plans à ordonnancer et de l'horizon temporel.

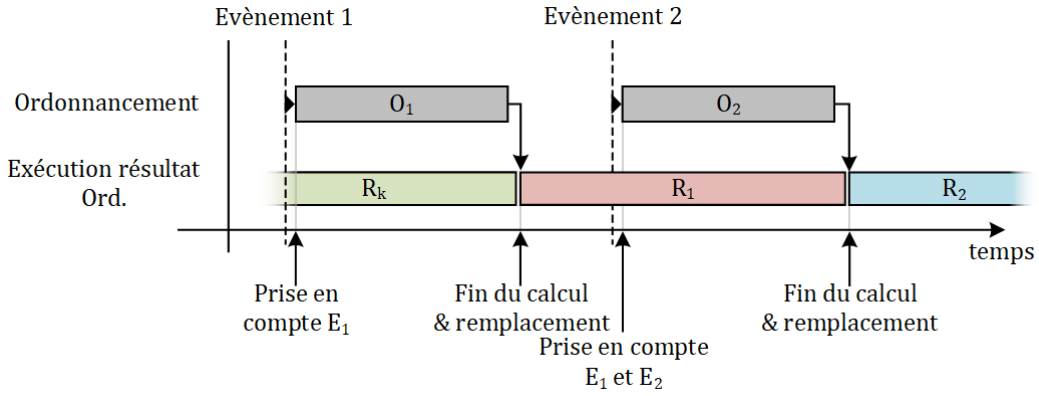


FIGURE 4.37 – Schéma des temps d'ordonnancement et exécutions de leurs résultats en fonction des évènements.

délais d'attente supplémentaires dans la boucle d'asservissement. Cette spécificité implique qu'un ordonnancement peut déjà être en cours alors qu'un plan prioritaire doit être pris en compte. Dans cette situation l'ordonnancement en cours doit être arrêté, son résultat invalidé et un nouveau calcul doit être lancé en prenant en compte les nouveaux plans, voir figure 4.38. Sur les figures 4.37 et 4.38, les rectangles O_1 et O_2 correspondent aux tâches d'ordonnancement. Les rectangles R_k correspondent à l'exécution normale des ordonnancements résultants des phases d'ordonnancement. L'ordonnancement actif à tout instant correspond au résultat de la dernière phase d'ordonnancement terminée. Une phase d'ordonnancement est préemptée si un évènement plus prioritaire doit être pris en charge avant la fin du calcul.

L'interruption et l'invalidation de l'ordonnancement systématique à chaque réception d'un plan agent peut engendrer une instabilité du système et empêcher de trouver des solutions dans le cas où les plans sont trop nombreux et arrivent à des intermédiaires réguliers. Pour résoudre ce problème, une solution est de préempter l'ordonnancement si le plan respecte certaines caractéristiques déterminées en fonction des plans présents dans l'ordonnancement en cours. Par exemple, si 10 plans d'une importance basse sont réceptionnés et qu'un 11ème plan de haute importance est reçu, alors l'ordonnancement peut être relancé en prenant en compte le nouveau

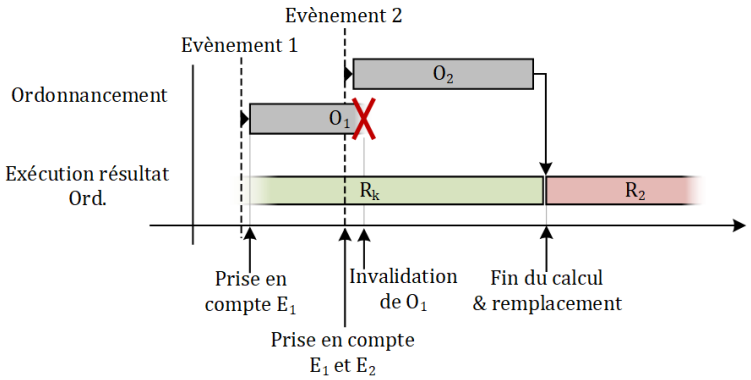


FIGURE 4.38 – Schéma des temps d'ordonnancement avec deux évènements rapprochés.

plan. Si le plan ne respecte pas les critères, alors l'ordonnancement n'est pas interrompu et le plan sera pris en compte lors du prochain lancement.

4.3.2 Plans senseurs

4.3.2.1 Tâche

Une tâche correspond à une réservation d'une ressource matérielle ou virtuelle pendant un intervalle de temps. Pour une ressource matérielle, une tâche renvoie vers une instruction ressource matérielle (pour les senseurs et URL), donc au travail précis à réaliser pendant l'intervalle de temps réservé. Une réservation de ressource a pour objectif de réserver le temps de fonctionnement de la ressource et d'interdire la réservation de la ressource par d'autres tâches. La figure 4.39 montre une tâche T_i appartenant à un plan P_k réservée sur une ressource r_n.

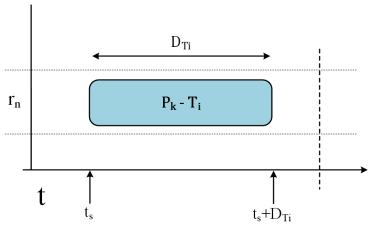


FIGURE 4.39 – Exemple d'une tâche.

La tâche T_i est caractérisée par un temps de début t_s, une durée D_{T_i} pouvant être variable, voir 4.3.2.2, une fin $t_e = t_s + D_{T_i}$ et une ressource r_n.

4.3.2.2 Durée des tâches

Les tâches représentent de réels travaux réalisés par les ressources. Cela implique que le temps de traitement peut être variable en fonction du contexte. Un exemple concret sera proposé en 4.3.2.5 lors du calcul de la durée des tâches du plan SAR.

4.3.2.3 Structure d'un plan

Un plan senseur est composé d'un ensemble de tâches ordonnées par un ensemble de contraintes liant les tâches les unes par rapport aux autres. La figure 4.40 montre un plan constitué de 4 tâches à exécuter sur 4 ressources distinctes.

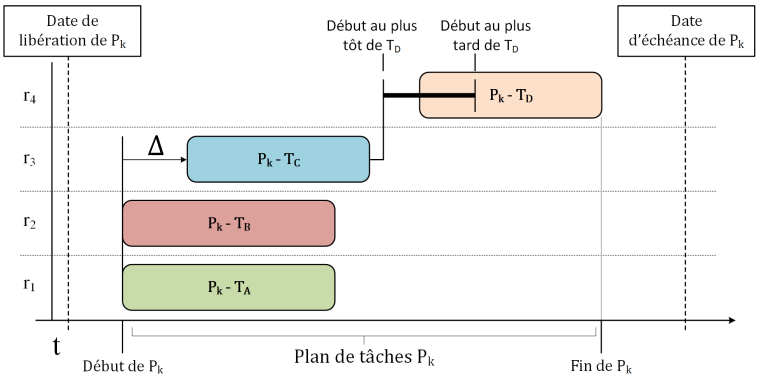


FIGURE 4.40 – Schéma exemple d'un plan senseur.

Cet exemple de plan présente toutes les contraintes pouvant être rencontrées sur l'ensemble des plans senseur :

- Les dates de libération et d'échéances du plan t_r et t_d ;
- Des contraintes de début de tâches début-début ;
- Des contraintes de début de tâches fin-début ;
- Des contraintes de délais exacts de début de tâches (Δ) ;
- Des contraintes de début de tâches au plus tôt et au plus tard ;
- Le plan est pondéré d'une priorité P.

4.3.2.4 Contraintes

Chacune des contraintes reflète une réalité fonctionnelle du SMS :

Les dates de libération et d'échéance du plan t_r et t_d correspondent aux dates auxquelles le plan est réalisable par le SMS. En dehors de cet intervalle, le plan ne pourra être accompli pour une raison de compatibilité des senseurs avec le contexte : la trajectoire de la plateforme et de l'objet à traiter peuvent forcer l'exécution du plan dans l'intervalle. La figure 4.41 montre les limites temporelles d'un plan.

Les contraintes de début de tâches début-début permettent de synchroniser l'exploitation de deux ressources en même temps, voir figure 4.42. La mise en place des ressources virtuelles (4.2.3.1, composant 11) explique la réservation synchrone d'une ressource et de la grandeur physique qu'elle exploite. Par exemple, un plan de brouillage par une antenne sur une bande RF donnée implique de construire un plan réservant de manière synchrone l'antenne et la bande de fréquence. Après allocation des ressources par l'ordonnanceur aucun autre plan qui requiert cette bande de fréquences ne pourra la réserver.

Les contraintes de début de tâches fin-début permettent d'imposer l'exécution d'une tâche avant une autre, voir figure 4.43. Par exemple, un plan de capture d'image optique nécessite une phase de capture par un capteur et une phase de traitement par

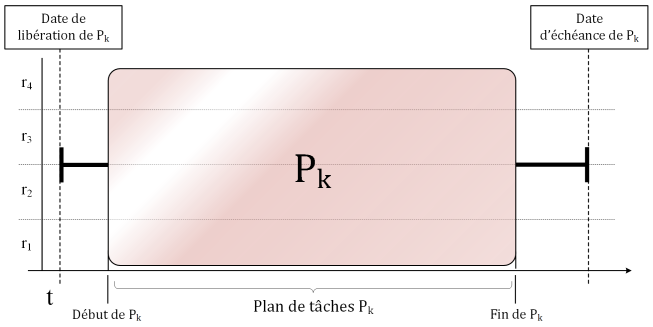


FIGURE 4.41 – Exemple du plan et de ses dates de libération et d'échéance.

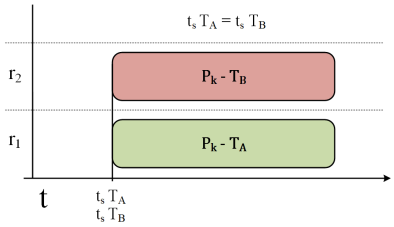


FIGURE 4.42 – Exemple de deux tâches synchrones, avec t_s égaux.

un processeur. Il va de soi que la phase de traitement ne peut être réalisée qu'après la phase de capture.

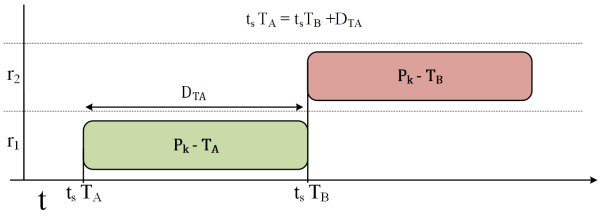


FIGURE 4.43 – Exemple de deux tâches successives

Les contraintes de délais de début de tâches (Δ) exacts sont utiles à l'exécution de plan pour lesquels le traitement du signal capturé peut être débuté suite à un délai exact après le début de la tâche, voir figure 4.44. Cette contrainte est par exemple utilisée par le plan SAR défilant, pendant lequel le signal RF réceptionné par l'antenne active est traité au fur et à mesure du défilement de la zone éclairée.

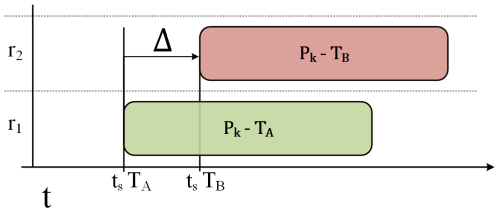


FIGURE 4.44 – Exemple de deux tâches aux temps de débuts retardés de Δ.

Il est dans certains cas nécessaires de ne pas contraindre une tâche à être réalisée directement à la suite de la précédente comme vu sur la figure 4.43 pour plusieurs

raisons :

- Un ordonnancement d'un ensemble de plans est fortement contraint et complexe à résoudre. Autoriser un intervalle de délai pour des tâches successives permet de relâcher les contraintes et d'obtenir plus de solutions au problème d'ordonnancement.
- Dans le cas d'une capture de signal et du traitement, un délai entre les deux tâches est acceptable et impacte peu le fonctionnement du système de senseurs.
- Un début au plus tôt de la seconde tâche peut être exigé s'il est nécessaire de prendre en compte les temps de transfert de données entre les instruments (ex. : capteur et URL de traitement) des deux opérations.

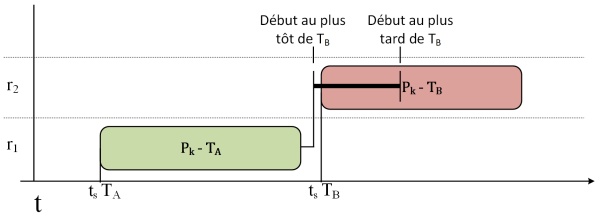


FIGURE 4.45 – Exemple d'une tâche avec la seconde tâche devant démarrer dans un intervalle de temps donné.

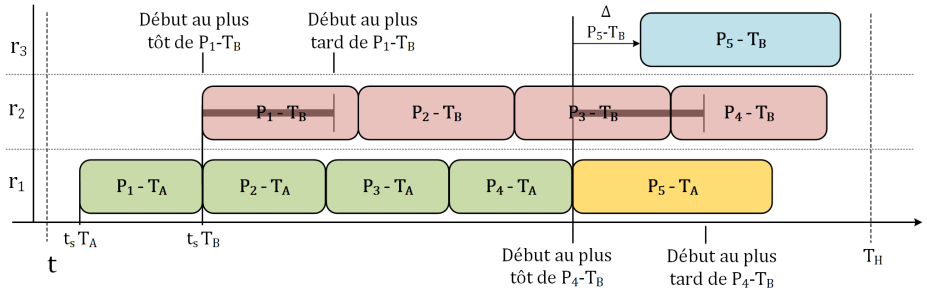


FIGURE 4.46 – Exemple d'un ordonnancement de 5 plans de tâches à contraintes flexibles.

Ce type de contrainte entre tâches est schématisé sur la figure 4.45. La figure 4.46 montre un ordonnancement de 5 plans : 4 plans de tâches avec des contraintes par intervalles de début P_1, P_2, P_3, P_4 et un plan de tâches avec un temps de début retardé. Résoudre un tel ordonnancement est possible à la fois grâce à la souplesse apportée par l'autorisation de retards entre les tâches et une définition stricte de l'ordre des tâches par les plans senseurs. Les espaces de stockages et différents buffers (mémoires tampons) étant limités au sein de chaque instrument face à la quantité de données en sortie de captures de senseurs, il est nécessaire de conserver un ordre strict entre les successions de tâches captures/traitements. En d'autres termes, il peut être impossible pour un type d'architecture matériel d'emmagasiner l'ensemble des captures en mémoire pour ensuite lancer le traitement dans un ordre quelconque.

4.3.2.5 Exemple concret de la construction d'un plan

Un exemple de plan présentant l'ensemble des contraintes va maintenant être présenté. Un plan du type SAR (Synthetic Apperture Radar, radar à ouverture synthétique), plus particulièrement le SAR-Pointé (ou SAR-Spot en anglais), est un plan

largement utilisé en mission. Il permet d'acquérir des images moyennes ou haute résolution de grandes zones du terrain à une distance pouvant aller jusqu'à 150km dans n'importe quelles conditions météorologiques.

La phase de transformation par un agent d'un plan provenant de la base de données des plans en un plan exploitable par les ressources et ordonnanceur du SMS est appelée *compilation de plan* puisqu'elle permet de transformer une donnée abstraite et générique en une donnée adaptée à la cible (ressources) et au contexte (environnement, données plateformes, etc.).

Un SAR-Spot est une action senseur exploitant :

- Une antenne active afin d'illuminer une zone du terrain, recueillir l'écho électromagnétique réfléchi par les matériaux du sol et des objets du terrain ;
- Un processeur de calcul qui analyse le signal capturé par l'antenne et transforme les ondes réfléchies affectées par l'effet Doppler en une image interprétable par un opérateur.

Ainsi la prise d'une image par SAR-Pointé est réalisée en deux phases, une phase de capture et une phase de traitement, voir figure 4.47. Lors d'une imagerie SAR, $T_i \approx T_p$ où T_i est le temps d'illumination et T_p le temps de traitement.

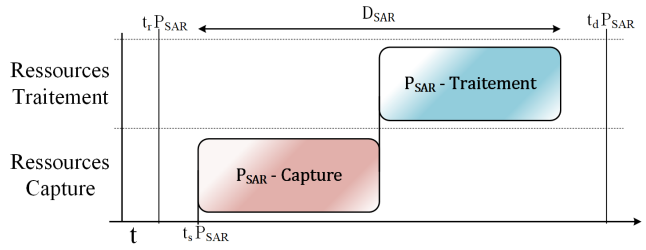


FIGURE 4.47 – Schéma des phases d'une prise d'image SAR-Pointé.

Phase de capture

Les signaux permettant de rendre l'image SAR doivent être acquis dans une plage d'angles particulière, afin que la vitesse de la plateforme et son défilement au fil de la trajectoire permettent de mesurer des distorsions du signal source une fois réfléchi par le terrain 4.48.

Dans [Merrill, 1990, Chapitre 17], la définition d'une image SAR est donnée par la formule 4.3

$$\delta_{rl} \approx \frac{R\lambda}{D} \tag{4.3}$$

Avec δ_{rl} la résolution latérale, R la distance plateforme/zone illuminée, λ la longueur d'onde du signal émis par l'antenne et D la largeur de l'antenne.

Pour une antenne de dimension $R = 1m$ fonctionnant en bande X, $\lambda = 3cm$, à une distance $R = 100km$, la définition de l'image SAR serait de $\delta_{rl} \approx 3km$ ce qui correspondrait à une définition permettant seulement de tracer les contours de grands bâtiments.

En capturant l'image sur toute la longueur du défilement, l'ouverture de l'antenne équivaut à celle d'une antenne beaucoup plus grande. Cette ouverture équivalente

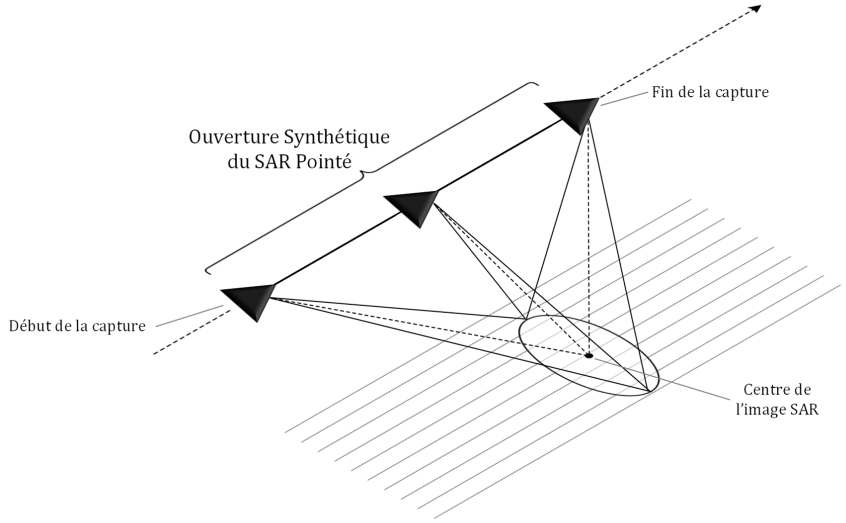


FIGURE 4.48 – Schéma d'une acquisition d'une image SAR-Spot.

permet d'acquérir des images d'une résolution bien plus fine qu'avec la même antenne sans défilement.

La définition de la nouvelle antenne équivalente est décrite par l'équation 4.4 :

$$\delta_{rl} \approx \frac{R\lambda}{2L_{os}} \tag{4.4}$$

Avec L_{SA} la distance de la prise de vue, aussi appelée l'ouverture synthétique, λ la longueur d'onde du signal, R la taille de l'antenne.

Nous pouvons comparer la définition obtenue avec la même antenne que précédemment avec une ouverture synthétique, soit $R = 1m$, $\lambda = 3cm$ à une distance $R = 100km$ sur une ouverture synthétique $L_{os} = 10km$ donc un temps de prise de vue $t = 36s$ pour une plateforme se déplaçant à $v = 277m.s^{-1}$ ($1000km.h^{-1}$). La définition de cette antenne à ouverture synthétique serait de $\delta_{rl} \approx 15cm$, une définition qui permet d'identifier finement les bâtiments et véhicules avec une Surface Equivalente Radar (SER).

Nous voyons que cette fonction dépend, entre autres, de la vitesse à laquelle la plateforme se déplace sur le théâtre et la distance avec l'objet à illuminer.

Un besoin opérationnel du type Identification d'objet sur le théâtre nécessite une définition constante indépendamment de la distance avec l'objet. Dans ce cas, la définition δ_{rl} est une constante pour le type d'identification à fournir par le SMS à l'opérateur. En reprenant l'équation 4.4, à définition constante δ_{rl} et trajectoire rectiligne à vitesse constante $v_p = \frac{L_{os}}{T_i} = 277m.s^{-1}$, le temps d'illumination T_i de la zone sur laquelle se trouve l'objet est de la forme de l'équation 4.5, voir figure 4.49 :

$$T_i \approx \frac{R\lambda}{2v_p\delta_{rl}} \tag{4.5}$$

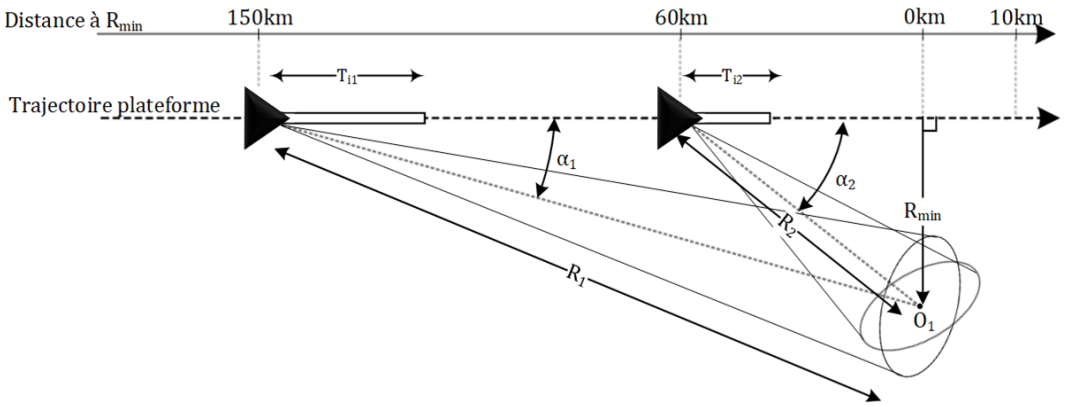


FIGURE 4.49 – Schéma représentant les temps d'illumination pour une prise d'image SAR en deux points d'une trajectoire rectiligne.

Le temps d'illumination dépend alors uniquement du moment de déclenchement de la capture du SAR. Il est possible de représenter ce temps de capture en fonction de l'instant t_d de déclenchement du SAR. La distance à l'objet correspond à la distance euclidienne entre la plateforme et l'objet. Le SAR étant incompatible avec une capture frontale, l'objet ne peut se trouver exactement sur la trajectoire et dans un angle inférieur à $\approx 15\,\mathrm{deg}$. Ce domaine de compatibilité du SAR implique que la plateforme ne sera jamais à une distance proche nulle de l'objet contrairement à une capture d'image optronique.

Le domaine de compatibilité du SAR est donné par le jeu de contraintes 4.6.

$$Dom_{SAR} = \begin{cases} 15 < \alpha < 120 & \alpha : \text{angle de gisement avec la cible en } \mathrm{deg} \\ -60 < \beta < 60 & \alpha : \text{angle de site avec la cible en } \mathrm{deg} \\ R < 150 & R : \text{distance à l'objet en } km \end{cases} \tag{4.6}$$

Pour une distance latérale à la trajectoire R_{min} et une distance à la projection de la position de l'objet sur la trajectoire D, l'expression de la distance à l'objet R est de la forme de l'équation 4.7. Il est alors possible de déterminer l'angle avec lequel l'objet est vu par la plateforme, la distance à l'objet en fonction de la progression de la trajectoire et en déterminer la compatibilité du plan avec la trajectoire. Voir figure 4.50.

$$R = \sqrt{D^2 + R_{min}^2} \tag{4.7}$$

Le domaine de compatibilité du plan permet de déterminer les contraintes principales du plan : ici l'angle d'observation de l'objet est le paramètre déterminant pour obtenir les dates de libération et d'échéance du plan. La distance D à l'objet est parcourue au fil du temps de vol sur la trajectoire de la plateforme ici rectiligne uniforme. La date de libération du plan sera le temps t_r à laquelle la plateforme est à $D = 75km$ de l'objet et la date d'échéance du plan t_d correspond au temps auquel la plateforme est à $D = 12km$ de distance.

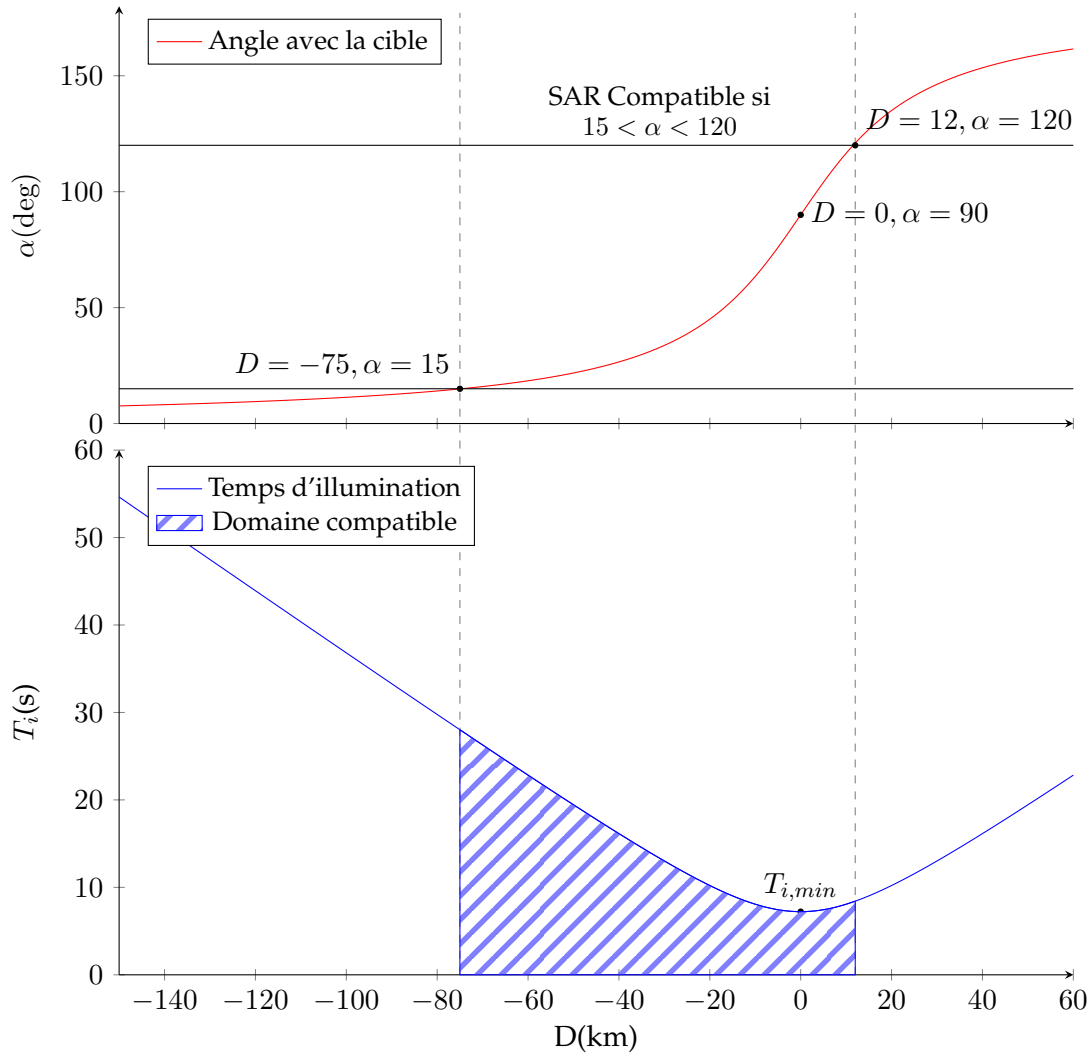


FIGURE 4.50 – Angle de vue et temps d'illumination en fonction de l'instant de déclenchement. Domaine de compatibilité de la capture SAR.

En termes de planification, la tâche à réserver est donc définie par les t_r t_d et la durée de la tâche dépendant de l'instant de déclenchement du SAR. Le positionnement précis du déclenchement est défini par l'ordonnanceur après transmission du plan par l'agent. Ceci implique que les instructions transmises à l'ordonnanceur pour la phase d'illumination contiennent l'ensemble des informations pour placer la tâche dans le temps.

La bande de fréquences des signaux permettant la capture du SAR dépend du type d'antenne et du matériel hyperfréquence disponible. Afin d'éviter des incompatibilités avec d'autres fonctions du SMS utilisant la même bande de fréquences une ressource virtuelle doit être réservée en même temps que la tâche d'illumination (contrainte de début-début ou tâches synchrones).

La phase d'illumination du terrain requiert un apport d'énergie électrique relativement élevé. Ainsi il peut être impossible lors de l'intervalle de temps capture du SAR de réaliser d'autres opérations aussi gourmandes en énergie. Cette contrainte peut être résolue de deux façons :

1. L'ordonnancement prend en charge la réservation quantitative des ressources et gère le niveau d'occupation maximal à ne pas dépasser. Dans ce cas il est possible d'attribuer une consommation énergétique à chacune des tâches senseurs. L'ordonnancement permettra de calculer automatiquement quelles tâches peuvent être exécutées en même temps. Cette solution peut aller jusqu'à doubler le nombre de tâches dans le pire des cas.
2. L'ordonnancement ne prend pas en charge la réservation quantitative. Il est possible de différencier les tâches fortement énergivores des autres tâches. Il faut ensuite introduire une ressource permettant d'exclure le fonctionnement des ressources mutuellement, à la façon d'une primitive de synchronisation de type Mutex.

Le premier cas est à préférer si les tâches réalisées par le système ont toute une consommation proche ou un spectre de consommation large. Cependant, au sein du SMS, une tâche telle que la veille GE a une consommation négligeable face à celle d'une tâche de brouillage et la capture d'un SAR. La deuxième solution a donc été privilégiée de façon à simplifier les premières versions de l'ordonnanceur et diminuer le nombre de tâches à ordonnancer.

La ressource introduite pour faire office de Mutex entre la fonction de brouillage et la capture d'un SAR est appelée « Energie > 50% » ou « Energie »/*Power* en version courte en anglais dans le reste du document. La réservation de cette ressource doit être synchronisée avec la tâche d'illumination, voir figure 4.51.

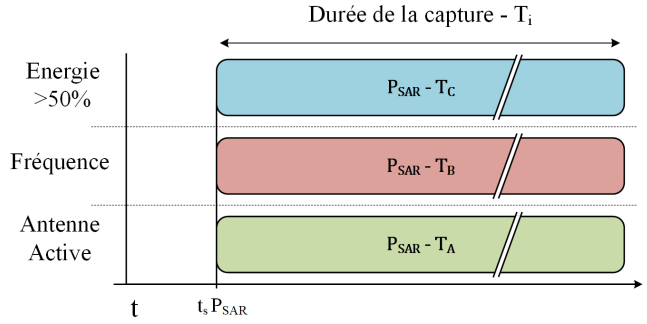


FIGURE 4.51 – Schéma des tâches de la phase de capture.

Phase de traitement

La phase de traitement consiste à traiter l'ensemble des signaux capturés par l'antenne lors de la phase d'illumination afin de les transformer en une image exploitable pour un opérateur. Voir figure 4.52.

La phase de traitement des signaux est réalisée par l'envoi d'une instruction matérielle à un processeur ainsi que la transmission des données acquises lors de la phase précédente. Un processeur appelé « SAR Processing Unit», unité de traitement SAR, est réservé pour réaliser le traitement, voir figure 4.53.

Les signaux acquis lors de la première phase peuvent être conservés, stockés et réutilisés indépendamment de la phase de traitement. En effet il pourrait être envisageable pour certaines missions de réaliser plusieurs acquisitions opportunistes à exploiter au sol en fin de mission. Dans ce cas un nouveau plan « SAR Acquisition » devra être écrit afin de disposer de cette possibilité pendant la mission. Néanmoins

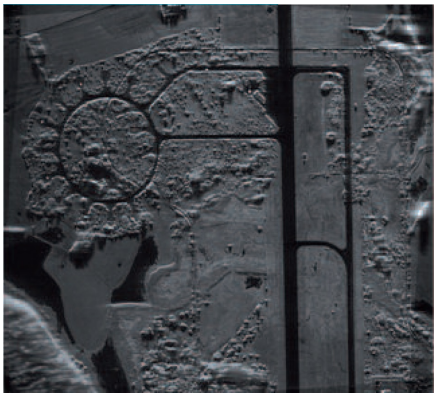


FIGURE 4.52 – Image SAR d'une base aérienne.

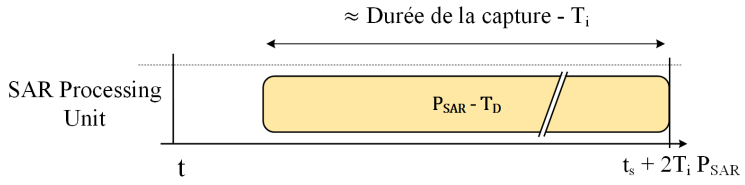


FIGURE 4.53 – Schéma de la phase de traitement du SAR.

il est possible de retarder le traitement de plusieurs secondes en fonction des capacités de stockage court-terme des composants de l'architecture, dont celle de l'unité de traitement SAR ou autres mémoires déportées.

Un plan de tâches SAR-Pointé complet peut alors se présenter comme un ensemble de 4 tâches avec des contraintes de trois tâches synchrones et une au temps de début retardé, voir figure 4.54.

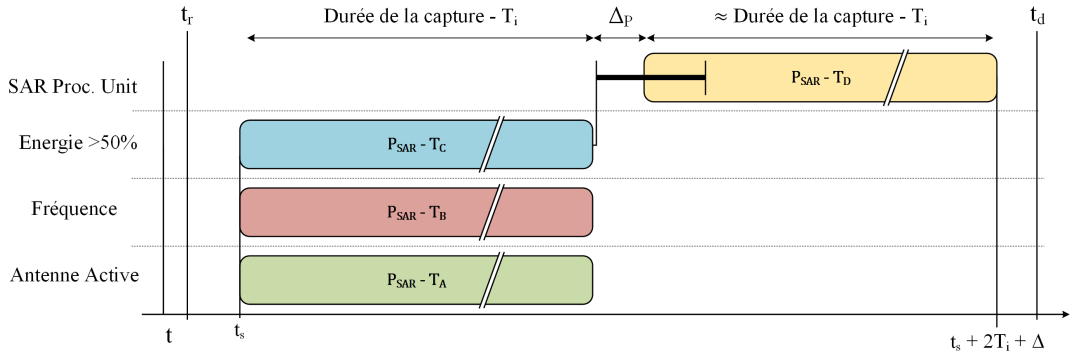


FIGURE 4.54 – Schéma des tâches d'un plan SAR.

Les plans SAR-Pointé et SAR-Défilant sont, à l'heure de l'écriture de ce manuscrit, les plans les plus complexes pour un SMS du point de vue du nombre de tâches, calculs de leurs durées et de contraintes. La sous-section 6.1.2.2 relative à la modularité il sera montré comment ces plans permettent de mettre en place des actions senseurs bien plus complexes.

Les contraintes du plan peuvent s'écrire sous la forme du set de contraintes 4.8 :

$$\mathcal{C}_{SAR} = \begin{cases} t_{s_A} > t_r \\ t_{s_A} = t_{s_B} = t_{s_C} \\ t_{e_A} < t_{s_D} < t_{e_A} + \Delta \\ t_{s_D} + D_D < t_d \end{cases} \tag{4.8}$$

Avec $t_{e_k} = t_{s_k} + D_k$.

Chaque calcul permettant de construire l'ensemble des contraintes est réalisé par des entités différentes du SMS. Le tableau 4.2 répertorie les différentes phases de construction des plans et de ses contraintes au sein du SMS.

Donnée calculée	Nom	Composant logiciel source
Priorité des tâches/plans	Π_k	Fixé par l'agent par des règles basées sur les règles opérationnelles
Contraintes de libération et d'échéances	t_r, t_d	Calculées par l'agent après projection de la trajectoire de l'objet virtualisé et de la plateforme
Durée des tâches	D_k	Fonction de calcul de durées mise à la disposition de l'agent. Le D_k est fonction de t_{s_k}
Temps de début des tâches	t_{s_k}	Calculés par l'ordonnanceur en fonction de la priorité de chaque plan, du temps ressource disponible, des contraintes du plan et de la durée de la tâche en chargeant le t_s dans le tableau des D_k.

TABLE 4.2 – Tableau des données calculées et des entités logicielles sources

4.3.3 Modèle d'ordonnancement choisi

4.3.3.1 Description de l'ordonnancement

La caractéristique asynchrone des agents implique que l'ensemble des plans créés par les agents peuvent être appelés à être ordonnancés à tout moment de l'exécution du SMS. L'ordonnanceur (aussi appelé scheduler) est une entité logicielle au fonctionnement mécanique qui prend en son entrée des plans à ordonnancer et délivre en sortie les dates de départ des tâches ordonnancées ainsi que la liste des tâches et plans n'ayant pu être ordonnancés, voir figure 4.55.

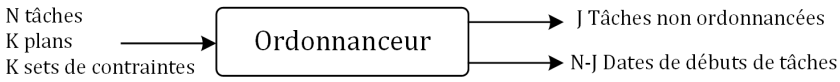


FIGURE 4.55 – Schéma de l'ordonnanceur.

Une entité de prise en compte et analyse des plans a été introduite sous le nom de gestionnaire d'ordonnancement. Cette entité a pour objectif de réceptionner l'ensemble des plans, comparer les priorités des nouveaux plans avec ceux déjà introduits dans le scheduler et lance un nouvel ordonnancement si la priorité des plans dépasse un seuil de priorité calculé à partir de la priorité moyenne des plans en cours de calcul. Ainsi si un plan d'importance critique doit être pris en compte directement après sa construction un nouvel ordonnancement permettra de l'insérer rapidement dans le flux de plans. Ce fonctionnement explique le workflow présenté en sous-section 4.3.1.2.

La qualité de l'ordonnancement sortant peut être mesurée par l'évaluation de deux critères :

1. Priorité statistique : La priorité globale et le nombre de plans ordonnancés doivent être maximisés.
2. Temps d'occupation des ressources : Le temps total pendant lequel chacune des ressources est non exploitée doit être minimisé.

4.3.3.2 Heuristique

Plusieurs méthodes permettent de mettre en place des heuristiques d'ordonnancements pour des problèmes du type RCPSP. Deux modèles de tâches ont été envisagés. Le premier modèle discrétise le temps et divise les tâches en parts élémentaires. La durée d'une part élémentaire est définie en fonction de l'action senseur la plus courte. Cette modélisation des tâches implique que l'ordonnanceur traite uniquement des lots de tâches élémentaires qu'il doit ranger dans le temps en respectant les contraintes. Une tâche est alors divisée en unités. Plusieurs spécificités du contexte rendent ce modèle complexe à appliquer :

- Les différences de durées entre la tâche la plus courte et les tâches les plus longues sont trop importantes. Pour une tâche élémentaire de 50ms (action de déclenchement des leurres pyrotechniques par exemple), la tâche la plus longue est de l'ordre de 15s (capture SAR). Ici la tâche la plus longue peut donc être divisée en 300 unités, insérant ainsi 300 tâches dans le processus d'ordonnancement ;
- La durée moyenne des tâches est beaucoup plus grande que la durée de la tâche élémentaire ;
- L'ensemble des tâches sont majoritairement indivisibles, ce qui implique qu'un grand nombre de tâches doivent être consécutives ;
- Le calcul des durées de tâches ne garantit pas un résultat multiple de la durée de la tâche élémentaire, mais s'en approche si la granularité est augmentée, augmentant ainsi le nombre d'unités maximal des tâches.

En conclusion le modèle de tâches en temps discrétisé apporte dans ce contexte un très grand nombre de tâches élémentaires à traiter et donc une forte complexité mémoire et calculatoire. Cependant, ce modèle, avec une granularité assez fine, peut aussi permettre de résoudre la problématique du micro-ordonnancement des senseurs.

Le modèle de tâches choisi respecte la description faite en 4.3.2.1. Une tâche est considérée continue, indivisible et définie par des variables temporelles réelles.

Formalisation du problème d'ordonnancement : Un plan de tâches Π_k est constitué d'un ensemble de tâches tel que :

$$\Pi_k = (J_1^k, \ldots, J_{n_k})$$

Le plan Π_k ne peut être ordonnancé si au moins une de ses tâches J_i^k ne peut être ordonnancée.

La relation de précédence et succession entre deux tâches/plans s'écrit par les opérateurs $\prec$ et $\succ$ respectivement. Exemple : $J_i^k \prec J_j^k$ ou $\Pi_k \succ \Pi_j$.

Une tâche J_i^k est une opération atomique qu'une ressource doit exécuter. Chaque tâche J_i^k appartient à un unique plan Π_k. Une tâche J_i^k est caractérisée dans cette heuristique par les éléments :

- Une durée d'exécution $p_i^k \in \mathbb{N}_0, p_i^k \geq 1$;
- Une date de libération $r_i^k \in \mathbb{N}$
- Une date d'échéance $d_i^k \in \mathbb{N}_0$
- Un délai de début $\delta_{ij}^k \in \mathbb{N} si J_i^k \prec J_j^k$
- Un set de ressources $\mathcal{R}_i^k = \{\rho_1, \ldots, \rho_n\}, \mathcal{R}_i^k \subseteq \mathcal{R}^k$

La spécificité des tâches pouvant être sur plusieurs ressources, c'est-à-dire que pour $\mathcal{R}_i^k = \{\rho_1, \ldots, \rho_n\},$ où $n > 1$ permet de traiter le cas des tâches synchrones présentées en figure 4.42 section 4.3.2.4.

Contraintes temporelles :

Lorsqu'un plan Π_k est ordonnancé pour chaque tâche $J_i^k \in \Pi_k$ le temps de début s_i^k et de fin/complétion $C_i^k = s_i^k + p_i^k$ doivent être dans la fenêtre temporelle $[W_s, W_e]$ ainsi que dans la fenêtre temporelle $[r_i^k, d_i^k]$ où :

$$(r_i^k \geq W_s) \wedge (d_i^k \geq W_e), \forall J_i^k \in \Pi_k \tag{4.9}$$

La valeur de s_i^k pour une tâche $J_i^k \in \Pi_k$ est calculée comme la valeur maximale de W_s, la date de libération r_i^k de la tâche J_i^k augmentée du délai de début δ_{ij}^k et le temps de complétion C_i^k pour chaque tâche précédant $J_j^k \in \Pi_k$. Soit formellement :

$$s_i^k = max(W_s, r_i^k, max_{j \in pred_i^k}(C_j^k + \delta_{ij}^k)), \tag{4.10}$$

où $pred_i^k$ est l'ensemble des tâches précédant la tâche J_i^k.

Pour un ensemble de plans $\mathcal{P}$ le plan $\Pi_k \in \mathcal{P}$ satisfait les contraintes temporelles si

$$\exists s_i^k \forall J_i^k \in \Pi_k : (s_i^k \geq r_i^k) \wedge (s_i^k + p_i^k \leq d_i^k). \tag{4.11}$$

Contraintes sur les ressources :

Le problème d'ordonnancement traité considère que chaque tâche J_i^k avec un temps de début dans un ordonnancement réalisable peut être exécutée sur un sous-ensemble $\mathcal{R}_i^k \subseteq \mathcal{R}^k$ de ressources si dans l'intervalle de temps $[S_i^k, C_i^k]$ l'utilisation de chaque ressource $\rho \in \mathcal{R}_i^k$ ne dépasse pas la disponibilité de la ressource B_ρ.

L'heuristique qui a été retenue est dite orientée évènements. Ce type d'heuristique symbolise l'ordonnancement en termes d'évènements de débuts et de fins de tâches plutôt que de gérer les lignes temporelles des ressources et les tâches de manière discrète. Lors de la construction de l'ordonnancement, une liste d'évènements $\mathcal{EL}$ est construite. Une liste d'évènements ayant pour entrée la date de chaque évènement, le début, la fin et la consommation des tâches pour chacune des ressources, voir [Guastella, 2017].

Un évènement e est un 4-uplet $(t(e), \mathcal{S}(e), \mathcal{C}(e), (q_\rho(e))_{\rho \in \mathcal{R}})$ où :

- $t(e)$ indique la date à laquelle l'évènement a lieu ;
- $\mathcal{S}(e)$ indique l'ensemble des tâches démarrant exactement à $t(e)$;
- $\mathcal{C}(e)$ indique l'ensemble des tâches terminant exactement à $t(e)$;
- $q_\rho(e)$ indique la consommation de la ressource ρ sur l'intervalle $[t(e), t_{next}(e)]$ où $t_{next}(e)$ est la date du premier évènement suivant e dans $\mathcal{EL}$.

La liste d'évènement 4.3 correspond aux évènements décrivant l'ordonnancement schématisé en figure 4.56.

L'ordonnanceur construit la liste d'évènements en y ajoutant et en modifiant des évènements déjà présents dans la liste.

Ici l'évènement $e1$ au temps 2 correspond au début de la tâche A consommant la ressource $q_2(e)$ (ici $B_\rho = 1$ et les ressources sont unitaires et indivisibles). À l'évènement $e2$ au temps 7 la tâche A prend fin et la ressource $q_2(e)$ est libérée (1/1 à 0/1).

Les figures 4.56, 4.57 et 4.58 décrivent trois étapes successives de la construction d'un ordonnancement par l'ordonnanceur. Les listes d'évènements correspondantes sont respectivement représentées par les tableaux 4.3, 4.4 et 4.5.

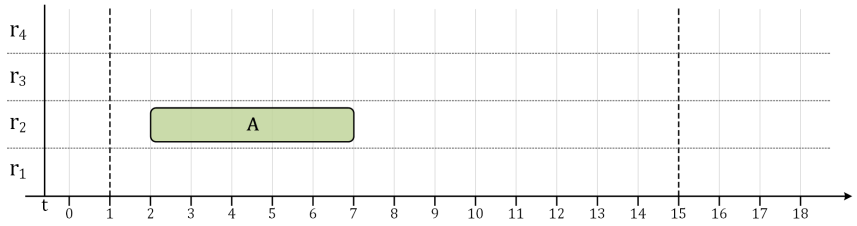


FIGURE 4.56 – Représentation d'une tâche ordonnancée sur un diagramme de Gantt.

e	$t(e)$	$\mathcal{S}(e)$	$\mathcal{C}(e)$	$q_1(e)$	$q_2(e)$	$q_3(e)$	$q_4(e)$
$e1$	2	$\{A\}$	$\emptyset$	0	1	0	0
$e2$	7	$\emptyset$	$\{A\}$	0	0	0	0

TABLE 4.3 – Liste d'évènements de l'ordonnancement présenté sur la figure 4.56.

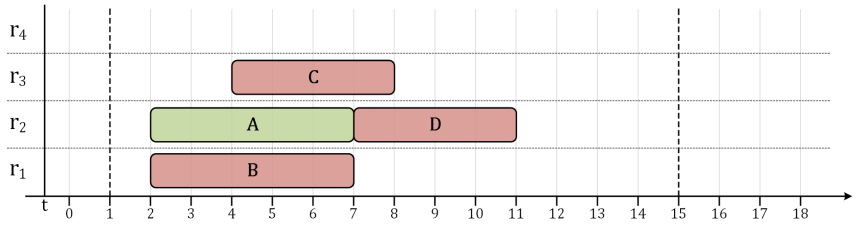


FIGURE 4.57 – Diagramme de Gantt de l'insertion d'un plan dans un ordonnancement.

L'ordonnanceur prend en entrées un ensemble de tâches constituant des plans. L'ensemble des contraintes de précédences et de démarrages des tâches sont prises en compte par le scheduler de la même façon que les contraintes de libérations et des dates limites de chaque plans.

e	$t(e)$	$\mathcal{S}(e)$	$\mathcal{C}(e)$	$q_1(e)$	$q_2(e)$	$q_3(e)$	$q_4(e)$
$e1$	2	$\{A, B\}$	$\emptyset$	1	1	0	0
$e2$	4	$\{C\}$	$\emptyset$	1	1	1	0
$e3$	7	$\{D\}$	$\{A,B\}$	0	1	1	0
$e4$	8	$\emptyset$	$\{C\}$	0	1	0	0
$e5$	11	$\emptyset$	$\{D\}$	0	0	0	0

TABLE 4.4 – Liste d'évènements de l'ordonnancement présenté sur la figure 4.57 après ajout des tâches B, C, D.

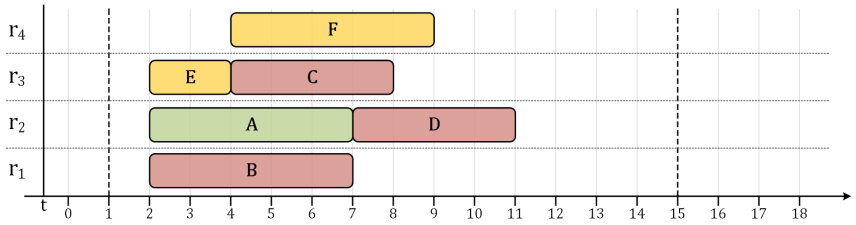


FIGURE 4.58 – Insertion d'un plan dans un ordonnancement réalisable.

e	$t(e)$	$\mathcal{S}(e)$	$\mathcal{C}(e)$	$q_1(e)$	$q_2(e)$	$q_3(e)$	$q_4(e)$
$e1$	2	$\{A, B, E\}$	$\emptyset$	1	1	1	0
$e2$	4	$\{C,F\}$	$\{E\}$	1	1	1	1
$e3$	7	$\{D\}$	$\{A, B\}$	0	1	1	1
$e4$	8	$\emptyset$	$\{C\}$	0	1	0	1
$e5$	9	$\emptyset$	$\{F\}$	0	1	0	0
$e6$	11	$\emptyset$	$\{D\}$	0	0	0	0

TABLE 4.5 – Liste d'évènements de l'ordonnancement présenté sur la figure 4.58 après ajout des tâches E,F.

L'ordonnancement consiste à optimiser la charge des ressources afin de perdre le moins possible de temps d'utilisation de ces dernières. Les termes « temps d'inactivité» ou Idle-time décrivent les intervalles de temps durant lesquels une ressource n'est pas exploitée. La figure 4.59 montre un ordonnancement réalisable de deux tâches j et i. L'intervalle de temps $[C_j, S_i]$ est un temps d'inactivité puisqu'une tâche l peut y être ordonnancée.

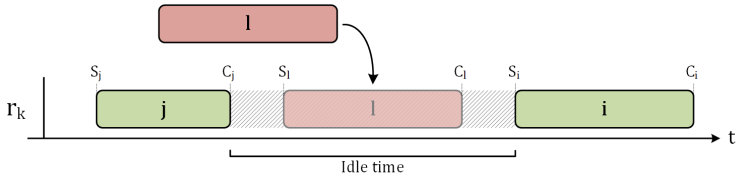


FIGURE 4.59 – Schéma de l'insertion d'une tâche dans un temps d'inactivité.

La nécessité de création de plans complexes demande la possibilité de combiner plusieurs plans élémentaires dans un ordre particulier. Une contrainte de précédence entre plans est alors demandée pour satisfaire cette possibilité.

La notion de plans de tâches implique un comportement complexe de la part de l'ordonnanceur puisque ce dernier devra être capable d'insérer chacune des tâches de chaque plan dans la liste d'évènements en vérifiant les contraintes temporelles et

de ressources. Si une tâche ne peut être placée, alors l'ensemble du plan, c'est-à-dire que toutes les tâches le constituant, ne peuvent être acceptées dans l'ordonnancement final. Le plan complet sera donc rejeté.

L'algorithme 1 présente l'heuristique de construction de l'ordonnancement. Cet algorithme est le point d'entrée de l'ordonnanceur. Il prend en entrée un ensemble de plans $\mathcal{P}$ à ordonnancer sur un ensemble de ressources $\mathcal{R}$. L'algorithme donne en sortie un ordonnancement faisable S_l qui est un vecteur de dates de démarrage des tâches $J_i^k \in \Pi_k, \Pi_k \in \mathcal{P}$ ainsi que deux ensembles de plans $\mathcal{P}_s$ et $\mathcal{P}_f$ correspondant aux plans acceptés (*scheduling success*) et refusés (*scheduling failed*) par l'ordonnanceur. Les premières lignes de l'algorithme permettent d'initialiser les structures de données utilisées tout au long de l'heuristique. Aux lignes 1 et 2 les variables $\mathcal{P}_s$ et $\mathcal{P}_f$ sont respectivement initialisées.

Ensuite aux lignes 3 et 4 les variables S_l et S_w sont initialisées et contiendront respectivement le dernier (*last*) ordonnancement réalisable et un ordonnancement temporaire de travail (*working*). La liste d'évènements $\mathcal{EL}$ est initialisée à la ligne 5.

Les contraintes du contexte opérationnel détaillées en 4.2.2.2 phase 6 §6 impliquent d'ordonnancer les plans les plus prioritaires en premier. Cette contrainte implique aussi qu'il est possible, en vue de satisfaire les contraintes présentées en section 4.1.2.2 besoin 5, d'interrompre l'ordonnancement après un temps fixé afin d'obtenir un ordonnancement complet cohérent avec la criticité du contexte, donc contenant les plans les plus prioritaires. La première phase de l'heuristique, à la ligne 6, est donc de trier les plans par ordre de priorité et précédence. Cette étape est accomplie par l'algorithme 2 avec en sortie un ensemble trié de plans non ordonnancés $\mathcal{P}_u$.

Ensuite, ligne 7, une boucle itère sur $\mathcal{P}_u$ jusqu'à ce que tous les plans aient été ordonnancés.

À la ligne 8 de l'algorithme 1 le plan le plus prioritaire est récupéré et assigné à la variable Π_k de priorité α_k. À la ligne 10 l'algorithme vérifie s'il existe plus d'un plan avec la même priorité α_k de Π_k. S'il existe un unique plan avec cette priorité, l'algorithme tente de l'ordonnancer à la ligne 11. Sinon un sous-ensemble de plan avec la même priorité $\mathcal{P}_{\alpha_k} \subset \mathcal{P}$ est calculé. Les plans $\mathcal{P}_{\alpha_k}$ sont ensuite ordonnancés par l'algorithme 5. L'algorithme 5 retourne un ensemble de plans ne pouvant être ordonnancés $\mathcal{U}$ enlevés de l'ensemble des plans de travail ligne 21. Les plans ordonnancés avec succès $\mathcal{P}_{\alpha_k} \backslash \mathcal{U}$ sont ajoutés aux plans ordonnancés $\mathcal{P}_u$ à la ligne 19. Tous les plans de $\mathcal{P}_{\alpha_k}$ sont supprimés de l'ensemble des plans non ordonnancés $\mathcal{P}_u$ à la ligne 20.

Finalement le dernier ordonnancement réalisable est mis à jour ligne 22. Ligne 23 les plans n'ayant pu être ordonnancés sont calculés $\mathcal{P}_f = \mathcal{P} \backslash \mathcal{P}_s$.

L'algorithme 1 est dans le pire des cas en $O(K^3 n^3)$, où $K = |\mathcal{P}|$ correspond au nombre de plans à ordonnancer. L'ensemble des structures de données utilisées pour stocker les plans, $\mathcal{P}$, $\mathcal{P}_u$, $\mathcal{P}_s$ et $\mathcal{P}_f$ ont été implémentées à base d'arbres de recherche binaires équilibrés afin d'assurer l'insertion, la suppression et l'accession en temps logarithmique [Cormen et al., 2009]. La liste d'évènements $\mathcal{EL}$ est elle aussi implémentée par un arbre binaire équilibré. La boucle *while* des lignes 7-23 est en $O(K^3 n^3)$ dans le cas où l'ensemble des plans sont sur la même frontière. Une frontière correspond à un sous-ensemble de plans ayant la même priorité et étant contraints par un ordre de précédence donné. Les frontières sont données par l'algorithme de tri topologique appelé après réception des plans. Ce tri, donc l'enchainement des frontières

décrivant les priorités et précédences des plans implique l'ordre dans lequel seront ordonnancés l'ensemble des plans en entrée de l'ordonnanceur.

L'opération *Get* ligne 8 est exécutée en temps logarithmique fonction du nombre de plans $\mathcal{P}_u$. Ligne 9 l'algorithme récupère le nombre de plans dans $\mathcal{P}_u$ qui ont une priorité de α_k. Une structure de données auxiliaire est utilisée et la donnée est acquise en $O(1)$.

Les opérations *Add* et *Remove* lignes 13 et 15 sont toutes les deux en $O(\log K)$. L'obtention des plans de priorité α_k ligne 17 est faite en temps constant puisque qu'une *Hashmap* est utilisée pour stocker les plans de même priorité : les clés de la *Hashmap* sont les priorités α_k et les valeurs une liste de plans ayant α_k pour priorité.

L'opération *Add* ligne 19 est en $O(K \log K)$ de même que les opérations *Remove* lignes 20 et 21.

Algorithme 1 : *BuildSchedule*, Construction de l'ordonnancement

Input : A set of plans $\mathcal{P}$
Output : A feasible schedule S_l, a set of scheduled plans $\mathcal{P}_s$, a set of unscheduled plans $\mathcal{P}_f$

```
1  P_s ← {}{Scheduled plans}
2  P_f ← {}{Discarded plans}
3  S_l ← {}{Feasible schedule}
4  S_w ← {}{Working schedule}
5  EL ← {}
6  P_u ← SortPlans(P)
7  while P_u ≠ ∅ do
8      Π_k ← Get(P_u)
9      N_αk ← number of plans in P_u with priority α_k
10     if N_αk = 1 then
11         success ← SchedulePlan(Π_k, S_w, EL)
12         if success then
13             Add(Π_k, P_s)
14         else
15             Remove(Π_k, S_w)
16     else
17         P_αk ← plans of P with priority α_k
18         U ← SchedulePlanSet(P_αk, S_w, EL)
19         Add(P_αk \ U, P_s)
20         Remove(P_αk , P_u)
21         Remove(U, S_w)
22     S_l ← S_w
23     P_f = P \ P_s
```

L'algorithme 2 (*SortPlans*) tri un set de plans $\mathcal{P}$ dans l'ordre décroissant des priorités tel que les contraintes de précédences entre les plans soient respectées. Ligne 1, un algorithme de tri topologique est exécuté pour l'ensemble de plans [Kahn, 1962,Cormen et al., 2009]. Un ensemble de couples (Π_i,F_i) est retourné, où Π_i est le *i*-ème plan du tri topologique et F_i est un sous-ensemble de la frontière f_i à laquelle le plan Π_i appartient. Ensuite, considérons F l'ensemble des sous-ensembles F_i des plans de chaque frontière f_i (ligne 2), une boucle sur chaque set F_i est exécutée (ligne 4 à 6) où chaque F_i est trié en fonction de la priorité des plans de la frontière f_i (ligne 5) et ensuite ajouté à l'ensemble des plans triés $\mathcal{P}_u$ ligne 6.

L'algorithme 2 est en $O(K\log K+n_k)$, où $K = |\mathcal{P}|$ est le nombre de plans à ordonnancer. Le tri topologique est réalisé en temps $O(K + n_k)$ où n_k est le nombre de prédécesseurs au plan Π_k. Ligne 2, l'ensemble des frontières découvertes par l'algorithme de tri topologique est assemblé : cet ensemble est implémenté par une *Hashtable* où les clés de la *Hashtable* sont les priorités α_k et les valeurs une liste de plans ayant α_k pour priorité.

La boucle lignes 4-6 itère sur chaque frontière et tri les plans appartenant à cette frontière en fonction de leur priorité. Cette boucle *for* est en $O(K\log K)$.

Algorithme 2 : *SortPlans*, tri des plans

Input : A set of plans $\mathcal{P}$
Output : A sorted set of plans $\mathcal{P}_u$

```
1 P_t ← TopologicalSort(P)
2 compute F_i for i = 1, ..., f
3 P_u = ∅
4 for i = 1, ..., f do
5 |   sort F_i according to α
6 |   Add(F_i, P_u)
7 return P_u
```

L'algorithme 3 ordonnance un plan unique. Il prend en entrée un plan Π_k, l'ordonnancement de travail S_w et une liste d'évènements $\mathcal{EL}$ et donne en sortie une valeur booléenne $vraie$ si le plan Π_k a été correctement ordonnancé, $faux$ sinon. L'algorithme tente à la ligne 2 d'ordonnancer chacune des tâches J_i^k de Π_k. Si une tâche ne peut être ordonnancée alors Π_k est marqué comme non-ordonnançable et l'algorithme retourne la valeur $false$ indiquant que le plan n'a pu être ordonnancé en satisfaisant les contraintes.

L'algorithme 3 est en $O(n^3)$, où $n = \sum_k n_k$, $\forall \Pi_k \in \mathcal{P}$ correspond au nombre total de tâches déjà ordonnancées de S_w. Dans la boucle 1-5, l'algorithme 4, ligne 2, est en $O(n_k)$.

Algorithme 3 : *SchedulePlan*, ordonnancement d'un plan

Input : A plan Π_k, a schedule S_w, an event list $\mathcal{EL}$
Output : $true$ if Π_k has been scheduled in S_w, $false$ otherwise.

```
1 foreach task J_i^k ∈ Π_k do
2 |   success ← ScheduleTask(J_i^k, S_w, EL)
3 |   if !success then
4 |   |   mark Π_k as unschedulable
5 |   |   return false
6 return true
```

L'algorithme 4 réalise le travail d'insertion d'une tâche dans l'ordonnancement S_w. Il prend en entrée une tâche J_i^k, l'ordonnancement de travail S_w, et la liste d'évènements $\mathcal{EL}$. Il donne en sortie une valeur booléenne $vraie$ si la tâche J_i^k a été correctement ordonnancée, $faux$ sinon. L'algorithme commence par calculer le temps de démarrage au plus tôt donné par l'équation 4.10 sans prendre en considération la disponibilité des ressources. À la ligne 2 l'algorithme cherche un évènement e dans la $\mathcal{EL}$ tel que $t(e) = s_i^k$. Si $\mathcal{EL}$ ne contient pas un tel évènement alors un nouvel

évènement e est créé à $t(e) = s_i^k$. À la ligne 4 l'algorithme lance une boucle afin de rechercher la date de début (e) et de fin (f) au plus tôt auxquelles la tâche J_i^k est faisable. La boucle itère tant que f n'est pas le dernier évènement de $\mathcal{EL}$ ou que la durée restante μ, fixée à p_i^k à l'origine, ne soit nulle. Si la condition d'exécution de la boucle est satisfaite en ligne 6 alors l'évènement suivant f est assigné à l'évènement g et les contraintes de J_i^k dans $t(e)$ sont vérifiées (ligne 6). Si les deux tests sont passés alors μ est réduit de $t(g) - t(f)$ et g est assigné à f. Si les tests échouent alors e n'est pas une position d'insertion valide, e est fixé à g et μ est réinitialisé à p_i^k. Après cette boucle une vérification finale des contraintes est faite (ligne 14). Si le test final échoue alors toutes les tâches de Π_k sont supprimées, tous les évènements de $\mathcal{EL}$ sont supprimés (ligne 15), Π_k est marqué non ordonnançable (*unschedulable*, ligne 15) et $false$ est retourné. Une fois les contraintes vérifiées pour e ligne 14 l'algorithme continue lignes 18 et 19 avec l'insertion de la tâche J_i^k dans l'ordonnancement de travail S_w et dans la liste des tâches démarrant à e, $\mathcal{S}(e)$. Ensuite ligne 21 l'évènement e est ajouté à la liste d'évènement $\mathcal{EL}$ si non présent. Les lignes 23 à 31 permettent de mettre à jour la liste d'évènements $\mathcal{EL}$. La dernière étape, ligne 32, permet de mettre à jour l'utilisation de la ressource par la tâche ordonnancée.

L'algorithme 4 a une complexité temporelle en $O(n^2)$, où $n = \sum_k n_k$. La recherche d'un temps de démarrage valide s_i^k, ligne 1, est au plus en $O(n-1)$ si la tâche J_i^k a $n-1$ prédécesseurs. La recherche de évènement e tel que $t(e) = s_i^k$ (ligne 2) est au plus $O(\log|\mathcal{EL}|)$ grâce à l'utilisation d'un arbre de recherche binaire équilibré. Dans le pire cas, pour un ensemble $\mathcal{P}$ de plans de cardinalité $|\mathcal{P}| = K$ le nombre maximal d'évènements dans la liste $\mathcal{EL}$ est de $2n$ si on suppose que chaque tâche est ordonnancée consécutivement avec un temps de retard entre chaque tâche.

La boucle 5-13 itère sur l'ensemble des évènements de $\mathcal{EL}$. Puisque $\mathcal{EL}$ contient au plus $2n$ évènements la boucle *while* est en $O(n)$. La recherche suivant f dans $\mathcal{EL}$ est au plus $O(log|\mathcal{EL}|)$. La vérification des contraintes est en temps constant. Les opérations restantes dans la boucle sont aussi en temps constant.

La suppression des tâches $J_i^k \in \Pi_k$ de chaque $e \in \mathcal{EL}$ (ligne 15) est en $O(\log|\mathcal{EL}|)$.

L'ajout d'une tâche J_i^k à l'ordonnancement S_w ligne 18 demande au plus un temps de $O(\log n)$. L'ajout de la tâche J_i^k à l'ensemble $\mathcal{S}(e)$ est réalisé en temps constant.

Ajouter l'évènement e à $\mathcal{EL}$ demande un temps en $O(\log|\mathcal{EL}|)$. La mise à jour de la liste d'évènements est de même en $O(\log|\mathcal{EL}|)$ si l'évènement g doit être inséré dans $\mathcal{EL}$ (lignes 25 et 28) sinon l'insertion de J_i^k dans $\mathcal{C}(f)$ est fait en temps constant.

Algorithme 4 : *ScheduleTask*, ordonnancement d'une tâche

Input : A task J_i^k, a schedule S_w, an event list $\mathcal{EL}$
Output : $true$ if J_i^k has been scheduled in S_w, $false$ otherwise.

1 $s_i^k = max(W_s, r_i^k, max_{j \in pred_i^k}(C_j^k + \delta_{ij}^k))$
2 $e \leftarrow$ GetEvent(s_i^k,$\mathcal{EL}$)
3 $\mu \leftarrow p_i^k$
4 $f \leftarrow e$
5 **while** (f *is not the last event of* $\mathcal{EL}$) $\wedge$ ($\mu > 0$) **do**
6 $\quad g \leftarrow next_f$
7 $\quad$ **if** ($q_\rho(f) = 0, \forall \rho \in \mathcal{R}_i^k$) $\wedge$ (*CheckConstraints*($t(e)$, J_i^k)) **then**
8 $\quad\quad \mu \leftarrow max(0, \mu - t(g) + t(f))$
9 $\quad\quad f \leftarrow g$
10 $\quad$ **else**
11 $\quad\quad \mu \leftarrow p_i^k$
12 $\quad\quad e \leftarrow g$
13 $\quad\quad f \leftarrow g$
14 **if** !*CheckConstraints*($t(e), J_i^k$) **then**
15 $\quad$ remove all tasks $J_i^k \in \Pi_k$ from $e, \forall e \in \mathcal{EL}$
16 $\quad$ mark Π_k as unschedulable
17 $\quad$ **return** $false$
18 Add(J_i^k, S_w)
19 Add($J_i^k, \mathcal{S}(e)$)
20 **if** $e \notin \mathcal{EL}$ **then**
21 $\quad \mathcal{EL} \leftarrow \mathcal{EL} \cup e$
22 **if** $t(e) + p_i^k = t(f)$ **then**
23 $\quad$ Add($J_i^k, \mathcal{C}(f)$)
24 **else if** $t(e) + p_i^k > t(f)$ **then**
25 $\quad$ insert $g = (t(e) + p_i^k, \emptyset, \{t\}, (q_\rho(f))_{\forall \rho \in \mathcal{R}_i^k})$ in $\mathcal{EL}$
26 $\quad f \leftarrow g$
27 **else**
28 $\quad$ insert $g = (t(e) + p_i^k, \emptyset, \{t\}, (q_\rho(pred_f))_{\forall \rho \in \mathcal{R}_i^k})$ in $\mathcal{EL}$
29 $\quad f \leftarrow g$
30 $q_\rho(g) = 1 \forall \rho \in \mathcal{R}_i^k, \forall g$ between e and $pred_f$ in $\mathcal{EL}$
31 **return** $true$

L'algorithme 5 permet d'ordonnancer un ensemble de plans P_{α_k} où chaque plan $\Pi_k \in P_{\alpha_k}$ a une priorité de α_k. Il prend en entrée $P_{\alpha_k} \subset \mathcal{P}$ un ordonnancement de travail S_w et la liste d'évènements $\mathcal{EL}$. Il donne en sortie un set de plans non ordonnancés $\mathcal{U}$. L'algorithme tente d'ordonnancer les plans en minimisant les temps d'inactivités entre les tâches des plans. L'algorithme commence par copier l'ensemble des plans V à la ligne 1 et initialiser la liste des plans non ordonnancés $\mathcal{U}$. La boucle aux lignes 3-25 est exécutée tant que V n'est pas vide. À chaque itération deux variables *minIdleTime* (le temps mini d'inactivité) et *bestPlan* sont utilisées pour conserver le meilleur plan candidat à ordonnancer. La boucle lignes 6-19 itère sur V afin de rechercher le plan candidat. Chaque plan Π_k est ordonnancé dans un ordonnancement temporaire S_{temp}. À chaque itération, ligne 11, l'algorithme choisi un plan $\Pi_k \in \mathcal{P}_{\alpha_k}$ qui minimise la somme des différences de temps entre $t(e)$ et $t(pred_e)$ pour chaque évènement e tel que $J_i^k \in \mathcal{S}(e), \forall J_i^k \in \Pi_k$.

L'algorithme 5 prend au plus un temps en $O(K^2 n^3)$. La boucle lignes 3-20 est répétée

tant qu'il reste des plans à ordonnancer au plus en temps $O(K)$. La boucle interne (lignes 6-16) est en $O(Kn^3)$

Algorithme 5 : *SchedulePlanSet*, ordonnancement d'un set de plans

Input : A set $P_{\alpha_k} \subset \mathcal{P}$ of plans, a schedule S_w, an event list $\mathcal{EL}$
Output : A set $\mathcal{U}$ of unscheduled plans

```
1  V ← P_{α_k}
2  U ← ∅
3  while V ≠ ∅ do
4      minIdleTime ← +∞
5      bestPlan ← ∅
6      foreach Π_k ∈ P_{α_k} do
7          S_temp ← S_w
8          EL_temp ← EL
9          success ← schedulePlan(Π_k, S_temp, EL_temp)
10         if success then
11             it = Σ_i(t(e) − t(pred_e)), ∀J_i^k ∈ S(e)
12             if it ≤ minIdleTime then
13                 minIdleTime ← it
14                 bestPlan ← Π_k
15         else
16             Remove(Π_k, V)
17     success ← schedulePlan(bestPlan, S_w, EL)
18     if !success then
19         Add(bestPlan, U)
20     Remove(bestPlan, V)
21 return U
```

L'algorithme 6 vérifie si un temps de début d'une tâche $t(e)$ satisfait les contraintes temporelles présentées en amont pour une tâche J_i^k. Si les contraintes temporelles sont satisfaites alors l'algorithme retourne $true$, sinon $faux$. Le tableau 4.6 synthétise les 6 algorithmes. L'algorithme 6 est exécuté en temps constant.

Algorithme 6 : *CheckConstraints*, vérification les contraintes

Input : A task J_i^k, a starting time $t(e)$ for J_i^k
Output : $true$ if J_i^k satisfies the temporal constraints, $false$ otherwise.

```
1  if (t(e) ∈ [r_i^k, d_i^k]) ∧ (t(e) ∈ [W_s, W_e]) ∧ (t(e) + p_i^k ∈ [W_s, W_e]) then
2      return true
3  return false
```

Algorithme	Entrées	Sorties	Fonctions	Complexité
BuildSchedule	Plans $\mathcal{P}$	Ordonnancement réalisable S_l Plans ordonnancés $\mathcal{P}_s$ Plans non ordonnancés $\mathcal{P}_f$	Point d'entrée de l'ordonnanceur. Construit l'ordonnancement final réalisable S_l	$O(K^3n^3)$
SchedulePlan	Plan Π_k Ordonnancement de travail S_w Liste d'évènement $\mathcal{EL}$	$true$ si ordonnancé, $false$ sinon	Ordonnance le plan Π_k dans S_w.	$O(n^3)$
SchedulePlanSet	Plans $\mathcal{P}_{\alpha_k}$ Ordonnancement de travail S_w liste d'évènement $\mathcal{EL}$	Plans non-ordonnancés $\mathcal{U}$	Ordonnance les plans de même priorité $\mathcal{P}_{\alpha_k}$ dans S_w .	$O(K^2n^3)$
ScheduleTask	Tâche J_i^k Ordonnancement de travail S_w Liste d'évènement $\mathcal{EL}$	$true$ si ordonnancée, $false$ sinon	Ordonnance J_i^k dans S_w.	$O(n^2)$
CheckConstraints	Tâche J_i^k Date de début $t(e)$ de J_i^k	$true$ si J_i^k satisfait les contraintes, $false$ sinon	Vérifie les contraintes temporelles d'une tâche avec pour date de début $t(e)$.	$O(1)$
SortPlans	Plans $\mathcal{P}$	Copie $\mathcal{P}_u$ de $\mathcal{P}$ triée	Tri un ensemble de plans $\mathcal{P}$ en utilisant l'algorithme *TopologicalSort* puis les tri par ordre des priorités.	$O(K\log K + n_k)$
TopologicalSort	Plans $\mathcal{P}$	Copie de $\mathcal{P}$ triée	Tri un ensemble de plans en fonction de leurs relations de précédences en utilisant un algorithme de tri topologique.	$O(K + n_k)$
GetEvent	Temps de début d'une tâche s_i^k Liste d'évènement $\mathcal{EL}$	Un évènement $e \in \mathcal{EL}$	Récupère l'évènement $e \in \mathcal{EL}$ tel que $t(e) = s_i^k$. S'il n'existe pas, un nouvel évènement sera créé à $t(e)$.	$O(\log\|\mathcal{EL}\|)$
Get($\mathcal{P}$)	Plans $\mathcal{P}$	Plan prioritaire Π_k	Trouve le plan le plus prioritaire de $\mathcal{P}$	$O(\log K)$

TABLE 4.6 – Liste des algorithmes et de leur fonction dans l'ordonnancement.

Conclusion

Nous avons présenté dans ce troisième chapitre l'ensemble des travaux réalisés. Dans une première section, nous avons exposé les méthodes utilisées pour l'étude des architectures, des contraintes opérationnelles et industrielles, notamment par des méthodes MBSE [6] et l'utilisation de métaphores du système étudié. Nous avons aussi montré des exemples de scénarios mettant en œuvre des systèmes de senseurs à titre d'expérimentation et de mise en situation.

Nous avons ensuite détaillé les différentes étapes de conception de l'architecture senseurs. Nous avons commencé par une présentation du paradigme multi-agent et de la notion de niveau de contrôle des systèmes. Nous avons ensuite détaillé la phase d'agentification du système multi-senseur. Nous avons vu lors de cette phase que la première approche d'agentifier les senseurs, c'est-à-dire de leur attribuer l'ensemble des capacités cognitives et de planification, ne permet pas de satisfaire les contraintes du contexte, mettant en place une forte complexité du point de vue de la synchronisation et représentation de l'état de l'environnement de la plateforme. Les senseurs ont alors été mis au rang de ressources, les agents correspondant alors à une virtualisation des objets observés par les senseurs. Les agents ont été définis en tant qu'agents tactiques, responsables de la planification des actions senseurs. Les structures de l'architecture et des agents ont ensuite été explicitées en détail. La boucle d'asservissement des senseurs et la dynamique complète de l'architecture ont été présentées.

Dans une troisième section nous avons vu que la problématique d'optimisation multi-senseur passe par l'ordonnancement de plans créés par les agents, pondérés d'une priorité fixe, calculée selon des règles opérationnelles. Nous avons explicité le problème d'ordonnancement comme étant un RCPSP du type optimisation d'un job-shop à M machines (M étant le nombre de senseurs) avec des contraintes de tâches du type délais inter-tâches début-début/fin-début exacts, possibilité d'exécutions synchrones de tâches sur plusieurs machines, sans retards possibles et des durées d'exécution variables en fonction du temps de début de la tâche. Nous avons ensuite montré que les critères à optimiser pour la construction de l'ordonnancement des senseurs sont la minimisation du makespan ainsi que la maximisation de la priorité relative des tâches ordonnancées. Les plans senseurs, issus de la phase de planification réalisée par les agents, ont été présentés et détaillés. Les plans senseurs, ses tâches et ses contraintes permettent ainsi la définition exacte de l'utilisation des senseurs. Les algorithmes des heuristiques d'ordonnancement prenant en entrée les plans senseurs ont été détaillés et expliqués. Nous avons montré que ces algorithmes, basés sur des évènements, permettent des temps d'exécution convenables de l'ensemble de l'architecture au regard du besoin.

6. MBSE

Chapitre 5

Expérimentation

5.1 Protocole expérimental

De la même façon que lors de la création d'une nouvelle architecture pour un bâtiment, le développement d'une nouvelle architecture système ne peut être fait au fur et à mesure de la construction du système réel. Il est possible de schématiser et détailler au maximum l'ensemble des interfaces entre chacun des composants et spécifier l'ensemble des exigences nécessaires à la satisfaction du cahier des charges

La construction d'une architecture système repose sur un certain nombre d'hypothèses faites à plusieurs niveaux. Dans le cas d'une architecture pour systèmes multi-senseur ces hypothèses portent sur les éléments suivants :

- Les senseurs, dont leurs interfaces, leurs capacités définitives et leur future autonomie ;
- Les puissances de calculs, de communications et la distribution matérielle des unités systèmes embarquées ;
- Le système de contrôle du SMS interne (Gestionnaire de mission embarqué) et externe (Gestionnaire de Mission du C2) dont ses capacités (éviter les redondances) et ses interfaces.

Alors qu'il est possible d'abstraire un certain nombre de paramètres lors de la simulation il est nécessaire dans d'autres cas d'établir des hypothèses sur le système final. Pour chacun des composants système intervenant au sein de la future architecture trois possibilités se présentent :

1. Les systèmes à y intégrer sont préexistants et les interfaces sont disponibles ;
2. Les systèmes sont en cours de développement ;
3. Les systèmes seront en développement à la suite de la création de l'architecture et pourront éventuellement correspondre à une évolution des systèmes existants.

Dans le premier cas, le travail d'architecture consiste à réunir l'ensemble des informations disponibles sur les composants afin de les assembler, créer des interfaces et décider des moyens de communication.

Dans le second cas, il est nécessaire de collaborer avec les équipes responsables de chaque système afin de comprendre quelles sont les exigences d'entrées, sorties et de fonctionnement attendu pour leurs systèmes.

Dans le dernier cas, soit les équipes responsables des futurs systèmes n'existent pas encore, soit elles sont à un stade très amont de l'étude.

L'architecture présentée dans ce manuscrit rentre dans la troisième catégorie : certains systèmes sont en cours de développements préliminaires et la plus grande partie des systèmes ne sont pas en développement. La technologie a cependant été choisie et se base sur les systèmes existants augmentés des technologies matérielles futures.

L'expérimentation d'une nouvelle architecture malgré les incertitudes citées ci-dessus est rendue possible par l'utilisation de la simulation. La simulation est aujourd'hui utilisée pour la conception et la fiabilisation de chacun des systèmes d'un aéronef dans son ensemble. De la simulation des impulsions électromagnétiques pour la conception des antennes radars, jusqu'à la simulation opérationnelle multi-acteur pour la coordination des tâches sur un théâtre opérationnel, chaque système ou système de systèmes peut être simulé.

Les simulations dites *HardWare In the Loop*, matériel dans la boucle (HWIL), *SoftWare In the Loop*, logiciel dans la boucle (SWIL) et *Man In The Loop*, homme dans la boucle (MITL) permettent de tester des composants réels du futur système dans des conditions représentatives. Par exemple, comme il est compliqué de tester une carte électronique faisant partie d'un ensemble de cartes encore non développées il est possible de réaliser une simulation logicielle complète du système final et de remplacer le code simulant la carte par la carte réelle. Les tests mis en place permettent de tester des aspects ciblés du système. Il est donc nécessaire de définir les aspects à tester préalablement à la construction d'une simulation et des tests. La simulation permet d'expérimenter et évaluer des systèmes sans nécessiter de les développer dans leur globalité pour ainsi réduire les coûts de conception.

Les expérimentations présentées dans ce chapitre permettent de démontrer que les besoins présentés en 4.1.2.2 et 4.1.2.3 peuvent être satisfaits par la nouvelle architecture. L'ensemble de la simulation a donc été adaptée à ces besoins. Chacun des aspects du système de senseurs a alors été étudié de façon à identifier les processus essentiels pour la simulation d'un système de senseurs en fonctionnement. Par exemple, de la création d'un ordre de pointage radar sur un objet jusqu'à la construction des données concernant cet objet plusieurs processus logiciels, matériels et physiques. La figure 5.1 montre une version simplifiée de la chaine d'acquisition d'une information par le radar.

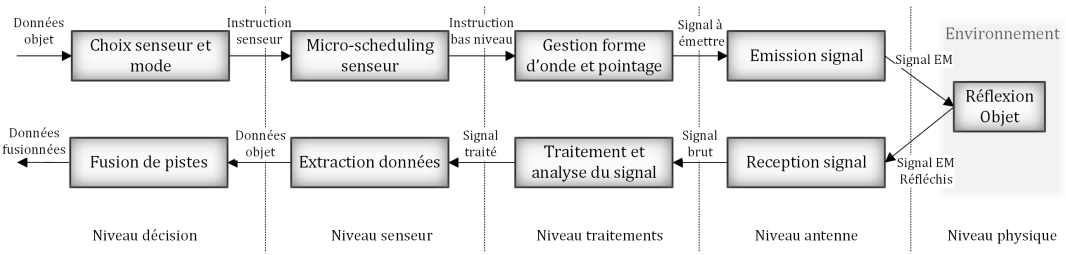


FIGURE 5.1 – Schéma simplifié d'une mesure radar.

Lors d'une simulation d'une architecture système mettant en œuvre des systèmes radars il n'y a pas besoin de simuler l'ensemble de la chaine d'acquisition, mais seulement son principe, en conservant les entrées et sorties des niveaux abstraits. Cependant, si nous avions besoin d'évaluer les interférences électromagnétiques entre

plusieurs radars nous aurions conservé les niveaux traitements, antenne et physique plutôt que les niveaux décision et senseur.

L'architecture multi-senseur multi-agent présentée dans ces travaux permet, entre autres, une corrélation des informations du théâtre, une autonomie décisionnelle décentralisée, une modularité accrue du fonctionnement du SMS et une décomposition des problématiques de planification et d'ordonnancement en problèmes simples. Ces aspects sont donc les principaux aspects à tester par la simulation.

Deux simulations distinctes ont été implémentées. Les deux simulations portent le nom de RAMSES I et RAMSES II. Ces deux simulations font office d'outil d'évaluation, de maturation et de démonstrateur.

5.2 Simulation

5.2.1 RAMSES I

5.2.1.1 Contexte

RAMSES I est le premier démonstrateur mettant en scène une implémentation d'agents tactiques pour le contrôle d'un système multi-senseur. L'objectif de cette simulation était donc de prouver le concept de la prise de décision autonome d'actions senseurs par des agents disposant des données représentatives d'un unique objet du théâtre. Afin de se concentrer sur l'aspect décision les senseurs implémentés dans RAMSES I étaient simplifiés, ne prenant pas en compte des échecs de tâches, des imprévus et les données à la disposition des agents étaient considérés parfaites de façon à faciliter les processus de décisions.

Cette simulation a été implémentée en Java 1.6 avec le framework multi-agent JACK© *: Intelligent Agent* développé par AOS Group. Le moteur multi-agent de JACK permet d'implémenter des agents BDI.

FIGURE 5.2 – Logo du framework JACK.

5.2.1.2 Fonctionnement du SMA

L'environnement de cette simulation met en situation différents types d'objets :

Le terrain est composé d'un espace 2D sur lequel se trouvent :

- La plateforme ;
- Des bâtiments civils simples ;
- Des bâtiments civils qui émettent des ondes RF ;
- Des véhicules transportant des missiles ;
- Des radars de conduite de tir ;
- Des radars de veille ;

- Des nuages.

Certains de ces objets sont mobiles (Plateforme, véhicules armés, etc.). Dans ce cas, ces objets ont un pilote qui possède une séquence de points par lesquels l'objet va faire son possible pour passer. Par défaut, le pilote garde une destination constante. Dans le cas de la plateforme, l'opérateur peut changer de destination quand il reçoit un ordre du GM ou si le pilote souhaite prendre la main sur le déplacement du RPAS. Les objets mobiles disposent aussi d'une vitesse. Par défaut la vitesse des objets est constante la plateforme se déplace à 200m/s (720 km/h), les camions se déplacent à 20m/s (72 km/h). Certains objets émettent des ondes radio (Radar de veille, bâtiment civil radar, etc.). Ces ondes RF peuvent être perçues par la fonction veille GE du SMS à une distance donnée. Cette distance est par défaut de 20km pour un bâtiment civil émettant des ondes RF et pour une arme équipée d'un radar et 53km pour un radar de veille.

L'environnement caractérise le « monde » autour du SMA. Il peut être aussi bien réel (ex. : une plateforme en vol) que virtuel (ex. : une simulation). Un environnement est dit être dans un certain état. L'environnement possède une dynamique, il évolue de lui-même (ex. : la position de la plateforme en vol change avec le temps) et suite aux actions effectuées par les agents (ex. : un agent active le brouillage). Les agents reçoivent des informations de leur environnement à travers leurs perceptions (ex. : flux vidéo). L'environnement est composé :

- D'un ensemble de fonctions qui peuvent être activées. Les fonctions peuvent émettre des données (ex. : la fonction prise d'image SAR indique l'image qu'elle a capturée) que les agents peuvent percevoir ;
- D'un Artefact d'Application des Propositions (AAP). Les agents peuvent agir sur l'AAP en lui indiquant une proposition d'activation. Dans ce cas, l'AAP active et désactive les fonctions adéquates. La dernière proposition qu'a reçue l'AAP est appelée proposition courante ;
- D'un environnement physique (ex. : *Unmanned Air Vehicle*, véhiculer aérien contrôlé à distance (UAV), objets au sol) ;
- D'un environnement organisationnel (ex. : gestionnaire de mission, opérateur). Cet environnement peut communiquer avec le SMA et le SMA peut communiquer avec cet environnement.

À noter qu'il est possible de connecter un SMA avec n'importe quel environnement compatible. Bien que pour des raisons évidentes de praticité nous avons choisi un environnement virtuel, l'environnement réel d'un SMS est compatible avec cette définition, il est tout à fait possible de connecter un SMA à un environnement réel.

Certains objets émettent des interférences (ex. : nuages). Les objets qui sont sous les nuages ne sont pas vus par une prise d'image ou la veille optronique. Le terrain possède la dynamique suivante. La dynamique est basée sur une boucle de simulation : chaque itération de la boucle fait « avancer le terrain dans le temps » de 0.1s. Pendant cette boucle, les objets mobiles sont mis à jour. Ils avancent à leur vitesse. S'ils ont une direction prédéfinie, ils tournent dans cette direction (max 3.6° par 0.1 seconde).

Le SMS possède la dynamique suivante :

Le SMS est mis à jour à l'aide de la même boucle itérative que pour le terrain (incrément de 0.1s de temps simulé à chaque tour de boucle). Lors de cette boucle, on

va interrompre les fonctions qui ont violé leurs contraintes, remettre à zéro la durée des fonctions interrompues, incrémenter la durée des fonctions activées et non interrompues de 0.1s. Puis, on récupère la donnée DRIL des fonctions actives et non interrompues, selon ses caractéristiques (ex. : si la fonction est restée active depuis assez longtemps, si les paramètres sont assez proches). Ces données sont envoyées au SDD. Le SMS est mis à jour quand l'agent décisionnaire envoie une nouvelle proposition d'activation à l'Artefact d'Application des Propositions (AAP). Dans ce cas, la proposition courante est mise à jour avec la nouvelle proposition d'activation. Les fonctions qui étaient actives et qui ne le sont plus sont désactivées. Celles qui ont reçu de nouveaux paramètres sont remises à zéro. Celles qui n'étaient pas actives et qui sont sur la nouvelle proposition sont maintenant activées, à condition que les ressources dont elle dépend soient utilisables. Le SMS est mis à jour quand nous voulons simuler une avarie et que l'on rend une ressource inutilisable manuellement. Si une fonction qui utilise cette ressource est activée, cette fonction est interrompue.

Le SMS est composé de l'AAP qui contient la proposition courante ainsi que l'état de chacune des fonctions et l'état de chacune des ressources. L'état de la fonction indique :

- Si cette fonction est activée (en cours d'utilisation) ou désactivée. Cette information dépend directement de la proposition courante ;
- Si cette fonction est interrompue (une fonction est interrompue si les contraintes sont violées, généralement parce que cette fonction a une portée limitée et que le RPAS en est sorti, ou qu'une de ces ressources est inutilisable) ;
- La durée de cette activation (depuis combien de temps elle est active sans interruption, utilisée par les fonctions basées sur une durée).

L'état de la ressource indique :

- Si cette ressource est utilisable
- Si cette ressource est en cours d'utilisation par une fonction

5.2.1.3 Mécanisme de coordination

L'architecture multi-agent réalisée diverge de l'architecture présentée dans le chapitre 4.2.2.2 et 4.2.3.1 puisque RAMSES I a été implémenté avant la finalisation de l'architecture. La principale différence entre RAMSES I et RAMSES II, outre un modèle des senseurs plus simple pour la première version, est la gestion temporelle des actions senseurs en fonction du contexte. L'architecture de RAMSES I planifie les actions les plus adéquates avec la situation à chaque instant tandis que RAMSES II dispose de l'ordonnanceur présenté en 4.3.3.

RAMSES I ne dispose pas d'un ordonnanceur mais d'un mécanisme d'enchères entre agents permettant de décider des actions senseurs à chaque nouvelle donnée dont les agents disposent.

Les mécanismes de coordination définissent une manière de faire travailler les agents ensemble afin qu'ils atteignent les objectifs du système. Notre architecture comporte deux mécanismes de coordination : une organisation et un protocole de décision.

Organisation :

Les organisations coordonnent les agents en leur donnant des rôles (ou responsabilités). Ces rôles influencent ce que doivent faire les agents (ex. : gérer les données

entrantes) et permettent de créer des attentes vis-à-vis des autres agents (ex. : un agent peut demander au gestionnaire des entrées de l'informer au sujet de certaines informations).

Ce mécanisme de coordination vise à simplifier la conception du SMA à l'aide de l'approche de conception classique « diviser pour régner ». Le problème se découpe naturellement en trois parties relativement indépendantes : récupération des données senseurs/organisationnelles, raisonnement (agents tactiques) et décision (agent décisionnaire). En allouant chaque partie à un rôle spécifique, on traite des sous-problèmes plus simples et par effet de non-linéarité de la complexité, on réduit la difficulté et le coût pour concevoir le système. De plus, cette organisation segmente le raisonnement en points d'intérêts tactiques (ex. : but, objet dangereux) : on associe un agent à chaque point d'intérêt tactique. Nous avons pu procéder à ce découpage, car les différents points d'intérêt tactique peuvent être considérés relativement indépendamment les uns des autres. Bien sûr, il reste possible de créer des liens entre ces différents points d'intérêt tactiques, ce qui peut être utile dans certains cas (ex. : croiser les informations «camion transportant des missiles » avec «radar désarmé» pour découvrir si un radar peut devenir dangereux). De plus, des liens supplémentaires sont requis afin de prendre une décision commune sur les fonctions à activer (voir section suivante).

Notre organisation est composée de quatre rôles principaux :

- Un gestionnaire des entrées : ce rôle demande de traiter les entrées du système (données reçues par les fonctions) et de fournir des informations aux agents qui en ont besoin ;
- Un gestionnaire des objectifs : ce rôle demande de traiter les ordres fournis par le GM ;
- Des agents tactiques : ce rôle abstrait demande de s'occuper d'un point d'intérêt tactique. Plusieurs rôles implémentent concrètement ce rôle ;
- Un agent décisionnaire : ce rôle demande de choisir quelle proposition d'activation fournir à l'AAP.

Le rôle abstrait de l'agent tactique est implémenté concrètement par les rôles suivants :

- Un agent opérateur : ce rôle demande de traiter les demandes d'utilisation de fonctions faites par l'opérateur et de les faire appliquer ;
- Agents menace (un par objet qui peut mener à une menace) : ce rôle demande de surveiller les objets du terrain et d'informer les autres agents menace si leurs objets peuvent devenir dangereux s'ils sont réunis (camion armé proche d'un radar) ;
- Agents zone (un par ordre de type «prise d'image » du GM) : ce rôle demande de traiter des objectifs envoyés par le GM jusqu'à leur accomplissement ;
- Un agent veille : ce rôle demande de veiller à la sécurité du RPAS ainsi qu'à la SiTac ;
- Agents interférence (un par nuage détecté) : ce rôle demande de surveiller les objets qui peuvent créer des interférences.

Protocole de décision :

Un protocole consiste en un ensemble de règles données aux agents dans le but d'accomplir une certaine interaction. Dans notre cas, l'interaction consiste à proposer et à décider d'une proposition d'activation à appliquer à l'AAP.

Ce protocole est défini ainsi :

- Chaque agent tactique peut, à partir de la proposition d'activation courante, générer un ensemble de propositions d'activation qu'il juge utiles vis-à-vis de son point d'intérêt tactique (ex. : un agent virtualisant un objet dangereux peut demander une prise d'image sur cet objet) ;
- Afin de mieux déterminer si cette proposition est utile, l'agent tactique peut demander l'avis d'autres agents tactiques en leur demandant d'étiqueter la proposition d'évaluation. Les étiquettes sont des informations objectives qui permettent de déterminer si une proposition est utile ou non (ex. : « cette proposition permet d'atteindre l'objectif O », « cette proposition implique une utilisation inappropriée d'une fonction ») ;
- Une fois qu'une proposition est étiquetée, il l'envoie avec ses étiquettes à l'agent décisionnaire ;
- L'agent décisionnaire doit utiliser les étiquettes pour évaluer la proposition reçue et la comparer avec la proposition courante ;
- Si la nouvelle proposition est meilleure, elle est soumise à l'AAP et l'agent décisionnaire avertit les agents tactiques que la proposition courante a changé.

Étiquetage :

Les agents tactiques peuvent allouer des étiquettes aux propositions qui leur sont faites. Ces étiquettes sont :

- « Cette proposition satisfait l'ordre O » ;
- « Cette proposition n'est plus utile pour atteindre l'ordre O » ;
- « Cette proposition permet de surveiller une cible dangereuse » ;
- « Cette proposition permet de surveiller une cible potentiellement dangereuse » ;
- « Cette proposition surveille un objet détruit » ;
- « Cette proposition respecte la demande de l'opérateur » ;
- « Cette proposition ne respecte plus la demande de l'opérateur » ;
- « Cette allocation brouille un radar ennemi » ;
- « Cette allocation est gênée par une interférence » ;
- « Cette proposition ne respecte pas les contraintes des fonctions » ;
- « Cette proposition offre une SiTac bonne/moyenne/faible ».

La décision d'affecter ou non une étiquette dépend de l'agent et est définie en Section 3.4. Notez que certaines étiquettes ne sont utilisées que pour « annuler » d'autres quand la situation change.

Ce protocole de décision est représenté par un diagramme de séquence en figure 5.3.

Fonction d'évaluation :

L'utilité est définie par une fonction qui permet de comparer deux propositions. Cette fonction dépend de la politique.

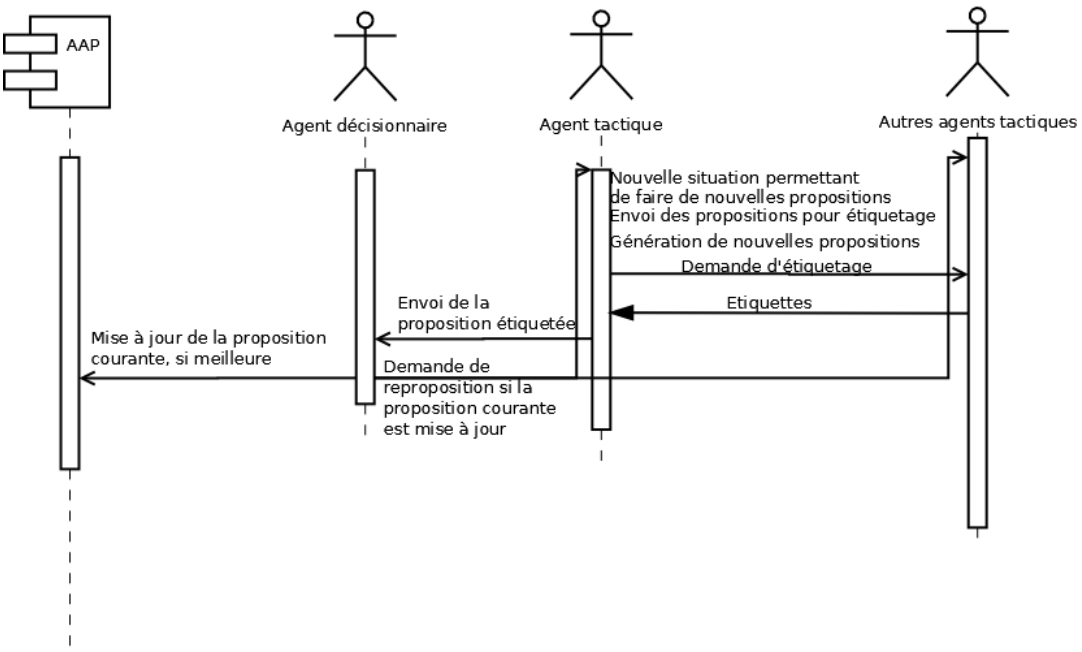


FIGURE 5.3 – Diagramme de séquence du protocole de décision.

Dans tous les cas, le paramètre le plus important est l'applicabilité : une proposition qui a l'étiquette : « ne respecte pas les contraintes des fonctions » sera toujours laissée de côté par rapport à une proposition qui n'a pas ce type d'étiquettes.

Le second paramètre le plus important est la demande de l'opérateur : les propositions qui ont les étiquettes « cette proposition respecte la demande de l'opérateur » seront toujours privilégiées par rapport à celles qui ne portent pas d'étiquettes ou l'étiquette « cette proposition ne respecte plus la demande de l'utilisateur ».

Le 3ème paramètre le plus important est la faisabilité : une proposition qui a l'étiquette : « cette proposition est gênée par une interférence » sera toujours laissée de côté par rapport à une proposition qui n'a pas ce type d'étiquettes.

Enfin, si ces contraintes dures sont franchies, la fonction d'évaluation évalue numériquement :

- La mission : en fonction du nombre d'objectifs remplis (logarithmique). Un malus proportionnel au temps investi est retranché si la nouvelle proposition implique de laisser tomber un SAR en cours ;
- La survie : si la proposition permet de suivre une cible dangereuse ; si la proposition brouille un radar ennemi ;
- La SiTac : à l'aide de l'étiquette adéquate ;
- La continuation : l'allocation courante reçoit un petit « bonus » pour éviter les changements intempestifs.

Ces évaluations numériques sont sommées pour fournir une évaluation globale. Cette somme est pondérée différemment en fonction de la politique. Si la politique est défensive, la survie et la SiTac ont un poids élevé. Si la politique est offensive, la mission a un poids élevé. La politique influence le poids donné à chaque contrainte.

Fonctionnement général du SMA

D'un point de vue général, si on fixe une situation donnée (supposons que le temps ne change pas), la décision de la proposition d'activation à activer est itérative et gloutonne : l'agent décisionnaire choisit une première proposition d'activation puis avertit les agents tactiques. Les agents tactiques prennent cette proposition et l'étendent de sorte à inclure leur point d'intérêt tactique. Ils font évaluer ces nouvelles propositions par les autres agents qui peuvent indiquer si une proposition permet de satisfaire plusieurs points d'intérêt tactique commun. Si les propositions qu'ils font sont plus utiles que la proposition courante, alors la proposition courante est changée pour une proposition plus utile et le cycle se répète. Sinon, les agents tactiques ne peuvent pas améliorer l'utilité globale en cherchant à améliorer leurs points d'intérêts tactiques locaux : l'utilité globale peut difficilement être améliorée. Dans une situation dynamique, la décision est relancée. En effet, si les agents peuvent faire de nouvelles propositions d'activation (parce que la situation a évolué dans ce sens) alors ils doivent les proposer. Par conséquent, le système s'adapte au changement de situation. Une vue d'ensemble du SMA abstrait est proposée en figure 5.4.

5.2.1.4 Simulation

La figure 5.5 présente la fenêtre principale du simulateur. Le scénario suivi par la plateforme correspond à une sous-partie du scénario présenté en 4.1.1.2 : Au début de la mission, la plateforme et ses senseurs sont en veille. La plateforme entre ensuite dans le rayon de détection du radar (objet A). Il avertit le GM de ce fait et active la fonction brouillage. Suite à cela, le GM donne l'ordre d'imager le point B. La plateforme image ce point et se remet en veille. Plus loin, sur la trajectoire définie par le GM, la plateforme détecte un bâtiment civil (objet B) qui émet un signal radar. Le SMS avertit le GM et le GM lui demande d'imager un point proche du bâtiment civil. Une fois que cette prise d'image a été effectuée, la plateforme envoie l'image capturée au GM. Le GM demande ensuite une exploration d'une zone large (Zone C) afin de trouver un objet en déplacement, suspecté être une arme mobile du type missile balistique. Au cours de l'exploration de cette zone, la plateforme découvre la présence d'un radar de conduite de tirs non armée (objet D) et d'un camion transportant des missiles en direction de ce radar (objet E). Suite à la découverte de ces deux objets, le SMS en informe le GM. Le GM demande de neutraliser l'objet en déplacement et demande la confirmation de la destruction de cet objet au SMS par l'intermédiaire d'une imagerie SAR-Spot.

La partie haute de la fenêtre montre l'ensemble des agents et l'évolution de leurs objets virtualisés sur le théâtre. La partie basse de la fenêtre montre l'ensemble des capteurs et fonctions engagées et réservées pendant le déroulement du scénario au gré des décisions des agents.

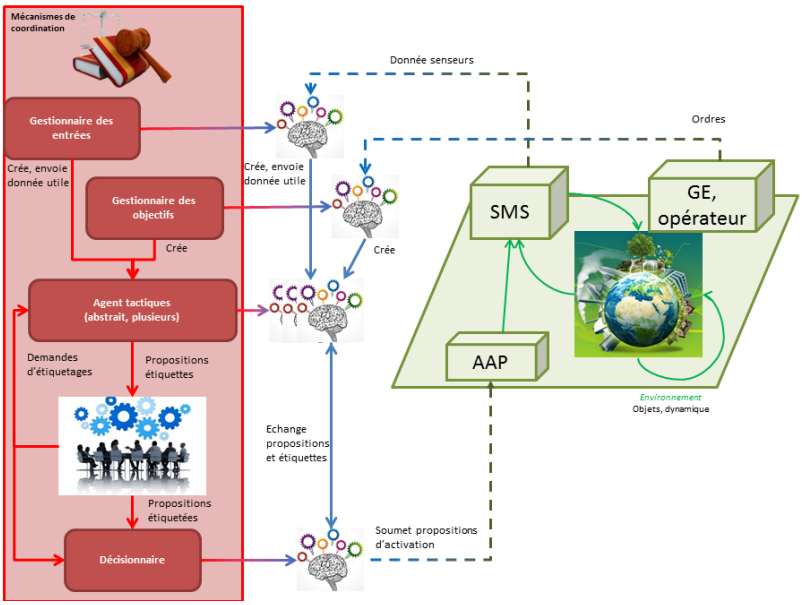


FIGURE 5.4 – Vue d'ensemble de la solution SMA du simulateur RAMSES I.

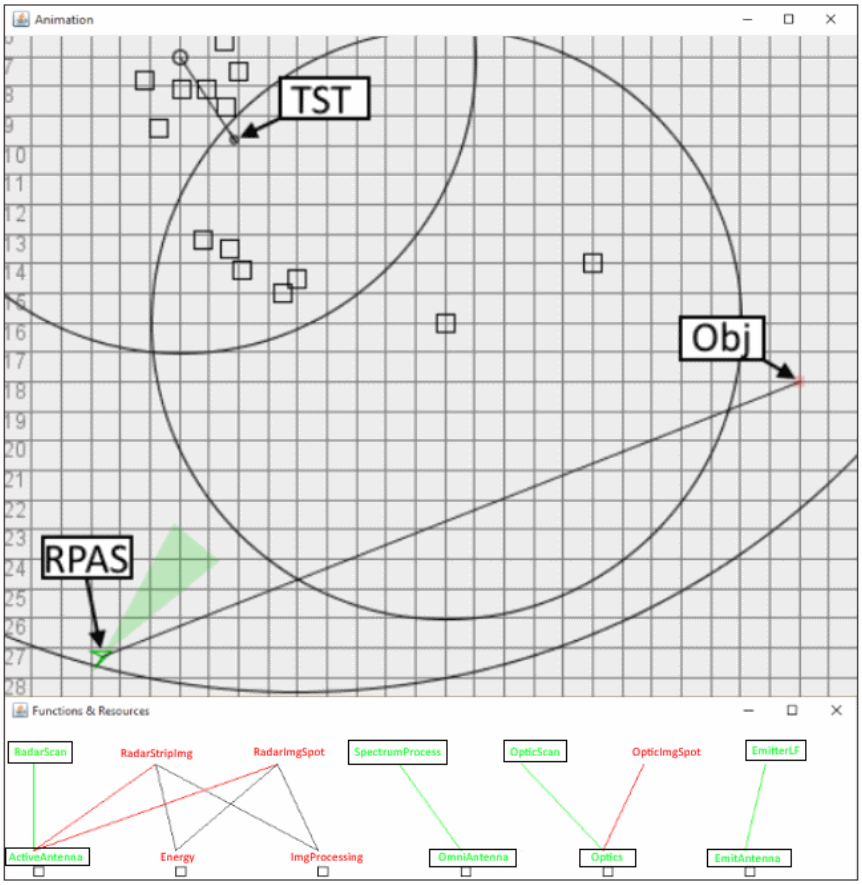


FIGURE 5.5 – Fenêtre principale du simulateur RAMSES I.

5.2.2 RAMSES II

5.2.2.1 Contexte et objectifs

Suite au développement de RAMSES I, ayant généré un fort intérêt, une démonstration plus représentative des systèmes de senseurs et des décisions opérationnelles a été implémentée. Cette simulation, portant le nom de RAMSES II, intègre plusieurs nouveautés et améliorations :

- Une simulation du théâtre améliorée, prenant en compte la propagation des signaux EM ;
- Des senseurs plus réalistes ;
- La prise en charge de modes capteurs ;
- Des actions senseurs détaillées avec la prise en charge de durées de tâches variables ;
- Une gestion temporelle des évènements, de la planification et de l'ordonnancement ;
- Un moteur multi-agent sur mesure.

Architecture logicielle et configuration matérielle :

Afin de simuler le théâtre efficacement un logiciel dédié a été implémenté. Ce logiciel, la partie *Génération terrain* de RAMSES II, permet de générer l'ensemble du monde dans lequel le système de senseurs est plongé. Le système de senseurs est simulé par un deuxième logiciel : la partie *Système* de RAMSES II. Pour des problèmes de puissance de calcul, les deux logiciels ont été pensés pour être exécutés sur deux machines différentes. La figure 5.6 montre la configuration matérielle et le rôle de chaque machine pour la simulation.

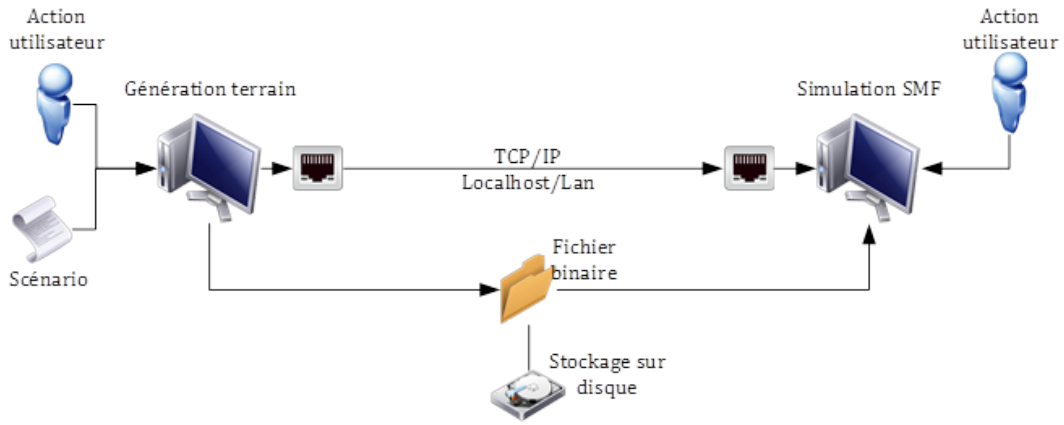


FIGURE 5.6 – Vue de la configuration matérielle pour la simulation de RAMSES II.

Puisque la trajectoire de la plateforme est contrainte au SMS, le plan de vol et les trajectoires/positions des objets du théâtre sont définis dans la partie génération. L'ensemble des signaux et leur propagation y sont simulés. Puisque l'attitude de la plateforme et les positions de chaque objet sont connues pour chaque instant du scénario lors de la génération il est possible de calculer l'ensemble des signaux émis par chaque objet et parvenant à la plateforme.

Après génération du théâtre et envoi des signaux et position de la plateforme au logiciel de simulation système pour chaque pas de temps du scénario, ces signaux sont récupérés et pris en compte par les senseurs. Au sein de la simulation de chaque senseur, une méthode permet de «trier» les signaux reçus en fonction de la configuration, du mode et de l'état du senseur à chaque instant. Cette partie diffère du fonctionnement d'un senseur réel, mais réalise le travail d'interface avec la simulation du théâtre. Le générateur du théâtre est considéré comme le serveur et la simulation système comme le client. Il est possible d'exécuter les deux logiciels sur la même machine en saisissant comme adresse de serveur *localhost* ou *127.0.0.1* lors du démarrage de la simulation système.

Afin de faciliter l'analyse d'un scénario donné, il est possible de générer l'ensemble du théâtre sur toute la durée du scénario et de l'enregistrer dans un fichier binaire. Ce fichier peut ensuite être chargé par la simulation système pour être joué/rejoué.

5.2.2.2 Génération du théâtre

La figure 5.7 montre l'interface de gestion de la simulation du théâtre et du scénario.

Le bandeau en haut de la fenêtre permet de contrôler la vitesse de simulation (par défaut 10ms donc 100Hz), les pauses, relancer la simulation, mesurer le temps de calcul de chaque step et le paramétrage de la visualisation.

La partie centrale de la fenêtre est une vue de la simulation courante. Cette visualisation est aussi appelée «vue de dieu» puisqu'elle décrit avec exhaustivité tous les éléments du théâtre, toute la réalité de la scène.

La partie basse de la fenêtre décrit l'ensemble des signaux calculés parvenant à la plateforme pour le step courant. Pour chacun des objets et chaque time-step le vecteur des données transmises à la simulation système sont :

- *ID* : identifiant de l'objet ;
- *Name* : nom de l'objet ;
- *Side* : la force d'appartenance ;
- *Cap* : le cap de l'objet ($0\mathrm{deg}$ = nord) ;
- *Speed* : vitesse de l'objet en nœuds ;
- *Alt* : altitude de l'objet en pieds ;
- *East* : position de l'objet (mile nautique) sur l'axe des abscisses (croît de l'ouest vers l'est) ;
- *South* : position de l'objet (mile nautique) sur l'axe des ordonnées (croît du nord vers le sud) ;
- *Distance* : Distance à la plateforme ;
- *Bearing* : Gisement de l'objet par rapport à la plateforme ;
- *Elevation* : Site de l'objet par rapport à la plateforme.

Les paramètres suivants sont des paramètres calculés en fonction des caractéristiques des objets :

- *OptVis* : visibilité optique ;
- *EMOpt* : visibilité par ondes radiofréquences ;

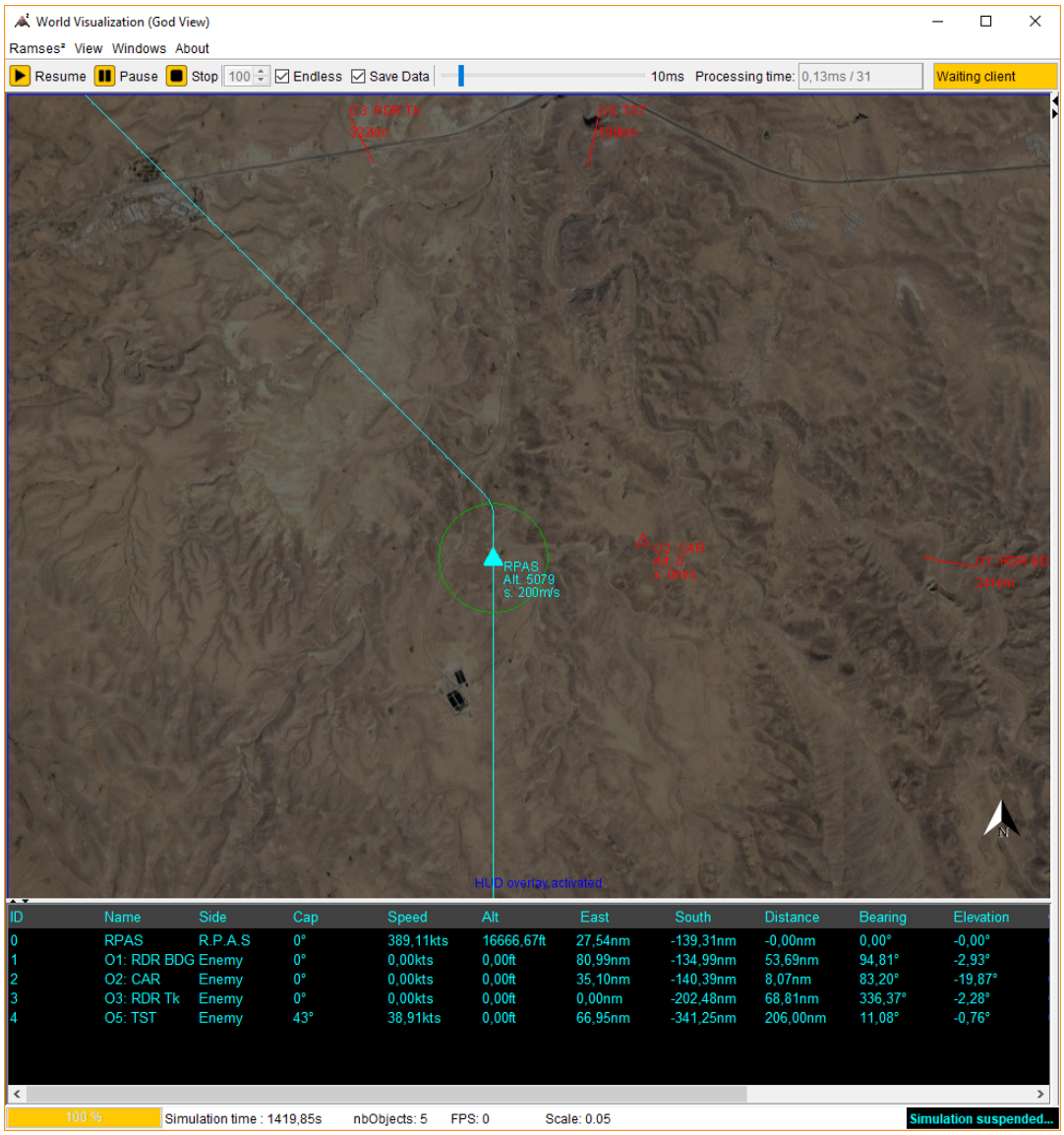


FIGURE 5.7 – Vue de la configuration matérielle pour la simulation de RAMSES II.

- *IREmit* : émissions d'ondes infrarouges ;
- *LazEmit* : émissions Laser ;
- *EMEmit* : liste d'émetteurs, chaque émetteur est constitué d'une fréquence et d'une puissance d'émission sur cette fréquence.

L'ensemble de ces valeurs représentent la capacité qu'un senseur aura à détecter un objet. Chacun des coefficients est calculé en fonction du contexte et des atténuations exercées par les éléments (nuages, atmosphère, etc.) situés entre l'objet et la plateforme. Les valeurs *OptVis* et *EMOpt* sont du type coefficients de visibilité alors que *IREmit*, *LazEmit* et *EMEmit* sont du type puissance de réception. Ceci permet de différencier les grandeurs physiques observables de manière passive avec celles nécessitant une stimulation extérieure. Par exemple, le coefficient de visibilité optique est multiplié par la source ambiante de luminosité afin d'en déduire le signal qui sera ensuite atténué par l'atmosphère. L'état actuel du développement de la simulation

considère que la luminosité de la scène est toujours à la valeur 1.0. La variation de cette luminosité fera varier le signal optique reçu par les senseurs de la plateforme. Les signaux IR, EM et Laser sont émis par l'objet puis sont atténués par l'environnement.

Les caractéristiques d'émission/réflexion des objets sont contenues dans un fichier chargé lors du lancement de la génération terrain.

La détection d'un objet par un senseur dépend de plusieurs facteurs :

- **Son domaine de fonctionnement :** le senseur doit fonctionner dans le domaine physique d'une des grandeurs émises par l'objet ;
- **Sa configuration :** le senseur doit avoir l'ouverture orientée vers l'objet (vérification des angles d'arrivée des signaux) ;
- **Sa sensibilité :** la sensibilité est la caractéristique inhérente à tout capteur, si le signal reçu est inférieur à la sensibilité du capteur alors le capteur ne pourra pas détecter le signal.

D'autres actions, telles que l'identification, sont plus complexes qu'une détection : l'identification optique par exemple, est basée sur l'interprétation d'une image. Les algorithmes des capteurs et boitiers de traitements ont été simplifiés afin de symboliser l'identification si nécessaire. Dans ce cas les données de simulation du théâtre, dont le type d'objet, sont utilisés. Ceci décrit l'unique mécanisme scripté et non réaliste du simulateur RAMSES II. L'identification par signature RF est quant à elle fonctionnelle. L'agent souhaitant identifier un signal RF décrit dans les *EMEmit* utilise la base de données des signatures RF mise à disposition des agents tactiques en vue d'identifier les objets.

5.2.2.3 Plateforme SMA

Une première version de RAMSES II utilisait le framework multi-agent *Jiac-V*. Ce framework avait été choisi puisque les sources sont disponibles et son utilisation est relativement simple. Jiac-V permet une implémentation rapide de SMA dans le langage Java. Après une intégration de Jiac-V au sein de RAMSES II il a été décidé de développer une plateforme multi-agent spécifique plus adaptée au contexte du SMS. Les raisons ayant motivé ce choix sont :

- **Une meilleure gestion du temps :**

  Les agents correspondants sont chacun lancés dans un Thread indépendant (processus). Laisser l'agent réaliser du travail sur les données dont il dispose sans en contrôler le déclenchement pose problème lors de l'analyse de son état à chaque instant. L'objectif de la plateforme multi-agent personnalisée est de faire fonctionner l'agent au tick de la simulation (évènement ayant lieu à l'ajout d'une unité élémentaire à l'horloge de la simulation), afin de contrôler précisément les traitements réalisés.

- **La possibilité de mettre les agents en pause :**

  Le dépouillement des données de simulation est un travail fastidieux dans le cas où l'ensemble des données sont tracées puis analysées. Disposer de la vue de la simulation à un instant en visualisant l'état de chaque agent et ses actions permet de contrôler le flux de travail global et de vérifier la cohérence du fonctionnement du SMS. Permettre la mise en pause de l'ensemble des agents et de la simulation en un clic permet de disposer de cette vision. Cette capacité

est appuyée par une gestion du temps améliorée. L'agent réagissant au tick de la simulation, arrêter le tick permet d'arrêter les agents.

- **Une gestion plus simple du nœud d'agents :**

  Le SMA nécessaire à l'établissement de la situation tactique dans cet état d'avancement de la simulation est simple, il n'y a pas d'exigence de mise en réseau des agents autrement qu'avec les autres agents de la situation tactique du SMS. Un nœud d'agents simple correspond à une liste actualisée régulièrement afin de lancer les processus agent au rythme de la simulation.

- **Des services d'annuaire et de supervision personnalisés :**

  Une implémentation simple des services d'annuaires et de supervision permet d'accélérer le fonctionnement de la plateforme multi-agent et de disposer d'une personnalisation totale du SMA.

- **Un fonctionnement bas-niveau :**

  Les agents tactiques traitent des données haut-niveau se déclinant en ordres bas-niveau. Contrôler à bas-niveau le fonctionnement des agents permet de disposer d'un meilleur contrôle des plans senseurs et une forte réactivité des agents en évitant des niveaux d'abstraction pouvant être lourds.

- **Des agents simples au fonctionnement et à l'instanciation rapides :**

  La simplification du nœud d'agent et une architecture logicielle agent simplifiée permettent de lancer une centaine d'agents en un temps dix fois plus court qu'avec une plateforme multi-agent de type Jiac-V. Cette caractéristique est essentielle pour la réactivité globale du SMS

- **Un fonctionnement plus proche d'une version embarquée :**

  Une version réelle d'un tel système multi-agent sera implémentée en C/C++ afin de s'adapter du mieux possible aux contraintes matérielles de la plateforme.

L'objectif du SMS étant d'exploiter des données élémentaires avec un minimum de latence, l'ensemble des fonctionnalités présentées ci-dessus permettent de satisfaire cet objectif. Les fonctionnalités des frameworks rencontrés habituellement sont adaptées à des simulations plus haut-niveau mettant en avant des processus cognitifs plus avancés (simulation de foules, de marché du travail, etc.).

Les agents tactiques implémentent la classe abstraite *agent*. Les agents respectent le cycle de vie de Jiac-V. La figure 4.22 présentée en 4.2.2.3 est celui implémenté au sein des agents du SMA. La fonction d'exécution est lancée à chaque tick de la simulation via le mécanisme Java d'observer. Permettant une construction souple et réactive de l'organisation des agents. Malgré cette synchronisation au tick un thread est lancé lors du lancement de l'exécution du processus principal des agents. Lors de la mise en pause de la simulation, donc l'arrêt de la production de ticks et l'incrémentation de l'horloge du simulateur le processus de calcul des agents peut continuer l'exécution de ses instructions jusqu'à terminer le cycle en cours.

5.2.2.4 Architecture SMS implémentée

L'architecture implémentée suit le modèle présenté en figure 4.26 de la section 4.2.3. Chacun des blocs du système est simulé et fonctionnel. L'ensemble de l'architecture est visualisable en temps réel au sein du simulateur du SMS. La figure 5.8 montre l'interface d'observation de l'architecture système simulée.

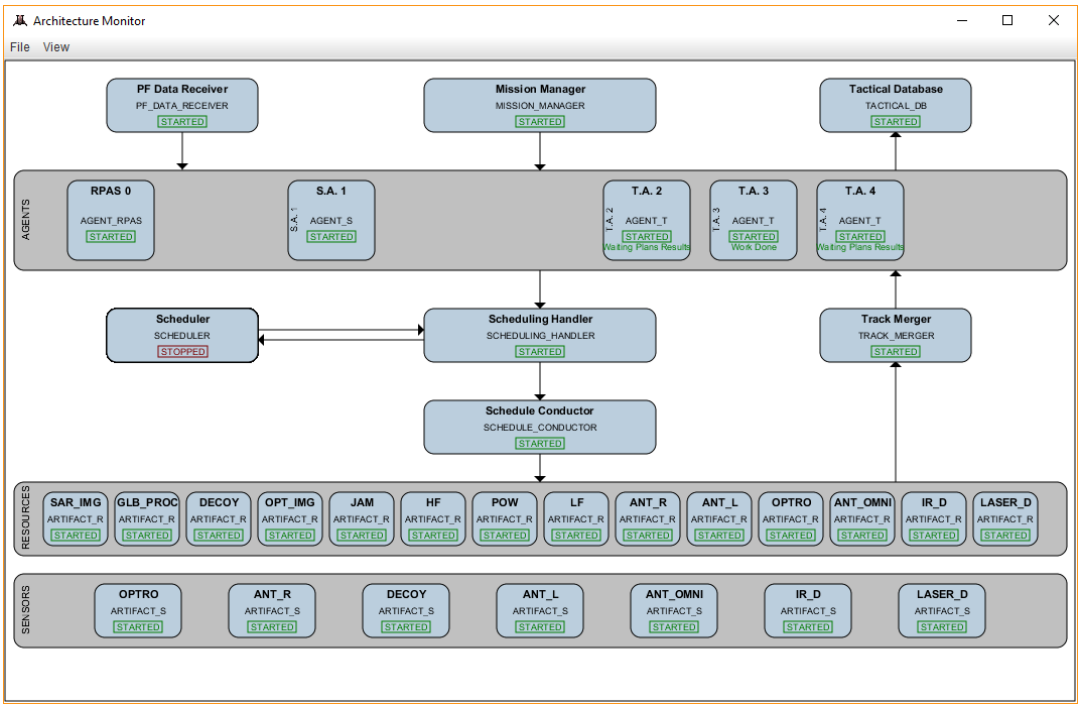


FIGURE 5.8 – Vue d'ensemble de la solution SMA du simulateur RAMSES I.

Chaque rectangle de l'interface correspond à un objet héritant de la classe *SystemBlock*. Cette classe dispose de plusieurs attributs/méthodes permettant de remonter les états de fonctionnement de chaque composant système.

Plusieurs composants ont été ajoutés à l'architecture présentée en 4.26 :

- ***Scheduling Handler* :**

  Le *Scheduling Handler* ou gestionnaire d'ordonnancement est l'entité gérant le déclenchement des ordonnancements. Lorsque les plans lui parvenant respectent certains critères (priorité supérieure à la priorité moyenne des plans ordonnancés en cours, nombre de plan suffisant, ressources ciblées non surchargées) le *scheduling handler* peut lancer une instance de l'ordonnanceur en ajoutant les plans les plus récents. Les autres instances d'ordonnanceur sont alors invalidées si le calcul n'est pas terminé lors du lancement d'un nouvel ordonnanceur, conformément au principe présenté en 4.3.1.2.

- **Agents services :**

  Les agents services, (S.A, *Service Agent*) ont pour objectif de fournir des fonctionnalités de support pour les agents tactiques. Autant d'agents services peuvent être instanciés afin de réaliser des tâches de traitement ou de suivi des agents tactiques, par exemple des fonctions de comptage des agents ou la surveillance de leur fonctionnement par envoi/réception de message tests (type ping/ACK).

- ***Tactical Database* :**

  La *Tactical Database* ou base de données tactique peut être considérée comme un dictionnaire permettant le rattachement de mesures à des informations dites ici "tactiques", comme celles par exemple permettant l'identification et la reconnaissance d'objets à travers l'analyse de leurs émissions RF.

Cette base de données est accessible à tout agent tactique au cours de leur cycle de vie afin de transformer les produits senseurs en informations sur les types d'objets.

5.2.2.5 Simulation des Systèmes

Afin de disposer de la meilleure fidélité possible en simulation, l'ensemble des systèmes présentés en 5.2.2.4 ont été simulées afin de constituer l'architecture SMS de simulation, que ce soit avec un fonctionnement simplifié, scripté ou fin.

La fenêtre architecture permet de visualiser en temps réel l'architecture mais non le fonctionnement et les sorties des systèmes. Afin d'observer le fonctionnement interne des systèmes de l'architecture plusieurs interfaces ont été développées, toutes adaptées au fonctionnement interne du système. Ces interfaces sont regroupées au sein d'une même fenêtre, visible sur la figure 5.9 (réduite pour la mise en page).

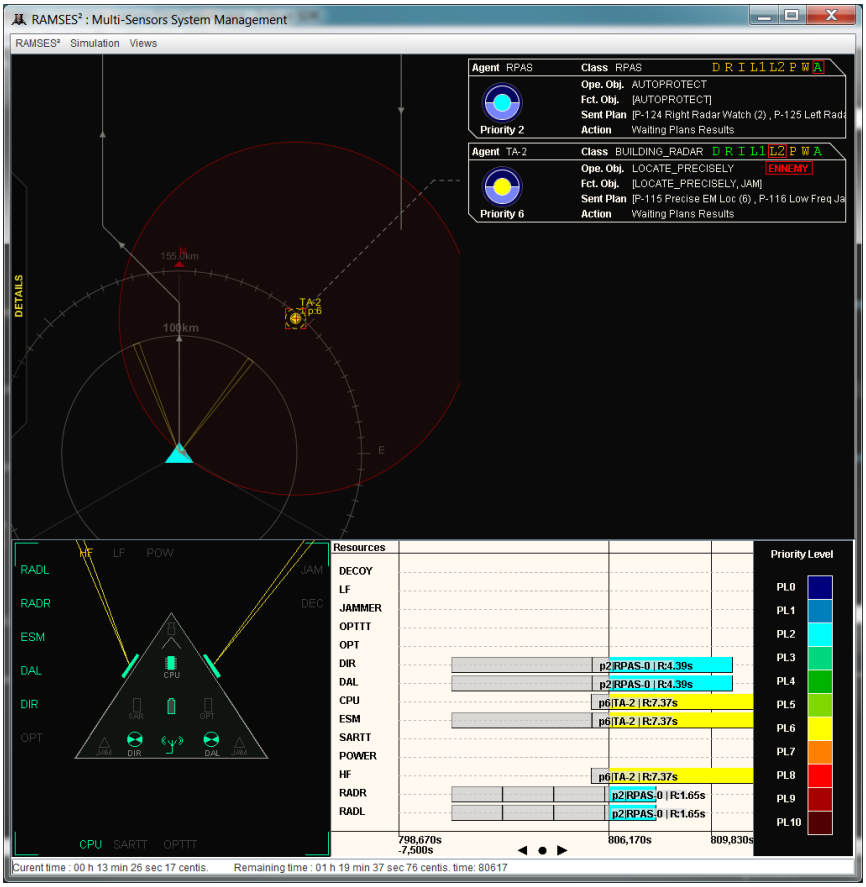


FIGURE 5.9 – Fenêtre de la simulation des systèmes.

Les interfaces permettant de visualiser et contrôler les systèmes vont maintenant être détaillées :

- **Vue senseurs :**

  La vue senseur diffère de la vue de dieu présentée dans le logiciel de génération du théâtre par le fait que les informations visibles sur l'interface sont uniquement celles remontées par les senseurs et interprétées par le SMS et les agents. Cette vue senseur place la plateforme toujours orientée vers le haut

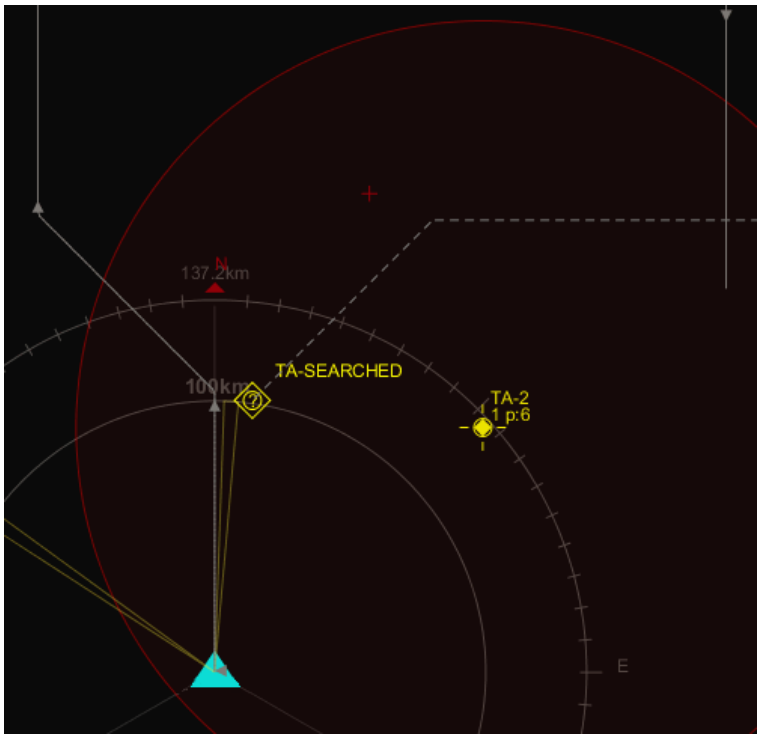
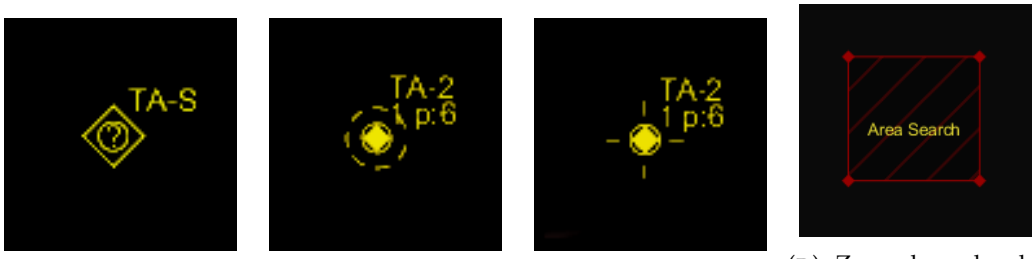


FIGURE 5.10 – Représentation du théâtre perçu par le SMS.

des graphiques. Ceux-ci tournent autour de la plateforme lorsqu'elle suit sa trajectoire. La vue senseur représente en temps réel la situation tactique et le groupe d'agent, avec les agents chargés de la recherche d'objets sur le théâtre, les agents virtualisant les objets recherchés (Fig. 5.11a) et localisés largement (Fig. 5.11b) ou précisément (Fig. 5.11c). La couleur des objets correspond à leur niveau de priorité.



(A) Objet recherché. (B) Localisé (large). (C) Localisé (précis). (D) Zone de recherche d'un objet.

FIGURE 5.11 – Représentation des objets sur la vue senseurs.

- **État des senseurs :**

  L'état de chaque ressource et leur activité est symbolisé via l'interface graphique présentée en figure 5.12. Cette interface regroupe les ressources de plusieurs types et leur activité :

  - Senseurs :

    RADL : Antenne active coté gauche,

    RADR : Antenne active coté droit,

    ESM : Antenne omnidirectionnelle,

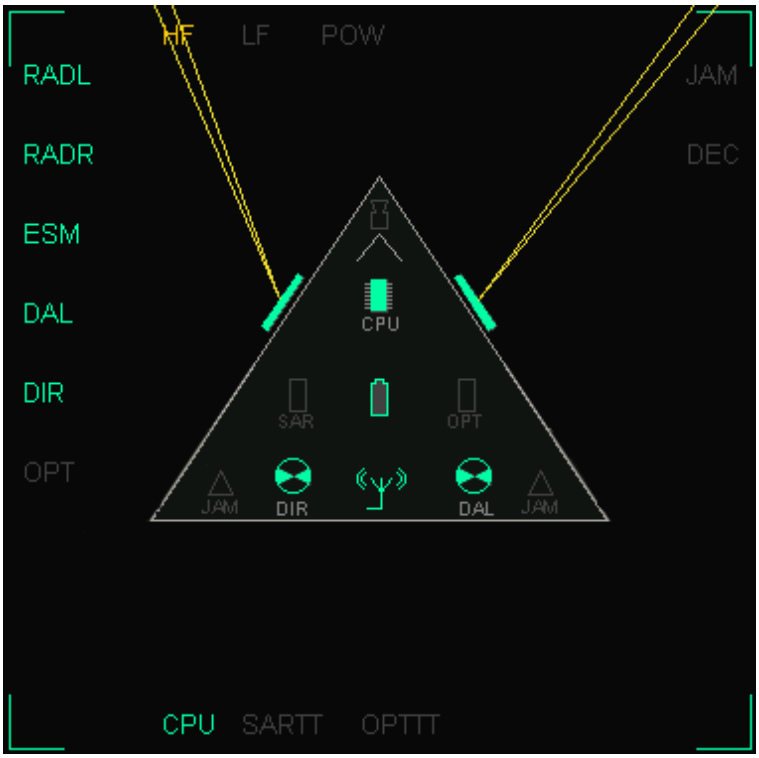


FIGURE 5.12 – Interface de visualisation des senseurs et ressources.

DAL : Détecteur d'alertes laser,
DIR : Détecteur infrarouge ;

- Effecteurs :
  JAM : brouilleurs,
  DEC : leurres ;
- Autres ressources physiques :
  CPU : ressource de calcul multi-usage,
  SARTT : Ressource de traitement SAR,
  OPTTT : Ressource de traitement optique ;
- Virtuelles :
  HF : Haute Fréquence,
  LF : Basse Fréquence,
  POW : Ressource haute consommation énergétique.

L'état des ressources est remonté en temps réel durant la simulation.

- **Interface Agents Tactiques :**
  Les agents tactiques présents au sein du système réalisent une part importante du traitement de l'information du SMS. L'interface visible en figure 5.13 contient plusieurs informations essentielles au contrôle du fonctionnement des agents. La priorité de l'agent calculée à partir des caractéristiques observées de l'objet, la classe et type de l'objet si disponibles (sinon une super-classe seulement), la progression dans la chaine opérationnelle DRIL-P et si des actions d'autoprotection ou de veille sont actives. Les plans compilés par l'agent et la phase actuelle du cycle de vie de décision sont aussi renseignés (*Sent plans* et *Action*).

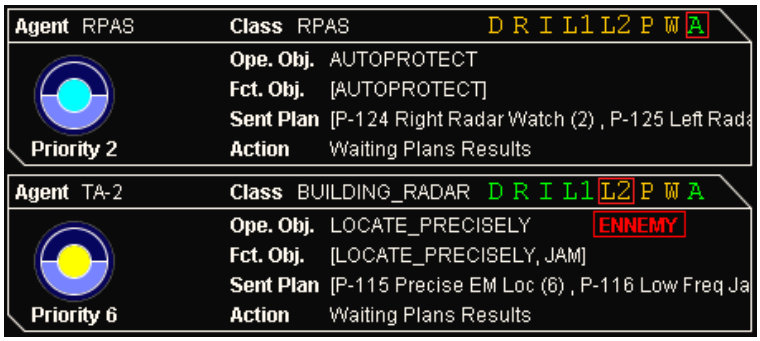


FIGURE 5.13 – Interface graphique représentant l'état des agents du SMA.

Un panneau de consultation des connaissances apparait lorsqu'un agent tactique est sélectionné sur la situation tactique, voir figure 5.14. Ce panneau permet de vérifier en temps réel les connaissances acquises, déduites et les plans générés par l'agent tactique sélectionné. L'ensemble de la carte des connaissances est alors accessible tout au long de la simulation. Cette visualisation ainsi que la vitesse de défilement réglable du moteur de la simulation permettent de vérifier l'ensemble des entrées/sorties des agents tactiques.

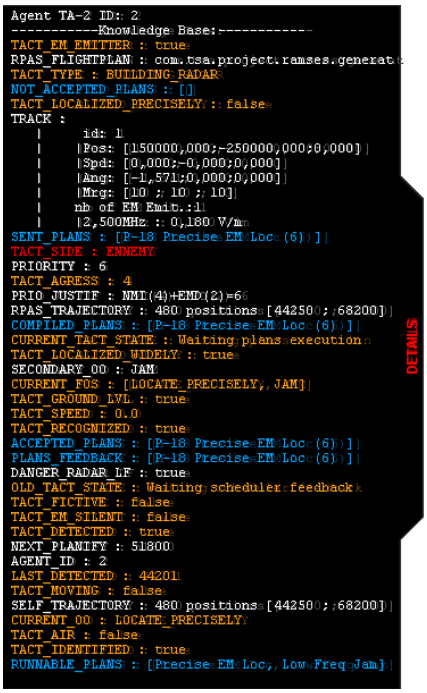


FIGURE 5.14 – Carte des connaissances de l'agent sélectionné.

- **Ordonnancement global :**

  Le panel d'ordonnancement sortant, représenté en figure 5.15 affiche la sortie de l'ordonnanceur global (*Global Scheduler*). La représentation est dynamique, la barre du temps courant de la simulation est fixe et les plans/tâches sont exécutés lorsque leurs dates de début coïncident avec celle-ci.

- **Niveaux de priorité**

  Le panel des niveaux de priorité permet d'identifier les différents niveaux de priorité des agents et plans produits par ces derniers présents à chaque

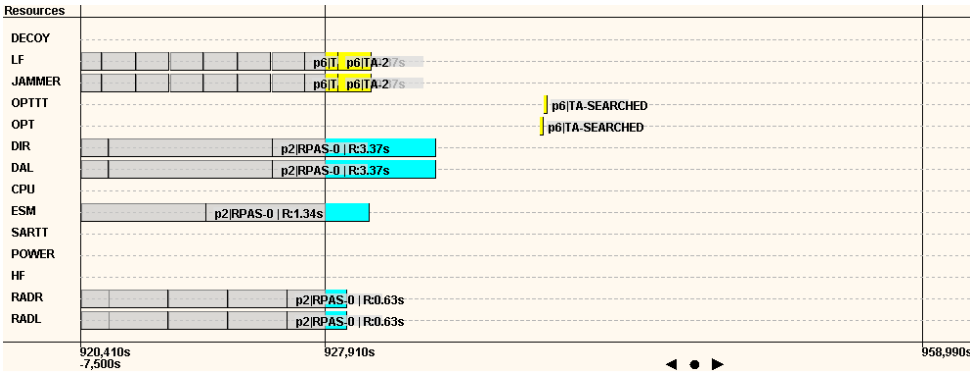


FIGURE 5.15 – Ordonnancement global.

FIGURE 5.16 – Niveaux de priorité.

instant dans le système. Ici 11 niveaux de priorité sont représentés, de 0 à 10, 0 étant le moins prioritaire. Voir figure 5.16.

- **Gestionnaire de mission :**

  Le gestionnaire de mission permet de lancer des requêtes de recherche d'objet au SMS, voir figure 5.17.

  Plusieurs options sont disponibles afin de saisir les caractéristiques de l'objet recherché. L'ensemble des caractéristiques saisies sont ensuite envoyées au SMS et un nouvel agent est créé et renseigné de ces informations. Si un objet est recherché, l'opérateur de la simulation peut saisir les coordonnées d'une zone de recherche et sélectionner l'objet de type *AREA*. Un agent de type zone est alors créé et sera chargé du lancement des plans les plus adaptés à la zone de recherche.

- **Interfaces des ressources physiques :**

  Plusieurs interfaces accessibles via les menus de l'application permettent de visualiser et contrôler les informations remontées par les senseurs du SMS ainsi que les modes de fonctionnement en temps réel (souple). Les figures 5.18 et 5.19 représentent respectivement le fonctionnement des antennes actives et de l'antenne omnidirectionnelle utilisées par les modes radars et guerre-électronique.

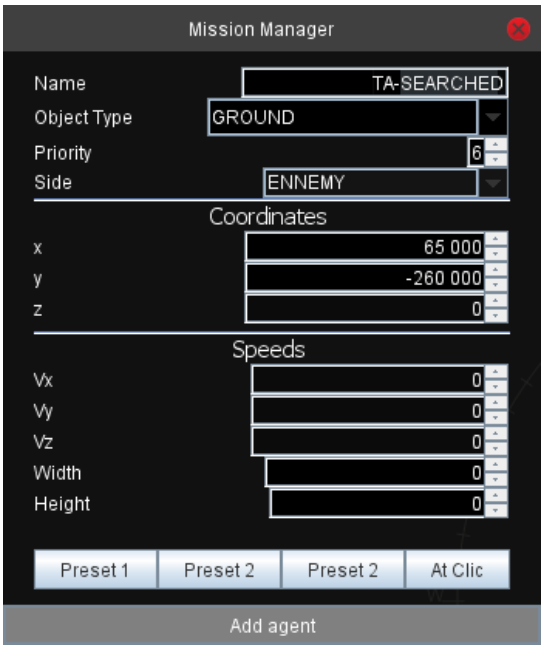


FIGURE 5.17 – Interface du gestionnaire de mission.

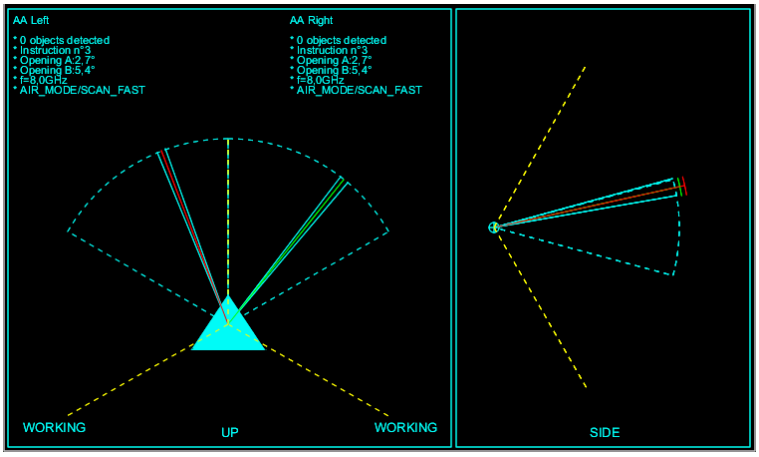


FIGURE 5.18 – Vue du fonctionnement des antennes actives.

5.2.2.6 Journalisation des évènements

Un grand nombre d'évènements prennent place au sein de l'architecture lors de l'exécution des systèmes simulés. Sortir les messages en console impacterait fortement les performances du système (la sortie standard ralentissant l'exécution du programme) et rendrait la consultation des évènements impossible face au nombre de messages créés à l'exécution. Pour garder trace des évènements, les visualiser efficacement et les stocker si désiré un outil de journalisation a été utilisé tout au long du développement du simulateur. Cet outil, appelé *logger*, journaliseur en français, est LOGMX développé par l'entreprise LightySoft. Cet outil est générique et permet d'ouvrir les flux de messages provenant de multiples sources et de dépouiller les données après et pendant exécution. L'outil a été interfacé avec le framework Java Log4j, permettant de sortir les messages d'évènements à LOGMX tout en limitant des retards d'exécution liés à la construction des messages, voir figure 5.20. L'ensemble des messages provenant des différentes classes du simulateur ont ainsi été

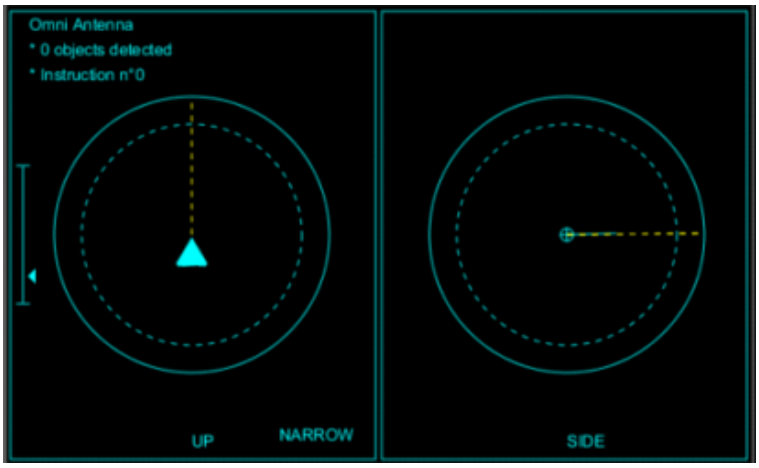


FIGURE 5.19 – Vue du fonctionnement de l'antenne omnidirectionnelle.

transmis à LOGMX afin de filtrer les messages visibles en sortie. Plusieurs niveaux de journalisation ont été exploités dans la simulation (*Trace*, *Debug*, *Warning*, *Error* et *Fatal*).

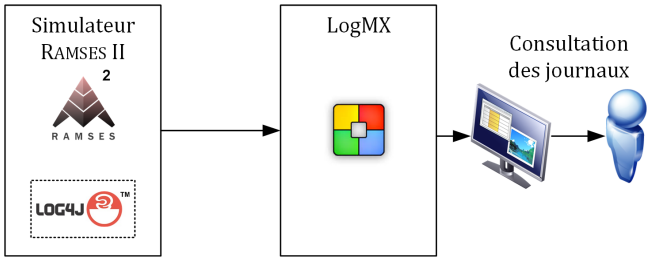


FIGURE 5.20 – Figure représentant le processus de journalisation des évènements du simulateur RAMSES II.

5.3 Résultats

Les deux simulateurs RAMSES I et RAMSES II ont tous les deux permis d'avancer le concept d'agents tactiques pour le contrôle d'un système multi-senseur.

5.3.1 Résultats RAMSES I

- **Autonomie face au contexte :**

  La première version de RAMSES permis de montrer la répartition du travail agents-tactiques/opérateur et plus particulièrement la délégation des tâches de calculs et de planification à l'agent tactique. Cette première version permit de montrer que l'ensemble des agents peuvent se coordonner afin d'optimiser la charge d'un senseur ou l'enchainement d'actions senseurs à exécuter afin de dérouler au mieux la mission sans l'intervention de l'opérateur.

- **Planification court terme :**

  Comme présenté en 5.2.1.3, RAMSES I est basé sur la proposition et l'optimisation des plans proposés par les agents tactiques. Cette planification est court terme (à l'instant pour le suivant) mais permet de disposer d'un plan

global optimisé au regard de l'ensemble des objets du théâtre et des objectifs de la mission.

- **Robustesse de la réplication d'agents :**
  Un mécanisme de sauvegarde des connaissances des agents tactiques a été implémenté afin d'illustrer le concept de réplication d'agent. À chaque instant de l'exécution de la simulation le comportement et les données sortant d'un agent sont définies à partir de l'ensemble de ses connaissances. Une sauvegarde et une transmission de ces données permettent de restaurer ou d'instancier l'agent dans le même état qu'avant la sauvegarde. Ce mécanisme permet une robustesse en cas de panne d'un agent tactique (machine à état dégradée ou données corrompues par exemple)

5.3.2 Résultats RAMSES II

- **Autonomie améliorée du SMS :**
  Les circuits de décision complexes des agents tactiques de RAMSES II permettent la construction de plans évolués et une bonne adaptation au théâtre lors du déroulement de la mission. Cette adaptation et l'autonomie des agents tactiques permettent une autonomie améliorée de l'ensemble du SMS.
- **Planification :**
  Le modèle de planification des plans senseurs permet une exécution fine des actions senseurs tout au long de la mission. La gestion du temps a été introduite avec RAMSES II et permet de planifier avec précision les actions senseurs en prenant en compte le plan de vol et les politiques de missions.
- **Ordonnancement :**
  L'heuristique d'ordonnancement permet un placement fin des tâches sur les timelines des ressources tout en conservant une réactivité élevée des prises de décisions des agents et plus généralement du SMS.
- **Synthèse et fusion intelligentes et adaptées de l'information :**
  La granularité élevée des informations remontées par les senseurs et de leur représentation au sein des agents tactiques permet de limiter la perte d'informations tout en synthétisant les informations sous une forme adaptée au regard des objets du théâtre.
- **Modularité et personnalisation :**
  L'ensemble des données permettant la prise de décision ainsi que les plans senseurs sont personnalisables. La machine à état tactique et les caractéristiques des senseurs et ressources représentées dans la simulation sont distinctes du cœur du moteur de simulation et permettent une modification du comportement général du simulateur avec un impact minimum sur le reste de la simulation.

5.3.2.1 Réactivité du SMS

La boucle d'asservissement présentée en section 4.2.3.3 exprime le délai $\mathcal{D}$ caractéristique de réactions de l'architecture, du traitement des évènements à la prise de décision s'en découlant. Les délais des différentes étapes sont schématisés de nouveau sur la figure 5.21.

Pour rappel, le délai $\mathcal{D}$ caractéristique de l'architecture est exprimé par l'équation 5.1 :

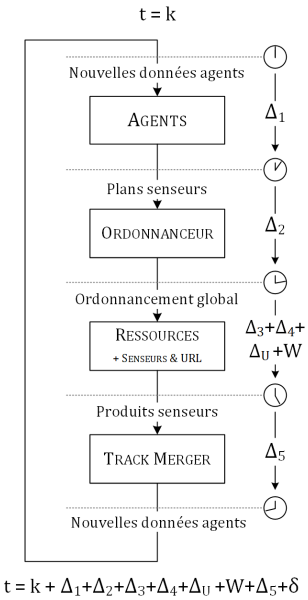


FIGURE 5.21 – Schéma du parcours et des temps de calcul des données dans le SMS.

$$\mathcal{D} = \sum_{N=1}^{5} \Delta_N + \delta \tag{5.1}$$

Ces différents délais sont relevables au sein de l'architecture lors de son exécution. Afin d'obtenir les dates des passages de chacune des étapes de la création des évènements aux prises de décision et exécution des plans par les ressources, un affichage de messages simples a été spécifiquement sorti en console, précédé du temps système en nanoseconde.

Le moment du scénario est situé autour de l'instant de détection de la menace radar de veille. La visualisation de la situation tactique à cet instant du scénario est visible sur l'affichage principal du simulateur, figure 5.24. Des exemples de messages relevés dans ce contexte sont visibles sur la figure 5.22.

Afin de mesurer les temps le plus efficacement possible, le simulateur a été configuré d'une manière permettant d'interpréter plus facilement les résultats. Ainsi, la boucle d'exécution principale du simulateur mettant en place un délai variable permettant de suivre un temps d'exécution à une échelle réaliste a été réduit à 0. Les algorithmes du simulateur s'exécutent alors au plus vite. Le temps de traitement Δ_U a été réglé sur 2ms. Les relevés des temps sont visibles dans le tableau 5.1. La colonne *Valeurs* a été calculée par la soustraction du temps précédent au temps de l'événement courant. L'événement relevé à Δ_1 prend place lors de la détection du radar de veille longue portée. L'agent le représentant est alors créé lors de la détection de son signal. Pour cette raison un temps supplémentaire à été introduit : Δ_c le délai nécessaire à la création d'un agent. Les délais de planification et de mise à jour des données tactiques au sein des agents ont été mesurés. Ces étapes sont les plus chronophages des processus agents. Dans le contexte de la détection d'un radar de veille la mise à jour des données agent prend en moyenne 3ms tandis que la planification (projection des trajectoires, vérification des compatibilités de plans, compilation des

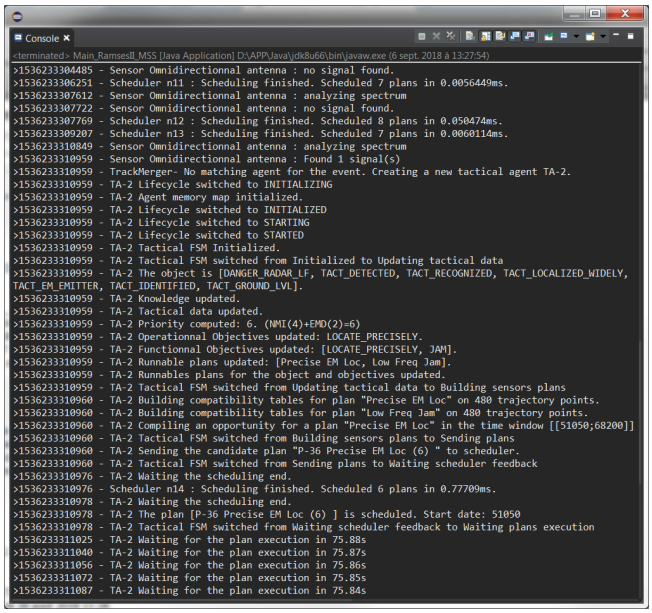


FIGURE 5.22 – Sortie console pour la mesure des délais d'exécution.

plans) occupe en moyenne 10ms. Les étapes de projection de la trajectoire de la plateforme et de construction des tables de compatibilités des plans occupent 90% du temps d'exécution de la phase de planification.

Les mesures réalisées permettent d'exprimer le délai caractéristique de l'architecture $\mathcal{D} = 19,368ms$. Le graphique figure 5.23 synthétise l'ensemble des délais rencontrés lors de l'exécution de l'architecture du SMS.

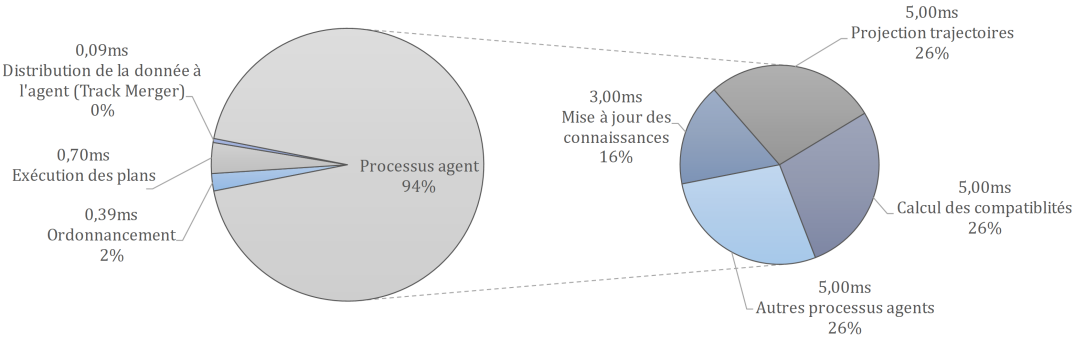


FIGURE 5.23 – Synthèse des temps d'exécution de l'architecture.

Conclusion

Les expérimentations réalisées pendant ces travaux et leurs résultats ont été présentés dans ce chapitre. Nous avons montré l'impossibilité de réaliser des tests sur des systèmes réels, les SMS étant des systèmes complexes encore en phase d'étude. Les simulateurs RAMSES I et RAMSES II mettant en œuvre les concepts, l'architecture et l'ordonnancement, détaillés en chapitre 4 ont été présentés. Nous avons montré par ces deux simulations l'efficacité des systèmes multi-agent et plus particulièrement des agents tactiques à générer des plans senseurs adaptés à leur environnement. RAMSES I permit de démontrer la faisabilité d'une architecture multi-senseur

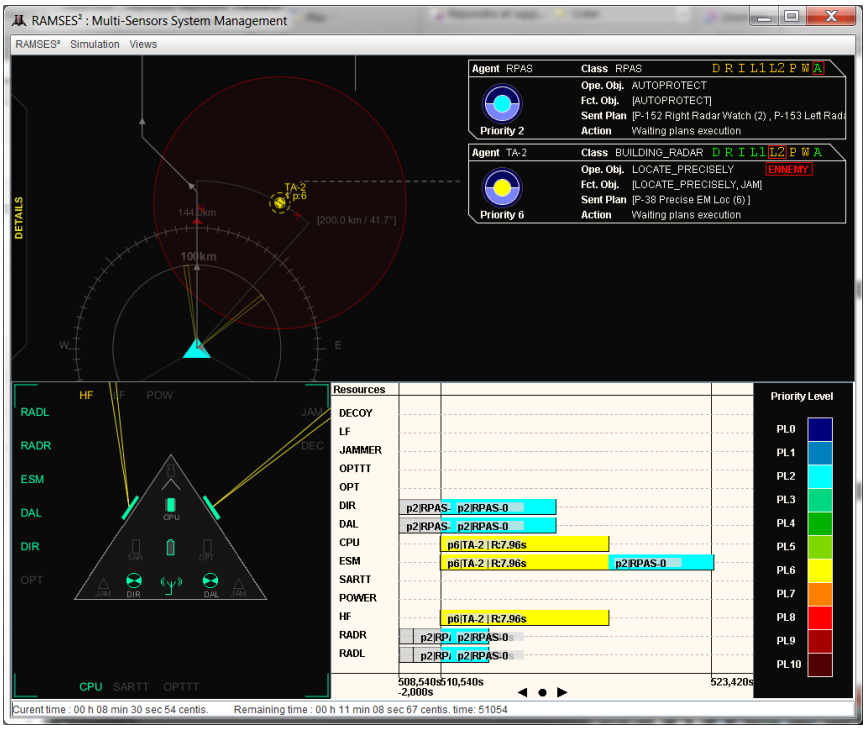


FIGURE 5.24 – Interface de RAMSES II après détection d'un radar de veille.

orientée agents avec négociation des ressources. RAMSES II a quant à lui démontré qu'un SMS, à base d'agents tactiques, doté d'un ordonnancement adapté permet l'ordonnancement de plans senseurs bas-niveau et ainsi de contrôler les senseurs avec précision en adéquation avec le théâtre d'opérations.

Événements	**Temps système** (ns de référence)	**Temps mesurés**	**Valeurs**
Nouvelles données agent récupérées	253 820 700 507 268	$t = k$	-
Données distribuées. Initialisation d'un nouvel agent	253 820 703 927 821	Δ_c	3,420ms
Agent créé et plan senseur généré	253 820 722 110 018	Δ_1	18,182ms
Ordonnancement global construit	253 820 722 504 372	Δ_2	0,394ms
Exécution de l'ordre senseur	253 821 027 480 714	W	304,976ms
Durée d'exécution fixée	-	Δ_U	2,00ms
Produits senseurs récupérés	253 821 030 181 246	Δ_3+Δ_4 $-\Delta_U$	0,700ms
Nouvelles données distribuées	253 821 030 272 529	Δ_5	0,091ms

TABLE 5.1 – Relevé des temps d'exécution des composants systèmes de l'architecture simulée au sein de RAMSES II.

Chapitre 6

Évaluation et Validation

6.1 Critères d'évaluation

6.1.1 Évaluation opérationnelle du SMS

6.1.1.1 Autonomie du SMS

Le comportement d'un agent reflète les connaissances et les actions pouvant être entreprises par un opérateur à l'égard de l'objet que l'agent virtualise. C'est-à-dire que pour chacun des objets connus et identifiés (ou classe d'objets) du théâtre des actions senseurs sont définies et applicables pendant la mission. Un des objectifs de l'architecture multi-agent pour système multi-senseur présentée dans ce manuscrit est de déléguer aux agents présents dans le SMS une part des connaissances et du travail actuel des opérateurs afin d'optimiser leur charge et concentrer leur attention sur des tâches de plus haut niveau.

Comme présenté en 4, les connaissances de l'opérateur mises à la disposition des agents sont sous la forme d'ontologies, de plans senseurs, de politiques de mission, de bases de données dites *tactiques* permettant l'identification ainsi qu'un processus complet (machine à états tactique) pour suivre l'évolution de l'objet et les requêtes de plans senseurs et données acquises le concernant.

La qualité des décisions prises par les agents, donc par l'ensemble du SMS (les agents réalisant la majorité des traitements, l'ordonnancement a un rôle d'arbitrage des plans senseurs construits) dépend donc des données mises à la disposition de ces derniers et de leur architecture interne. L'architecture globale présentée dans ce manuscrit a pour objectif de démontrer l'efficacité de l'agent tactique pour la prise de décision, mais ne propose pas une version finale/exhaustive des processus de décision de chacun des agents ni des ontologies qu'ils exploitent. La possibilité de personnalisation et la modularité de l'ensemble de l'architecture permettent son adaptation à différents types de missions, de contextes, voire de plateformes.

De la même façon qu'il est nécessaire aujourd'hui d'adapter les bibliothèques permettant l'identification de signatures RF lors de l'arrivée sur base d'un appareil il sera au minimum nécessaire avec un SMS à base d'agents d'adapter les différents plans et ontologies aux théâtres d'opérations sur lequel la plateforme sera amenée à opérer.

6.1.1.2 Délai de réponse du SMS

Le délai des traitements impose une limite aux processus de décisions appliqués par les agents. La caractéristique multi-processus par le biais des agents permet de répartir le calcul des décisions de l'ensemble des plans senseurs pour un théâtre sur plusieurs cœurs/cartes de calculs si besoin sans modifier le comportement général du SMS.

Les démonstrations RAMSES I et RAMSES II ont toutes les deux été exécutées sur des machines du commerce (processeur Intel i7), mais présentaient néanmoins des délais de calculs acceptables à ce stade des recherches. La complexité calculatoire algorithmique est maitrisée à partir d'algorithmes de tri et coupe, de sélection de plans, d'un mécanisme de gestion des priorités et d'une adaptation de l'ordonnancement au contexte (fenêtre d'ordonnancement ajustable) présentés en section 4.3.3.1. L'ensemble de l'architecture pourra par la suite être matériellement optimisée, afin d'adapter du mieux possible le matériel aux processus logiciels.

6.1.1.3 Synthèse de l'information

La synthèse des informations remontées par les senseurs est un enjeu majeur de la future génération de SMS. Le SMS présenté dans ce manuscrit permet une synthèse active de l'information, c'est-à-dire que les informations déduites et calculées sont à la fois un produit/une sortie du SMS, mais aussi une source/entrée pour d'autres processus internes de ce dernier, à la différence d'algorithmes de synthèse permettant uniquement le calcul de nouvelles informations non exploitées en interne par la suite. Au cœur de ce processus se trouve les agents et leurs connaissances : mises à jour en permanence, les agents infèrent à partir de ces dernières, les actualisent et demandent des actions senseurs afin de les enrichir. Cette boucle vertueuse permet le maintien et la synthèse des données d'état d'un objet et plus généralement de toute la vue senseurs pour l'ensemble du SMA.

6.1.2 Évaluation système du SMS

6.1.2.1 Quantité de re-travail pour évolution

La modularité du SMS est un des besoins essentiels exprimés par le monde industriel afin de réduire les coûts, les durées de développements et le taux de reprise de l'existant, voir besoin 9 section 4.1.2.3.

L'architecture à base d'agent présentée dans ce manuscrit compartimente dans différents composants l'interprétation, la décision et la planification des actions senseurs ainsi que l'ordonnancement et la fusion des produits senseurs. Compartimenter les traitements permet de limiter l'impact de l'ajout de nouveaux composants sur le reste du système ou d'identifier et d'en maitriser plus simplement les conséquences.

Hypothèses :

- Le senseur K n'est dépendant d'aucune ressource encore non représentée dans le SMS.
- Le senseur K est dépendant de la ressource virtuelle HF.
- Plusieurs plans peuvent être définis mettant en œuvre le nouveau senseur K :

- Un premier plan d'utilisation simple, le senseur K doit être réservé de manière synchrone avec la ressource virtuelle HF.
- Un deuxième plan, plus complexe, fait intervenir le senseur «Antenne Active», la ressource «SAR Processing» et la ressource virtuelle «HF-Band».

Étudions l'impact de l'ajout d'un nouveau senseur *K*, dans le SMS selon une approche ascendante (ou *bottom-up*, voir section 4.1.2.1). La première étape est d'ajouter matériellement le senseur au SMS :

1. **Ajout de la tête senseur**

   Les têtes senseurs sont les parties matérielles physiquement actives du SMS, permettant de recueillir les informations du théâtre. Le senseur K est composé d'une tête senseur matérielle unique positionnée sur la plateforme et faisant partie du réseau de senseurs disponibles au sein du SMS.
2. **Ajout du driver**

   Le driver est un composant logiciel qui permet de contrôler la ressource physique afin de la piloter et de regrouper/encapsuler les données de manière compatible avec le SMS englobant la ressource. Le driver est le composant passerelle entre le monde logiciel et matériel.
3. **Création de la ressource**

   La ressource (voir section 4.2.3.1 composant 12) est un composant logiciel permettant de traduire les plans et tâches en instructions senseurs. La ressource gère les modes de fonctionnement du senseur, sa direction, les durées d'exécution précises en fonction de l'ensemble des informations spécifiées dans les tâches qu'elle reçoit.

La chaine constituée des trois éléments *Tête senseur/Driver/Ressource* permet d'intégrer un nouveau senseur au SMS, mais n'est pas suffisante afin de l'utiliser de manière autonome. Une étude *top-down* ou descendante permet d'identifier les éléments nécessaires :

4. **Ajout de la définition du senseur et de ses modes**

   Cette étape consiste à ajouter la définition du senseur et de ses modes, c'est-à-dire de donner un nom unique à chacun des modes qu'il fournit, et de les ajouter à l'ensemble des ontologies accessibles aux agents. Les modes mettant en jeu plusieurs senseurs sont aussi définis lors de cette étape. L'intégration des ontologies au sein des agents permet à ces derniers d'être en mesure de désigner plusieurs modes fournis par le nouveau senseur pouvant atteindre les objectifs opérationnels qu'ils se sont fixés, ou par le biais du gestionnaire de mission.
5. **Définition de la portée des modes du senseur**

   Une fois que les agents sont capables de choisir les modes exploitant le nouveau senseur, ceux-ci doivent être capable de planifier leur exécution et de vérifier leur compatibilité avec le contexte (trajectoire, météo, politiques). Les contraintes contextuelles (distances, plages de visions exploitables, etc.) sont alors définies pour chacun des modes. Les agents sont capables de choisir un sous-ensemble de modes permettant d'accomplir leurs objectifs opérationnels.

6. **Écriture des plans senseurs**

   Les plans décrivant les réservations de ressources et les contraintes temporelles sont définis de façon à ce que les agents puissent planifier leur utilisation et les placer dans le temps. L'agent retient le mode préféré/optimal pour l'objectif opérationnel entre tous les modes senseurs compatibles avec le contexte retenus précédemment. Si un des modes du nouveau senseur K est retenu alors il sera envoyé à l'ordonnancement.

7. **Ajout des données d'interprétation des produits du senseur**

   Après exécution des tâches et des instructions au niveau matériel les données remontées par la tête senseur et le driver sont ensuite fusionnées par le track-merger (voir section 4.2.3.1 composant 9). Après fusion l'agent à l'origine de la requête est enrichi des données remontées si besoin. Les données remontées par la ressource peuvent être de natures différentes en fonction des modes et senseurs présents sur le SMS. Afin de donner aux agents la capacité d'interpréter correctement les données remontées par les senseurs des données d'interprétation doivent leur être mises à disposition, en enrichissant par exemple les bibliothèques et bases de données (voir *Tactical Database* en section 5.2.2.4).

La figure 6.1 permet de visualiser les ajouts nécessaires à l'architecture afin d'exploiter le nouveau senseur K.

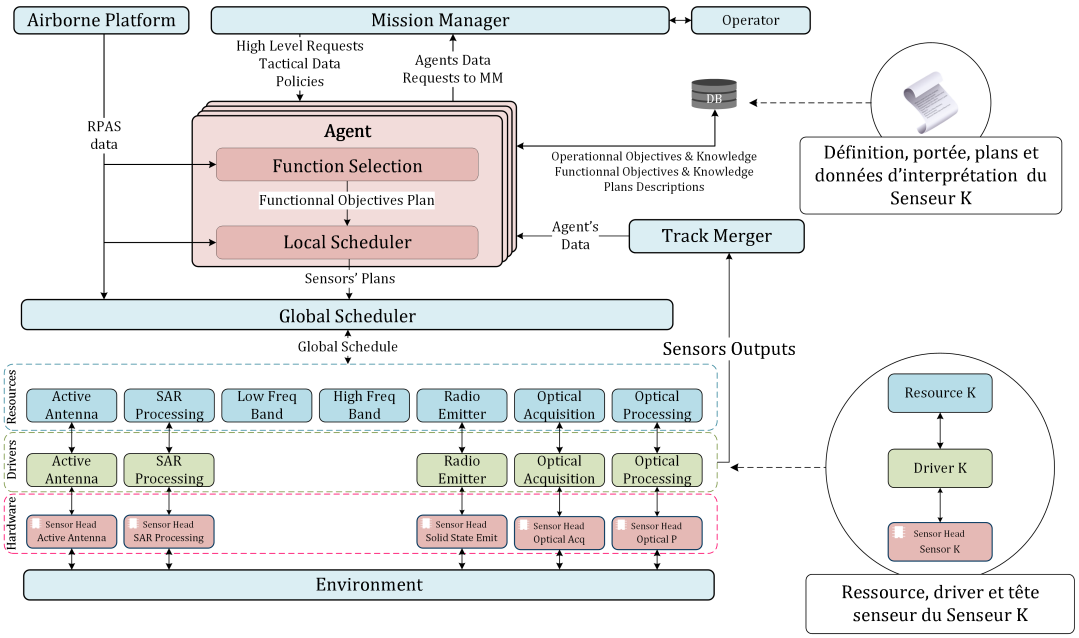


FIGURE 6.1 – Schéma de l'ajout d'un senseur au sein de l'architecture SMS.

L'aspect modulaire de l'architecture peut être optimisé si l'ensemble des composants logiciels et définitions nécessaires à l'exploitation d'un senseur sont empaquetés afin de simplifier le chargement/ajout d'un senseur à un SMS. De cette façon, des modules, sous la forme de senseurs et de leurs dépendances, peuvent être ajoutés/-déplacés sur différentes plateformes afin de disposer, par exemple, d'une flotte de plateformes avec des SMS à géométries variables.

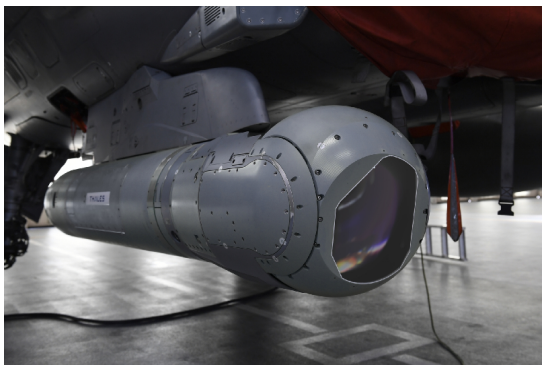


FIGURE 6.2 – Un pod multifonction Talios.

La figure 6.2 montre une nacelle senseur pouvant être ajoutée de manière modulaire sur une plateforme aéroportée. Ce type de nacelle est appelée un pod et permet d'embarquer des instruments spécifiques en mission. Ce pod est multifonction et met à disposition un ensemble de fonctions au pilote en vol. L'installation de ce type de matériel à bord d'une plateforme ayant un SMS à bord est un bon exemple de besoin de modularité et de mise à disposition de fonctions d'un nouveau senseur à un SMS existant. Il est possible d'interfacer un tel pod avec un SMS orienté SMA si l'ensemble des plans, définitions et données permettant l'analyse sont embarqués dans la nacelle et partagés au SMS lors de sa connexion. Les agents tactiques pourront planifier l'utilisation du pod au même titre que n'importe quel autre senseur du SMS.

Afin de conserver une flexibilité et modularité dans le système de senseurs l'ordonnanceur est indépendant de la création des plans et des agents, les paramètres passés à l'ordonnanceur, nécessaires au lancement du processus d'ordonnancement ne dépendant pas des agents ni des composants amont. Ceci permet de découpler ce dernier du système multi-agent et ainsi de limiter le taux de rework de l'ensemble du SMS en cas de modification de l'ordonnanceur.

6.1.2.2 Flexibilité et adaptabilité de l'architecture

La section précédente, 6.1.2.1, a permis d'identifier les contours des éléments à ajouter au SMS afin de disposer des fonctions/modes apportés par un nouveau senseur. La modularité du SMS permet une flexibilité de haut niveau de ce dernier aux différents types de missions, environnements et plus généralement contextes. Cette opération permet une flexibilité moyen terme et à tout instant du cycle de vie du SMS.

À moyen terme, la modification des plans et noyaux agents permet de réaliser des mises à jour et des modifications en profondeur maitrisées du SMS. L'agentification réalisée permet de limiter l'apparition d'effets de bord lors de la modification du code interne des agents (agents asynchrones et autonomes). Il est par exemple possible de modifier la machine à état tactique d'un agent afin de réaliser des traitements très différents des précédents au sein de l'agent sans impacter le reste du SMS. La création de nouveaux plans dans la base de données des plans pouvant être choisis par les agents permet une adaptation de haut niveau de l'ensemble du SMS. Par exemple, mettre en place un plan complexe exploitant 2 senseurs et 5 ressources est possible uniquement en suivant les étapes 4/5/6 présenté en section 6.1.2.1.

Sur le court terme, sur l'échelle d'une opération complète, les plans et les données des bibliothèques permettent une adaptation du SMS au contexte lors de la préparation de mission via le chargement ou déchargement de plans et de données permettant l'exploitation des senseurs et de leurs produits.

Sur le très court terme, les politiques et les requêtes SMS permettent de régir le fonctionnement du SMS avec autorité (via l'agent d'autorité absolue, voir section 4.2.4.2) tout en conservant une autonomie tout au long de la mission.

La place des senseurs, des ressources, du track merger et de l'ordonnanceur au sein de l'architecture permet d'imaginer rapidement des agents avec une portée supérieure de celle présentée dans ce manuscrit. La capacité communicante des agents tactiques ici présentés permet à ces derniers de coexister avec de nouveaux types d'agents. Cette expérience a été réalisée une première fois lors de l'ajout de l'agent tactique « zone » au sein du SMA. L'agent zone est un agent tactique héritant de l'ensemble des caractéristiques d'un agent tactique ainsi que de la machine à états tactique modifiée de façon à accomplir du mieux possible des plans optimaux pour ce type d'objet. Des connaissances propres à l'agent zone sont instanciées en son sein et la boucle de traitement de l'information lui est propre, permettant de lancer des plans senseurs à la suite dans l'ordre qu'il désire sans que l'agent ne requière la compréhension de ses semblables.

De la même façon, certains agents pourront être soutenus par des algorithmes de Machine Learning (apprentissage par la machine) afin d'apprendre les résultats des simulations, des entrainements et missions. Donner la capacité à un agent, d'observer et d'apprendre les requêtes qui lui sont soumises, les résultats des actions senseurs qu'il a passées et les transformations des caractéristiques de l'objet qu'il virtualise représentent permettrait de disposer des bases d'un agent tactique apprenant, intelligent et évolutif. Le schéma figure 6.3 montre les entrées sorties de l'agent pouvant être apprises par l'agent.

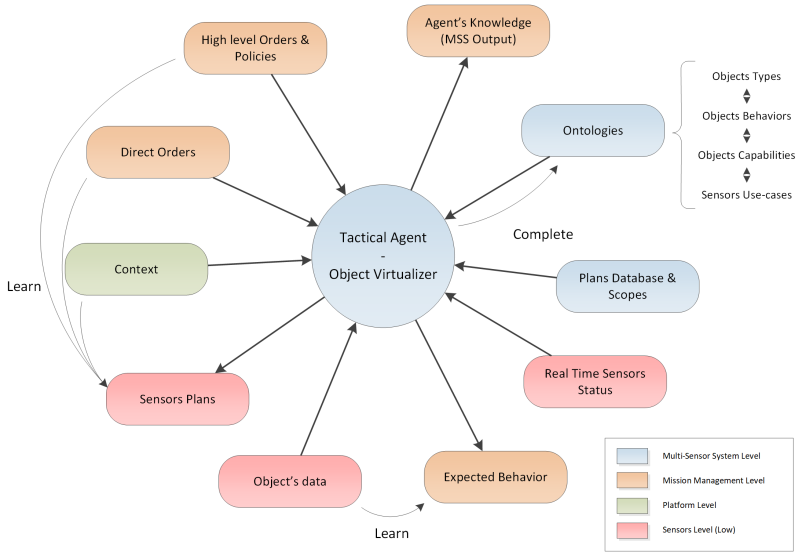


FIGURE 6.3 – Schéma des possibilités d'apprentissages au sein de l'agent tactique

6.1.2.3 Finesse de spécification des plans

Les plans de tâches présentés en section 10 permettent une définition fine des réservations des ressources pour l'exécution des plans senseurs. Face à la précision demandée par l'exécution des actions senseurs la description des plans doit accepter des jeux de contraintes strictes. Les contraintes de précédences de tâches présentées dans ce manuscrit en section 4.3.2.4 définissent le schéma de réservation des ressources avec exactitude si nécessaire. Pour les plans demandant le plus de précision il est possible de contraindre chacun des débuts des tâches afin de fixer exactement les enchainements de tâches et les délais entre celles-ci (contrainte présentée sur la figure 4.44 montrant une contrainte temporelle de type délai Δ exact entre 2 débuts de tâches).

La possibilité de faire varier la durée des tâches au sein des plans de tâches est rendue possible par la spécification des contraintes adoptée pour l'ordonnancement ainsi que par une écriture précise des plans senseurs. Les durées de tâches sont obtenues lors de la compilation des plans par les agents, ces derniers accèdent aux fonctions de calculs de durées spécifiques aux plans sélectionnés.

La complexité des plans exécutés au sein de la plateforme est sans limites : il est possible dans les cas les plus complexes de construire un ensemble de plans élémentaires puis de les assembler au sein de plans regroupant l'ensemble des tâches afin de construire un plan unique indissociable. Ceci permettra au SMS de mettre en place de futures actions senseurs complexes tout en conservant sa capacité à planifier les actions au regard du contexte.

La précédence de plans permet de définir des séquences capteurs personnalisées, pouvant évoluer tout au long de la vie de la suite de senseurs. Les précédences de plans sont définies lors de la compilation des plans par les agents puis sont ensuite prises en compte par l'ordonnanceur (voir section 1).

6.1.2.4 Généricité de l'architecture

L'architecture présentée dans ce manuscrit est générique de par l'ensemble des mécanismes permettant la modularité/flexibilité et peut être instanciée à des systèmes de senseurs de plateformes différentes. Nous pouvons citer par exemple une plateforme du type frégate. Le système de senseurs d'une frégate est un bon candidat à une architecture à base d'agents. Composé de radars, de sonars, de senseurs électromagnétiques, les frégates sont des plateformes amenées à intervenir dans des cadres opérationnels proches des plateformes aéroportées du type RPAS ayant des chaines opérationnelles proches : besoin de détecter, reconnaître, identifier, localiser et de s'autoprotéger, etc.

L'ensemble des plans peuvent être redéfinis pour correspondre aux contraintes des ressources et senseurs de la frégate. Le SMS de la frégate peut conserver une majeure partie des caractéristiques des agents et une partie des ontologies du SMS aéroporté. Les caractéristiques physiques d'une frégate telles que ses dimensions/masse et donc les dimensions des réseaux et systèmes de son SMS peuvent néanmoins mener à des adaptations de l'architecture. Les limites de masse des boitiers permettront ainsi de mettre en place des calculateurs plus performants afin de supporter des algorithmes plus puissants, parallélisés et de la redondance.

6.1.2.5 Robustesse

Le SMA dispose de plusieurs mécanismes permettant de récupérer en cas d'erreurs provoqués par une faute logicielle, une panne matérielle ou une perte d'information de type corruption de données :

- **Cycle de vie des agents :**
  Le cycle de vie de l'agent tactique permet à tout moment à l'agent d'observer son état courant afin d'adapter son fonctionnement ou d'alerter en cas de défaut et d'en informer des services dédiés du SMA. Les caractéristiques de réflexivité de l'agent peuvent être apportées par le biais du langage de développement de l'agent ou activement via une expression complète lors des changements d'état au sein des connaissances de l'agent si le langage dans lequel l'agent est implémenté n'est pas réflexif. L'observation par l'agent du déroulement de son propre code peut permettre d'activer des procédures de récupérations ou sauvegardes automatiquement lors de la détection de divergences.
- **Redondance d'agents :**
  Les agents étant des entités relativement légères et autonomes il est simple de les dupliquer au sein d'un SMA. Les règles de basculement d'un agent à son agent redondant peuvent varier en fonction du type de faute détectée et de l'entité l'ayant détectée.
- **Watchdog agent**
  Un watchdog fonctionnant en parallèle du cycle de vie des agents permet de réinitialiser l'agent lorsqu'un signal envoyé régulièrement lors de l'exécution du cycle de vie principal met trop de temps à lui parvenir.
- **Connaissances de l'agent :**
  Les connaissances de l'agent sont toutes regroupées au sein d'un même objet logiciel : la base de connaissances de l'agent aussi appelée carte des connaissances. Les données manipulées par l'agent pendant son exécution sont stockées et mises-à-jour au sein d'un même objet. Ceci implique qu'il est simple de sauvegarder et réinstancier les connaissances de l'agent localement ou sur une autre plateforme. Les connaissances de l'agents accumulées lors de l'exécution représentent l'ensemble des données utilisées, déduites lors de l'exécution. Une réinstanciation de sa mémoire permet d'instancier l'agent de nouveau. En cas de faute, en fonction de la gravité de celle-ci, l'agent peut donc être tué et réinstancié au sein du même SMS ou un différent.
- **Services du SMA :**
  Lorsque l'agent diverge de son fonctionnement nominal, mais que ce dernier ne lève pas d'alerte des services spécifiques du SMA peuvent se charger de la détection des erreurs et de la réinstanciation des agents en cas de fautes. Ces services peuvent être basés sur différents principes : respect de normes de valeurs de sorties, comparaisons de résultats sortants d'agents redondés ou observation de la cohérence générale du contexte.

La réplication des connaissances des agents est un mécanisme puissant apportant plus que de la robustesse : la transmission de ces connaissances permet de recréer des copies conformes à l'agent source sur d'autres plateformes ou au sein du C2.

Cette démarche permet de disposer de jumeaux numériques d'agents au sol afin de vérifier la cohérence des décisions prises par la plateforme.

Aussi, transmettre les connaissances des agents tactiques à des plateformes disposant de capacités différentes construites sur la même architecture permet un partage des données sans limite entre les différentes plateformes d'une même force. Les ontologies et données à l'origine de la modularité et flexibilité des SMS présentées en section 6.1.2.1 seront différentes en fonction des plateformes et senseurs présents sur le SMS : le fonctionnement des agents tactiques et les ontologies opérationnelles peuvent néanmoins être communs à toutes les plateformes s'échangeant les données afin d'assurer un fonctionnement correct en réseau.

6.1.2.6 Autonomie

La montée en autonomie des systèmes d'une plateforme aéroportée dont son système de senseurs implique des prises de décisions du niveau opérationnel de la part de ses systèmes. Des systèmes permettant la prise de ce niveau de décisions n'ont jamais été embarqués jusqu'à présent. La comparaison de la nouvelle architecture avec l'existant est donc non pertinente. La réelle comparaison de performance peut donc être réalisée uniquement entre un opérateur et la nouvelle architecture. Une expérience spécifique peut être réalisée afin d'évaluer l'apport de l'architecture en termes d'autonomie. Celle-ci reprend le concept fondateur de la méthode d'évaluation d'algorithmes cognitifs présentés en section 4.2.1.5. L'ensemble des systèmes composant la suite de senseur virtuelle et de son opérateur interfacé au GM (circuit de décision A) sont placés en parallèle d'un opérateur senseur isolé ayant l'ensemble du contrôle de sa suite de senseurs (circuit de décision B). Les deux circuits de décision A et B sont interfacés tous deux avec le même environnement synthétique composé d'un théâtre d'opérations synthétique et de senseurs virtuels.

Une métrique permet de mesurer le nombre d'actions réalisées par chacune des chaines de décisions tout au long de la mission.

L'évaluation de l'autonomie du SMS à base d'agent peut être appréciée en comparant les résultats de la métrique sur les deux chaines de décisions.

La première chaine de décision comporte deux types de décisions : les décisions humaines (opérateur du GM) et les décisions système (agents tactiques). La deuxième chaine de décision est, quant à elle, contrôlée par l'opérateur : les décisions sont donc uniquement d'origine humaine.

6.2 Validation

6.2.0.1 Validation des besoins opérationnels et industriels

Pour rappeler la section 4.1.2, le contexte opérationnel et industriel demande la vérification de plusieurs caractéristiques essentielles à l'acceptation de l'architecture comme solution pouvant servir en situation réelle. Reprenons ci-dessous les besoin opérationnels déjà présentés et tentons de montrer en quoi la solution présentée en 4.2 et 4.3 satisfont ces besoins.

Validation des besoins opérationnels : (Besoins exprimés en section 4.1.2.2)

Besoin 1 : Prédictibilité des résultats

Plusieurs composants et mécanismes ont été conçus afin de garantir une prédictibilité des résultats. L'architecture de l'agent tactique, basé sur deux machines à états permet de prédire en fonction de l'état en cours lors de l'exécution les différentes phases du cycle de vie de l'agent et donc des phases de calculs, projections, déduction de connaissances et compilation de plans. La mémoire de l'agent dont sa carte des connaissances permet d'avoir à chaque instant une vue complète de l'état de fonctionnement de l'agent tactique. Couplée à ses machines à état, composées de son cycle de vie et sa machine à états tactique, chacune des sorties de l'agent est vérifiable et prédictible.

La priorité déterminée par les règles, politiques et algorithmes de l'agent tactique résume la perception et la connaissance d'un objet en une valeur (ici entière) évoluant au cours du temps. La priorité découle de la carte des connaissances de l'agent à un instant donné. Les mécanismes de justification du niveau de priorité permettent à un opérateur de comprendre à tout instant quelles données sont exploitées et remontées lors du calcul de priorité.

La compilation des plans senseurs est réalisée par les agents à partir de leurs connaissances et des structures de plans définies dans les bases de données de plans chargées en préparation de mission. Les calculs réalisés lors de la construction des plans, de la projection de l'attitude de la plateforme à partir du plan de vol jusqu'aux durées des tâches exécutées par les ressources sont déterministes. Les modèles de performances permettant le calcul des durées des actions senseurs sont aussi des calculs déterministes.

L'ordonnancement est basé essentiellement sur la priorité des plans compilés par les agents tactiques. L'ordonnancement n'accepte pas de retard d'exécution de tâches. L'ordonnancement sortant est prédictible dans le sens où l'ensemble des plans les plus prioritaires du set de plan soumis seront ordonnancés tant que les contraintes peuvent être satisfaites.

L'ensemble de la chaine de décision est donc déterministe et donne des résultats prédictibles tant que les données permettant la prédiction sont accessibles. Il est possible à partir de l'ensemble des données, modèles de performances, bases de données et signaux (état des senseurs, variables environnementales, données plateformes) disponibles au SMS de reproduire le comportement du SMS au sol à des fins de diagnostic et monitoring des systèmes (jumeau numérique du SMS).

Besoin 2 : Fonctionnement par règles

Comme détaillé dans la section précédente, le fonctionnement des agents est basé sur les ontologies, les plans senseurs, les bases de données permettant la prise de décision des actions senseurs. De pair avec le fonctionnement interne des agents (machines à état et cycle de vie) ces données constituent ensemble des règles suivies par le SMS lors de son fonctionnement. Les règles peuvent être changées et modifiées afin de correspondre au contexte opérationnel.

Certaines règles opérationnelles telles que l'autorité maximale du gestionnaire de mission sur le SMS a été instanciée au sein de l'architecture via des mécanismes dédiés, l'agent de priorité absolue dans cet exemple. La nature modulaire des agents tactiques et leur placement dans l'architecture permettent d'apporter de nouveaux mécanismes avec un taux de reprise minimal tout en permettant des changements en profondeur de l'architecture.

Les politiques données par le gestionnaire de mission au SMS permettent de contraindre sous la forme de règles simples l'ensemble des décisions prises par les agents du SMA.

Besoin 3 : Flexibilité de fonctionnement

D'un point de vue opérationnel, la flexibilité du SMS est atteinte par la capacité qu'a l'architecture à accepter à la fois un contrôle autonome des senseurs tout en intégrant des requêtes senseurs directes et haut-niveaux. La prise en compte des ordres en plus des objectifs déterminés en autonomie des requêtes bas et haut niveau est rendue possible par la caractéristique exhaustive de la représentation des objets du théâtre en plus d'une gestion des priorités par l'ordonnanceur. Voir section 6.1.2.2.

Besoin 4 : Modularité

Du point de vue opérationnel, la modularité présentée en 6.1.2.1 permet de disposer d'un SMS modulaire. Pouvoir personnaliser les SMS de chaque plateformes permet aux opérationnels de s'adapter du mieux possible à chaque missions, tout en disposant d'un nombre de senseurs restreint. Pouvoir connecter un pod plug&play compatible SMA permettrait de disposer d'un SMS à géométrie variable, reconfigurable rapidement.

Besoin 5 : Faible latence

Les résultats de l'expérimentation en section 5.3.2.1 montrent que dans un contexte de développement tel que décrit dans le chapitre 5, c'est-à-dire avec des ressources matérielles de calcul limitées et un langage fonctionnant sur machine virtuelle, les mesures des délais de la simulation 5.2.2 sont en adéquation avec les besoins de réactivité actuels et moyen terme.

Besoin 6 : Exploitation du plan de vol

Le plan de vol est une entrée exploitée par les agents afin de projeter l'attitude de la plateforme sur les prochains instants du vol de la plateforme. Ces projections permettent de définir quels sont les plans senseurs acceptables pour accomplir ses objectifs opérationnels propres. Un exemple de projection de plan de vol a été présentée en section 4.3.2.5, cas d'étude de la compilation d'un plan SAR.

Besoin 7 : Robustesse et résilience

Les sections 5.3.1 et 6.1.2.5 ont montré en quoi les agents et leur structure permettent une robustesse et résilience adaptée au besoin opérationnel. Plusieurs mécanismes en plus des mécanismes de récupération internes aux agents peuvent être introduits au sein du SMA, tels que des agents services dédiés, une copie systématique de l'ensemble des données sur une plateforme déportée, etc. La redondance et l'emplacement de chaque mécanisme peuvent être adaptés aux architectures matérielles et logicielles (langage d'implémentation et technologies) choisies afin de disposer d'une robustesse adaptée.

Validation des besoins industriels (Besoins exprimés en section 4.1.2.3)

Besoin 8 : Décentralisation de la conception

La capacité modulaire de l'architecture apporte une possibilité de développement elle aussi par module (voir section 6.1.2.1). Ceci permet de décentraliser la conception par modules. Certains modes senseurs/fonctions nécessitent la coopération de plusieurs senseurs distincts, demandant dans ce cas une coopération de plusieurs modules senseurs, avec des données propres aux modes individuels de chaque module et des données communes. Par exemple, lors du développement de deux nouveaux senseurs, deux équipes de développement distinctes peuvent considérer indépendamment un senseur, ses modes, ses données de plans et d'interprétation. Dans le cas où des fonctions communes sont envisagées, donc des plans senseurs dans lesquels les deux senseurs sont exploités, alors des données dédiées à ces plans doivent être définies.

Besoin 9 : Modularité

Le besoin industriel en modularité est différent de celui de la modularité opérationnelle. La configuration générale du SMS par la modification du réseau de capteurs et des plans permet aux opérationnels de personnaliser le SMS. La modularité industrielle quant à elle permet à différentes équipes d'étudier et développer des portions du SMS par modules. L'existence des plans, des agents tactiques, de leurs machines à états et des ontologies permet aux équipes de développement logiciel et matériel de maitriser les tenants et aboutissants de la prise de décision au sein du SMS. Les autres composants système tels que l'ordonnanceur et les ressources donnent aux ingénieurs la maîtrise du workflow général de l'architecture.

L'ensemble des interfaces permettent les échanges de données au sein de l'architecture sont robustes et pérennes vis-à-vis du cycle de vie de l'architecture, permettant de faire varier la configuration générale du SMS tout en conservant son workflow original : la modification des systèmes tout en exploitant les interfaces disponibles au sein de l'architecture permet de construire plusieurs configurations, afin par exemple de mettre en place plusieurs version d'architectures tout en conservant le squelette standard du SMS. La gestion de configuration, la maintenance et les évolutions sont alors mieux gérées et maitrisées.

6.2.1 Performances générales de l'architecture SMS

La performance générale d'un SMS et de son architecture telle que celle présentée dans ce manuscrit peut se décomposer selon plusieurs axes :

- **Autonomie**

  L'autonomie du SMS est dynamique et ajustable en fonction des préférences choisies par l'opérateur. Le système est capable d'optimiser la charge de travail opérateur en le suppléant dans les actions les plus fastidieuses en cours de mission. L'opérateur peut ainsi se consacrer à des tâches essentielles. Voir section 6.1.1.1.

- **Robustesse**

  Les possibilités de redondance, de transmission et de partage des données recueillies pendant la mission sont facilitées par la structure interne des agents ainsi que par l'ensemble des mécanismes présentés en section 6.1.2.5.

- **Réactivité**

Les estimations de délai de prise de décisions présentées en 6.1.1.2 montrent que la première version de l'architecture simulée avec du matériel du commerce permet d'atteindre des capacités acceptables à ce stade des recherches.

- **Durée de vie**

  La durée de vie de la suite de senseurs est un facteur important pris en compte pour la performance du SMS. Les possibilités offertes en termes d'évolution et personnalisation de la suite de senseurs permettent à celle-ci d'évoluer avec l'apparition des prochaines technologies (stockages, communications, puissance de calcul) et du développement de nouveaux senseurs. L'impact d'ajout de senseurs a été étudié en section 6.1.2.1.

Conclusion

Nous avons évalué la nouvelle architecture par projection de chaque besoin présenté au chapitre 4.2 sur l'architecture. Nous avons testé sa modularité en identifiant les modifications à réaliser afin d'ajouter un senseur à une architecture existante. Nous avons ainsi validé l'architecture et l'ordonnancement en mettant en phase les tests et les résultats des simulations avec le besoin. Nous avons vu que l'architecture orientée agent permet une modularité et une autonomie ne pouvant être atteinte avec les architectures actuelles. Nous avons vu que les délais engendrés lors de l'ensemble des calculs de la planification à l'ordonnancement sont compatibles des exigences du contexte au regard du matériel utilisé pour les tests et l'effort d'optimisation à ce stade. Nous avons montré comment l'architecture et l'ordonnancement permettent d'interfacer de nouvelles technologies ou de nouveaux types d'agents sans retravailler l'ensemble de l'architecture.

Nous pouvons noter que l'architecture est composée de 3 éléments principaux :

- **Les agents :**

  Les agents représentent le cœur actif de l'architecture. Ces derniers dotent l'architecture d'une autonomie satisfaisante au regard du contexte. Ils forment avec l'ordonnancement le squelette de l'architecture.

- **L'ordonnancement :**

  L'ordonnancement permet de satisfaire une part importante des exigences, notamment en termes de prédictibilité et de réactivité de l'architecture. Son fonctionnement est indépendant des agents puisqu'il prend en son entrée un ensemble de plans construits par les agents.

- **Un ensemble de mécanismes :**

  Les mécanismes présentés en section 4.2.4.2 (politiques, entrées opérateur, agent de priorité absolue, etc.) prennent une part importante dans le fonctionnement de l'architecture. Elles soutiennent les processus agents et mettent en phase le squelette de l'architecture avec les besoins opérationnels et industriels. Celles-ci constituent les points d'entrées du contrôle de l'opérateur sur le système.

Ensemble, les agents, l'ordonnancement et les mécanismes constituent l'architecture d'une suite de senseurs compatible avec les exigences opérationnelles et industrielles court et moyen terme. La modularité que cette architecture propose permet d'accepter de futures exigences encore non identifiées.

Chapitre 7

Conclusion

L'ensemble des contraintes, exigences, besoins industriels et opérationnels évoluent et imposent le développement de nouveaux systèmes multi-senseur, capables de survivre aux prochaines évolutions du contexte technico-opérationnel. Le développement de nouveaux systèmes complexes confrontés à des environnements dynamiques et évolutifs tel que le système multi-capteur présenté dans ce manuscrit s'étend au-delà des senseurs. Les travaux réalisés et présentés dans ce manuscrit se concentrent sur la clé de voute et la colonne vertébrale du système, situées au centre de chacun des senseurs. L'architecture proposée constitue la colonne vertébrale du SMS tandis que l'ordonnancement sa clé de voute.

7.1 Une première architecture à base d'agents pour les SMS

L'architecture proposée dans ce manuscrit suit un paradigme différent des approches naturelles consistant à considérer chaque senseur du système de senseurs comme une entité autonome possédant ses propres objectifs. Dans l'architecture proposée, les entités autonomes incarnent les objets du terrain virtualisés après détection. Les senseurs et autres ressources matérielles ou virtuelles sont mis au rang de ressources, et sont négociés par un mécanisme complet de gestion de plans et d'ordonnancement inscrit en profondeur dans le squelette de l'architecture.

Les entités autonomes, des agents logiciels au design spécifique, constituent ensemble un système multi-agent purement communicant, traduisant en chaque instant, et en toute autonomie, la volonté d'un opérateur d'exécuter des actions senseurs en fonction du contexte : la situation de la plateforme au regard du terrain et des évènements qui s'y produisant. Les agents disposent d'un grand nombre de données afin de déterminer quelles sont les actions les plus adaptées afin de satisfaire les attentes de l'opérateur conformément au contexte. À partir de ces données, des objectifs d'un rang élevé, appelés ici objectifs opérationnels, sont déduits en fonction des données relatives à l'objet que l'agent incarne. La faculté de prise de décision de niveaux opérationnels et fonctionnels des agents explique le terme choisi d'*Agent Tactique* pour ce type d'agent.

Avec les senseurs, les agents forment une boucle de rétroaction dans laquelle chacune des actions senseurs décidées par les agents permet d'alimenter leurs futures décisions. Chaque décision prise par les agents, se traduit en plan senseurs, qui, une fois exécuté créé de la donné. Cette donnée est ensuite redirigée vers l'agent tactique concerné par la donnée afin d'alimenter ses algorithmes de décision pour construire de nouveaux plans senseurs, etc.

Un ensemble de mécanismes permet de contrôler le système multi-senseur, d'y ancrer les contraintes de l'opérateur et de le colorer de sa philosophie. Ainsi, la priorité des actions senseurs, point de départ du mécanisme d'ordonnancement, est déterminée selon des règles opérationnelles strictes, en phase avec le contexte opérationnel dans lequel un système multi-senseur et sa plateforme sont amenés à opérer.

L'autorité de l'opérateur sur le système, malgré la caractéristique autonome de ce dernier, est primordiale pour l'acceptation d'un tel système multi-senseur. Les décisions prises au cours des différentes missions mènent l'opérateur à diriger plusieurs plateformes et leurs systèmes embarqués en plus d'autres entités opérationnelles (hommes, véhicules, etc.) sur un même théâtre. La criticité des missions et la prépondérance des systèmes de senseurs embarqués à bord de plateformes au fort potentiel (vitesse, altitude, rayon d'action) écartent la possibilité d'autorité de la machine sur l'homme et la prise de décisions non pré-identifiées par l'opérateur. Afin de respecter l'ensemble de ces contraintes, des mécanismes à base de règles opérationnelles ont été construits et intégrés à l'architecture. Ces mécanismes permettent aussi d'enrichir les systèmes des connaissances opérationnelles acquises au cours des expériences passées.

Ces mécanismes sont des mécanismes d'inférence simples basés sur les données. Ces mécanismes, un ensemble d'algorithmes, prennent place au sein des agents et sont appelés lors des phases de réflexion des agents tactiques. Ces algorithmes utilisent en entrée des ontologies, définissant les objets du théâtre pouvant être rencontrés durant une opération. Ces ontologies définissent avec précision les objectifs opérationnels à appliquer en fonction des connaissances acquises de l'objet et les actions senseurs à appliquer pour les atteindre. Les actions senseurs à appliquer à l'objet dépendent du type d'objet et de son comportement.

Les objets rencontrés en opération pouvant être très divers, un mécanisme de définition et spécification des agents tactiques a été construit : l'agent tactique est générique à sa création et se spécifie en fonction des données acquises le concernant. Le type de l'agent tactique après spécification dépend donc des données perçues de l'objet, en fonction de la partialité des données senseurs le concernant. Le type de l'agent tactique permet par la suite de disposer des actions senseurs relatives à la nature de l'objet, en héritant l'ensemble des plans senseurs des types parents.

Une part importante des recherches permit de définir des structures de plans senseurs, définition des objets et des objectifs de manière flexibles, afin de réduire le taux de reprise de l'existant en cas de modifications de l'architecture ou simplement de l'ajout de définitions d'objets et de plans dans le système de senseurs. Considérer les senseurs comme les ressources du système de senseurs permet d'accroître sa modularité. De paire avec la définition flexible des plans et des objets l'ajout de senseurs peut être réalisée par module, un module embarquant à la fois les données d'interprétations, les plans et les cadres d'utilisation du senseur.

Une architecture multi-senseur telle que celle décrite dans ce manuscrit n'a jamais été réalisée ou n'a du moins pas été rendue publique. Les agents tactiques s'inspirent d'agentifications vues dans des travaux de recherches antérieurs selon lesquelles les agents représentent des objets du monde réel et réalisent des actions de contrôle afin de mesurer certains paramètres et d'alerter/informer de la situation. Le système multi-agent décrit dans ce manuscrit met en place une agentification par laquelle les agents représentent aussi des objets de l'environnement, effectuent aussi des actions

de contrôle et d'alerte et ont un rôle supplémentaire : la création de plans senseurs permettant d'enrichir leurs connaissances.

7.2 Un ordonnancement par heuristique rapide et en phase avec les besoins opérationnels

Les exigences en termes de réactivité et de robustesse du système multi-senseur engendrées par le contexte hautement critique dans lequel la plateforme est amenée à opérer ont mené au développement d'un ordonnanceur spécifique.

L'ordonnanceur décrit dans ce manuscrit accepte des contraintes temporelles strictes et complexes. Le contexte peut s'apparenter à un ordonnancement du type job-shop à plusieurs machines, avec des travaux ordonnés, l'exécution de tâches en parallèle sur plusieurs machines, l'impossibilité de retards entre les travaux et la possibilité de précédences de travaux (tâches) et d'ensemble de travaux (plans de tâches).

Le caractère hautement dynamique du contexte peut amener l'ensemble de l'ordonnancement passé calculé à être invalidé lors de l'apparition de certains évènements, plus prioritaires. L'ordonnancement doit donc être un processus rapide et respectant des critères opérationnels définis.

L'ordonnancement est un aspect critique d'un système mettant plusieurs senseurs au fonctionnement précisément temporisé. L'ordonnancement prend en son entrée un ensemble de plans provenant des agents. Ces agents pouvant être nombreux, le nombre de plans ainsi créés et le temps d'occupation des ressources qu'ils requièrent peuvent dépasser la limite acceptable de l'ordonnanceur. Ce dernier se doit d'ordonnancer les plus prioritaires et de reporter ou annuler les autres. La récupération des échecs d'ordonnancement et des refus se fait par les agents, en amont de l'ordonnanceur, leur autonomie et pro-activité permet l'analyse du contexte senseur et l'ajustement de leurs requêtes.

Contrairement aux travaux réalisés sur la gestion de senseurs par un système multi-agent au sein duquel la négociation des ressources permettait d'allouer à un agent du temps senseur, les travaux de ce manuscrit mettent en place une heuristique d'ordonnancement basée sur la priorité représentative des objets réels selon des règles opérationnelles. Le fonctionnement de l'ordonnanceur est à la fois conforme au contexte opérationnel (réactivité, gestion de la priorité, etc.) et adapté au système multi-agent émettant les plans senseurs.

L'ordonnancement présenté dans ce manuscrit est basé sur les évènements, une méthode efficace et rapide de placement des tâches et de vérification des contraintes. L'heuristique ici décrite met en place un ordonnancement à base d'évènements adapté aux contraintes strictes et complexes exigées par le contexte.

Les différents mécanismes et heuristiques composant l'ordonnanceur permettent de conserver la flexibilité et la modularité de l'architecture (définition des plans, gestion des contraintes et ordonnanceur compartimenté du SMA) tout en exprimant des caractéristiques de réactivité et de fiabilité accrues tout au long de la chaîne décisionnelle (ordonnancement selon la priorité, ordonnancement rapide et lancement de plusieurs processus d'ordonnancement parallèles).

7.3 Perspectives

Les travaux présentés dans ce manuscrit offrent de nombreuses possibilités, soit essentielles pour l'approfondissement des recherches et l'adoption d'une telle architecture, soit auxiliaires, pour en enrichir certains aspects et l'adapter à d'autres contextes.

7.3.1 Perspectives principales

Les perspectives principales concernent de nombreux points à approfondir de l'architecture pouvant être améliorés au travers de travaux permettant l'amélioration de son fonctionnement.

- **Enrichissement des données exploitées par le SMS**
  L'architecture SMS proposée dans ce manuscrit est alimentée par un ensemble de données essentielles à son fonctionnement. Compléter ces données permettra au SMS de présenter des décisions plus fines, en adéquation avec les évènements observés par le biais des senseurs.
- **Développement d'une architecture physique**
  Suite à la création de l'architecture logique présentée dans ce manuscrit, sa projection sur une architecture physique permettra d'approfondir certains aspects du fonctionnement du SMS. Les communications, les protocoles, les ressources de calculs et l'interfaçage avec les senseurs seront entre autres des axes de développement important.
- **Maturation et démonstration sur maquette physique**
  Les phases de développement de tout système simulé passent successivement par des tests et des évaluations en conditions réelles ou s'en approchant. Les évaluations du type SWIL (SMA logiciel/Senseurs matériels) et MITL seront essentielles aux suites de développement du SMS.
- **Recherches associées**
  Un grand nombre de recherches associées aux senseurs ou s'en approchant permettront d'améliorer le fonctionnement global du SMS. Par exemple, la reconnaissance automatique d'objets sur des images SAR ou optique permettrait de disposer de données senseurs transformées à bord de la plateforme. Une amélioration des senseurs, une réduction des bruits de mesures ainsi que les recherches concernant la fusion de piste amélioreront les bases de fonctionnement des agents.

7.3.2 Perspectives auxiliaires

- **Création d'outils de spécification du SMS**
  Les données de fonctionnement du SMS sont exprimées au sein de fichiers facilement éditables. Cependant, le paramétrage et l'enrichissement du système par des expérimentateurs/opérationnels via des outils d'édition graphique des bases de données de plans, des ontologies et des procédures agents permettrait d'enrichir le SMS.
- **Extension du SMA**
  Les agents tactiques permettent de compartimenter les décisions prises par chacun d'entre eux. Ainsi, il est aisé d'imaginer des processus spécifiques pour de nouveaux agents tactiques ou d'un autre type.

Annexe A

Exemple de senseurs du SMS type étudié

TABLE A.1 – La collection de senseurs et leurs domaines respectifs.

Senseur	**Abréviation**	**Type**	**Domaine physique**	**Position sur la plateforme**	**Domaine angulaire complet**	**Domaine distance**
Radar à antenne active droit	RADR	Actif	EM-RF	Normale antenne direction gisement +60°	Gisement : ±60° Site : ±60°	≈ 150km*
Radar à antenne active gauche	RADL	Actif	EM-RF	Normale antenne direction gisement -60°	Gisement : ±60° Site : ±60°	≈ 150km*
Optronique	OPT	Passif	EM-Optique	Boule optronique Sous la plateforme	Gisement : ±60° Site : ±60°	< 100km†
Antennes omnidirectionnelles Récepteur superhétérodyne***	ESM	Passif	EM-RF	Plusieurs antennes réparties	Gisement : ±360° site : ±180°	< 200km‡
Détecteur d'Alerte Laser	DAL	Passif	EM-Laser	3 détecteurs répartis tous les 120°	Gisement : ±60° Site : ±60°	<100m**
Détecteur d'Alerte Infrarouge	DIR	Passif	EM-IR	2 détecteurs droite/ gauche	Gisement : ±60° Site : ±60°	<100km**
Brouilleurs	JAM	Actif	EM-RF	3 émetteurs répartis tous les 120°	Gisement : ±360° Site : ±180°	Portée < 50km**
Leurres	DEC	Actif	Thermiques Paillettes EM	Cartouches pyrotechniques à l'arrière de la plateforme	-	Portée ≈5km**

* Dépend de la SER de l'objet cible.
† Dépend des caractéristiques physiques de l'objet et du contexte (exemple : météo).
‡ Dépend de la puissance d'émission du signal cible.
** Dépend de la menace.
*** Les antennes omnidirectionnelles et les récepteurs superhétérodynes seront regroupés sous le même terme, ESM (*Electronic Support Measures*, mesures de soutien électronique), dans le reste du document.

L'ensemble des domaines d'utilisation des senseurs sont des valeurs types provenant de la littérature ouverte et n'ont pas pour objectif de refléter les performances réelles de senseurs précis.

Annexe B

Exemple de fichier de description de plans

Exemple de plans provenant de la base de données d'exemple de plans, le fichier *plans.xml*.

L'ensemble des domaines d'utilisation des senseurs sont des valeurs types provenant de la littérature ouverte et n'ont pas pour objectif de refléter les performances réelles de senseurs précis.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- EXAMPLES PLANS DATABASE -->

<!-- TEMPLATE -->
<!-- <plan constraintType=""> -->
<!-- <task resource="" duration="" /> -->
<!-- <constraint start="" end="" /> -->

<!-- </plan> -->
<!-- <variable name="" value="" /> -->
<!-- variables should -->
<!-- be separated by a pipe followed by a sign -->

<!-- time unit: 100 = 1 second -->

<!-- constraints: constraintType="1" designs that the start and end constraint
  are applied at the previous task end and 0 for beginning of previous task. -->

<plans>

<!-- Plans for the two antennas are separate since they are different
    and independant sensors-->
  <plan type="SAR_SPOT_L" minDuration="spotMinDuration">

    <task resource="ANT_L" duration="spotMinDuration" fixedDuration="no" />
    <task resource="POW" duration="spotMinDuration" fixedDuration="no" />
    <task resource="HF" duration="spotMinDuration" fixedDuration="no" />
    <task resource="SAR_IMG" duration="spotMinDuration" fixedDuration="no" />

    <constraint start="0" end="0" constraintType="1" />
    <constraint start="0" end="0" constraintType="0" />
    <constraint start="0" end="0" constraintType="0" />
    <constraint start="0" end="2000" constraintType="1" />

    <!-- AND logic means all constraints should be met -->
    <requirements logic="AND">
```

```
    <!-- DISTANCE in m (platform-object distance range) -->
    <requirement name="A" type="DISTANCE" range="INTERVAL">
      <reqvalue value="0" />
      <reqvalue value="150000" />
    </requirement>
    <!-- ALPHA for bearing relative to sensor's heading in rad -->
    <requirement name="B" type="ALPHA" range="INTERVAL">
      <reqvalue value="-0.174" />
      <reqvalue value="-1.919862" />
    </requirement>
    <!-- BETA for elevation relative to sensor's pitch in rad -->
    <requirement name="C" type="BETA" range="INTERVAL">
      <reqvalue value="-1.396263" />
      <reqvalue value="1.396263" />
    </requirement>
  </requirements>

</plan>

<plan type="SAR_STRIP_L" minDuration="spotMinDuration">

  <task resource="ANT_L" duration="stripMinDuration" fixedDuration="no" />
  <task resource="POW" duration="stripMinDuration" fixedDuration="no" />
  <task resource="HF" duration="stripMinDuration" fixedDuration="no" />
  <task resource="SAR_IMG" duration="stripMinDuration" fixedDuration="no" />

  <constraint start="0" end="0" constraintType="1" />
  <constraint start="0" end="0" constraintType="0" />
  <constraint start="0" end="0" constraintType="0" />
  <constraint start="delayStripProcessingMin" end="delayStripProcessingMax"
    constraintType="0" />


  <requirements logic="AND">
    <requirement name="A" type="DISTANCE" range="INTERVAL">
      <reqvalue value="0" />
      <reqvalue value="150000" />
    </requirement>
    <requirement name="B" type="ALPHA" range="INTERVAL">
      <reqvalue value="-0.174" />
      <reqvalue value="-1.48353" />
    </requirement>
    <requirement name="C" type="BETA" range="INTERVAL">
      <reqvalue value="-1.396263" />
      <reqvalue value="1.396263" />
    </requirement>
  </requirements>
</plan>


<!-- the pipe means an operation should be applied on the value
 This is used for relative delays construction -->
<plan type="DECOY" minDuration="20|+decoyDelay|+20">

  <task resource="DECOY" duration="20" />
  <task resource="DECOY" duration="20" />

  <constraint start="0" end="0" constraintType="1" />
  <constraint start="decoyDelay" end="decoyDelay"
    constraintType="1" />

  <requirements logic="AND">
```

```
      <requirement name="A" type="DISTANCE" range="INTERVAL">
        <reqvalue value="0" />
        <reqvalue value="400000" />
      </requirement>
    </requirements>
  </plan>

  <plan type="RDR_SPOT_L" minDuration="rdrSpotDuration">

    <task resource="ANT_L" duration="rdrSpotDuration" />

    <constraint start="0" end="0" constraintType="1" />

    <requirements logic="AND">
      <requirement name="A" type="DISTANCE" range="INTERVAL">
        <reqvalue value="0" />
        <reqvalue value="200000" />
      </requirement>
      <requirement name="B" type="ALPHA" range="INTERVAL">
        <reqvalue value="-0.0" />
        <reqvalue value="-2.007129" />
      </requirement>
      <requirement name="C" type="BETA" range="INTERVAL">
        <reqvalue value="-1.570796" />
        <reqvalue value="1.570796" />
      </requirement>
    </requirements>
  </plan>


<!-- 100 = 1 second -->

  <variable name="IRWatchDelay" value="500" />
  <variable name="optroWatchDuration" value="500" />
  <variable name="rdrWatchDuration" value="180" />
  <variable name="spotMinDuration" value="1000"/>
  <variable name="stripMinDuration" value="1000"/>
  <variable name="delayStripProcessingMin" value="500" />
  <variable name="delayStripProcessingMax" value="3000" />
  <variable name="rdrSpotDuration" value="10" />
  <variable name="decoyDelay" value="10" />

</plans>
```

Liste des tableaux

Table des figures

Liste des publications

[1] Grivault, Ludovic, Amal El Fallah-Seghrouchni et Raphaël Girard-Claudon: *Agent-Based Architecture for Multi-sensors System Deployed on Airborne Platform*. Dans *2016 IEEE International Conference on Agents (ICA)*, pages 86–89, septembre 2016. DOI :10.1109/ICA.2016.028.

[2] Grivault, Ludovic, Amal El Fallah-Seghrouchni et Raphaël Girard-Claudon: *Coordination Of Sensors Deployed On Airborne Platform : A Scheduling Approach*. Dans *EUMAS-AT2016*, Valencia, Spain, décembre 2016. DOI :10.1007/978-3-319-59294-7.

[3] Grivault, Ludovic, Amal El Fallah-Seghrouchni et Raphaël Girard-Claudon: *Coordination Of Sensors Deployed On Airborne Platform : A Scheduling Approach*. Dans *Multi-Agent Systems and Agreement Technologies*, tome LNCS/LNAI. Springer, juillet 2017, ISBN 978-3-319-59293-0. https://hal.archives-ouvertes.fr/hal-01528754.

[4] Grivault, Ludovic, Amal El Fallah-Seghrouchni et Raphaël Girard-Claudon: *Multi-Agent System to Design Next Generation of Airborne Platform*. Dans *11th International Symposium on Intelligent Distributed Computing*, Belgrade, Serbia, octobre 2017. https://hal.archives-ouvertes.fr/hal-01630999.

[5] Grivault, Ludovic, Amal El Fallah-Seghrouchni et Raphaël Girard-Claudon: *Next Generation of Airborne Platforms : From Architecture Design to Sensors Scheduling*. Dans *IEEE ICA-2017*, Beijing, China, juillet 2017. https://hal.archives-ouvertes.fr/hal-01539119v1.

Bibliographie

[Arlabosse et al., 2004] Arlabosse, F., Gleizes, M.-P., and Occello, M. (2004). Méthodes de Conception de Systèmes Multi-agents. In *Systèmes Multi-Agents - Collection ARAGO - N° 29*, page 32p. Tec&doc.

[Artigues et al., 2007] Artigues, C., Demassey, S., and Neron, E. (2007). *Resource-Constrained Project Scheduling : Models, Algorithms, Extensions and Applications*. ISTE.

[Artigues et al., 2013] Artigues, C., Demassey, S., and Neron, E. (2013). *Resource-constrained project scheduling : models, algorithms, extensions and applications*. John Wiley & Sons.

[Błażewicz, 1986] Błażewicz, J. (1986). *Scheduling under resource constraints : Deterministic models*, volume 7. JC Baltzer.

[Bajcsy, 1988] Bajcsy, R. (1988). Active perception.

[Ballot et al., 2013] Ballot, G., Kant, J.-D., and Goudet, O. (2013). Modeling both sides of the French labor market with adaptive agents under bounded rationality *. In *The 25th Annual Conference of the EAEPE (European Association for Evolutionary Political Economy)*, Bobigny, France. EAEPE (European Association for Evolutionary Political Economy).

[Bellifemine et al., 2007] Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. Number p. 3.

[Bellifemine et al., 2010] Bellifemine, F. L., Caire, G., and Greenwood, D. (2010). *Jade Programmer's Guide*.

[Benfield et al., 2006] Benfield, S. S., Hendrickson, J., and Galanti, D. (2006). Making a strong business case for multiagent technology. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 10–15. ACM.

[Bezouwen et al., 2010] Bezouwen, H. v., Feldle, H. P., and Holpp, W. (2010). Status and trends in AESA-based radar. In *2010 IEEE MTT-S International Microwave Symposium*.

[Bishop et al., 1994] Bishop, D., Sturm, P., and Ybarra, K. (1994). *Method of track merging in an aircraft tracking system*. Google Patents.

[Blanchard, 2004] Blanchard, Y. (2004). *Le Radar, 1904-2004 : Histoire d'un siècle d'innovations techniques et opérationnelles*. Collection scientifique et technique Thales. Ellipses.

[Blazewicz et al., 1983] Blazewicz, J., Lenstra, J. K., and Kan, A. R. (1983). Scheduling subject to resource constraints : classification and complexity. *Discrete Applied Mathematics*, 5(1) :11–24.

[Bürkle et al., 2011] Bürkle, A., Segor, F., and Kollmann, M. (2011). Towards autonomous micro uav swarms. *Journal of intelligent & robotic systems*, 61(1-4) :339–353.

[Brown and Porcello, 1969] Brown, W. M. and Porcello, L. J. (1969). An introduction to synthetic-aperture radar. *IEEE spectrum*, 6(9) :52–62.

[Callantine, 2002] Callantine, T. J. (2002). CATS-based Air Traffic Controller Agents. *San Jose State University*.

[Cavalcante et al., 2012] Cavalcante, R. C., Bittencourt, I. I., da Silva, A. P., Silva, M., Costa, E., and Santos, R. (2012). A survey of security in multi-agent systems. *Expert Systems with Applications*, 39(5) :4835–4846.

[Center for History and New Media,] Center for History and New Media. Guide rapide pour débuter.

[Chabod and Chamouard, 2009] Chabod, L. and Chamouard, E. (2009). Low cost moving target tracking and fire control. *International Radar Conference*.

[Chabod and Galaup, 2014] Chabod, L. and Galaup, P. (2014). Shared Resources for Airborne Multifunction Sensor Systems. *IET International Conference on Radar Systems*.

[Chabod et al., 2014] Chabod, L., Girard-Claudon, R., and Segura, E. (2014). Taking benefit of Radar resources in integrated RF. *2014 International Radar Conference*.

[Cohen and Levesque, 1990] Cohen, P. R. and Levesque, H. J. (1990). Intention is choice with commitment. *Artificial intelligence*, 42(2-3) :213–261.

[Cong and Liang, 2009] Cong, S. and Liang, Y. (2009). PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems. *IEEE Transactions on Industrial Electronics*, 56(10) :3872–3879.

[Coppin and Marchalot, 2009] Coppin, G. and Marchalot, G. (2009). *Systèmes multi-agents pour l'identification de cibles par un système multicapteur*.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.

[Cornilleau et al., 2014] Cornilleau, T., Linard, P., Moxon, P., and Nicholas, C. (2014). ECOA - A New Architecture Concept for Complex Military Software Systems. *SAE Int. J. Aerosp.*, 7 :214–221.

[Davies, 2012] Davies, G. J. (2012). *Agent based impact analysis of air traffic management systems*. phdthesis, Queen's University Belfast.

[Demange et al., 2011] Demange, J., Galland, S., and Koukam, A. (2011). Towards the agentification of a virtual situated environment for urban crowd simulation. In *IET International Conference on Smart and Sustainable City (ICSSC 2011)*, pages 1–5.

[Endsley, 1995] Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human factors*, 37(1) :32–64.

[Eryigit and Uyar, 2008] Eryigit, C. and Uyar, S. (2008). Integrating agents into data-centric naval combat management systems. In *2008 23rd International Symposium on Computer and Information Sciences*, pages 1–4.

[Evertsz et al., 2004] Evertsz, R., Fletcher, M., Jones, R., Jarvis, J., Brusey, J., and Dance, S. (2004). Implementing Industrial Multi-agent Systems Using JACKTM. In Dastani, M. M., Dix, J., and El Fallah-Seghrouchni, A., editors, *Programming Multi-Agent Systems : First International Workshop, PROMAS 2003, Melbourne, Australia, July 15, 2003, Selected Revised and Invited papers*, pages 18–48. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Fagin et al., 2004] Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. (2004). *Reasoning about knowledge*. MIT press.

[Ferber, 1995] Ferber, J. (1995). *Les Systèmes Multi Agents : vers une intelligence collective*.

[Ferber, 1999] Ferber, J. (1999). *Multi-agent systems : an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

[Ferber and Gutknecht, 1998] Ferber, J. and Gutknecht, A. (1998). Un méta-modèle organisationnel pour l'analyse, la conception et l'execution de systèmes multi-agents. In *Proceedings of Third International Conference on Multi-Agent Systems ICMAS*, volume 98, pages 128–135.

[Fleszar and Hindi, 2004] Fleszar, K. and Hindi, K. S. (2004). Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, 155(2) :402–413.

[Franklin and Graesser, 1997] Franklin, S. and Graesser, A. (1997). Is It an agent, or just a program ? : A taxonomy for autonomous agents. In Müller, J. P., Wooldridge, M. J., and Jennings, N. R., editors, *Intelligent Agents III Agent Theories, Architectures, and Languages : ECAI'96 Workshop (ATAL) Budapest, Hungary, August 12–13, 1996 Proceedings*, pages 21–35. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Garey and Johnson, 1975] Garey, M. R. and Johnson, D. S. (1975). Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4) :397–411.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). Computers and Intractability : A Guide to the Theory of NP-Completeness.

[Garey et al., 1976] Garey, M. R., Johnson, D. S., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2) :117–129.

[Gascueña and Fernández-Caballero, 2011] Gascueña, J. M. and Fernández-Caballero, A. (2011). Agent-oriented modeling and development of a person-following mobile robot. *Expert Systems with Applications*, 38(4) :4280 – 4290.

[Gaud, 2007] Gaud, N. A. (2007). *Systèmes multi–agents holoniques : de l'analyse à l'implantation : méta-modèle, méthodologie, et simulation multi-niveaux*. phdthesis.

[Georgeff and Rao, 1998] Georgeff, M. and Rao, A. (1998). Rational Software Agents : From Theory to Practice. In Jennings, N. R. and Wooldridge, M. J., editors, *Agent Technology : Foundations, Applications, and Markets*, pages 139–160. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Georgeff and Lansky, 1987] Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. In *AAAI*, volume 87, pages 677–682.

[Gorodetsky et al.,] Gorodetsky, V., Karsaev, O., Samoylov, V., and Skormin, V. Multi-Agent Technology for Air Traffic Control and Incident Management in Airport Airspace. page 8.

[Graham et al., 1979] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of discrete mathematics*, 5 :287–326.

[Grivault et al., 2016a] Grivault, L., El Fallah-Seghrouchni, A., and Girard-Claudon, R. (2016a). Agent-Based Architecture for Multi-sensors System Deployed on Airborne Platform. In *2016 IEEE International Conference on Agents (ICA)*, pages 86–89.

[Grivault et al., 2016b] Grivault, L., El Fallah-Seghrouchni, A., and Girard-Claudon, R. (2016b). Coordination Of Sensors Deployed On Airborne Platform : A Scheduling Approach. In *EUMAS-AT2016*, Valencia, Spain.

[Grivault et al., 2017a] Grivault, L., El Fallah-Seghrouchni, A., and Girard-Claudon, R. (2017a). Coordination Of Sensors Deployed On Airborne Platform : A Scheduling Approach. In *Multi-Agent Systems and Agreement Technologies*, volume LNCS/LNAI. Springer.

[Grivault et al., 2017b] Grivault, L., El Fallah-Seghrouchni, A., and Girard-Claudon, R. (2017b). Multi-Agent System to Design Next Generation of Airborne Platform. In *11th International Symposium on Intelligent Distributed Computing*, Belgrade, Serbia.

[Grivault et al., 2017c] Grivault, L., El Fallah-Seghrouchni, A., and Girard-Claudon, R. (2017c). Next Generation of Airborne Platforms : From Architecture Design to Sensors Scheduling. In *IEEE ICA-2017*, Beijing, China.

[Grosch, 2009] Grosch, T. (2009). *Computational Intelligence in Integrated Airline Scheduling*, volume 173 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg. DOI 10.1007/978-3-540-89887-0.

[Guastella, 2017] Guastella, D. (2017). Rapport de stage de Master 2 de Davide Guastella - Sorbonne Université.

[Gutiérrez et al., 2011] Gutiérrez, C., García-Magariño, I., and Fuentes-Fernández, R. (2011). Detection of undesirable communication patterns in multi-agent systems. *Engineering Applications of Artificial Intelligence*, 24(1) :103–116.

[Gutknecht and Ferber, 2001] Gutknecht, O. and Ferber, J. (2001). The MadKit Agent Platform Architecture. In Wagner, T. and Rana, O. F., editors, *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems : International Workshop on Infrastructure for Scalable Multi-Agent Systems Barcelona, Spain, June 3–7, 2000 Revised Papers*, pages 48–55. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Hall and Llinas, 1997] Hall, D. L. and Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1) :6–23.

[Heinze et al., 1998] Heinze, C., Smith, B., and Cross, M. (1998). Thinking quickly : Agents for modeling air warfare. *Advanced topics in artificial intelligence*, pages 47–58.

[Herroelen et al., 1997] Herroelen, W., Demeulemeester, E., and De Reyck, B. (1997). A classification scheme for project scheduling problems.

[Hoogeboom et al., 2004] Hoogeboom, P., Joosse, M., Hodgetts, Straussberger, S., and Schaefer, D. (2004). Does the 'silent cockpit' reduce pilot workload ? *Digital Avionics Systems Conference*.

[Hovareshti et al., 2007] Hovareshti, P., Gupta, V., and Baras, J. S. (2007). Sensor scheduling using smart sensors. In *Decision and Control, 2007 46th IEEE Conference on*, pages 494–499. IEEE.

[Hughes and Choe, 2000] Hughes, P. K. and Choe, J. Y. (2000). Overview of advanced multifunction RF system (AMRFS). In *Phased Array Systems and Technology, 2000. Proceedings. 2000 IEEE International Conference on*, pages 21–24. IEEE.

[Ibrahim et al., 2013] Ibrahim, Y., Higgins, P., and Bruce, P. (2013). Evaluation of a collision avoidance display to support pilots' mental workload in a free flight environment. *IEEE International Conference on Industrial Engineering and Engineering Management*.

[Ignall and Schrage, 1965] Ignall, E. and Schrage, L. (1965). Application of the branch and bound technique to some flow-shop scheduling problems. *Operations research*, 13(3) :400–412. Citation Key : ignall1965application bibtex[publisher=INFORMS].

[Jennings, 2000] Jennings, N. R. (2000). On agent-based software engineering. *Artificial intelligence*, 117(2) :277–296.

[Jennings et al., 1998] Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1) :7–38.

[Joubert et al., 1995] Joubert, T., Salle, S. E., Champigneux, G., Grau, J. Y., Sassus, P., and Le Doeuff, H. (1995). The Copilote Electronique project : First lessons as exploratory development starts. page 188.

[Juan,] Juan, N. *Une infrastructure pour l'optimisation de systèmes embarqués évolutifs à base de composants logiciels*. PhD thesis.

[Kahn, 1962] Kahn, A. B. (1962). Topological sorting of large networks. *Communications of the ACM*, 5(11) :558–562.

[Kemkemian et al., 2009] Kemkemian, S., Nouvel, M., Cornic, P., Le Bihan, P., and Garrec, P. (2009). Radar systems for sense and avoid on UAV. *International Radar Conference*.

[Kemkemian and Nouvel-Fiani, 2010] Kemkemian, S. and Nouvel-Fiani, M. (2010). Toward common radar & EW multifunction active arrays. In *2010 IEEE International Symposium on Phased Array Systems and Technology*, pages 777–784.

[Koestler, 1969] Koestler, A. (1969). Holons and Holarchy of Arthur Koestler.

[Kolisch and Hartmann, 2006] Kolisch, R. and Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling : An update. *European journal of operational research*, 174(1) :23–37.

[Koné et al., 2011] Koné, O., Artigues, C., Lopez, P., and Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, 38(1) :3–13.

[Konolige, 1986] Konolige, K. (1986). A Deduction Model of Belief.

[Labrou et al., 1999] Labrou, Y., Finin, T., and Peng, Y. (1999). Agent communication languages : The current landscape. *IEEE Intelligent Systems and Their Applications*, 14(2) :45–52.

[Lalanda, 1998] Lalanda, P. (1998). Shared repository pattern. In *Proceedings of the 5th Pattern Languages of Programs Conference (PLoP 1998)*.

[Lalanda et al., 1992a] Lalanda, P., Charpillet, F., and Haton, J. P. (1992a). A Real Time Blackboard Based Architecture. In *ECAI*, pages 262–266.

[Lalanda et al., 1992b] Lalanda, P., Charpillet, F., and Haton, J.-P. (1992b). *Une architecture temps réel à base de tableau noir*. Centre de Recherche en Informatique.

[Lesser et al., 2012] Lesser, V., Ortiz Jr, C. L., and Tambe, M. (2012). *Distributed sensor networks : A multiagent perspective*, volume 9. Springer Science & Business Media.

[Ljungberg and Lucas, 1992] Ljungberg, M. and Lucas, A. (1992). The OASIS air traffic management system. volume 92, Seoul, Korea.

[Längle and Wörn, 2001] Längle, T. and Wörn, H. (2001). Human–Robot Cooperation Using Multi-Agent-Systems. *Journal of Intelligent and Robotic Systems*, 32(2) :143–160.

[Lützenberger et al., 2013] Lützenberger, M., Küster, T., Konnerth, T., Thiele, A., Masuch, N., He\s sler, A., Burkhardt, M., Tonn, J., Kaiser, S., and Keiser, J. (2013). Engineering industrial multi-agent systems—the JIAC V approach. In *Proceedings of the 1st International Workshop on Engineering Multi-Agent Systems (EMAS 2013)*, pages 160–175.

[Lützenberger et al., 2016] Lützenberger, M., Küster, T., Masuch, N., and Fähndrich, J. (2016). Multi-Agent System in Practice : When Research Meets Reality. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 796–805, Singapore, Singapore. International Foundation for Autonomous Agents and Multiagent Systems.

[Luo et al., 2002] Luo, R. C., Yih, C.-C., and Su, K. L. (2002). Multisensor fusion and integration : approaches, applications, and future research directions. *IEEE Sensors Journal*, 2(2) :107–119.

[Maes, 1994] Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM*, 37(7) :30–40.

[Mandiau et al., 2002] Mandiau, R., Grislin-Le Strugeon, E., and Péninou, A. (2002). *Une méthode de modélisation et de conception d'organisations multi-agents holoniques*. Hermès Science Publications, Paris.

[McCarthy, 1979] McCarthy, J. (1979). Ascribing Mental Qualities to Machines. Technical report, DTIC Document.

[McCarthy, 1995] McCarthy, J. (1995). Making Robots Conscious of Their Mental States. In *Machine Intelligence 15*, pages 3–17.

[Merrill, 1990] Merrill, I. S. (1990). *Radar handbook*.

[Merritts, 2013] Merritts, R. A. (2013). Online Deception Detection Using BDI Agents.pdf.

[Mora et al., 1999] Mora, M. C., Lopes, J. G., Viccariz, R. M., and Coelho, H. (1999). BDI Models and Systems : Reducing the Gap. In Müller, J. P., Rao, A. S., and Singh, M. P., editors, *Intelligent Agents V : Agents Theories, Architectures, and Languages : 5th International Workshop, ATAL'98 Paris, France, July 4–7, 1998 Proceedings*, pages 11–27. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Navarro et al., 2011] Navarro, L., Flacher, F., and Corruble, V. (2011). Dynamic Level of Detail for Large Scale Agent-Based Urban Simulations. page 8, Taipei, Taiwan.

[Nguyen-Duc et al., 2008] Nguyen-Duc, M., Guessoum, Z., Marin, O., Perrot, J.-F., and Briot, J.-P. (2008). A multi-agent approach to reliable air traffic control. In *2nd International Symposium on Agent Based Modeling and Simulation*, Vienna, Austria.

[Omicini et al., 2006] Omicini, A., Ricci, A., and Viroli, M. (2006). Agens Faber : Toward a Theory of Artefacts for MAS. *Electr. Notes Theor. Comput. Sci.*, 150(3) :21–36.

[Omicini et al., 2008] Omicini, A., Ricci, A., and Viroli, M. (2008). Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3) :432–456.

[Pappas et al., 1996] Pappas, G. J., Tomlin, C., and Sastry, S. S. (1996). Conflict resolution for multi-agent hybrid systems. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 2, pages 1184–1189 vol.2.

[Park and Sugumaran, 2005] Park, S. and Sugumaran, V. (2005). Designing multi-agent systems : a framework and application. *Expert Systems with Applications*, 28(2) :259–271.

[Pavón et al., 2007] Pavón, J., Gómez-Sanz, J., Fernández-Caballero, A., and Valencia-Jiménez, J. J. (2007). Development of intelligent multisensor surveillance systems with agents. *Robotics and Autonomous Systems*, 55(12) :892–903.

[Picard, 2004] Picard, G. (2004). *Methodology for developing adaptive multi-agent systems and designing software with emergent functionality*. Theses, Université Paul Sabatier Toulouse III.

[Pinedo, 2016] Pinedo, M. L. (2016). *Scheduling : theory, algorithms, and systems*. Springer.

[Pipattanasomporn et al., 2009] Pipattanasomporn, M., Feroze, H., and Rahman, S. (2009). Multi-agent systems in a distributed smart grid : Design and implementation. In *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pages 1–8. IEEE.

[Poslad, 2009] Poslad, S. (2009). Using Multi-agent Systems to Specify Safe and Secure Services for Virtual Organisations. In Barley, M., Mouratidis, H., Unruh, A., Spears, D., Scerri, P., and Massacci, F., editors, *Safety and Security in Multiagent Systems*, volume 4324, pages 258–273. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Potiron et al., 2009] Potiron, K., Taillibert, P., and El Fallah-Seghrouchni, A. (2009). Autonomous Agents : When the Mailbox Remains Empty. pages 293–296. IEEE.

[Rao, 1996] Rao, A. (1996). AgentSpeak (L) : BDI agents speak out in a logical computable language. *Agents breaking away*, pages 42–55.

[Rao and Georgeff, 1995] Rao, A. S. and Georgeff, M. P. (1995). BDI Agents : From Theory to Practice. In *In proceedings of the first international conference on multi-agent systems - ICMAS-95*, pages 312–319.

[Rao and Murray, 1994] Rao, A. S. and Murray, G. (1994). Multi-agent mental-state recognition and its application to air-combat modelling. *Proc. Work. Distributed AI*.

[Reynaud and Corruble, 2013] Reynaud, Q. and Corruble, V. (2013). Un mécanisme de composition de comportements pour agents virtuels. page 11.

[Richard R. Brooks, 2008] Richard R. Brooks (2008). Book review : Distributed Sensor Networks : A Multiagent Perspective, Edited by V. Lesser, C. L. Ortiz, Jr., and M. Tambe (2003).

[Roberts, 1988] Roberts, L. (1988). The Arpanet and computer networks. In *A history of personal workstations*, pages 141–172. ACM.

[Rodriguez et al., 2005] Rodriguez, S., Hilaire, V., and Koukam, A. (2005). Holonic modeling of environments for situated multi-agent systems. In *International Workshop on Environments for Multi-Agent Systems*, pages 18–31. Springer.

[Schulte et al., 2015] Schulte, A., Donath, D., and Honecker, F. (2015). Human-system interaction analysis for military pilot activity and mental workload determination. *IEEE International Conference on Systems, Man, and Cybernetics*.

[Shen et al., 2006] Shen, W., Wang, L., and Hao, Q. (2006). Agent-based distributed manufacturing process planning and scheduling : a state-of-the-art survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 36(4) :563–577.

[Skolnik, 1962] Skolnik, M. I. (1962). Introduction to radar. *Radar Handbook*, 2.

[Smith, 1980] Smith, R. G. (1980). The Contract Net Protocol : High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, C-29(12) :1104–1113.

[Soh and Tsatsoulis, 2001] Soh, L.-K. and Tsatsoulis, C. (2001). Reflective negotiating agents for real-time multisensor target tracking. In *International Joint Conference On Artificial Intelligence*, volume 17, pages 1121–1127. LAWRENCE ERLBAUM ASSOCIATES LTD.

[Taylor and Reising, 1995] Taylor, R. M. and Reising, J. (1995). The Human-Electronic Crew : Can We Trust the Team ? Technical report, DTIC Document.

[Teutsch and Krüger, 2012] Teutsch, M. and Krüger, W. (2012). Detection, Segmentation, and Tracking of Moving Objects in UAV Videos. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*, pages 313–318.

[Thales - Systèmes Aéroportés,] Thales - Systèmes Aéroportés. Radar for the Rafale, the omnirole fighter AESA. Fiche descriptive produit. www.thalesgroup.com /[...]/aesa_rbe2_5_juin_val_def_bat_ok.pdf.

[Thao Le et al., 2009] Thao Le, Timothy J. Norman, and Wamberto Vasconcelos (2009). Agent-based Sensor-Mission Assignment for Tasks Sharing Assets. *IFAAMA*.

[Tumer and Agogino, 2007] Tumer, K. and Agogino, A. (2007). Distributed Agent-Based Air Traffic Flow Management. *The Sixth Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems - AAMAS*.

[Uetz, 2001] Uetz, M. (2001). *Algorithms for deterministic and stochastic scheduling*. Cuvillier.

[Voirin, 2013] Voirin, J.-L. (2013). Modelling languages for Functional Analysis put to the test of real life. In *Complex Systems Design & Management*, pages 139–150. Springer.

[Voirin, 2017] Voirin, J.-L. (2017). *Model-based System and Architecture Engineering with the Arcadia Method*. Elsevier.

[Weglarz, 2012] Weglarz, J. (2012). *Project scheduling : recent models, algorithms and applications*, volume 14. Springer Science & Business Media.

[Winikoff, 2005] Winikoff, M. (2005). Jack™ Intelligent Agents : An Industrial Strength Platform. In Bordini, R. H., Dastani, M., Dix, J., and El Fallah-Seghrouchni, A., editors, *Multi-Agent Programming : Languages, Platforms and Applications*, pages 175–193. Springer US, Boston, MA.

[Winter, 2008] Winter, E. (2008). *Des outils d'optimisation combinatoire et d'ordonnancement pour la gestion des radars embarqués*. phdthesis, Palaiseau, Ecole polytechnique.

[Wooldridge, 1997] Wooldridge, M. (1997). Agent-based software engineering. *IEE Proceedings-software*, 144(1) :26–37.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. R. (1995). Agent theories, architectures, and languages : A survey. In Wooldridge, M. J. and Jennings, N. R., editors, *Intelligent Agents : ECAI-94 Workshop on Agent Theories, Architectures, and Languages Amsterdam, The Netherlands August 8–9, 1994 Proceedings*, pages 1–39. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Wooldridge, 1992] Wooldridge, M. J. (1992). *Intelligent Agents V. Agents Theories, Architectures, and Languages*. PhD thesis, Citeseer.