Conception et modélisation de systèmes de systèmes : une approche multi-agents multi-niveaux

Jean Baptiste Soyez

Thèse de doctorat de l'Université Lille 1

École Polytechnique Universitaire de Lille

École Doctorale Sciences pour l'Ingénieur

Laboratoire d'Automatique Génie Informatique et Signal

Spécialité : Automatique et Génie Informatique

présentée et soutenue par

Jean-Baptiste Soyez

intitulée

Conception et modélisation de systèmes de systèmes : une approche multi-agents multi-niveaux

Soutenue le 03 Décembre 2013

Avec pour jury :

Rapporteurs :	
Ali Charara	Professeur, Université de Technologie de Compiègne, Heudiasyc
François Charpillet	Directeur de recherche à INRIA-Lorraine
Examinateurs :	
Olivier Simonin	Professeur, INSA de Lyon
Dominique Luzeaux	Ingénieur général de l'armement, HDR, Ministère de la Défense
Fabien Michel	Maître de conférences, Université de Montpellier 2
Directeur et co-directeur :	
Rochdi Merzouki	Professeur, Université de Lille 1
Daniel Dupont	Enseignant-chercheur, HDR, Université Catholique de Lille, HEI
Co-encadrant :	
Gildas Morvan	Maître de conférences, Université d'Artois

Université Lille1 Sciences et Technologies

InTraDE Intelligent Transportation for Dynamic Environment

Remerciements

Pour leur soutien, leur bienveillance et leurs conseils je remercie mes trois encadrants de thèse : Gildas Morvan - maître de conférences au Laboratoire de Génie Informatique et d'Automatique de l'Artois (LGI2A, Béthune), Rochdi Merzouki - professeur des universités à l'École Polytechnique Universitaire de Lille (Université de Lille1) et chercheur au Laboratoire d'Automatique et Génie Informatique et Signal (LAGIS, Lille) et Daniel Dupont - enseignant-chercheur (HDR) à l'école des Hautes Études d'Ingénieur (HEI, Lille). Ils ont, chacun à leur manière, contribué à l'achèvement de ces travaux qui constituent pour moi un enrichissement personnel qui va au delà du simple cadre de la recherche scientifique.

Je tiens à exprimer mes profonds respects aux membres du jury, ayant accepté d'évaluer mes travaux.

Je remercie Philipe Kubiak - maître de conférence au LAGIS (Lille) - qui a été un élément décisif pour la concrétisation de cette thèse.

Ce projet a été rendu possible grâce au co-financement du projet européen : Intelligent Transportation for Dynamic Environment (InTraDE) et de l'école des Hautes Études d'Ingénieur (HEI, Lille). Merci donc, à tous les membres du projet InTraDE et à HEI pour le support qu'ils ont représenté.

J'adresse mes remerciements les plus amicaux à mes collègues des différents laboratoires LAGIS et LGI2A et à HEI dans lesquels j'ai bénéficié d'un cadre de travail agréable et productif. Je remercie également toutes les personnes qui par leurs productions ont contribué, à cette thèse et qui démontrent que la science n'est faite que de nains juchés sur des épaules de géants. En particulier je remercie Wissam Khalil, Shahin Gelareh, Alexandre Veremme, Denis Deranton, Julie Boulenguez, Aude Morel, Gauthier Bedek, Yoann Kubera et Saïd El Hmam .

Je remercie ma famille, mes parents : Jean-Pierre et Carmen, ma soeur : Elvire et mon amie : Céline qui m'ont encouragé et soutenu tout au long de cette thèse. Je pense aussi à tous les membres de mon entourage qui comme eux, ne comprennent pas forcément mon travail dans son intégralité mais donnent de la valeur au fait que je fasse un travail qui me plaît et dont l'intérêt du sujet se renouvelle constamment à mes yeux.

Enfin je remercie toutes les personnes que j'oublie et dont les conversations sur l'informatique en général, les systèmes complexes ou les systèmes biologiques m'ont stimulé et inspiré au cours de ces trois années de thèse.

Abréviations

ABM Modèle à Base d'Agents (Agent-Based Model).

AGV Véhicule guidé et automatisé (Automated Guided Vehicle).

CS Composant Systèmes, sous parties d'un SdS.

FdS Fédération de Systèmes (Federation of Systems).

IAV Véhicule intelligent autonome (Intelligent Autonomous Vehicle).

InTraDE Intelligent Transportation for Dynamic Environment, projet européen co-financeur de cette thèse.

IRM4MLS Modèle Influence-Réaction pour la Simulation Multi-Niveaux (Influence Reaction Model for Multi-Level Simulation).

ITS Système de Transport Intelligent (Intelligent Transportation System).

ML-ABM Modèle Multi-Niveaux à Base d'Agents (Multi-Level Agent-Based Model).

SdS Système de Systèmes, système constitué d'une hiérarchie de sous systèmes indépendants.

SI Système d'Information.

SMA Système Multi-Agents.

SOA Simulation Orientée Agents.

Sommaire

Chapitre 1

Introduction

Sommaire

La thématique principale des recherches présentées dans cette thèse est la modélisation et le contrôle de systèmes complexes organisés hiérarchiquement et plus spécifiquement de systèmes de systèmes (SdS). Pour pallier le manque d'outils génériques dans ce domaine, nous proposons de développer une méthodologie. Cette méthodologie repose sur les tâches suivantes, que nous détaillerons plus tard dans ce manuscrit : 1) formaliser de manière rigoureuse le concept de SdS, 2) mettre en place une définition générique de modèle respectant cette formalisation, 3) créer une implémentation qui permet d'instancier ces modèles sur ordinateur, 4) développer un cadre de simulation pour exécuter de manière cohérente ces modèles instanciés, en procurant au modélisateur les moyens pour évaluer, prédire et contrôler ces modèles et indirectement les SdS réels, et enfin 5) appliquer cette approche au cas concret du projet européen InTraDE qui concerne la gestion de la logistique portuaire, côté terre, par un SdS. Ce système de transport intelligent (ITS) est constitué de véhicules autonomes intelligents (IAV).

Le développement de ce travail s'appuie sur la simulation de modèles multi-niveaux, en particulier son optimisation en termes de quantité d'information manipulée par cette simulation comparée aux ressources informatiques (microprocesseur et mémoire) requises pour exécuter cette simulation. Cette optimisation repose sur des améliorations d'ordre conceptuel appliquées au modèle de simulation et non sur des améliorations matérielles facilitant la médiation entre la simulation et les composants de l'ordinateur l'exécutant.

1.1 Contexte et motivations

1.1.1 Les systèmes de systèmes

Nous pouvons décrire un *système* comme "un ensemble ou un arrangement d'éléments [personnes, produits (matériels et logiciels) et processus (installations, équipements, matériaux et procédures)] qui sont reliés et dont le comportement satisfait les besoins utilisateurs/opérationnels et supporte les besoins du cycle de vie des produits", d'après le standard IEEE 1220 [IEE 2005].

Un *système complexe* est défini par [Aniorté 2006] comme "un ensemble d'éléments en interaction entre eux et avec l'extérieur. Par essence il s'agit de systèmes ouverts, hétérogènes dans lesquels les interactions sont non linéaires et dont le comportement global ne peut être obtenu par simple composition des comportements individuels". Ces critères sont subjectifs, cependant nous définissons habituellement la complexité d'un système en fonction du nombre d'entités, du nombre d'interactions entre entités, de la taille de l'environnement et du niveau de détail de tous ces éléments représentés. Traditionnellement, un système complexe peut être appréhendé selon deux points de vue : comme un tout ou comme un ensemble d'éléments. Au cours de cette thèse, nous chercherons à concilier les deux points de vue, systémique et analytique, nous prendrons en compte la forme et la structure des relations entre éléments ainsi que les propriétés isolées des éléments individuels [Bertalanffy 1968].

Un *système de systèmes* (SdS) est un système complexe doté de certaines caractéristiques. Le SdS est un concept organisationnel qui décrit un système, généralement de grande taille, composé de sous-systèmes, appelés composants systèmes (CS), imbriqués les uns dans les autres comme des poupées gigognes, autonomes, souvent hétérogènes, pouvant communiquer et s'adapter de manière locale ou globale pour répondre aux changements de leur environnement [Maier 1996, Boardman 2006, Jamshidi 2008a]. Trois éléments clés servent à distinguer les SdS des autres systèmes : 1) l'**autonomie** des CS et 2) l'**émergence** de nouvelles propriétés au sein du SdS qui permettent 3) la **robustesse** de ce dernier face aux évolutions internes (structure du SdS) ou externes (environnement).

L'intérêt principal des SdS repose sur un double paradoxe. D'une part, le modélisateur veut s'assurer du contrôle du SdS tout en laissant une grande liberté de décision aux CS. D'autre part, la conception d'un SdS se fait par un mélange d'intégrations horizontales et verticales. L'intégration horizontale revient à ajouter de nouveaux systèmes qui vont interagir avec ceux déjà présents dans le SdS en échangeant des flux (par exemple information). L'intégration verticale revient à faire apparaître plusieurs niveaux hiérarchiques de modélisation. Il en résulte certaines caractéristiques propres aux SdS, telles qu'un contrôle décentralisé, des besoins conflictuels entre systèmes et une évolution continuelle de ce type de systèmes.

Voici quelques exemples de SdS que l'on rencontre de nos jours : internet avec tous ses serveurs indépendants disposés en réseaux s'échangeant des informations ou des services pour assurer une qualité de services aux utilisateurs où qu'ils se trouvent, des colonies de robots dressant une carte partagée de l'environnement en mettant en commun le fruit de leur exploration locale ou encore un système de transport intelligent, tel qu'il est présenté dans la section suivante.

1.1.2 Le projet InTraDE

Entraîné par la mondialisation et le développement des pays émergents, le commerce maritime mondial n'a cessé de croître durant cette dernière décennie. Pour gérer cette hausse du flux de marchandises et éviter la saturation ou la perte de compétitivité par rapport à leurs voisins, les ports doivent augmenter leur productivité. Pour cela il existe deux moyens : augmenter la surface des terminaux portuaires ou augmenter le nombre d'opérateurs intervenant simultanément. Cependant ces solutions sont trop contraignantes pour les ports de petites et moyennes tailles qui parsèment la zone NWE (North West Europe). La seule stratégie acceptable pour ces ports est donc d'optimiser l'espace opérationnel et la gestion du trafic routier et maritime.

Le projet européen Intelligent Transportation for Dynamic Environment (InTraDE)[int] se déroule entre 2009 et 2013 pour répondre aux problématiques liées à cette stratégie. InTraDE a pour finalité de concevoir un système de transport intelligent (ITS) qui gère le transport du fret à l'intérieur des terminaux portuaires de la zone NWE, tout en s'adaptant à leur environnement évolutif. Les critères d'optimisation qui guident la démarche d'InTraDE sont socio-économiques, écologiques et sécuritaires.

Un moyen d'atteindre ces buts est la mise en place du déplacement automatisé des containers dans les ports, géré par une flotte de véhicules autonomes intelligents (IAV). Ces véhicules se présentent sous la forme d'une plate-forme pouvant accueillir un container, dotée de 4 roues omnidirectionnelles et pouvant communiquer grâce au wi-fi. Pour ce faire, InTraDE va produire deux livrables innovants : 1) des prototypes d'IAVs porte-conteneurs pouvant être pilotés manuellement ou automatiquement, de façon individuelle ou en train de véhicules et 2) un simulateur 3D, temps réel, reprenant les données géographiques du terrain et intégrant la dynamique réaliste des véhicules. Ce dernier outil sera utilisé lors de la phase de conception et lors de la phase de fonctionnement pour superviser les différentes entités de l'ITS.

Les ITS présents dans inTraDE sont décrits comme des systèmes indépendants qui peuvent coopérer pour réaliser un but commun tout en s'adaptant à l'évolution de l'environnement. Ces systèmes peuvent être considérés comme des SdS. Ainsi, InTraDE constitue un cas d'application concret permettant d'illustrer notre approche généraliste de modélisation de SdS.

1.2 Problématique et positionnement scientifique

1.2.1 Problématique

Le concept de SdS, défini dans les années 90 [Eisner 1991], est aujourd'hui encore trop peu formalisé, entre autres à cause du manque de consensus sur une définition universellement reconnue. Ainsi, il n'existe pas encore de modèle généraliste permettant de contrôler un SdS en assurant le respect de ses caractéristiques fondamentales. D'une part, la littérature relative aux SdS contient peu de formalismes de modélisation et se contente bien souvent de définitions théoriques ou philosophiques. D'autre part, les modèles proposés sont souvent très spécifiques car liés à une application ou à un domaine précis (simulation militaire, sociologie, biologie...) et donc peu réutilisables dans un autre cadre. De même, les organismes de recherches qui traitent spécifiquement de ce sujet sont peu nombreux. Le laboratoire Heudiasyc (Heuristic

et Diagnostic des Systèmes Complexes) de l'Université de Technologie de Compiègne (UTC) dirigé par le professeur Ali Charara est un des rares laboratoires, et le plus, reconnu dans ce domaine au niveau national. Ce laboratoire est associé à un réseau du laboratoire d'excellence Labex dans la Maîtrise des Systèmes de Systèmes Technologiques MS2T).

En dehors du manque de généricité et de réutilisabilité des modèles proposés, la littérature qui traite spécifiquement de la modélisation des SdS souffre de plusieurs lacunes. Un grand nombre de SdS possèdent des interactions entre CS qui sont des contraintes par leur genre ou par une structure. Ainsi, des systèmes d'information où les interactions entre CS se limitent à des échanges d'informations et où ils sont gérés comme des nœuds d'un réseau ne pouvant interagir qu'avec leurs voisins [Held 2008, Bar-Yam 2004, Gorod 2008]. Beaucoup de modèles proposés ne détaillent que les aspects statiques des SdS en occultant leurs aspects dynamiques, dont les mécanismes de reconfiguration assurant le respect des caractéristiques fondamentales des SdS au cours de leur fonctionnement [Sage 2007, Simpson 2009]. Les rares articles offrant un modèle généraliste, prenant en compte les aspects statiques et dynamiques des SdS, voient ce modèle appliqué à un exemple théorique qui ne peut pas être directement transposé dans la réalité [Gezgin 2012, Zhou 2011a].

Pour montrer en quoi un modèle multi-niveaux à base d'agents (ML-ABM) permet de représenter un SdS, il convient, d'abord, de définir ce concept. Lors de la modélisation d'un système complexe, le concept d'**échelle** correspond à un outil de mesure du cadre de l'évolution des entités du système, en termes d'espace, de temps et/ou de granularité [Coyrehourcq 2011]. Dans la littérature les termes d'échelle et de niveau sont souvent employés l'un à la place de l'autre. Cependant, le concept de **niveau** est utilisé pour caractériser les abstractions choisies, leur échelle et leur position dans une hiérarchie qui organise et structure le système. Un **ML-ABM** est, donc, un modèle à base d'agents dans lequel les agents sont organisés par le modélisateur entre différents niveaux [Vo 2012a].

Les principales raisons d'utiliser un ML-ABM sont : 1) la modélisation des interactions entre différents niveaux, avec par exemple la prise en compte d'entités à différentes échelles spatio-temporelles ou organisationnelles ainsi que l'apparition de structures ou de modèles émergeant à partir des interactions entre entités à un autre niveau, 2) le couplage de modèles hétérogènes représentant différentes échelles ou domaines d'intérêt et 3) l'adaptation dynamique du niveau de détail des simulations pour économiser des ressources informatiques [Morvan 2012a]

Un grand nombre de modèles multi-niveaux ont été proposé dans la littérature [Davis 1993, Gil Quijano 2009, Morvan 2011, Navarro 2011, Picault 2011, Vo 2012a]. Cependant peu d'entre eux sont assez généralistes ou possèdent les concepts fondamentaux (entités récursives, représentation explicite des niveaux, structure organisationnelle...) pour pouvoir modéliser toute instance de systèmes multi-niveaux ou constituer une solution réutilisable. Le modèle présenté dans cette thèse et le formalisme qui y est attaché ont été conçus dans ce but et sont donc particulièrement adaptés pour représenter et simuler des instances de systèmes complexes dont les nombreuses entités s'étalent sur plusieurs niveaux de représentation, comme les SdS par exemple.

La modélisation multi-niveaux peut-être utilisée pour adapter dynamiquement le niveau de détail dans les simulations de systèmes complexes. L'étude des simulations de systèmes complexes effectuant un compromis entre le niveau de détail des entités simulées et les ressources informa-

tiques utilisées pour exécuter cette simulation est un sujet de recherche assez jeune. En effet, il n'existe pas de méthode générique pour décider pour quelles entités il est nécessaire de changer de niveau de détail, ni quand, ni comment. Bien souvent, dans les travaux existants, les modèles de simulation proposée se limitent à deux ou trois niveaux de détail (microscopique, mésoscopique et macroscopique) et offre des mécanismes de détection et de changements de niveau de détail liés à la structure du problème traité par la simulation [Gil Quijano 2009, Navarro 2011, David 2011].

1.2.2 Contribution scientifique

Cette thèse propose une méthodologie de modélisation multi-niveaux des SdS. Nous proposons une définition des SdS en précisant explicitement les caractéristiques que doivent respecter les CS ou le SdS dans sa globalité. Cette définition sert ensuite de base pour concevoir un formalisme généraliste qui retranscrit aussi bien les aspects statiques que dynamiques des SdS. En particulier, des algorithmes sont proposés pour contrôler la création ou la réorganisation du SdS lorsque les buts ou capacités des CS évoluent.

Ensuite, ce formalisme est exploité pour créer un modèle de SdS. Nous utilisons un modèle à base d'agents (ABM) pour profiter de l'autonomie et de la capacité à raisonner des agents qui est comparable à celle des CS. Ainsi un CS est incarné, de façon intuitive, par un agent. les aspects organisationnels des SdS sont gérés par le modèle agent groupe rôle (AGR) [Ferber 1998]. Les aspects fonctionnels, qui consistent à guider les SdS dans l'accomplissement de leur but global, sont pris en charge par une spécification fonctionnelle [Hübner 2002b]. Enfin, les aspects multi-niveaux sont modélisés en utilisant le méta-modèle multi-agents multi-niveaux IRM4MLS (Influence/Reaction Model for Multi-Level Simulations) [Morvan 2011].

Enfin, nous développons un cadre de simulation pour les SdS. Nous intégrons à la plate-forme multi-agents MadKit[mad] d'une part les concepts du méta-modèle IRM4MLS, d'autre part les concepts organisationnels et fonctionnels de notre formalisme de représentation des SdS, ainsi que les algorithmes de création et de réorganisation de SdS.

Les outils proposés dans cette thèse ont été exploités pour produire une co-simulation d'un SdS sous forme de Système de Transport Intelligent (ITS) composé d'IAVs affectés au transport de conteneurs dans des terminaux portuaires. Les données réalistes issues des interactions entre les IAVs gérant le fret et leur environnement (cinématique des véhicules, adhérence au sol...) sont générées en temps réel par la plate-forme de simulation routière SCANeRstudio [sca], tandis que toute la partie qui ne concerne pas les éléments physiques (hiérarchie des organisations, ordonnancement des tâches...) est gérée par la plateforme multi-agents MadKit. MadKit est doté des concepts organisationnels d'agent, de groupe et de rôle qui permettent de représenter les aspects organisationnels intrinsèques aux SdS. Ainsi toutes les interactions et événements qui se déroulent lors de la co-simulation forment une représentation fidèle de l'évolution du système physique qui est simulé. De cette manière, cette co-simulation assure la conduite, l'évaluation ou la prévision de l'évolution de l'état du système physique.

Le cas d'application lié à InTraDE présenté dans cette thèse est unique au sein de la littérature portant sur la simulation des SdS. En effet, il vise à simuler un système réel (et non d'un exemple jouet) avec des données réalistes traitées en temps réel. Il permet de modéliser un SdS en respectant les caractéristiques fondamentales de ce type de systèmes et la simulation des modèles produits va permettre de contrôler la création d'un système réel, son fonctionnement

normal ou sa reconfiguration en tenant compte des aspects statiques et dynamiques inhérents aux SdS, en conditions réalistes. De plus les interactions entre les CS du SdS représenté sont généralistes et ne se limitent pas à l'échange d'informations.

Cependant, InTraDE demeure un cas d'application pratique de notre travail. L'utilisation de la méthodologie présentée dans ce manuscrit peut être étendue pour modéliser tout SdS. Cet usage peut même s'étendre à tout système où les entités sont présentes à différents niveaux de représentation organisés hiérarchiquement.

En plus de la modélisation des systèmes complexes, cette thèse aborde les problématiques liées à leur simulation, en particulier, le fait que les ressources informatiques (mémoire et microprocesseur) nécessaires pour simuler avec précision de tels systèmes sont particulièrement importantes. Nous proposons une méthodologie pour tirer partie de la capacité des simulations multi-niveaux à produire un compromis entre la précision de la simulation et les ressources informatiques utilisées. Cette méthodologie consiste à appliquer dynamiquement des changements de niveau de détail à la représentation des entités simulées. Elle est dotée d'un outil qui évalue la qualité des simulations en comparant le gain de ressources produit par rapport à la perte d'information générée, cette mesure est appelée la *cohérence* [Davis 1993].

1.2.3 Principaux domaines de recherche associés à cette thèse

Cette thèse aborde différents domaines de recherche qui ne sont pas liés entre eux d'ordinaire. Cette section présente ces différents domaines qui ont été réunis au cours de cette thèse.

Cette thèse traite de la modélisation multi-agents multi-niveaux [Gaud 2007, Scerri 2010, Picault 2011, Morvan 2011] en tirant partie de modèles organisationnels existants pour faciliter le contrôle des modèles simulés [Ferber 1998, Hübner 2002b].

Cette approche est utilisée pour représenter et exécuter des systèmes de systèmes [Maier 1996, Bar-Yam 2004, Boardman 2006, Sage 2007, Jamshidi 2008a, Sauser 2010, SoS 2010] et en particulier la logistique portuaire effectuée par une flotte semi-autonome d'IAVs [Li 2004, Mathew 2005, Contet 2011, Khalil 2011b, Khalil 2011a, Winikoff 2011].

Dans une moindre mesure, cette thèse fait, aussi, appel à certains algorithmes génériques utilisés couramment dans le cadre de la logistique et du transport [Polacek 2004, Pisinger 2007, Li 2009, Fleszar 2009]. Ces algorithmes permettent de résoudre des variantes du problème du voyageur de commerce : trouver un chemin optimal sur un réseau, en fonction de critères fixés à l'avance (chemin le plus court ou le plus rapide) ou en évolution (encombrement des routes).

1.3 Organisation de la thèse

En dehors de l'introduction et de la conclusion, cette thèse est découpée en quatre chapitres principaux de plus en plus appliqués et concrets.

Le **chapitre 2** dresse un état de l'art de la modélisation des SdS. Il décrit et précise les caractéristiques fondamentales liées à ce concept organisationnel et présente plusieurs domaines d'application. Puis il introduit les systèmes multi-agents et leur importance dans la simulation

et enfin les systèmes multi-agents multi-niveaux et montre en quoi ces derniers sont adaptés pour modéliser des SdS.

Le **chapitre 3** présente le méta-modèle, multi-agents, multi-niveaux IRM4MLS qui servira, par la suite, de cadre à la modélisation des SdS. Après avoir montré les avantages spécifiques à ce méta-modèle, ce chapitre montre comment développer IRM4MLS pour tirer partie de son pouvoir de représentation. Enfin ce chapitre donne des éléments pour optimiser l'usage des ressources informatiques utilisées lors des simulations en faisant varier dynamiquement le niveau de détail.

Le **chapitre 4** présente une représentation graphique d'un SdS et une taxonomie des différents types de composants systèmes (CS) possibles. Il fournit ensuite un formalisme générique utilisant IRM4MLS qui permet de modéliser et de simuler les SdS tout en respectant les caractéristiques fondamentales liées à ce concept. Puis il donne deux algorithmes assurant le respect de ces caractéristiques de manière dynamique, en particulier lorsque le SdS est initialisé ou lorsque ses capacités évoluent.

Enfin, le **chapitre 5** dévoile les applications concrètes tirées de cette thèse sous forme de co-simulations implémentées sur un couple de plate-formes ; l'une multi-agents (MadKit[mad]) et l'autre spécialisée dans le domaine de la simulation routière réaliste (SCANeRstudio[sca]). MadKit doté des concepts d'IRM4MLS sert ici à simuler tous les aspects non physiques du modèle comme la hiérarchie d'organisation des CS, la répartition des tâches ou les modules de prise de décision des IAVs. SCANeRstudio est utilisé pour produire des données physiques réalistes en conformité avec la physique et la cinématique des IAVs, qui pourront être exploitées dans MadKit.

Finalement, le **chapitre 6** dresse le constat de ce que cette thèse a permis de produire, puis évoque un certain nombre de perspectives liées à ce travail.

Chapitre 2

Modélisation des systèmes de systèmes : État de l'art

Sommaire

2.1 Introduction

On peut traduire la définition de la **simulation** donnée par [Shannon 1998] comme "le processus permettant de concevoir un modèle d'un système réel et de mener des expérimentations sur la base de ce modèle pour comprendre le comportement du système ou évaluer diverses stratégies pour son fonctionnement (par les limites imposées par un ou plusieurs critère)." D'après [Fichwick 1997], l'objectif de la simulation est de faciliter la compréhension de la dynamique d'un système réel et tenter d'en prédire l'évolution. Pour satisfaire cet objectif, il est nécessaire d'élaborer un modèle du système à étudier, de l'exécuter sur un calculateur et d'analyser les résultats de cette exécution.

Un **modèle** est une abstraction de la réalité, qui tente de la décrire au mieux sans pouvoir la retranscrire parfaitement [Muller 2000]. Un modèle est constitué d'une représentation fidèle

du système et de son comportement avec les éléments (propriétés et comportements) les plus importants représentés en détails, alors que les éléments les moins importants sont ignorés. L'intérêt majeur du modèle est que la représentation et la simulation des éléments modélisés restent moins coûteuses que les tests effectués sur ces éléments dans le système réel. Un modèle doit 1) représenter la complexité du système en prenant en compte toutes les informations issues de ce dernier. 2) représenter fidèlement le comportement du système, pour offrir des services tels que l'évaluation de ce système et de ses performances, le contrôle du système en temps réel ou la prédiction de son évolution.

[Fichwick 1997] définit ainsi la **simulation multi-niveaux** : "La simulation multi-niveaux est un type particulier de simulation où le modèle proposé du système intègre différents niveaux d'abstraction (au moins deux) et où les outils nécessaires à son exécution permettent de faire cohabiter ces différents niveaux d'abstraction au sein d'une même exécution et d'assurer la transition dynamique entre eux en fonction de contraintes définies (dépendantes du modèle ou du cadre expérimental)." Ainsi la simulation multi-niveaux pose deux problématiques dans lesquelles peuvent s'inscrire la simulation et la modélisation des SdS. 1) comment gérer les interactions entre niveaux ou CS ? 2) comment définir la transition dynamique entre niveaux ? Ou comment gérer le regroupement d'entités ou de CS créant des organisations qui sont instanciées dans des niveaux supérieurs par d'autres entités ou CS ?

Dans ce chapitre, nous commençons par poser une définition exploitable des SdS basée sur les caractéristiques fondamentales des CS ou du SdS dans son intégralité. Celles-ci nous permettront, par la suite, de produire un formalisme solide et générique des SdS. Nous établissons également le positionnement de notre contribution dans le domaine de la modélisation dynamique et la simulation des SdS par rapport aux autres approches de modélisation des SdS. Dans un second temps nous allons présenter les concepts fondamentaux et les capacités uniques liés aux SMA. Nous allons, également, situer les outils que nous utilisons par rapport aux approches de modélisation multi-agents multi-niveaux existantes. Puis, nous montrons en quoi ces outils sont utiles à la modélisation des systèmes complexes tels que les SdS.

2.1.1 Clarification

Dans ce manuscrit nous utilisons souvent les termes de système, système de systèmes (SdS) et système multi-agents (SMA). Nous prenons l'opportunité de cette introduction pour clarifier le sens de ces termes dans le contexte de cette thèse.

Nous considérons un **système** de la manière la plus générale possible. Le terme système vient du mot grec *sistèma* : différentes entités qui 1) restent (istèma, de stase : "une stagnation", "un arrêt") 2) ensemble (syn). En prenant en compte deux des définitions du dictionnaire Larousse pour décrire ce terme, on considère le fonctionnement du système en plus de ses éléments constitutifs : "Ensemble d'éléments considérés dans leurs relations à l'intérieur d'un tout fonctionnant de manière unitaire", et aussi le fait que si ce système est construit, il l'est généralement pour poursuivre un objectif ou une fonction : "Appareillage, dispositif formé de divers éléments et assurant une fonction déterminée" [Lar].

Un **système de systèmes** est un concept organisationnel qui décrit le fait qu'un regroupement de systèmes, précédemment décrits, produit des services et peut accomplir des tâches que les systèmes seuls ne peuvent proposer ou accomplir. Ce concept est relié à celui d'émergence et

se retrouve dans la célèbre maxime d'Aristote : "la totalité est plus que la somme des parties" [Aristote 2008] [1] .

En informatique, un **système multi-agents** est un outil constitué d'un ensemble d'agents autonomes dont l'activité produit des simulations qui sont utilisées pour représenter l'état et les interactions de systèmes décrits précédemment [Ferber 1995].

2.2 Les Systèmes de Systèmes

De plus en plus de projets traitant des systèmes complexes font appel au concept de SdS [Held 2008, Zhou 2011b, Gerst 2012, Khalil 2012b, Weyns 2013]. Malheureusement il n'existe pas de consensus fort dans la littérature concernant ce concept. Bien que de nombreuses définitions aient été proposées dans la littérature, aucune d'elle n'est universellement acceptée. Pour mieux appréhender les SdS nous commençons, dans la première partie cette section, par présenter la genèse de ce concept et son évolution dans le domaine de la recherche. Ensuite, dans la seconde partie, nous posons une définition des SdS basée sur leurs caractéristiques fondamentales. Nous comparons cette caractérisation à d'autres existantes. Puis, nous montrons en quoi cette caractérisation permet de capturer la nature des SdS, en particulier, les aspects statiques et dynamiques de tels systèmes et de les différencier des autres systèmes. Finalement, dans la dernière partie, nous établissons une classification des approches de modélisation des SdS les plus récentes pour nous positionner par rapport à ces dernières.

2.2.1 Historique des systèmes des systèmes

La première mention du terme remonte à 1956. Boulding et al. ont imaginé un SdS comme une "gestalt" dans le domaine de la construction théorique créant un "spectre de théories" plus grand que la somme de ses parties [Boulding 1956]. Selon le gestaltisme ou la psychologie des formes " les processus de la perception et de la représentation mentale traitent spontanément les phénomènes comme des ensembles structurés (les formes) et non comme une simple addition ou juxtaposition d'éléments."[wik] Cette première définition sous-entend une approche holistique des SdS.

Ackoff [Ackoff 1971], Jacob [Jacob 1974], Jackson et Keys [Jackson 1984] ont également fait mention des SdS dans les domaines de la recherche opérationnelle, du management et de la biologie, avant qu'une définition formelle ne soit vraiment posée.

Eisner et al. ont introduit dans leurs travaux en 1991 la première définition des SdS considérée comme moderne. En désignant un SdS comme "un ensemble de plusieurs systèmes acquis indépendamment, chacun sous un processus nominal d'ingénierie des systèmes. Ces systèmes sont indépendants et forment, dans leur opération combinée, une solution multifonctionnelle à une mission globale cohérente. L'optimisation de chaque système ne garantit pas l'optimisation du SdS dans sa globalité"[Eisner 1991].

En 1994, Shenhar décrit les SdS comme des tableaux en insistant sur sa vision de leur structure plutôt que de leur comportement. Pour lui un SdS est "une large collection ou un

1. Cette maxime a été complétée par Morin : "Le tout est à la fois plus et moins que la somme des parties". Il montre ainsi la dualité émergence/contrainte des organisations [Morin 1990].

réseau de systèmes fonctionnant ensemble afin de réaliser une tâche commune" [Shenhar 1994].

En 1995, Holland a proposé de voir les SdS adaptatifs en changement constant grâce à leurs capacités d'auto-organisation issues de règles de gouvernance locales [Holland 1995]. Cette même année, l'amiral Owens introduira le concept de SdS dans le domaine militaire en utilisant cette approche [Owens 1995].

En 1996, Maier a apporté une contribution majeure dans le domaine des SdS, en proposant une caractérisation des SdS pour différencier ceux-ci des systèmes monolithiques. Elle comprend les caractéristiques suivantes : indépendance opérationnelle et indépendance managériale des éléments, développement évolutionnaire, comportement émergent et distribution géographique [Maier 1996]. En 1998, Maier a inclus dans ses travaux une nouvelle définition des SdS : "un SdS est un assemblage de composants qui peut être vu individuellement comme un système qui possède deux propriétés additionnelles : indépendance opérationnelle des composants [et] indépendance managériale des composants"[Maier 1998].

La même année, Manthorpe a proposé de combiner dans un SdS les aspects de commande, contrôle, computation, communication et d'information (C4I) et ceux d'intelligence, de surveillance et de reconnaissance (ISR) afin d'assurer une domination dans les opérations militaires de grande ampleur [Manthorpe 1996]

En 1997, Kotov a été l'un des premiers à proposer une approche pour modéliser et synthétiser les SdS. Il a également proposé une des définitions des SdS les plus concises. Les SdS sont des "systèmes de grande échelle, distribués et concurrents, dont les composants sont des systèmes complexes eux-mêmes (e.g. entreprise, intranets). Les structures communicantes sont des structures hiérarchiques qui représentent un SdS d'une manière uniforme, systématique"[Kotov 1997].

En 1998, Luskasik a considéré l'ingénierie de SdS dans le domaine de l'éducation. Il a étudié un processus d'intégration de systèmes dans des SdS qui contribuent à l'évolution de l'infrastructure sociale [Luskasik 1998].

En 2000, Pei a créé une méthode d'intégration pour les SdS. Elle a pour but de développer l'intégration, l'interopérabilité et l'optimisation des systèmes pour améliorer leurs performances sur les champs d'opérations militaires [Pei 2000].

En 2001, Sage et Cuppan ont proposé un "nouveau fédéralisme" pour concevoir et gérer les SdS. Ces principes peuvent aussi être appliqués aux fédérations de systèmes ou aux SdS fédérés [Sage 2001].

En 2003, Keating et al. ont produit une définition des SdS difficilement exploitable en dehors de l'ingénierie des SdS mais néanmoins intéressante. Pour eux un SdS est un "méta-système comprenant de multiple systèmes complexes intégrés et autonomes qui peuvent être diverses par leur contexte technologique, opération, géographie, et patron conceptuel" [Keating 2003].

En 2004, Bar-Yam et al. ont étudié des applications des SdS dans les domaines de la biologie, de la sociologie ou de l'armée [Bar-Yam 2004]. Pour améliorer la compréhension des SdS, plutôt que de proposer une nouvelle définition, ils ont doté la caractérisation de Maier de deux ajouts. Ces nouvelles caractéristiques sont : l'auto-organisation, l'adaptation, des systèmes

complexes, la spécialisation individuelle, la synergie, l'interdépendance, de multiples taxonomies et la recherche de buts ou de besoins.

En 2005, DeLaurentis a étudié les SdS dans le domaine du transport et en a déduit certains traits distinctifs de ces SdS. Il a également proposé une méthode théorique pour faciliter la modélisation et la simulation des SdS grâce au paradigme agent et à la programmation orientée objet. [DeLaurentis 2005]

En 2006, Boardman et Sauser ont articulé une nouvelle caractérisation des SdS qui permet de les désigner et de les différencier des autres systèmes. Cette caractérisation est issue de l'étude comparative de nombreuses définitions des SdS. Les caractéristiques qui constituent cette caractérisation sont l'autonomie, l'appartenance, la connectivité, la diversité et l'émergence [Boardman 2006].

En 2007, Sage et Biemer ont proposé un processus formel d'ingénierie des SdS, fédérations de systèmes et familles de systèmes [Sage 2007].

En 2007, Sloane et al. ont proposé une modélisation des SdS avec une architecture orienté service. Ainsi, les CS sont des pourvoyeurs de services ce qui permet de guider l'émergence au sein du SdS tout en garantissant l'autonomie des CS [Sloane 2007].

En 2008, Gorod et al. ont dressé une chronologie qui retrace l'état de la recherche dans le domaine des SdS [Gorod 2008]. Ils ont également proposé une méthode théorique pour définir et délimiter les SdS.

En 2008, Jamshidi a écrit les deux premiers livres dédiés aux SdS [Jamshidi 2008a, Jamshidi 2008b] dans lesquels il donne une définition des SdS reconnue dans cette communauté : "Systems of systems are large-scale integrated systems that are heterogenous and independently operable on their own, but are networked together for a common goal." Que l'on peut traduire par :

> "Les Systèmes de Systèmes sont des systèmes intégrés de grande échelle qui sont hétérogènes et opérationnels de manière indépendante, mais sont structurés en réseaux pour accomplir un but commun."

Depuis 2008, un certains nombre de livres ont été écrits concernant les SdS : [Luzeaux 2008a, Luzeaux 2008b, Cantot 2009, Nanayakkara 2010].

2.2.2 Définitions fondamentale des SdS

Dans cette thèse nous nous appuyons principalement sur la définition de Sage et Coppan [Sage 2001], inspirée par la caractérisation de Maier [Maier 1996], qui peut être résumée comme suit :

> Un SdS est un ensemble de Composants Systèmes (CS) organisés hiérarchiquement et dotés d'un but global. Les CS peuvent être hétérogènes et composés d'autres CS. Ils gèrent leurs propres ressources et leurs sous CS d'une manière indépendante et peuvent coexister et coopérer pour accomplir une mission qu'un CS ne pourrait réaliser seul. Les CS sont distribués géographiquement sans aucun lien physique

entre eux. Un SdS doit être robuste et adaptatif : son environnement, son but ou sa structure (en ajoutant ou supprimant des CS) peuvent évoluer sans modifier ses capacités à atteindre son but principal.

Un **sous CS** d'un CS donné représente un des CS présent dans un niveau inférieur qui forment ce CS. Pour un CS donné, son **super CS** est le CS présent dans un plus haut niveau auquel il appartient.

[DeLaurentis 2005] explique qu'une définition ne permet pas de capturer l'essence des SdS. Pour définir un formalisme de modélisation des SdS, on a donc besoin d'établir une série de traits qui permettent de distinguer un système classique d'un SdS. Dans la suite de cette partie nous présentons une caractérisation des SdS reconnue de la communauté qui va nous servir de base pour établir notre formalisme de modélisation des SdS.

Caractérisation des SdS

La définition précédente respecte les cinq caractéristiques fondamentales énoncées par Maier [Maier 1996], que nous illustrons avec un exemple tiré du projet InTraDE.

1. **Indépendance opérationnelle** : Chaque CS possède ses propres ressources nécessaires à l'accomplissement de ses missions. Par exemple, un IAV possède sa propre réserve d'énergie et son propre statut diagnostiqué en temps réel qu'il gère .

2. **Indépendance managériale** : Une fois qu'une mission est attribuée à un CS, Il se gère lui-même ainsi que ses sous CS, sans aide extérieure, pour accomplir cette mission. Par exemple, une flotte d'IAVs attachée à un quai décide de son propre chef comment organiser ses IAVs pour accomplir les missions qui lui ont été allouées. Au niveau individuel un IAV auquel on a attribué une tâche de livraison d'un conteneur décidera comment réaliser cette tâche cette livraison et choisira l'itinéraire qu'il empruntera pour effectuer cette livraison.

3. **Distribution géographique** : Les CS peuvent échanger des informations, mais ne peuvent pas échanger des éléments physiques ou de l'énergie. Par exemple, lorsque deux IAVs sont rattachés pour former un train de véhicules pour transporter des containers de 40 pieds de long, ils perdent leur distribution géographique et ne sont plus considérés comme des CS.

4. **Comportement émergent** : L'accomplissement du but global d'un SdS est seulement possible à travers l'action conjointe de plus d'un de ses CS. Par exemple, si un IAV est suffisant pour atteindre le but global du SdS qui est d'accomplir toutes les tâches de transport du port, alors ce SdS n'a pas de raison d'être.

5. **Développement évolutif** : Un SdS peut s'adapter à l'ajout ou à la suppression de CS ou à l'évolution de l'environnement ou encore au changement du but global pour rester capable d'accomplir ce dernier par l'action de ses CS. Par exemple, quand un IAV doté d'une mission perd sa capacité à l'accomplir, il en informe son super CS. Ce super CS

peut ainsi résoudre le problème de perte de capacité en allouant cette mission a un autre CS avec des capacités intactes.

Ces cinq caractéristiques doivent être respectées en permanence pour définir un SdS. Prenons l'exemple d'un IAV décrit précédemment. Un tel système n'est pas considéré comme un SdS car toutes les ressources nécessaires (moteur, batterie) pour accomplir une tâche utile sont partagées par tout le véhicule, l'indépendance opérationnelle n'est donc pas respectée. Supposons que l'on rende les moitiés avant et arrière autonomes et disposant de leurs ressources en propre, avec deux roues, un moteur et une batteries par exemple, un tel système n'est toujours pas un SdS car l'indépendance managériale n'est pas respectée. En effet ces moitiés avant et arrière répondent aux commandes d'un seul microprocesseur qui est commun à tout le véhicule. Il n'y a donc pas de prise de décision autonome. Supposons que l'on dote le véhicule de deux microprocesseurs, dédiés à chaque moitié du véhicule. Ces deux moitiés sont toujours liées physiquement donc la caractéristique de dispersion géographique n'est pas respectée. Donc nous ne sommes toujours pas face à un SdS. À partir de ce moment les deux dernières caractéristiques sont impossible à respecter. On peut ainsi conclure que l'élément de base (CS élémentaire) d'un SdS est une unité de ressources, de décision et d'action non contrainte géographiquement.

Il existe des caractérisations similaires, comme celle de Bordman et Sauser [Boardman 2006]. Cette caractérisation fournit les caractéristiques suivantes qui permettent de différencier un SdS d'un système monolithique :

1. *Autonomie* - les CS exercent leur autonomie en vue de réaliser un but global ;
2. *Appartenance* - les CS choisissent d'appartenir au SdS d'après un calcul basé sur le rapport coût/bénéfice ;
3. *Connectivité* - les CS fournissent une connectivité dynamique pour améliorer les capacités générales du SdS ;
4. *Diversité* - un produit caractéristique d'un SdS qui n'est pas disponible dans des systèmes seuls ;
5. *Émergence* - l'apparition de capacités non prévues à l'origine est rendue possible, ainsi que la détection et à l'élimination précoce des comportements indésirables.

[Gorod 2008] a établi une revue des méthodes d'ingénierie des SdS (SoSE en anglais) et s'appuie sur la caractérisation précédente des SdS pour créer un cadre de gestion pour l'ingénierie des SdS.

Dans ces travaux nous n'adoptons pas la caractérisation de [Boardman 2006] car ses éléments sont redondants avec les caractéristiques de [Maier 1996], ou alors sortent du champs d'application de ce manuscrit. Ainsi, l'Autonomie peut contenir les indépendances opérationnelles et managériales des éléments. L'Appartenance est un cas à part car pour nous les CS ne choisissent pas d'appartenir ou non au SdS mais ils offrent plutôt des capacités pour réaliser un but et si le coût pour atteindre ce but est trop élevé les CS "refusent" de fournir cette capacité. Cependant, dans les SdS que nous abordons, la Connectivité et de Diversité sont traitées implicitement. Notre but étant de proposer un outil de modélisation générique et unifié on considère que la communication entre CS est une capacité des CS comme les autres et que les CS peuvent interagir entre eux sans besoin de nouvelles interfaces. On considère que le travail d'interfaçage a été fait auparavant. On peut voir notre SdS comme un ensemble de

jeux de construction Lego© : chaque nouvel élément (boîte) est indépendant et peut offrir ses propres capacités, on sait qu'il peut interagir avec les autres éléments (car les éléments font partie d'un système unifié), et de nouvelle capacités non déductibles des différents éléments peuvent émerger lors de l'interaction entre éléments.

Dans le cas du projet InTraDE nous sommes face à un système constitué a) de nombreux véhicules différents, b) dont une flotte de véhicules autonomes (pouvant détecter les obstacles, s'auto-diagnostiquer,...), c) dans un environnement dynamique et d) ayant pour mission d'assurer le transport de biens en temps réel en prenant en compte la sécurité. Ces systèmes autonomes doivent communiquer pour collaborer. On peut poser le fait que ce système est un SdS. Donc notre but est d'implémenter et de valider un modèle de SdS pour contrôler et piloter l'ITS du projet InTraDE. Si la collaboration est affectée les capacités des véhicules autonomes (CS) doivent être utilisées pour réorganiser le système pour permettre de nouveaux aux CS de collaborer. Si cela est impossible alors le système ne peut pas être considéré comme un SdS.

D'autres définitions semblent être trop imprécises pour être exploitables dans le cadre de cette thèse. Dans de nombreux cas [Maier 1996, Dauby 2011, Acheson 2012] les SdS décrits se réduisent à des systèmes d'informations (SI) où les interactions entre systèmes sont limitées à l'échange d'information. Dans ce cas de figure d'un SI, la gestion du SdS se borne, en général, à assurer la transmission et le partage des données. De plus si le SdS est représenté par un réseau et les CS en sont les noeuds et les arcs sont leurs communications la configuration du SdS se réduit à établir les bons arcs. Ceci est trop restrictif pour représenter des systèmes avec des interactions fortes [Michel 2003], e.g., deux robots joignant leur force pour porter un objet trop lourd pour un robot seul.

Un grand nombre de définitions ne sont pas exploitables car elles sont adaptées à l'ingénierie des SdS. L'ingénierie des SdS s'apparente à la gestion de projet en traitant un nombre important d'aspects que nous ignorons car nous nous focalisons sur la conception et le contrôle du SdS. Les différentes phases d'ingénierie d'un système ou d'un SdS sont les suivantes [SoS 2010] :

1. développement des besoins (cahiers des charges),
2. analyse logique,
3. conception de solutions,
4. implémentation,
5. intégration,
6. vérification,
7. validation,
8. transition (livraison du produit fini),
9. analyse des décisions,
10. planification technique,
11. évolution technique,
12. établissement des besoins managériaux,
13. gestion des risques,
14. configuration managériale,
15. gestion des données,

16. gestion des interfaces .

Notre approche est généraliste et a pour but de contrôler un SdS une fois que ses éléments sont déployés. C'est pourquoi nos travaux peuvent être utilisés et intégrés par ceux qui font de l'ingénierie de SdS. Cependant, la méthodologie que nous proposons s'inscrit seulement dans les phases d'ingénierie 4 à 7.

Les différents types de SdS

Les familles de systèmes ou fédérations de systèmes (FdS) sont des systèmes qui sont eux-mêmes composés de systèmes complexes. Les FdS peuvent être différenciés des systèmes conventionnels par leur haut degrés d'autonomie, d'hétérogénéité et la dispersion spatiale de leurs composants[Sage 2001, Sage 2007, SdS 2004]. Les SdS peuvent être considérés comme un type particulier de FdS : les CS opèrent indépendamment et sont mues par leurs buts locaux, néanmoins ils doivent coopérer pour accomplir des buts globaux potentiellement contradictoires avec leurs buts locaux [Huhn 2011].

Le concept de SdS possède de nombreuses applications dans différents domaines comme les simulations maritimes ou militaires, la biologie, la climatologie ou la sociologie [Bar-Yam 2004, Gerst 2012, SdS 2004, Mahulkar 2009]. En sociologie, les comportements grégaires des humains peuvent entraîner l'émergence naturelle d'organisations ou de sociétés considérées comme des SdS. Dans un contexte militaire, les SdS sont parfois la seule approche disponible pour représenter les influences entre organisations à différents niveaux, avec leur propre aire d'intervention (air, mer, terre) dans des opérations de grande ampleur.

Dans son guide d'ingénierie des SdS, le département de défense des États Unis d'Amérique (DoD en anglais) décrit quatre types de SdS, suivant leur degrés de centralisation [SoS 2010]. Ils peuvent être : 1) **Virtuels**. Le SdS ne possède pas de gestion centralisée ni d'objectif communément admis comme central. 2) **Collaboratifs**. Les CS à l'intérieur du SdS interagissent de manière plus ou moins volontaire pour réaliser un objectif communément admis comme central. 3) **Reconnus**. Le SdS a des objectifs reconnus, un décideur désigné, et des ressources, alors que les CS conservent leurs ressources indépendantes, leurs objectifs et leurs routines de création, de développement et d'entretien. 4) **Dirigés**. Le SdS est construit pour accomplir un but spécifique. Les CS opèrent indépendamment, mais leur mode opératoire est subordonné à des objectifs de gestion centralisée.

Les SdS virtuels sont généralement utilisés pour étudier l'évolution de systèmes complexes naturels tels que les systèmes sociaux ou biologiques [Bar-Yam 2004]. Les auteurs de ce dernier article mettent en avant l'hypothèse selon laquelle les SdS purement collaboratifs émergent seulement si l'environnement influence les CS, créant ainsi un problème inédit qui nécessite l'existence du SdS pour être résolu. Ces cas ne correspondent pas au sujet de cette thèse, qui se concentre sur l'ingénierie de systèmes dotés d'une hiérarchie et d'une structure de niveaux emboîtés les uns dans les autres. Ainsi, cette thèse traite principalement des SdS dirigés ou reconnus. De plus, les outils que nous proposons dans ce manuscrit permettent de modéliser tout type de SdS. Les SdS dirigés étant les plus contraints il suffit de relaxer ces contraintes, en définissant un but moins directif ou une structure plus lâche pour le SdS.

2.2.3 Applications et modélisation des SdS

Pour illustrer le fait que les SdS sont appliqués dans beaucoup de systèmes d'ingénierie nous allons effectuer une revue des approches de modélisation de SdS existants. En invoquant en premiers les plus appliqués qui peuvent difficilement être réutilisée. Ensuite nous allons présenter les approches plus généralistes. Puis nous allons déterminer les outils nécessaires pour élaborer notre propre modélisation des aspects statiques et dynamiques des SdS.

Parce que les SdS ont été introduits dans divers domaines, des outils dédiés ont été proposés pour les modéliser et les contrôler. Cependant, jusqu'à récemment les SdS sont restés un concept théorique sans vrai formalisme de simulation générique. Plusieurs approches de modélisation des SdS ne sont pas génériques et se concentrent sur des problèmes liés à un domaine particulier.

Il existe dans la littératures de nombreux articles proposant des pistes intéressantes pour modéliser les SdS. Mais on rencontre des problèmes récurrents dans certains de ces articles, par exemple le non-respect des caractéristiques fondamentales des SdS, comme leur autonomie. C'est le cas de [Yang 2011] qui propose une approche centralisée pour gérer la reconfiguration d'un SdS. Cependant, [Yang 2011] ne traite pas du contrôle du SdS lors de la reconfiguration et l'existence des CS n'est pas abordée. C'est le cas également de [Huhn 2011] qui modélise une flotte d'AGVs sous forme de SdS. Les AGVs ne sont pas autonomes au sens des CS car ils sont "pilotés" par le marquage au sol.

Un autre problème fréquent est le manque de détails dans le modèle proposé. Dans [Flanigan 2012] une méthodologie est proposée pour initialiser la construction d'un SdS, basée sur l'allocation d'exigences et de capacités, sans que les communications et les interactions entre CS ne soient considérées. Enfin, un certains nombre d'article modifient les caractéristiques des SdS en ajoutant des contraintes supplémentaires à la structure des SdS, en contradiction avec leurs définitions d'origine. [Weyns 2013] propose ainsi trois styles d'architecture pour contrôler l'auto-adaptation des SdS. Ce contrôle est effectué en créant de manière arbitraire deux types de CS : des actionneurs et des contrôleurs. C'est ce que fait également [Adler 2012] en plaçant les systèmes opérants et les systèmes habilitants (qui conditionnent les capacités des premiers) dans des niveaux différents.

Dans les tableaux suivants (2.1, 2.2 et 2.3) nous présentons une étude comparative des tentatives pour modéliser les SdS de manière générique. Ce tableau contient les références des différents travaux, les caractéristiques et définitions qui sont utilisées pour distinguer les SdS, le but visé par ces travaux, les outils qui sont utilisés ou produits pour atteindre ce but, les critiques qui permettent de mettre en avant les aspects qui sont traités dans ce manuscrit et ignorés dans ces travaux et enfin le domaine d'application de ces travaux. Les approches considérées ici comme les plus génériques d'entre elles sont situées dans les lignes grisées.

Référence	Caractéristiques ou définitions	But	Outils utilisés ou proposés	Critiques	Domaine
[Simpson 2009]	CS indépendants opérationnellement [Boardman 2006] et acquis indépendamment [Eisner 1991].	Faciliter la compréhension des SdS, en réduisant la complexité cognitive et perceptuelle.	N-Squared charts	Améliore uniquement la perception humaine du SdS mais pas son comportement qui reste à la charge du modélisateur.	ingénierie des SdS
[Mahulkar 2009]	5 Caractéristiques de [Maier 1998]	Intégrer de nouvelles technologies au sein d'un système existant.	SOA	Pas d'assurance des caractéristiques d'un SdS, contrôle un SdS pour effectuer une routine et non des tâches allouées dynamiquement.	simulation interne de bâtiment maritime
[Parker 2010]	Définition de [DeLaurentis 2005], 5 caractéristiques de [Maier 1998]	Améliorer les systèmes de transport en utilisant les SdS.	New mobility hub et smart grid	Pas de formalisme de modélisation proposé, les méthodes proposées ne réunissent pas approche ascendante et descendante.	transport, ITS
[Sauser 2010]	5 caractéristiques de [Boardman 2006]	Créer une science des SdS, la systomique, faisant l'analogie avec les systèmes biologiques pour garantir "l'évolution naturelle" du SdS.	SOA, systèmes holoniques	Un formalisme de modélisation des SdS n'est pas détaillé, ici les holons respectent partiellement les caractéristiques du SdS.	ingénierie des SdS

TABLE 2.1 – Comparaison des approches de modélisation des SdS.

Référence	Caractéristiques ou définitions	But	Outils utilisés ou proposés	Critiques	Domaine
[Dauby 2011]	Définition de [Jamshidi 2008a] et connectivité des CS [Boardman 2006]	Créer une méthodologie pour comprendre les principes de gouvernance des SdS existants.	ABM, simulation de réseaux sans fil	SdS vus sous forme de SI, systèmes déjà existants désignés comme SdS, pas de contrôle direct sur les SdS.	étude des SdS
[Zhou 2011b, Zhou 2011a]	Définition de [Jamshidi 2008a], 5 caractéristiques de [Boardman 2006]	Créer une méthode de modélisation générique des SdS respectant leurs caractéristiques.	SOA, définition des interfaces et des caractéristiques des SdS.	Formulation non générique, les CS ne sont pas vraiment autonomes, le problème en exemple peut être réduit à une gestion de flux sur un circuit fixe.	systèmes de production manufacturiers
[Acheson 2012]	Pas de définition formelle	Créer un modèle de développement générique des SdS.	ABM, assesseurs en logique flou, algorithme génétique	Algorithme centralisé qui redéfinit régulièrement les interconnexions entre CS, gestion de SI uniquement.	simulation militaire
[Gezgin 2012]	5 caractéristiques de [Maier 1998] et [Boardman 2006] et définition de [Jamshidi 2008a]	Proposer un formalisme de modélisation des SdS en assurant la flexibilité et la sûreté du SdS dans un contexte critique.	SOA, graphe d'état paramétré, grammaire des graphes	Pas de convergence assurée lors de la reconfiguration, défaillance et ajout des CS non prise en compte.	simulation de lutte contre les incendies.

Table 2.2 – Comparaison des approches de modélisation des SdS (2).

Référence	Caractéristiques ou définitions	But	Outils utilisés ou proposés	Critiques	Domaine
[Gerst 2012]	Définition de [Agusdinata 2008]	Proposer une architecture, ENGAGE, pour modéliser les effets politiques climatiques au niveau local sur la consommation et production de biens, d'énergie et de pollution.	ABM, simulation multi-agents multi-niveaux	Outil de modélisation non générique, les caractéristiques des SdS ne sont pas vraiment nécessaires pour le modèle proposé.	gestion climatique globale
[Khalil 2012a]	5 caractéristiques de [Maier 1998]	Modéliser un SdS de manière générique avec ses aspects dynamiques et statiques pour piloter sa supervision et reconfiguration.	hypergraphes valués ou non	La reconfiguration des SdS n'est pas formalisée, avec un algorithme générique par exemple.	transport, ITS, projet IntraDE [int]

TABLE 2.3 – Comparaison des approches de modélisation des SdS (3).

En complément de ces travaux, il est nécessaire de présenter plus en détails deux thèses qui traitent spécifiquement de la modélisation des SdS [Khalil 2012b] et [Held 2008].

[Held 2008] pose une définition plutôt ouverte des SdS. Pour l'auteur "un SdS peut être considéré comme tel *ssi* : 1) le SdS peut être subdivisé en systèmes opérant indépendamment ; 2) le SdS ne dépend pas de tous ses éléments pour survivre (accomplir sa mission) ; 3) les systèmes dans le SdS possèdent des formes de communication ; 4) les éléments ont une mission commune." Ces quatre critères issus de la caractérisation de [Maier 1998] sont pour moi insuffisants pour capturer l'essence d'un SdS, cependant, ils apportent des précisions utiles que nous utilisons par la suite. L'auteur veut modéliser un SdS qui représente un ensemble de drones aériens qui effectuent de la visualisation et du suivi en groupe de cibles. [Held 2008] cherche à répondre à trois questions : quelles sont les métriques correctes pour un SdS ? Comment un seul modèle topologique du SdS peut être formé par ces métriques ? Comment cette représentation de SdS peut convenir aux besoins de la mission ? Les métriques sont ici des attributs quantitatifs et normalisés du SdS. [Held 2008] utilise des SMA et des ensembles structurés de métriques pour modéliser et simuler son SdS. En définitive, cette thèse s'intéresse plus à la représentation des données et à leur utilisation pour évaluer un SdS, plutôt que de se focaliser sur le contrôle dudit SdS. D'une certaine manière cette thèse est complémentaire de ce manuscrit car elle formalisme comment fixer le contenu des attributs et des capacités des différents CS d'un SdS, alors que c'est une tâche complexe qui est laissée à l'appréciation du modélisateur. Cependant il faut garder à l'esprit que la méthodologie proposée est adaptée aux SI, avec toutes les limitations que cela induit.

[Khalil 2012b] a une importance particulière pour ces travaux. En effet, nos travaux s'inscrivent dans le même projet européen InTraDE, et dans le même domaine : le transport portuaire côté terre par une flotte d'IAVs. Nous reprenons une définition très similaire des SdS basée sur la caractérisation de [Maier 1998] et nous souhaitons également contrôler un SdS lors de sa mise en place et de son fonctionnement en mode normal ou face à des incidents impliquant sa réorganisation. Ce contrôle et sa validation prennent place dans des simulations portées par la plate-forme de simulation routière SCANeRstudio. [Khalil 2012b] propose une modélisation des aspects statiques et dynamiques des SdS en respectant en permanence les caractéristiques dictées par [Maier 1998]. Cependant, l'ajout, le retrait ou la défaillance de CS donne lieu à des reconfigurations pour rendre l'accomplissement du but global possible. Ce mécanisme de reconfiguration qui est essentiel pour modéliser les aspects dynamiques du SdS est évoqué à travers un exemple très simple (retrait d'un CS défaillant du SdS), mais il n'est pas formalisé, avec un algorithme par exemple. Pour être plus précis seule la partie ascendante de la reconfiguration est formalisée, c-a-d, la détection d'une défaillance et son isolation. Dans ce manuscrit nous proposons des algorithmes génériques pour exposer de manière formelle cette reconfiguration, en particulier, la partie descendante lorsqu'un CS restructure et redistribue les tâches entre ces sous CS.

De plus [Khalil 2012b] utilise comme formalisme de modélisation les hypergraphes. L'intérêt des approches graphiques est qu'elles offrent un outils qui permet à un utilisateur d'appréhender facilement un système en déroulant une représentation visuelle de ce système. Les hyperarêtes permettent de représenter autant de CS formant un super CS. Cependant, de notre point de vue, les approches graphiques sont des bons outils de représentation pour les SdS mais sont inadaptées pour capturer toute la complexité d'un SdS. C'est particulièrement vrai pour modéliser les comportements complexes de super CS issus des comportements de leurs sous

CS. Pour juger un modèle on observe ses valeurs représentatives et prédictives. La valeur représentative du modèle se mesure en observant la quantité et la qualité des informations que nous donne le modèle du système réel. La valeur prédictive d'un modèle peut se tester en comparant l'évolution du modèle et du système réel face à un événement donné. À mon avis le modèle à base d'hypergraphes présenté dans [Khalil 2012b] est une bonne représentation des SdS mais il lui manque du pouvoir prédictif. En effet, les approches graphiques seules ne sont pas suffisantes pour retranscrire l'évolution d'un système il faut les doter d'un ensemble de règles qui dictent les changements de structure des graphes lors de la survenue d'un événement. Ce genre d'éléments n'est pas complètement formalisé dans [Khalil 2012b] et de toute manière un système de règles qui captureraient toutes possibilités comportementales d'un SdS semble fastidieux à créer. Les SMA semblent particulièrement adaptés pour modéliser ce type de comportements complexes, c'est pourquoi j'utilise un tel système pour modéliser les SdS.

Cette étude comparative de la littérature traitant de la modélisation de SdS montre que les tentatives les plus génériques de modélisation des SdS s'appuient sur les ABM. En effet, il existe de nombreuses similarités (et des différences également) entre les ABM et les SdS, telles que la propriété d'autonomie des agents et des CS.

Le comportement global d'un SdS est dérivé des interactions de ses CS. L'action collective des agents produit un comportement émergent pouvant fournir des données réalistes même à partir de modèles vraiment simples. C'est une méthode simple et versatile pour étudier les systèmes complexes non linéaires. Le degrés de réalisme ou d'abstraction est entièrement modulable (pour capturer des comportements individuels ou comprendre l'essence d'un problème). De plus les ABM peuvent révéler des propriétés quantitatives et qualitatives de systèmes réels, en servant de laboratoires computationnels en permettant de tester toute sorte d'hypothèse sous forme de simulation [DeLaurentis 2005]. Toutes ces raisons et le fait que la littérature des SMA est abondante et possède de nombreux algorithmes pour résoudre des problèmes génériques nous ont conduit à adopter un formalisme à base d'agents pour modéliser les SdS. Dans la section suivante nous présentons les SMA plus en détails en montrant les ajouts nécessaires à la modélisation de SdS.

2.3 Systèmes Multi-Agents

La définition des SdS que nous avons adopté est assez générique pour être acceptée par la communauté et assez concrète et précise pour être opérationnelle dans un contexte de modélisation et de simulation. Le formalisme que nous allons proposer pour bâtir notre modèle doit permettre d'appréhender facilement la structure et les organisations du système. Les modèles générés en utilisant ce formalisme permettent de représenter une flotte d'IAVs du projet InTraDE, qui sont des SdS dirigés par un but global.

Pour rendre facile son appréhension et pour diviser sa complexité computationnelle, nous allons découper le SdS par échelles ou par aspects indépendants en utilisant le méta-modéle multi-agents multi-niveaux IRM4MLS [Morvan 2011].

Durant la reconfiguration du SdS, causée par des changements de but global ou de l'environnement, dans les modèles de [Zhou 2011b] et [Gezgin 2012], de nouvelles configurations de SdS sont établies en appliquant des règles contenues dans les CS ou dans des obligations extérieures au système. Il n'y a pas de représentation indépendante du but global pour le décomposer et établir plusieurs plans pour l'atteindre. De même, dans ces modèles, il n'y a pas de représentation organisationnelle des groupes de CS indépendante qui permette de choisir la meilleure structure de SdS pour accomplir les plans précédents.

Dans [Held 2008], l'auteur souligne une conséquence résultant des cinq caractéristiques fondamentales des SdS. C'est le fait que tous les CS ne sont pas nécessaires pour accomplir le but global. Cela signifie que en cas d'échec, un SdS doit être capable de réallouer la mission défaillante à un CS disponible ou de créer un CS dans ce dessein. Cela induit moins l'obligation de la faisabilité d'une telle reconfiguration que la présence de mécanismes de reconfiguration qui apportent l'assurance que si cette reconfiguration est possible, elle sera effectuée.

De plus, d'après l'aspect évolutif d'un SdS, un formalisme pour pouvoir représenter un SdS doit posséder deux mécanismes : n'importe quel CS peut détecter qu'il n'est plus capable de mener à bien sa mission et, s'il sert le but global, il transmet cette information au CS qui possède l'autorité sur celui-ci et qui ensuite fait de même. L'autre mécanisme concerne ces CS défaillants par la reconfiguration du SdS avec la réallocation des missions et/ou les changements de structure. Ces mécanismes et concepts seront exposés dans le chapitre 4.

Cette section présente les outils de modélisation des SMA et comment ils sont appliqués pour représenter des systèmes complexes comme les SdS. En particulier le méta-modèle multi-niveaux IRM4MLS, qui permet de représenter des systèmes complexes à différents niveaux de granularité, sera introduit par la suite.

2.3.1 Définitions fondamentales

La plupart des auteurs s'accordent généralement à définir un SMA comme un système composé d'agents communiquant et collaborant, qui ont des objectifs (personnels ou collectifs) et des ressources pour les accomplir. La communication implique l'existence d'un espace partagé pour la supporter. Cet espace est généralement appelé environnement [Michel 2009]. Ferber a défini les agents et les SMA comme suit [Ferber 1995].

On appelle **agent** une entité physique ou virtuelle

1. qui est capable d'agir dans un environnement,
2. qui peut communiquer directement avec d'autres agents,
3. qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
4. qui possède des ressources propres,
5. qui est capable de percevoir (mais de manière limitée) son environnement,
6. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
7. qui possède des compétences et offre des services,
8. qui peut éventuellement se reproduire,
9. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

On appelle **système multi-agents** (ou SMA), un système composé des éléments suivants :

1. Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.
2. Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.
3. Un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système.
4. Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.
5. Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.
6. Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.

Cette définition présente les différents éléments d'un SMA. Cependant elle ne détaille pas son fonctionnement, pour cela on peut s'appuyer sur la définition de [Treuil 2008] qui décrit de manière très concise un SMA :

> C'est un système composé d'entités multiples ou agents qui évoluent dans un environnement, conçu comme une entité particulière, dans lequel ils sont localisés. Ces agents sont dotés d'attributs, de comportements et de capacités de perception et de communication. L'ensemble des valeurs des attributs d'une entités à un instant donné constitue l'état de cette entité, et la réunion de l'ensemble des états des entités forme l'état microscopique - ou dit plus simplement - l'état du système. Les capacités de perception des entités leur permettent de consulter un sous-ensemble de cet état microscopique, habituellement de façon localisé dans l'environnement. Les comportements sont des règles contrôlant à chaque instant l'évolution de cet état, en intervenant sur les états des entités qui les portent ou sur leur existence même (création et destruction), ainsi que sur les états et existences des autres entités intervenant dans les éventuelles actions, communications ou interactions décrites dans les comportements.

2.3.2 Architecture des agents

Il existe plusieurs types d'agents dans la littérature. De manière très générale on peut distinguer agents *réactifs* et *cognitifs*.

Un agent réactif est un agent que ne possède aucune représentation de lui même ou de son environnement. Le module de décision d'un tel agent est très limité car ses actions dépendent directement de ses perceptions. Le processus d'action prend la forme stimulus/réponse ou action/réaction. Un moyen commun de communiquer pour les agents réactifs est d'utiliser leur environnement comme support de communication, ce procédé est appelé communication indirecte ou *stigmergie* [Weyns 2004]. Les signaux émis peuvent prendre la forme de sons ou de phéromones par exemple.

Un agent cognitif est un agent qui dispose d'une représentation symbolique de lui-même et de son environnement ce qui lui permet de planifier ses actions. La collaboration au sein d'un SMA cognitif est généralement plus complexe que dans un SMA réactif. En effet les agents possèdent des mécanismes de communication directe Il existe de nombreux protocoles pour formaliser la communication entre agents, basée sur la théorie des actes du langage comme FIPA-ACL ou KQML.

À titre d'exemple nous allons présenter deux architecture d'agents cognitifs pour montrer leur complexité : les agents BDI et les MDP. Un revue plus complète des architectures d'agents peut être trouver dans [Wooldridge 2009]. Un agent "Belief-Desire-Intention" (BDI) est un agent rationnelle rationel dotée d'une représentation de son Les croyances (Belief) d'un agent sont les informations que l'agent possède sur l'environnement et sur les autres agents. Les désirs (Desire) d'un agent représentent les états de l'environnement qu'il souhaite voir réalisés. Les intentions (Intention) les actions qu'il a décidé de faire pour accomplir ses désirs. Un processus de décision markovien (MDP) est un modèle stochastique issu de la théorie de la décision et de la théorie des probabilités. Le modèle MDP peut être vu comme une chaîne de Markov à laquelle on ajoute une composante décisionnelle. Avant chaque action un agent MDP calcule une espérance de gain pour tous les états du système atteignable afin de choisir l'action possèdant la plus grain espérance de gain.

[Ferber 1995] distingue également agents *hystérétiques* et *tropistiques*, i.e., ayant ou n'ayant pas de mémoire.

Nous souhaitons résoudre des problèmes complexes sans alourdir la structure des agents ce qui nuirait à la lisibilité du SMA. Dans notre cas nous ne posons pas de contrainte sur le choix d'architecture des agents utilisés pour simuler un SdS cependant, nous voulons faire en sorte que le modèle que nous proposons permet de capturer la complexité d'un SdS et de le contrôler avec des agents très simples. Pour cela il faut déplacer la complexité en dehors de l'agent.

Deux éléments peuvent guider les SMA cognitifs : l'environnement ou les structures d'interaction entre agents (groupes). Comme l'ont souligné [Drogoul 1993, Veremme 2010] une telle résolution de problèmes est analogue à celle qui est présente dans les systèmes biologiques ou sociaux. Dans de tels systèmes, l'intelligence et le contrôle repose sur deux processus distincts : l'auto-organisation et l'émergence de capacités. Les deux prochaines sections montrent comment

ces processus peuvent être présents dans les SMA.

2.3.3 SMA centrés organisation

Les SMA sont largement utilisés pour simuler les interactions entre des entités autonomes, agissant en parallèle. Ce genre d'entités est commun dans des systèmes complexes comme les SdS. Comme le souligne [Gaud 2008] :

> Un SMA est une société organisée d'agents dans laquelle un certain nombre de phénomènes peuvent émerger comme la résultante des interactions entre agents. Cette notion d'émergence est essentielle dans les SMA, car c'est l'une des propriétés qui les rendent si aptes à modéliser les systèmes complexes.

Dans un premier temps, les SdS devraient être considérés comme un concept organisationnel, et ainsi ils peuvent être comparés aux modèles organisationnels SMA tels que Agent Groupe Rôle (AGR) [Ferber 1998, Ferber 1999]. Ce formalisme organise un système en groupes en fonction des rôles joués par les agents.

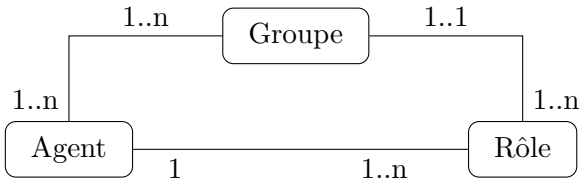

FIGURE 2.1 – Concepts Centraux d'AGR (cardinalités respectant le formalisme UML)

[Hübner 2002b, Hübner 2002a] a étudié les SMA orientés organisations. Dans leur modèle : $\mathcal{M}oise^{+}$, ils utilisent une spécification déontique pour lier les groupes et l'allocation des missions. Cette spécification a pour but d'assurer que le système va atteindre son but global, tout en garantissant l'autonomie en termes de comportement et de création d'organisations. D'une certaine manière cette approche est similaire à la nôtre. Mais dans notre approche, l'existence des groupes est fortement contrainte par la nécessité de réaliser le but global du système.

2.3.4 Simulation multi-agents

[Drogoul 1993] définit ainsi la simulation :

> On nomme simulation la démarche scientifique qui consiste à réaliser une reproduction artificielle, appelée modèle, d'un phénomène réel que l'on désire étudier, à observer le comportement de cette reproduction lorsqu'on en fait varier certains paramètres, et à en induire ce qui se passerait dans la réalité sous l'influence de variations analogues.

L'intérêt à simuler des systèmes est triple. La simulation permet de valider des modèles de systèmes réels, de faire des expérimentations de contrôle sur un système réel à un faible

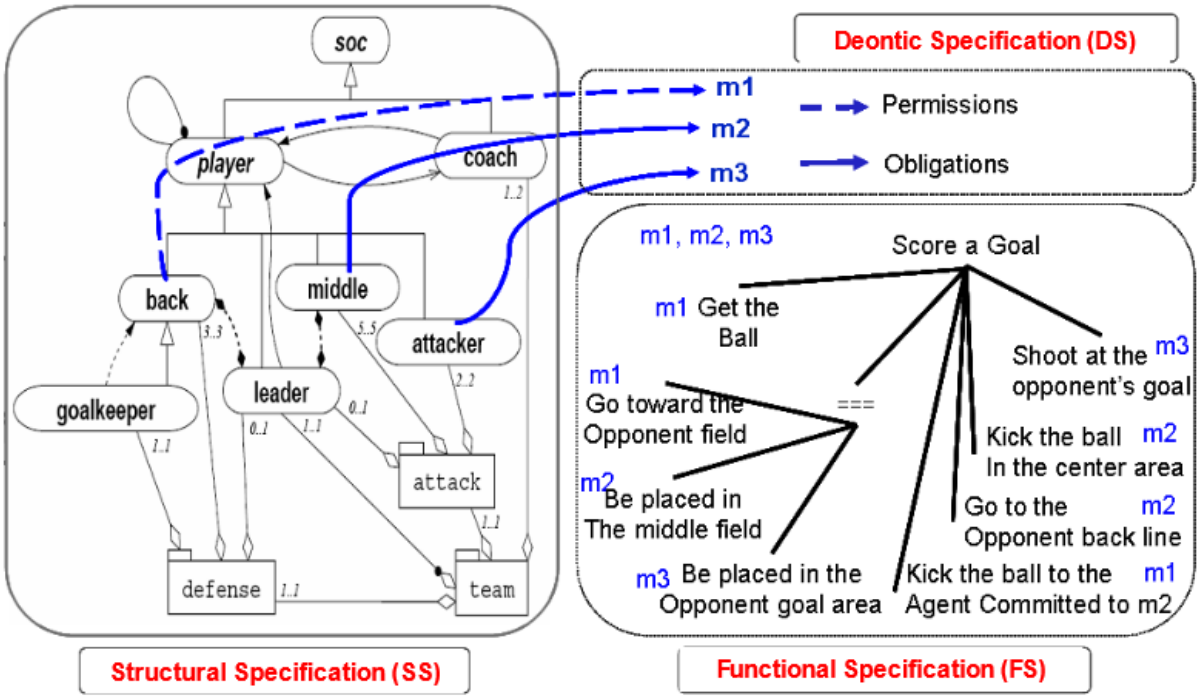

FIGURE 2.2 – Les trois éléments principaux de $\mathcal{M}oise^+$: 1) la spécification structurel qui définit les rôles, 2) la spécification fonctionelle qui définit les buts et 3) la spécification déontique qui associe rôle et but à accomplir.

coût comparé à l'expérimentation sur le système réel, et de pouvoir manipuler les conditions d'évolution du systèmes en changeant les paramètres de la simulation à volonté.

Il y a deux raisons majeures qui nous poussent à utiliser les SMA modéliser et simuler des systèmes complexes :

1. il est difficile de représenter un système complexe dans son ensemble à l'aide d'un modèle analytique et
2. les SMA permettent de modéliser les interactions entre composants du système grâce aux comportements des agents qui peuvent être estimés.

De plus, les SMA ont été utilisés avec succès pour modéliser et simuler des systèmes complexes dans de nombreux domaines comme la biologie, la sociologie ou encore l'écologie [Ferber 1995, Epstein 2006, Gilbert 2007, Railsback 2011, Resnick 1994, Treuil 2008].

La définition de [Drogoul 1993] montre les trois éléments majeures d'une simulation :

1. le système réel modélisé,
2. le modèle théorique qui représente ce système et
3. le simulateur qui sert de base à l'exécution de ce modèle au cours d'une simulation.

Les trois aspects d'une simulation relevés par Drogoul correspondent, dans une approche multi-agents, à trois types d'agents différents [Drogoul 2003] :

– les **agents réels** qui sont les entités pouvant être observées dans le système (acteurs),

- les **agents conceptuels** qui sont une formalisation des agents réels dans les SMA,
- les **agents informatiques ou computationnels** sont les implémentations des agents conceptuels exécutables dans un environnement de simulation (sur ordinateur).

Dans ce manuscrit on s'intéresse uniquement aux agents conceptuels et computationnels. Dans la section suivante nous allons voir qu'elles on les architectures possibles pour ces agents.

2.4 Modélisations à base d'agents multi-niveaux

[Morvan 2012a] donne la définition suivante des moélisation à base d'agents multi-niveaux (ML-ABM) :

> Une intégration d'ABMs hétérogènes, représentant des points de vue complémentaires, appelés niveaux, d'un même système.

[Fichwick 1997] définit ainsi une simulation multi-niveaux :

> La simulation multi-niveaux est un type particulier de simulation où le modèle proposé du système intègre différents niveaux d'abstraction (au moins deux) et où les outils nécessaires à son exécution permettent de faire cohabiter ces différents niveaux d'abstraction au sein d'une même exécution et d'assurer la transition dynamique entre eux en fonction de contraintes définies (dépendantes du modèle ou du cadre expérimental).

Quand on évoque différents niveaux de représentation on pense tout d'abord au multi-échelle ou multi-résolution On peut définir une **échelle** comme une mesure du temps, de l'espace ou de la granularité d'un environnement. Un **niveau** est ici vu comme un concept plus général qui permet de définir un point de vue sur un système. Ainsi, différents niveaux peuvent être à la même échelle mais concerner des aspects différents d'un même système. Par exemple, un IAV peut être considéré selon deux points de vue à la même échelle spatio-temporelle : dans un premier niveaux on considère la fonction de déplacement du véhicules et dans l'autre sa fonction d'auto-diagnostic. Un tel découpage fait sens uniquement si les différents niveaux concernent des aspects relativement indépendants.

2.4.1 Plate-formes et moteurs de simulations multi-niveaux

Un certain nombre de méta-modèles et de moteurs de simulation dédiés aux ML-ABM a été proposé dans la littérature. Dans cette section nous présentons les principaux méta-modèles, plate-formes et moteurs de simulations multi-niveaux que l'on peut trouver dans la littérature. Cette section est basée sur une revue très complète de la littérature et les enjeux des simulation multi-agents multi-niveaux [Morvan 2012a].

HLA [HLA , Petty 2002, Ham 2006, Chen 2011](High Level Architecture) est une architecture orientée objet qui a pour but la modélisation et la simulation. HLA est utilisé pour réunir plusieurs simulations à l'implémentation hétérogène et à les faire fonctionner et communiquer ensemble. HLA n'est pas un moteur de simulations multi-niveaux et en raison de la mise en commun d'implémentations hétérogènes il est moins efficace qu'un cadre de simulation unifié. Cependant, HLA peut permettre de coupler des simulations dont les modèles sont à différents niveaux de représentation.

HLA-Repast [Scerri 2010] est un modèle à base d'agents qui veut offrir un cadre de simulation global intégrant plusieurs simulations basé sur HLA. HLA-Repast comprend des données partagées plus ou moins globalement et des simulations encapsulées dans des modules. Ce modèle se focalise sur le partage équitable de l'accés aux données partagées entre les différents modules en fonction de l'activité des agents présents dans ces modules.

GEAMAS [Marcenac 1998](GEneric Architecture for MultiAgent Simulation) est un des tous premiers cadres de simulation pour les ML-ABM, il intègre les trois niveaux classiques de description (micro, méso, macro). Les niveaux micro et macro représentent respectivement les points des agents et du système tandis que le niveau méso (ou du milieu) représente une agrégation d'agents dans un contexte spécifique. La communication entre niveaux se fait de manière asynchrone. **GEAMAS-NG** est une amélioration de ce cadre qui propose des outils pour détecter et réifier les phénomènes émergent, tout en leur donnant un statut particulier qui modifie l'évolution des niveaux concernés [David 2011].

CRIO [Gaud 2007, Gaud 2008] (Capacity Role Interaction Organization) est un méta-modèle organisationnel utilisé pour la ML-ABM. Ce méta-modèle est basé sur le concept d'holon [Koestler 1967]. Les holons sont des entités récursives dotées d'une structure contrainte. La plateforme multi-agents **Janus** basée sur le langage Java, a été utilisé pour implémenter ces simulations. Janus contient nativement le concept d'holon. Une des applications développées traite la simulation multi-échelles de foules de piétons.

PADAWAN [Picault 2011] (Pattern for Accurate Design of Agent Worlds in Agent Nests) est un méta-modèleABM basé sur la représentation des interactions : IODA (Interaction-Oriented Design of Agent simulations) [Kubera 2008], qui offre un cadre de simulation simple et élégant. PADAWAN fournit des agents avec plusieurs spécificités qui nous intéresse : les agents peuvent contenir des environnements où peuvent être situés d'autres agents et les agents possèdent plusieurs "visages" et une partie centrale. Le "visage" d'un agent est une représentation situé de cet agent dans un environnement donné permettant à un agent d'être présent dans plusieurs environnements simultanément (voir 3.2.2).

GAMA [Drogoul 2013, Vo 2012a, Vo 2012b] est une plateforme ABM dotée d'un langage de modélisation dédié **GAML**, offrant des capacités de modélisation multi-niveaux. GAMA se focalise sur la représentation spatiale de ces agents en intégrant de nombreux modèles pour les agents (géométrie) et pour les environnements (topologies, réseaux, grilles, données géographiques réalistes) en 2D et en 3D. De ce point de vue GAMA est la plateforme la plus avancée intégrant le multi-niveaux. La notion de niveau n'apparaît pas explicitement mais le concept d'espèce définit des attributs et des comportements d'une classe d'un même type d'agent et la structure du modèle, c-à-d, comment les espèces sont imbriquées les unes dans les autres. De ce fait les niveaux dans GAMA font obligatoirement référence à des niveaux de représentation des organisations. Les agents agrégés dans des groupes existants voient leurs attributs et comportements modifiés mais ne génèrent pas de pas de nouvelles structures inédites.

AA4MM [Camus 2013](Agent and Artifact for Multi-Modeling) est un méta-modèle de multi-modélisation ou de couplage de modèles multi-agents. Des niveaux encapsulés dans des agents interagissent à travers des artefacts Ce méta-modèle étend des méta-modèles existants, voir [Bonneaud 2008], distribuant l'ordonnancement entre niveaux.

Dans [Michel 2004] l'auteur présente les différents modèles temporels utilisés pour la représentation des systèmes dynamiques : les modèles continus, les modèles discrets et les modèles événementiels. Les variables d'état du système changent de valeur de manière continue dans un modèle continu. L'évolution des variables d'état du sysème se fait par temps de temps continu dans un modèle discret. Dans les modèles événementiels, les variables d'état du système changent de manière discrète (instantaée) à des instants précis qui sont appelés événements. Dans le cadre de l'exécution d'une simulation on considère les modèles dont la temporalité est mesurée par pas de temps continus ou par la survenue d'événements discrets. Jusqu'ici nous avons évoqué principalement des modèles dont la temporalité est mesurée par pas de temps continus.

DEVS [Zeigler 2000] est un cadre générique de simulation à base d'évènements, qui a été étendu pour supporter les ABM [Muller 2009]. Une revue de la littérature sur les extensions multi-niveaux DEVS peut être trouvée dans [Duhail 2013].

ML-DEVS [Uhrmacher 2007] est une extension de DEVS qui permet la simulation de modèles multi-échelles (et non seulement des modèles couplés dans lesquels le comportement d'un modèle est déterminé par les comportements de ses sous-modèles). Deux types de relations enter niveaux sont définis : propagation d'information et activation d'évènement. Cependant, ML-DEVS se focalise sur la modélisation multi-échelles et ainsi, ne permet que des modèles purement hiérarchiques où les graphes d'interactions sont vus comme des arbres [Maus 2008].

Le modèle **Seck & Honig** [Seck 2012] est une autre extension de DEVS qui permet de créer des modèles de simulations multi-niveaux qui en sont pas forcément couplés de manière hiérarchique. Le couplage entre niveaux se fait au travers modèles DEVS réguliers.

IRM4MLS [Morvan 2012b, Morvan 2011] (Influence Reaction Model for Multi-Level Simulation) est une extension multi-niveaux de **IRM4S** (Influence Reaction Model for Simulation) [Michel 2007b], est un méta-modèle ABM basé sur le modèle Influence Réaction qui voit l'action des agents qui un processus en deux étapes. Dans la plupart des ABM l'action des agents produit immédiatement des modifications dans leur environnement, (1) les agents produisent des "influences" qui peuvent figurer des décisions individuelles d'après leur état interne et leurs perceptions (2) le système "réagit" en calculant les conséquences de ces influences [Ferber 1996]. Les relations d'influence et de réaction sont spécifiés par un digraphe.

Une des raisons pour lesquelles nous adoptons IRM4MLS comme base de nos travaux et le fait que les interactions entre agents ou entre niveaux ne sont pas contraintes. Il en résulte que l'ordonnancement de l'exécution des agents et le calcul du résultat des actions est géré de manière implicite. De même il n'y a pas de contraintes sur les temporalités des différents niveaux niveaux. Ainsi certains peuvent être ordonnancés par pas de temps constants et d'autres selon l'occurrence d'évènements discrets. De plus la résolution d'actions conccurentes simultanées est aussi implicitement traitée.

2.4.2 Passage dynamique entre niveaux de représentation

Les modèles de simulation multi-niveaux permettent de représenter un système à différents niveaux d'abstraction. Cela permet de "zoomer" ou "dézoomer" pendant l'exécution de la simulation pour voir plus en détails les éléments de base du modèle ou au contraire appréhender

de manière plus global un phénomène. Un autre avantage du passage dynamique entre niveaux de représentation repose sur l'économie de ressources informatiques en ne représentant que ce qui est utile à un moment donné.

[Davis 1993] est un des pionniers dans ce domaine et a travaillé sur la modélisation multi-résolution dans un cadre militaire. Les premiers modèles proposés, très simples, représentaient des bataillons à l'aide de rectangle uniforme avec différentes variables pour les caractériser et ils permettaient de passer à une vue détaillée des soldats qui constituent ces bataillons. [Davis 1993] a proposé un outils pour mesurer la qualité des simulations intégrant un niveau de détail qui peut varier dynamiquement. Cet outil de mesure est appelé la *cohérence*, il est décrit plus en détail dans le chapitre 3.

[Servat 1998] est un des premiers travaux sur le sujet dans le domaine des ABM. Cet article présente le projet RIVAGE qui propose de modéliser la formation de flaques, l'érosion et la formation de ravines. La modélisation proposé est découpée entre niveaux microscopique et macroscopique. Au niveau microscopique les agents qui représentent des gouttes d'eau sont considérées avec leur position et leur direction. Ces agents sont regroupés en fonction de leur similiraté pour représenter les flaques ou les ravines en formation présents au niveau macroscopique.

[Yilmaz 2004, Yilmaz 2005] traitent des multi-modèles en particulier de la mise à jour dynamique et du remplacement d'un modèle par un autre. Les modèles proposés sont des ABM utilisant DEVS. Même si des mécanismes de changement de modèle plus ou moins souples sont proposés, la limitation de l'approche proposée repose sur le fait que les systèmes modélisés sont uniquement représentés dans un modèle actif à la fois.

[El hmam 2006] a présenté un modèle multi-agents multi-niveaux hybride permettant de simuler la gestion de traffic de flux routier sur un réseau autoroutier de grande taille. Le modéle macroscopique est appliqué aux tronçons d'autoroute où le comportement des automobilistes est prévisibles. Ce modèle représente les véhicules comme des déplacements de fluides dans des tuyaux. Le modèle microscopique est appliqué aux zones où le comportement des automobilistes influencent le traffic à proximité (à proximité d'une intersection par exemple). Dans ce modèle chaque véhicule est représenté par un agent individuel. Ce modèle propose un mécanisme de couplage générique entre modèles.

[Navarro 2011] a proposé une modélisation multi-résolution à base d'agents où le niveau de détail de représentation des entités simulés est un des paramètres qui varie au cours de la simulation. Ce modèle a été utilisé pour simuler une foule de piétons où les individus sont regroupés en fonction de leur similarité physique aussi bien que psychique. Ce modèle présente les piétons au niveau microscopique et des foules au niveau macroscopique. Le passage d'un niveau à l'autre est permis par l'agrégation et la désagrégation des agents. La qualité de la simulation (cohérence) est comparée au gain en terme de ressources informatiques utilisées. IL apparaît une limite qui dégrade la qualité de la simulation en même temps que la taille des foules augmentent : les foules ont le même comportement que les piétons. Ce défaut a été en partie corrigé dans [Navarro 2013] où un niveau mésoscopique a été ajouté. Ce niveau permet de représenter uniquement certaines parties ou processus des agents qui sont partiellement agrégés.

2.5 Conclusion

Après un tour d'horizon de cette thématique on peut constater que la modélisation des SdS est encore un sujet en construction. Certains outils ou certaines techniques permettent de créer des modèles. Cependant il reste encore plusieurs points clés à préciser et à formaliser qui permettent de positionner notre travail doctoral. Tout en gardant à l'esprit que notre travail se veut généraliste, modulable et réutilisable.

1. **Comment représenter les CS quand ceux-ci peuvent être considérés à la fois comme des individus et comme des organisations ?** Dans ce chapitre nous avons posé une définition solide des SdS basé sur 5 caractéristiques fondamentales. Ces caractéristiques qui concernent les CS ou le SdS dans son intégralité sont assez formelles pour déduire les caractéristiques et la structure des entités qui doivent représenter ces CS. Entre autre chose elles mettent en avant le besoin d'une spécification organisationnelle des CS et le besoin des concepts explicites de niveau et de capacité. Nous faisons le choix de travailler avec un formalisme multi-agents car ces derniers sont dotés de caractéristiques similaires à celles des CS, comme leur autonomie.
2. **Comment la représentation d'un SdS peut-elle être rattachée à la réalisation d'un objectif ?** Pour répondre à cette question il faut élaborer des algorithmes qui gèrent la structure du SdS au niveau des CS, pour que celui-ci soit toujours en mesure d'atteindre son but malgré un environnement dynamique. Ces algorithmes s'appuient sur une spécification fonctionnelle et la notion de capacité évolutive des CS et groupes de CS.
3. **Quelles sont les caractéristiques que devraient présenter les outils de modélisation et de simulation des SdS ?** Il existe déjà des outils de modélisation multi-niveaux, cependant ils ne présentent pas l'ensemble des caractéristiques suivantes : une représentation récursive des agents, des interactions possibles entre niveaux, des mécanismes d'auto-organisation des systèmes et potentiellement une représentation du niveau de détail dynamique. Cependant, ces caractéristiques sont permises par le méta-modèle IRM4MLS qui se veut très généraliste. Ainsi en utilisant IRM4MLS nous pouvons réunir les caractéristiques suffisantes pour modéliser fidèlement un SdS, qui dépassent celles minimales requises dans certains cas.

Dans le chapitre suivant nous présentons plus en détails le méta-modèle multi-agents multi-niveaux IRM4MLS, ce qu'il permet de modéliser, comment il est adapté à la simulation et en particulier comment il permet d'économiser des ressources informatiques lors de la simulation.

Chapitre 3

Contribution au développement d'un modèle multi-niveaux : IRM4MLS

Sommaire

3.1 Le méta-modèle IRM4MLS

Après un état de l'art où nous avons introduit les SdS et les SMA, ce chapitre va présenter les outils que nous utilisons pour modéliser les SdS. Dans la première section de ce chapitre, nous présentons le méta-modèle multi-agents multi-niveaux IRM4MLS qui va servir de base pour développer nos outils de modélisation des SdS. IR4MLS est tiré du modèle IR4MS s'appuyant sur le principe d'Influence/Réaction et étant spécifiquement adapté à la simulation [Michel 2007b, Michel 2007a]. Nous allons détailler son fonctionnement et comment il est utilisé pour simuler des SMA multi-niveaux. Dans la seconde section nous démontrons l'efficacité d'IRM4MLS pour modéliser et simuler toute instance valide de SMA multi-niveaux et nous présentons un outil qui permet de tirer avantage de la possibilité pour les agents d'être présents dans plusieurs niveaux à la fois. Enfin, dans la troisième section nous présentons les différents mécanismes qui permettent de régler le niveau de détails d'une simulation pour établir un compromis entre la qualité de la simulation et les ressources pour l'exécuter.

3.1.1 Spécifications des niveaux et leurs interactions

Un modèle multi-niveaux est défini par un ensemble de niveaux, L, et une spécification des relations entre ceux-ci. Deux types de relations sont considérées dans IRM4MLS : l'influence (les agents d'un niveau l sont capables de produire des influences dans un niveau $l' \neq l$) et la perception (les agents d'un niveau l sont capables de percevoir l'état dynamique d'un niveau $l' \neq l$). Ces relations sont représentées par des graphes orientés notés respectivement $< L, E_I >$ et $< L, E_P >$, où E_I et E_P sont deux ensembles d'arcs, *i.e.*, des paires ordonnées d'éléments de L. Les relations d'influence et de perception dans un niveau sont systématiques et ne sont donc pas spécifiées dans E_I et E_P (cf. eq. 3.1 et 3.2). *E.g.*,$\forall l, l' \in L^2$, si $E_P = \{ll'\}$ alors les agents de l sont capables de percevoir les états dynamiques de l et l' et les agents de l' seulement celui de l'.

La relation de perception représente la capacité, pour les agents d'un niveau, d'être "conscient" de l'existence d'autres niveaux de représentation du système dans lequel ils interagissent. Ainsi dans un modèle composé d'agents réactifs, $E_P = \emptyset$. Par ailleurs, E_P représente ce que les agents sont capables de percevoir, non ce qu'ils perçoivent effectivement : cela est défini par une fonction de perception, propre à chaque agent.

Les voisinages entrants et sortant dans $< L, E_I >$ (respectivement $< L, E_P >$) sont notés N_I^- et N_I^+ (resp. N_P^- et N_P^+) et sont définis comme suit :

$$\forall l \in L, N_I^-(l) \text{ (resp. } N_P^-(l)) = \{l\} \cup \{l' \in L : l'l \in E_I \text{ (resp. } E_P)\}, \tag{3.1}$$

$$\forall l \in L, N_I^+(l) \text{ (resp. } N_P^-(l)) = \{l\} \cup \{l' \in L : ll' \in E_I \text{ (resp. } E_P)\}, \tag{3.2}$$

E.g., $\forall l, l' \in L^2$ si $l' \in N_I^+(l)$, alors l'environnement et les agents de l sont capables de produire des influences dans le niveau l' ; réciproquement, si $l \in N_I^-(l')$, alors l' peut être influencé par l.

3.1.2 Populations d'agents et environnements

L'ensemble des agents du système au temps t est dénoté $A(t)$. $\forall l \in L$, l'ensemble des agents appartenant au niveau l à t est noté $A_l(t) \subseteq A(t)$. Un agent appartient à un niveau si et seulement si un sous-ensemble de son état physique ϕ_a appartient à l'état du niveau :

$$\forall a \in A(t), \forall l \in L, a \in A_l(t) \text{ ssi } \exists \phi_a^l(t) \subseteq \phi_a(t) | \phi_a^l(t) \subseteq \sigma^l(t). \tag{3.3}$$

Ainsi, un agent appartient à zéro, un ou plusieurs niveaux et un environnement peut appartenir à différents niveaux, comme indiqué dans la fig. 3.1.

Il est important de voir qu'ici un environnement n'a pas la signification que l'on peut lui donner dans la plupart des modèles agents. En effet ici, un environnement désigne un ensemble de lois qui régissent les actions réalisables par les agents présents dans un niveau. Le support des actions des agents se trouve dans les niveaux. Par exemple si on modélise le déplacement de voitures sur une route avec le modèle IRM4MLS sur un seul niveau, les agents seront les voitures, les lois physiques qui régissent les déplacements des voitures seront présentes dans l'environnement du niveau et le niveau contiendra toutes les caractéristiques qui seront liées à ces lois pour déterminer le résultat des déplacements des voitures, comme l'adhérence de la route, la présence de nids de poules ou encore les dimensions de la route.

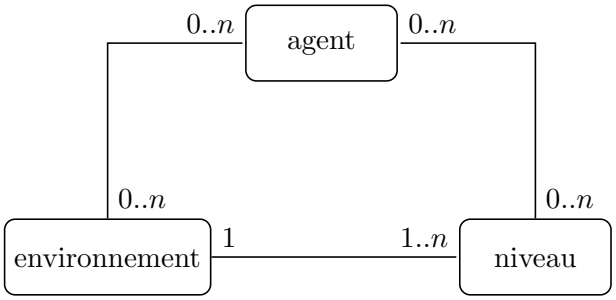

FIGURE 3.1 – Concepts centraux de IRM4MLS (les cardinalités sont spécifiées à la manière d'UML)

3.1.3 Production d'influences

Contrairement à un certain nombre de SMA où l'action des agents produit immédiatement une modification de leur environnement, les modèles respectant le principe d'Influence/Réaction voient la résolution des actions se dérouler en deux temps. D'abord tous les agents produisent des influences qui représentent leur tentative d'effectuer une action. Ensuite, la réaction de toute ces influences combinées est calculée et leur résultat est retranscrit dans l'environnement. Dans IRM4MLS c'est niveau par niveau que les influences sont produites et collectées.

L'état dynamique d'un niveau $l \in L$ à t, noté $\delta^l(t) \in \Delta^l$, est un tuple

$$\delta^l(t) =< \sigma^l(t), \gamma^l(t) >, \tag{3.4}$$

où $\sigma^l(t) \in \Sigma^l$ et $\gamma^l(t) \in \Gamma^l$ représentent respectivement les ensembles des propriétés environnementales et des influences de l.

Le comportement d'un agent $a \in A_l$ est défini par une fonction :

$$Behavior_a^l : \prod_{l_P \in N_P^+(l)} \Delta^{l_P} \mapsto \prod_{l_I \in N_I^+(l)} \Gamma^{l_I\prime}. \tag{3.5}$$

Cette fonction est décrite comme une composition de fonctions. Comme deux types d'agent sont considérés (des agents *tropistiques*, *i.e.*, sans mémoire et des agents *hystérétiques*, *i.e.*, possédant un état interne), deux types de fonctions comportementales sont définies [Ferber 1995].

Un agent *hystérétique ha* dans un niveau l agit selon son état interne et des percepts construits par une fonction de perception. Ainsi sa fonction comportementale est définie comme :

$$Behavior_{ha}^l = Decision_{ha}^l \circ Memorization_{ha} \circ Perception_{ha}^l, \tag{3.6}$$

où

$$Perception_{ha}^l : \prod_{l_P \in N_P^+(l)} \Delta^{l_P} \mapsto \prod_{l_P \in N_P^+(l)} P_{ha}^{l_P}, \tag{3.7}$$

$$Memorization_{ha} : \prod_{l \in L | ha \in A_l} \prod_{l_P \in N_P^+(l)} P_{ha}^{l_P} \times S_{ha} \mapsto S_{ha}, \tag{3.8}$$

$$Decision^{l}_{ha} : S_{ha} \mapsto \prod_{l_I \in N^{+}_{I}(l)} \Gamma^{l_I\prime}. \tag{3.9}$$

La fonction de mémorisation n'est pas spécifique à un niveau : IRM4MLS suppose qu'un agent peut avoir plusieurs corps (dans différents niveaux), mais un seul esprit, ou en d'autres termes, un seul état interne.

Un agent *tropistique ta* dans un niveau l agit seulement selon des percepts :

$$Behavior^{l}_{ta} = Decision^{l}_{ta} \circ Perception^{l}_{ta}, \tag{3.10}$$

où $Perception^{l}_{ta}$ suit la définition de l'eq. 3.7 et

$$Decision^{l}_{ta} : \prod_{l_P \in N^{+}_{P}(l)} P^{l_P}_{ta} \mapsto \prod_{l_I \in N^{+}_{I}(l)} \Gamma^{l_I\prime}. \tag{3.11}$$

L'environnement ω d'un niveau l produit des influences selon une fonction

$$Natural^{l}_{\omega} : \Delta^{l} \mapsto \prod_{l_I \in N^{+}_{I}(l)} \Gamma^{l_I\prime}. \tag{3.12}$$

Une fois les influences produites, les interactions entre niveaux n'importent plus. La fonction de réaction aux influences d'un niveau l, $Reaction^{l}$ est donc définie classiquement :

$$Reaction^{l} : \Sigma^{l} \times \Gamma^{l\prime} \mapsto \Delta^{l}. \tag{3.13}$$

ici $Reaction^{l}$ est la fonction de réaction propre à chaque niveau.

3.1.4 Modèles de simulation pour IRM4MLS

Dans cette section deux modèles de simulation pour IRM4MLS sont proposés. Le premier est directement basé sur IRM4MLS. Il suppose que tous les niveaux ont la même dynamique temporelle. Le second a une vision plus générale, mais est aussi plus compliqué et coûteux en temps de calcul. Ces modèles sont compatibles avec les différentes méthodes classiques d'évolution du temps (d'évènement à événement ou pas de temps fixes) utilisées dans la simulation multi-agents. Dans la suite, t_0 et T désignent respectivement le premier et dernier temps de simulation.

Un modèle de simulation simple

Dans cette section, un modèle doté d'une seule dynamique temporelle est introduit. Comme il n'y a pas de problème de synchronisation, ce modèle est très similaire à celui de IRM4S [Michel 2007b, Michel 2007a]. $HA(t)$ et $TA(t)$ désignent respectivement l'ensemble des agents hystérétiques et tropistiques dans le système.

Premièrement, les sous-fonctions comportementales sont exécutées pour chaque agent :

$$\forall l \in L, p_a(t) =< Perception^{l}_{a}(< \delta^{l_P}(t) : l_P \in N^{+}_{P}(l) >) : a \in A_l(t) >, \tag{3.14}$$

$$\forall a \in \mathrm{HA}(t), s_a(t+dt) = Memorization_a(p_a(t)), \tag{3.15}$$

$$\forall l \in L, \forall a \in \mathrm{HA}_l(t), < \gamma^{l_I\prime}_{a}(t) : l_I \in N^{+}_{I}(l) >= Decision^{l}_{a}(s_a(t+dt)), \tag{3.16}$$

$$\forall l \in L, \forall a \in \mathrm{TA}_l(t), < \gamma_a^{l_I\prime}(t) : l_I \in N_I^+(l) >= Decision_a^l(p_a(t)). \tag{3.17}$$

Ensuite, les influences environnementales sont produites :

$$\forall l \in L, < \gamma_\omega^{l_I}(t) : l_I \in N_I^+(l) >= Natural_\omega^l(\delta^l(t)). \tag{3.18}$$

L'ensemble des influences temporelles dans un niveau $l \in L$ au temps t sont définies comme suit :

$$\gamma^{l\prime}(t) = \{\gamma^l(t) \bigcup_{l_I \in N_I^-(l)} \gamma_\omega^{l_I\prime}(t) \bigcup_{a \in A_{l_I}} \gamma_a^{l_I\prime}(t)\}. \tag{3.19}$$

Finalement, le nouvel état du système peut être calculé :

$$\forall l \in L, \delta^l(t + dt) = Reaction^l(\sigma^l(t), \gamma^{l\prime}(t)). \tag{3.20}$$

L'algorithme 1 résume le modèle de simulation.

Algorithm 1: Modèle de simulation simple d'IRM4MLS

Input: $< L, E_I, E_P >, A(t_0), \delta(t_0)$
Output: $\delta(T)$

$t = t_0$;
while $t \leq T$ **do**
  foreach $a \in A(t)$ **do**
    $p_a(t) =< Perception_a^l(< \delta^{l_P}(t) : l_P \in N_P^+(l) >) : a \in A_l >$;
    if $a \in HA(t)$ **then**
      $s_a(t + dt) = Memorization_a(p_a(t))$;
    end
  end
  foreach $l \in L$ **do**
    $< \gamma_\omega^{l_I\prime}(t) : l_I \in N_I^+(l) >= Natural_\omega^l(\delta^l(t))$;
    foreach $a \in HA_l(t)$ **do**
      $< \gamma_a^{l_I\prime}(t) : l_I \in N_I^+(l) >= Decision_a^l(s_a(t + dt))$;
    end
    foreach $a \in TA_l(t)$ **do**
      $< \gamma_a^{l_I\prime}(t) : l_I \in N_I^+(l) >= Decision_a^l(p_a(t))$;
    end
  end
  foreach $l \in L$ **do**
    $\gamma^{l\prime}(t) = \{\gamma^l(t) \bigcup_{l_I \in N_I^-(l)} \gamma_\omega^{l_I\prime}(t) \bigcup_{a \in A_{l_I}} \gamma_a^{l_I\prime}(t)\}$;
    $\delta^l(t + dt) = Reaction^l(\sigma^l(t), \gamma^{l\prime}(t))$;
  end
  $t = t + dt$;
end

Un modèle de simulation doté de dépendances entre dynamique temporelle de différents niveaux

Dans cette section, un modèle de simulation doté de dépendances entre les dynamiques temporelles des différents niveaux est introduit. Par la suite, t^l et $t^l + dt^l$ désignent respectivement le temps courant et le prochain pas de temps d'un niveau $l \in L$. De plus $t =< t^l : l \in L >$

and $t + dt =< t^l + dt^l : l \in L >$ désignent respectivement l'ensemble des temps courants et le prochain pas de temps pour tous les niveaux. Il est nécessaire d'introduire des règles qui encadrent la production des influences et des réactions et le calcul des réactions. Ces règles se basent en premier sur le *principe de causalité.*

– Un agent ne peut pas percevoir le futur, *i.e.*,

$$\forall l \in L, l_P \in N_P^+(l) \text{ est perceptible par } l \text{ si } t^l \geq t^{l_P}, \tag{3.21}$$

– un agent ou un environnement ne peut pas influencer le passé, *i.e.*,

$$\forall l \in L, l_I \in N_I^+(l) \text{ peut être influencé par } l \text{ si } t^l \leq t^{l_I}. \tag{3.22}$$

Cependant, le principe de causalité n'est pas suffisant pour assurer un bon agencement. Un *principe de cohérence* doit aussi guider la conception du modèle de simulation, pour éviter le calul inutile de perceptions qui seront réécrites avant d'être considérées durant la production d'influences des agents :

– un agent peut seulement percevoir les derniers états dynamiques disponibles, *i.e.*,

$$\forall l \in L, l_P \in N_P^+(l) \text{ est perceptible par } l \text{ si } t^l < t^{l_P} + dt^{l_P}, \tag{3.23}$$

– comme un agent hystérétique peut appartenir à plus d'un niveau, Son état interne doit être calculé pour le prochain pas de temps où il sera utilisé, *i.e.*,

$$\forall l \in L, s_a(t_a + dt_a) = Memorization_a(p_a(t^l)), \tag{3.24}$$

Tel que

$$\begin{gathered} t_a + dt_a = t^l + dt^l | \forall t^{l\prime} + dt^{l\prime}, t^l + dt^l \geq t^{l\prime} + dt^{l\prime} \\ \Rightarrow t^l + dt^l = t^{l\prime} + dt^{l\prime} \wedge a \in A_l, \end{gathered} \tag{3.25}$$

– un agent ou un environnement peut influencer un niveau selon son dernier état, *i.e.*,

$$\forall l \in L, l_I \in N_I^+(l) \text{ peut être influencé par } l \text{ si } t^l + dt^l > t^{l_I}, \tag{3.26}$$

– la réaction doit être calculée pour le prochain pas de temps de la simulation, *i.e.*,

$$\forall l \in L, Reaction^l \text{ est calculée si } t^l + dt^l \in min(t + dt). \tag{3.27}$$

De plus, un *principe d'utilité* doit aussi être appliqué pour optimiser l'éxécution de la simulation en minimisant le nombre d'étapes de calcul :

– les perceptions doivent être calculées en une seule fois, *i.e.*,

$$\begin{gathered} \forall l \in L, \forall a \in A_l, Perception_a^l \text{ est calculée} \\ \text{si } \forall l_P \in N_P^+(l), t^l \geq t^{l_P}. \end{gathered} \tag{3.28}$$

– de même que les influences, *i.e.*,

$$\begin{gathered} \forall l \in L, Natural_\omega^l \text{ and } \forall a \in A_l, Decision_a^l \text{ sont calculées} \\ \text{si} \forall l_I \in N_I^+(l), t^l \leq t^{l_I} \vee t^l + dt^l < t^{l_I} + dt^{l_I}. \end{gathered} \tag{3.29}$$

Il est facile de montrer que la règle définie dans l'eq. 3.28 inclut la règle définie dans l'eq. 3.21. De plus, la règle définie dans l'eq. 3.27 implique la règle définie dans l'eq. 3.23.

Selon l'eq. 3.29, les influences ne sont pas nécessairement produites à chaque pas de temps par un niveau l pour un niveau $l_I \in N_I^+(l)$. Ainsi, une fonction c_I, définit la production des influences à partir des règles définies par les eq. 3.26 et 3.29 :

$\forall l, \in L, \forall l_I \in N_I^+(l)$,

$$c_I(l, l_I) = \begin{cases} \gamma^{l_I\prime}(t^{l_I}) & \text{si} \quad t^l \leq t^{l_I} \wedge t^l + dt^l > t^{l_I} \\ \emptyset & \text{sinon.} \end{cases} \tag{3.30}$$

Le modèle de simulation peut être défini comme suit. Premièrement, si la condition définie dans l'eq. 3.28 est respectée, les agents construisent leurs perceptions et de manière consécutive les agents hystérétiques calculent leurs prochain état interne :

$\forall a \in A(t)$,

$$p_a(t^l) =< Perception_a^l(< \delta^{l_P}(t^{l_P}) : l_P \in N_P^+(l) >) : l \in L_P >, \tag{3.31}$$

$$s_a(t_a + dt_a) = Memorization_a(p_a(t^l)) \text{ si } a \in \text{HA}(t), \tag{3.32}$$

avec $L_P = \{l \in L : a \in A_l(t) \wedge \forall l_P \in N_P^+(l), t^l \geq t^{l_P}\}$.

Ensuite, si la condition définie dans l'eq. 3.29 est respectée, les agents et environnements produisent des influences :

$\forall l \in L_I$,

$$< c_I(l, l_I) : l_I \in N_I^+(l) > = Natural_\omega^l(\delta^l(t^l)), \tag{3.33}$$

$$\forall a \in \text{HA}_l, < c_I(l, l_I) : l_I \in N_I^+(l) > = Decision_a^l(s_a(t_a + dt_a)), \tag{3.34}$$

$$\forall a \in \text{TA}_l, < c_I(l, l_I) : l_I \in N_I^+(l) > = Decision_a^l(p_a(t^l)), \tag{3.35}$$

avec $L_I = \{l \in L : \forall l_I \in N_I^+(l), t^l \leq t^{l_I} \vee t^l + dt^l < t^{l_I} + dt^{l_I}\}$.

L'ensemble des influences temporelles dans un niveau $l \in L$ au temps t^l est défini comme :

$$\gamma^{l\prime}(t^l) = \{\gamma^l(t^l) \bigcup_{l_I \in N_I^-(l)} c_I(l_I, l)\}. \tag{3.36}$$

Finalement, les réactions sont calculées pour les niveaux qui rassemblent les conditions définies dans l'eq. 3.27 :

$\forall l \in L_R$,

$$\delta^l(t^l + dt^l) = Reaction^l(\sigma^l(t^l), \gamma^{l\prime}(t^l)), \tag{3.37}$$

avec $L_R = \{l \in L : t^l + dt^l \in min(t + dt)\}$.

L'algorithme 2 résume ce modèle de simulation.

Algorithm 2: Modèle de simulation d'IRM4MLS avec des dynamiques temporelles différentes entre niveaux

Input: $< L, E_I, E_P >, A(t_0), \delta(t_0)$
Output: $\delta(T)$

foreach $l \in L$ **do**
  $t^l = t_0$;
end
while $\exists t^l \leq T$ **do**
  foreach $a \in A(t)$ **do**
    $L_P = \{l \in L : a \in A_l(t) \wedge \forall l_P \in N_P^+(l), t^l \geq t^{l_P}\}$;
    $p_a(t^l) =< Perception_a^l(< \delta^{l_P}(t^{l_P}) : l_P \in N_P^+(l) >) : l \in L_P >$;
    if $a \in HA(t)$ **then**
      $s_a(t_a + dt_a) = Memorization_a(p_a(t^l))$;
    end
  end
  $L_I = \{l \in L : \forall l_I \in N_I^+(l), t^l \leq t^{l_I} \vee t^l + dt^l < t^{l_I} + dt^{l_I}\}$;
  foreach $l \in L_I$ **do**
    $< c_I(l, l_I) : l_I \in N_I^+(l) >= Natural_\omega^l(\delta^l(t^l))$;
    foreach $a \in HA_l(t)$ **do**
      $< c_I(l, l_I) : l_I \in N_I^+(l) >= Decision_a^l(s_a(t_a + dt_a))$;
    end
    foreach $a \in TA_l(t)$ **do**
      $< c_I(l, l_I) : l_I \in N_I^+(l) >= Decision_a^l(p_a(t^l))$;
    end
  end
  $L_R = \{l \in L : t^l + dt^l \in min(t + dt)\}$;
  foreach $l \in L_R$ **do**
    $\gamma^{l\prime}(t^l) = \{\gamma^l(t^l) \bigcup_{l_I \in N_I^-(l)} c_I(l_I, l)\}$;
    $\delta^l(t^l + dt^l) = Reaction^l(\sigma^l(t^l), \gamma^{l\prime}(t^l))$;
    $t^l = t^l + dt^l$;
  end
end

3.2 Pouvoir de représentation de IRM4MLS

3.2.1 Démonstration de l'universalité de IRM4MLS

Dans cette section, nous démontrons que l'algorithme 2 permet de simuler toute instance de IRM4MLS. La démonstration repose sur l'analyse de l'exécution des différents types de situations possibles pour chacune des phases d'observation/mémorisation, influence et enfin réaction [Soyez 2011]. Du point de vue d'un niveau donné, l_1, il n'est pas nécessaire de représenter plus de deux autres niveaux. On pose donc $L = \{l_1, l_2 \text{ et } l_3\}$.

Calcul de perceptions et des mémorisations

Le calcul de la perception dans un niveau l_1 ne dépend pas de $t^{l_1} + dt^{l_1}$. On doit donc traiter 3 cas qui sont illustrés dans la figure fig. 3.2.

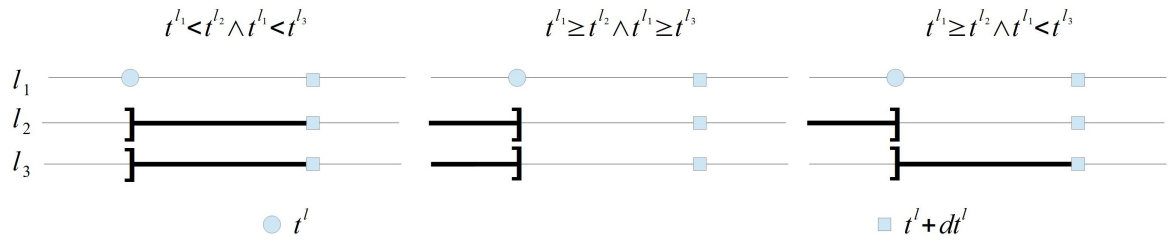

FIGURE 3.2 – Les différents cas de figure à traiter lors du calcul des perceptions. Les traits en gras donnent la position relative possible pour t^{l_2} et t^{l_3}.

$$t^{l_1} < t^{l_2} \wedge t^{l_1} < t^{l_3} \tag{3.38}$$

les agents de l_1 ne perçoivent pas l_2 et l_3.

$$t^{l_1} \geq t^{l_2} \wedge t^{l_1} \geq t^{l_3} \tag{3.39}$$

les agents de l_1 perçoivent l_2 et l_3.

$$t^{l_1} \geq t^{l_2} \wedge t^{l_1} < t^{l_3} \tag{3.40}$$

les agents de l_1 perçoivent l_2 mais pas l_3.

Calcul des influences

Calcul de L_I Le calcul de L_I dépend de t^{l_1} et de $t^{l_1} + dt^{l_1}$. On doit donc traiter 12 cas qui sont illustrés dans la figure fig. 3.3. On pose :

I $t^{l_1} \leq t^{l_2} \wedge t^{l_1} \leq t^{l_3}$

II $t^{l_1} > t^{l_2} \wedge t^{l_1} > t^{l_3}$

III $t^{l_1} > t^{l_2} \wedge t^{l_1} \leq t^{l_3}$

et

a $t^{l_1} + dt^{l_1} < t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} < t^{l_3} + dt^{l_3}$

b $t^{l_1} + dt^{l_1} \geq t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} \geq t^{l_3} + dt^{l_3}$

c $t^{l_1} + dt^{l_1} < t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} \geq t^{l_3} + dt^{l_3}$

d $t^{l_1} + dt^{l_1} \geq t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} < t^{l_3} + dt^{l_3}$

Dans les cas de **I**$\wedge$**a**, **I**$\wedge$**b**, **I**$\wedge$**c**, **I**$\wedge$**d**, **II**$\wedge$**a**, **III**$\wedge$**a** et **III**$\wedge$**c**, $l_1 \in L_I$ car l_1 peut potentiellement influencer l_2 et l_3. Dans le cas de **II** $\wedge$ **c**, $l_1 \notin L_I$ car l_1 peut potentiellement influencer l_2 mais pas l_3. Dans les cas de **II** $\wedge$ **d**, **III** $\wedge$ **b** et **III** $\wedge$ **d**, $l_1 \notin L_I$ car l_1 peut potentiellement influencer l_3 mais pas l_2. Dans le cas de **II** $\wedge$ **b**, $l_1 \notin L_I$ car l_1 ne peut pas influencer l_2 et l_3.

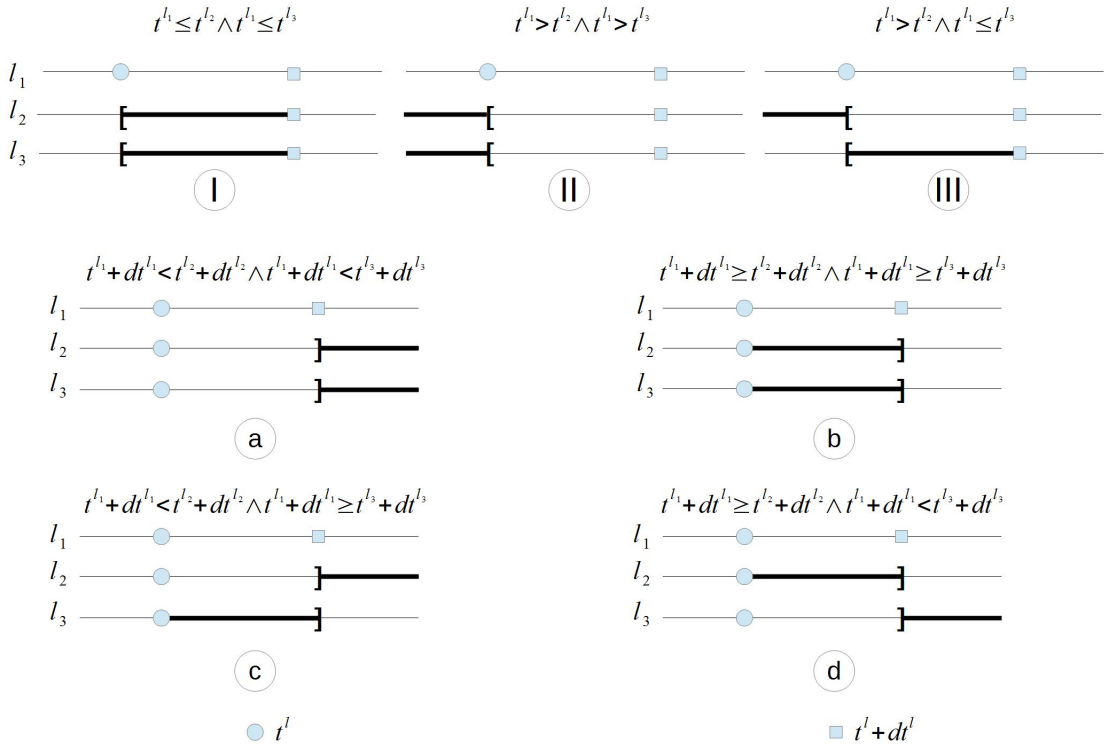

FIGURE 3.3 – Les différents cas de figure à prendre en compte lors du calcul de L_I. Les traits en gras donnent la position relative possible pour t^{l_2} et t^{l_3} (I, II, III) et pour $t^{l_2} + dt^{l_2}$ et $t^{l_3} + dt^{l_3}$ (a, b, c, d).

Calcul des influences dans l_1 Le calcul de l'influence dans l_1 dépend de t^{l_1} et de $t^{l_1} + dt^{l_1}$. On doit donc traiter 12 cas qui sont illustrés dans la figure fig. 3.4. On pose :

A $t^{l_1} + dt^{l_1} > t^{l_2} \wedge t^{l_1} + dt^{l_1} > t^{l_3}$
B $t^{l_1} + dt^{l_1} \leq t^{l_2} \wedge t^{l_1} + dt^{l_1} \leq t^{l_3}$
C $t^{l_1} + dt^{l_1} \leq t^{l_2} \wedge t^{l_1} + dt^{l_1} > t^{l_3}$
D $t^{l_1} + dt^{l_1} > t^{l_2} \wedge t^{l_1} + dt^{l_1} \leq t^{l_3}$

I $\wedge$ **A**, l_1 produit des influences vers l_1, l_2 et l_3. **I** $\wedge$ **B**, l_1 produit des influences uniquement vers l_1 car l_1 évoluera vers le prochain pas de temps avant de pouvoir influencer l_2 et l_3. **I** $\wedge$ **C**, l_1 produit des influences vers l_1 et l_3 car l_1 évoluera vers le prochain pas de temps avant de pouvoir encore influencer l_2. **I** $\wedge$ **D**, l_1 produit des influences vers l_1 et l_2 car l_1 évoluera vers le prochain pas de temps avant de pouvoir encore influencer l_3.

II $\wedge$ **A**, l_1 produit des influences uniquement vers l_1 car l_1 est dans futur par rapport à l_2 et l_3. On ne rencontre pas les cas **II** $\wedge$ **B**, **II** $\wedge$ **C** et **II** $\wedge$ **D** dans le corps de la boucle des influences car c'est en contradiction avec **a** et le fait que $\forall l \in L, t^l < t^l + dt^l$, la seule possibilité pour l_1 d'appartenir à L_I, lorsque l'on prend **II**.

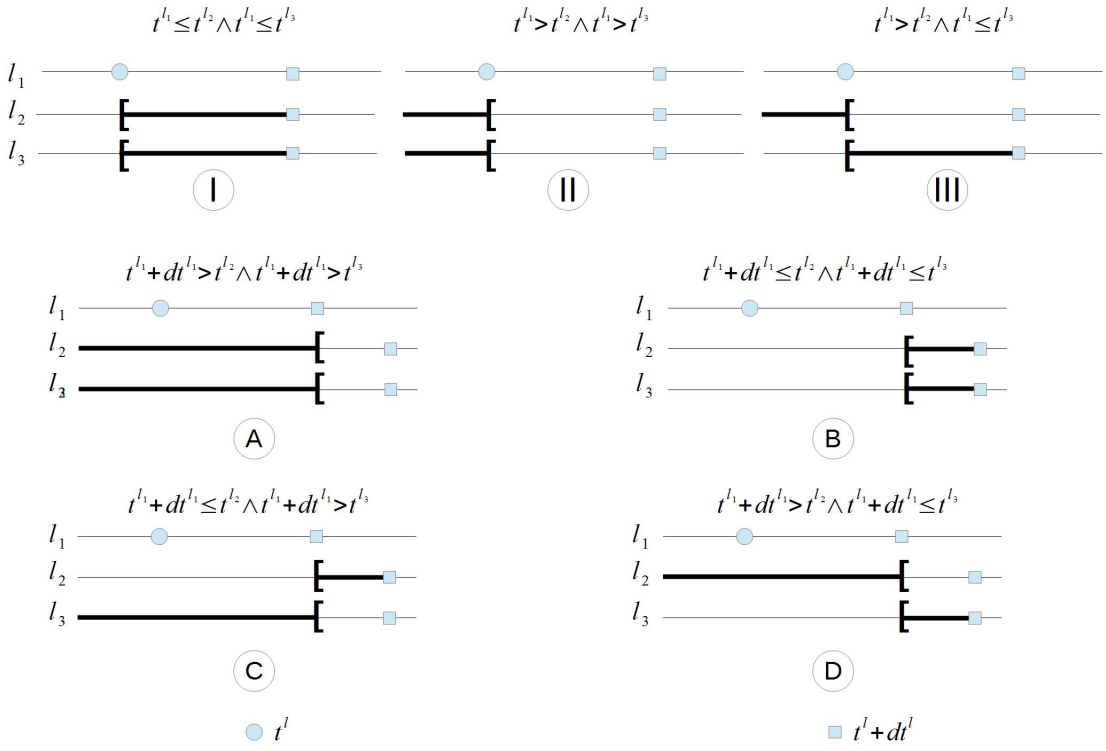

FIGURE 3.4 – Les différents cas de figure à prendre en compte lors du calcul des influences dans l_1. Les traits en gras donnent la position relative possible pour t^{l_2} et t^{l_3}.

III$\wedge$**A**, l_1 produit des influences vers l_1 et l_3 mais pas vers l_2, car l_1 est dans futur par rapport à l_2. On ne rencontre pas les cas **III** $\wedge$ **B**, **III** $\wedge$ **C** dans le corps de la boucle des influences car c'est en contradiction avec $\mathbf{a} \vee \mathbf{c}$, les deux seules possibilités pour l_1 d'appartenir à L_I et le fait que $\forall l \in L, t^l < t^l + dt^l$, lorsque l'on prend **III**. **III** $\wedge$ **D**, l_1 produit des influences uniquement vers l_1, car l_1 est dans futur par rapport à l_2 et évoluera vers le prochain pas de temps avant de pouvoir encore influencer l_3.

Calcul de la réaction

Le calcul de la réaction dans un niveau l_1 ne dépend pas de t^{l_1}. On doit donc traiter 6 cas qui sont illustrés dans la figure fig. 3.5.

$$t^{l_1} + dt^{l_1} < t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} < t^{l_3} + dt^{l_3}, \tag{3.41}$$

l_1 évoluera vers le prochain pas de temps avant l_2 et l_3.

$$t^{l_1} + dt^{l_1} > t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} > t^{l_3} + dt^{l_3} \tag{3.42}$$

l_1 évoluera vers le prochain pas de temps après l_2 et l_3.

$$t^{l_1} + dt^{l_1} < t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} > t^{l_3} + dt^{l_3} \tag{3.43}$$

l_1 évoluera vers le prochain pas de temps avant l_2 et après l_3.

$$t^{l_1} + dt^{l_1} = t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} = t^{l_3} + dt^{l_3} \tag{3.44}$$

l_1 évoluera vers le prochain pas de temps en même temps que l_2 et l_3.

$$t^{l_1} + dt^{l_1} = t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} > t^{l_3} + dt^{l_3} \tag{3.45}$$

l_1 évoluera vers le prochain pas de temps en même temps que l_2 et après l_3.

$$t^{l_1} + dt^{l_1} = t^{l_2} + dt^{l_2} \wedge t^{l_1} + dt^{l_1} < t^{l_3} + dt^{l_3} \tag{3.46}$$

l_1 évoluera vers le prochain pas de temps en même temps que l_2 et avant l_3.

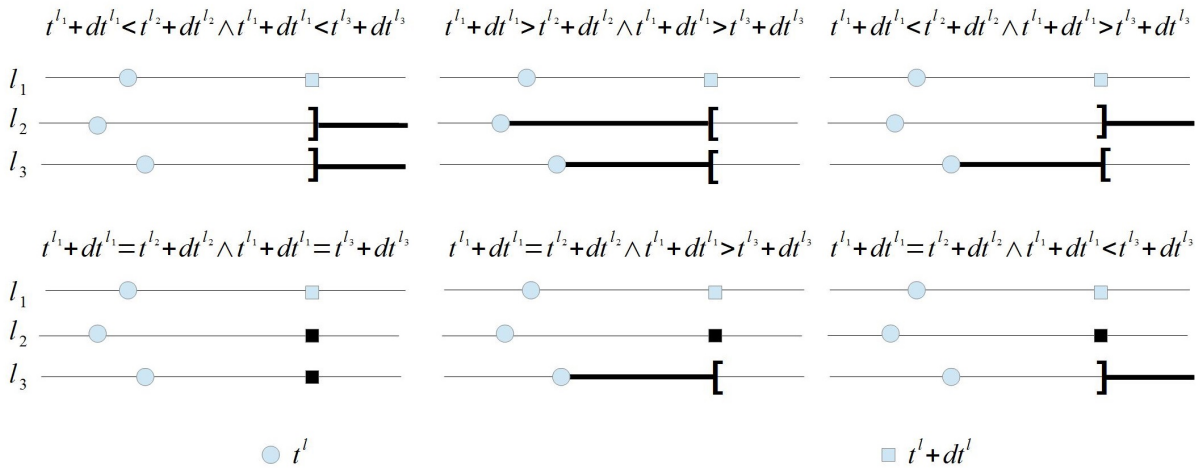

FIGURE 3.5 – Les différents cas de figure à prendre en compte lors du calcul de la réaction. Les traits et points en gras donnent la position relative possible pour $dt^{l_2} + t^{l_2}$ et $t^{l_3} + t^{l_3}$.

Tous les cas pouvant être traités par l'algorithme donnent donc des résultats cohérents. On peut donc conclure à la validité de l'algorithme 2.

3.2.2 "Un esprit, plusieurs corps"

Dans cette section, nous proposons un cadre de travail pour améliorer l'intégration des agents localisés dans différents niveaux simultanément [Soyez 2011]. Par la suite nous montrons comment tirer partie de ce concept pour simuler des systèmes complexes en optimisant l'utilisation des ressources informatiques.

Dans notre approche inspirée par [Picault 2011], les agents peuvent être présents dans plusieurs niveaux en même temps. Nous proposons de décomposer les agents en une partie "centrale" non située et un ensemble de n parties "périphériques" chacune située dans un niveau différent. Ainsi nous appelons *spiritAgent* la partie non située des agents qui contient son état interne, ses processus de décisions et qui ne peut agir dans un niveau. Les parties des agents situées dans des niveaux, (*BodyAgents*), contiennent son état externe et les actions possibles dans ses niveaux, comme la perception de l'environnement. Ces différents aspects sont illustrés

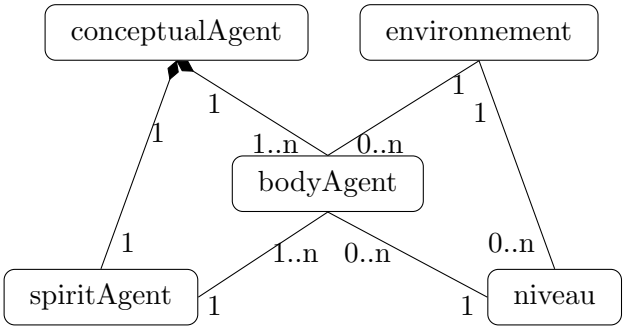

FIGURE 3.6 – Diagramme de classes des concepts centraux d'IRM4MLS avec séparation des parties situées ou non des agents

dans la fig. 3.6.

Les **ConceptualAgents** représentent les agents dans les simulations classiques. Les **SpiritAgents** contiennent seulement l'*état interne* de l'agent et son *module de décision*. Les **BodyAgents** doivent être situés dans un seul niveau. Ils contiennent l'*état externe* de l'agent spécifique à sa localisation dans un niveau donné et un *module d'action* qui indique : 1) quelles sont les actions disponibles à un moment donné et 2) quel est leur résultat en termes d'influences produites. Le *processus de perception* doit se trouver dans le module d'action. Les **Niveaux** contiennent tous les objets inactifs et tout ce qui supporte l'action des agents. Les **Environnements** est de produisent des *influences naturelles* du niveau (comme la gravité pour un niveau physique).

Pour obtenir des simulations valides avec un tel modèle, un spiritAgent doit pouvoir accéder à l'état externe de son conceptualAgent contenu dans ses bodyAgents quand ils sont actifs (durant l'exécution du niveau de ces derniers). Ainsi, on peut considérer les différentes étapes du cycle de vie des agents illustrées dans la fig. 3.7. À chaque fois qu'un bodyAgent est actif, 1) il perçoit son niveau et les autres perceptibles depuis celui-ci L_p, 2) il envoie une part de ses perceptions et les actions possibles au spiritAgent, 3) le spiritAgent modifie son état interne et 4) indique l'action la plus appropriée à faire pour le bodyAgent, 5) le bodyAgent accomplit cette action qui produit des influences sur son niveau et les autres potentiellement influençables depuis ce dernier, L_i.

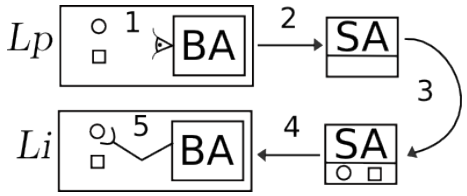

FIGURE 3.7 – Cycle de vie d'un agent conceptuel composé d'un spiritAgent (SA) et d'un bodyAgent (BA)

3.3 Changement dynamique de niveau de détails

Dans cette section, nous donnons une méthodologie pour appliquer des changements dynamiques de LOD dans une simulation. Pour commencer, nous présentons le *graphe de niveaux hiérarchique*, qui indique les liens entre niveaux et les fonctions de dés/agrégation attachées pour changer le LOD des entités simulées. Finalement, nous précisons quand et dans quelles conditions les fonctions de dés/agrégation peuvent s'appliquer [Soyez 2012]. Dans la prochaine section nous donnons une méthode pour tester la qualité des mécanismes de dés/agrégation exposés ici en mesurant la cohérence globale des simulations

3.3.1 Graphe hiérarchique de niveaux

Les relations entre niveaux sont formalisés par un digraphe, $\langle L, E_H \rangle$ où E_H est un ensemble d'arcs, i.e., de paires ordonnées d'éléments de L. Ce digraphe, dont les sommets sont les niveaux, est appelé *graphe hiérarchique de niveaux*. Il indique comment les niveaux sont imbriqués entre eux et quelles paires de niveaux traitent les différents aspects d'un même phénomène.

Un *arc simple* représente un *lien d'inclusion* entre deux niveaux. Par exemple, un arc (l_1, l_2) signifie que l_2 est à une échelle spatio-temporelle plus grande que l_1. Ainsi, les bodyAgents situés dans l_1 peuvent être agrégés et l'agrégat résultant peut être instancié dans l_2. Nous notons que $l_1 \prec l_2$.

Une *paire d'arcs symétriques* signifie qu'il y a un *lien de complémentarité* entre deux niveaux. Par exemple, les arcs (l_1, l_3) et (l_3, l_1) signifient que l_1 et l_3 sont à la même échelle. De plus un spiritAgent peut contrôler plusieurs bodyAgents simultanément présents et actifs dans l_1 et l_3. Nous notons que $l_1 \equiv l_3$.

Une *boucle* sur un sommet indique un niveau dont les bodyAgents peuvent adopter un comportement similaire. Par exemple, un arc (l_1, l_1) signifie que les spiritAgents, de certains bodyAgents situés dans l_1, peuvent être agrégés pour former un seul spiritAgent qui contrôlera ces bodyAgents inchangés dans l_1. Ces bodyAgents auront le même comportement quand ils seront face aux mêmes situations, mais garderont leur autonomie.

Les règles suivantes doivent être appliquées pour obtenir un modèle cohérent.

Règle 1 *Les liens d'Inclusion et de Complémentarité sont transitifs.*

$$l_1 \prec l_2 \wedge l_2 \prec l_3 \rightarrow l_1 \prec l_3,$$
$$l_1 \equiv l_2 \wedge l_2 \equiv l_3 \rightarrow l_1 \equiv l_3.$$

Règle 2 *Un niveau ne peut être inclus dans lui même de manière directe ou indirecte. Cette règle se traduit par le fait que si on supprime toutes les paires d'arcs symétriques, il ne doit pas y avoir de* cycle orienté *dans un graphe hiérarchique.*

$$\nexists l_1 \in L \wedge l_1 \prec l_1$$

Règle 3 *Deux niveaux ne peuvent pas partager simultanément un lien d'inclusion et de complémentarité, de manière directe ou transitive.*

$$l_1 \prec l_2 \rightarrow l_1 \not\equiv l_2,\ l_1 \equiv l_2 \rightarrow l_1 \not\prec l_2.$$

Chaque arc qui n'est pas dans une paire d'arcs symétriques est étiqueté avec le nom d'au moins une fonction d'agrégation. Le nom d'une fonction d'agrégation peut être placé sur plusieurs arcs.

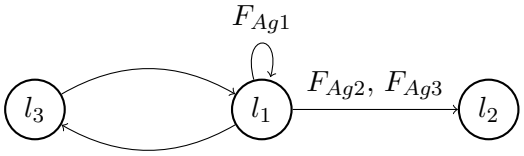

FIGURE 3.8 – Un exemple de graphe hiérachique de niveaux.

La figure fig. 3.8 montre un exemple de graphe hiérarchique de niveaux. L'arc (l_1, l_1), étiqueté F_{Ag1}, indique que les spiritAgents contrôlant certains bodyAgents présents dans l_1 peuvent s'agréger pour former un seul spiritAgent contrôlant tous ses bodyAgents, à travers la fonction F_{Ag1}. L'arc (l_1, l_2), étiqueté F_{Ag2}, F_{Ag3}, signifie que les spiritAgents contrôlant certains bodyAgents présents dans l_1 peuvent s'agréger pour former un seul spiritAgent contrôlant un seul bodyAgent agrégé situé dans l_2, au travers de la fonction F_{Ag2} ou F_{Ag3}. Ces deux fonctions concernent des combinaisons différentes de corps. Et la paire d'arcs symétriques entre l_1 et l_3, non étiquetés, représentent le fait que certains spiritAgents peuvent contrôler simultanément des bodyAgents situés dans ces deux niveaux.

3.3.2 Fonctions d'agrégation ou de désagrégation

Contenu

Il existe deux types d'agrégations. Le premier concerne l'agrégation de spiritAgents et le second l'agrégation de spiritAgents et de leurs bodyAgents associés. Le premier type d'agrégation est utilisé pour représenter un ensemble d'agents avec le même état interne, ce qui conduit à des agents qui agissent de manière similaire confrontés à la même situation mais qui peuvent être placés dans des situations différentes. L'agrégation de plusieurs bodyAgents sans celle conjointe de leur spiritAgent est impossible car un corps ne peux pas être contrôlé de manière concurrente par plusieurs esprits simultanément.

Une fois que le graphe hiérarchique de niveaux a été fixé, le modélisateur doit indiquer dans quels niveaux se situent chaque classe de bodyAgent présent dans le modèle et quelle classe de spiritAgents contrôlent ces bodyAgents. Pour chaque fonction d'agrégation, le modélisateur doit également préciser combien d'agents doivent fusionner, la classe des agents agrégés et agrégats et comment générer l'état interne et/ou externe de l'agent agrégat.

Dans ce manuscrit nous ne donnons pas d'indication pour concevoir le module de décision ou celui d'action des agents agrégats ou non mais nous indiquons comment agréger l'état interne et externe des agents, respectivement contenus dans les spiritAgents et les bodyAgents. Chaque fonction d'agrégation peut être divisée en plusieurs sous-fonctions. Ces sous-fonctions peuvent être de deux types. Premier type : une sous-fonction prend la même variable dans chaque agent concerné (spiritAgents ou bodyAgents) et les agrège pour obtenir une seule valeur à placer dans l'état de l'agent agrégat. Par exemple, un agent représentant un train de véhicules (platoon)

possède la position moyenne de tous ses agents véhicules. Second type : une sous-fonction similaire au premier type fait une agrégation sur plusieurs variables dans les agents à agréger mais ne produit qu'une valeur. Ceci peut être illustré par l'agent platoon décrit plus haut. Il possède une seule variable dans son état interne appelée "priorité", dont la valeur est générée avec la composition des variables "énergie" et "vitesse" des chaque agent véhicule du platoon. Certaines variables des agents à agréger peuvent être ignorées pour construire un agrégat.

Notation

Une fonction d'agrégation consiste à créer un agent composite à partir de plusieurs agents. Nous donnons ici la forme générale d'une fonction d'agrégation F_{Ag} utilisant comme arguments n classes de conceptualAgents, *caa* (classe à agréger), doté d'un intervalle, $[min_i, max_i]$, indiquant combien d'instances de ces classes sont nécessaires pour accomplir cette agrégation. pour chaque classe de conceptualAgents, il est précisé si l'agrégation concerne les bodyAgents en plus des spiritAgents avec l'indication du niveau dans lequel sont situés les bodyAgents. La classe de l'agent produit par l'agrégation, CAA (Classe d'Agent Agrégat), est la sortie de F_{Ag} avec son niveau l si l'agrégation concerne des bodyAgents. Si l'agrégation concerne uniquement des spiritAgents alors $l = l_i = \varnothing$.

$$F_{Ag}(\prod_{i \in n} \langle [min_i; max_i] caa_i, l_i \rangle) = (CAA, l) \tag{3.47}$$

Par exemple, considérons la fonction F_{Ag2} décrite dans le graphe hiérarchique de niveaux au dessus. Disons que, F_{Ag2} agrège un bodyAgent de la classe *Leader* et entre 4 et 9 bodyAgents de la classe *Follower*, tous situés dans le niveau l_1, et leur spiritAgents associés pour créer un bodyAgent de la classe *Platoon* (qui représente un train de véhicules formé d'un meneur et de plusieurs suiveurs), situé dans l_2 et son spiritAgent associé.

$$\begin{aligned} F_{Ag2}(\langle [1;1], Leader, l_1 \rangle, \langle [4;9], Follower, l_1 \rangle) \\ = (Platoon, l_2) \end{aligned} \tag{3.48}$$

Les sous-fonctions d'agrégation ont à peu près la même notation que les fonctions d'agrégation. Il n'est plus nécessaire de préciser le nombre d'agents concernés mais les variables, dans les agents concernés, qui seront fusionnées doivent être connues. Par exemple, la sous-fonction décrite dans la section précédente peut s'écrire ainsi :

$$\begin{aligned} f_{Ag2,1}\big((Leader.stamina, Leader.speed, l_1), \\ (Follower.stamina, Follower.speed, l_1)\big) \\ = (Crowd.priority, l_2) \end{aligned} \tag{3.49}$$

Fonctions de désagrégation et de mémorisation

Chaque fonction d'agrégation possède une fonction d'agrégation et éventuellement une fonction de mémorisation associée. Une fonction de désagrégation permet de créer plusieurs instances des agents agrégés à partir de l'agent agrégat. Une fonction de mémorisation est utilisée pour retenir les informations les plus significatives. Chaque fonction de mémorisation est associée à une fonction de désagrégation pour générer plusieurs agents, représentant les agents agrégés

initialement, en prenant en compte l'état de ces agents avant l'agrégation et l'évolution du système depuis l'agrégation. nb_i indique le nombre d'agents de chaque classe concernées par cette agrégation.

$$\begin{aligned} & F_{Disag}(AAC, l, \\ F_{Memorization}(\textstyle\prod_{i\in n}\langle nb_i, cta_i, l_i\rangle)) & \\ = (\textstyle\prod_{i\in n}\langle nb_i, cta_i, l_i\rangle) & \end{aligned} \tag{3.50}$$

Ces deux fonctions sont divisées en sous-fonctions de la même manière qu'une fonction d'agrégation. Prenons un train de véhicules dotés de deux variables de positionnement, X et Y, représentant les variables de positionnement, x et y, de tous les véhicules qui le constituent. La fonction de mémorisation conserve la positions de tous ces véhicules. La mémorisation n'est pas active durant l'exécution de l'agent représentant le train de véhicules (Platoon). Après que l'agent Platoon se soit déplacé à la position (X', Y'), il peut être désagrégé en recréant les agents véhicules, en calculant la valeur de leurs variables x et y à partir de X' et Y' et en appliquant la répartition mémorisée.

3.3.3 Test d'agrégation et de désagrégation

[Navarro 2011] propose une méthode permettant de déterminer quand des agents doivent être agrégés. Il utilise une fonction d'affinité qui mesure la similarité de l'état interne et externe des agents. Lorsque la similarité atteint un certain seuil prédéfini, entre deux agents, il les relie. Les agents reliés ayant la plus grande valeur de similarité sont agrégés ensemble.

Nous pouvons utiliser un mécanisme similaire pour décider quand utiliser une fonction d'agrégation, mais dans notre cas il est nécessaire d'avoir une fonction d'affinité Aff par fonction d'agrégation F_{Ag}. S'il y a plusieurs fonctions d'agrégation qui concernent les mêmes spiritAgents ou bodyAgents dans les mêmes niveaux, il faut décider quand appliquer une fonction d'agrégation plutôt qu'une autre.

Il y a trois possibilités pour motiver ce choix. 1) Le choix de F_{Ag} est fait après la mesure de l'affinité d'un groupe d'agents avec toutes les Aff et les agrégats sont instanciés à chaque fois, en choisissant le groupe avec l'affinité la plus grande, jusqu'à ce qu'il ne reste plus de groupe. 2) Il est aussi possible d'imposer un ordre pour tester les différentes F_{Ag}. Tous les groupes avec une grande affinité pour une F_{Ag} sont agrégés, ensuite la prochaine F_{Ag} est testée jusqu'à ce qu'il ne reste plus de F_{Ag}. 3) Le choix des F_{Ag} peut être fait avec un mélange des deux méthodes précédentes. Un ordre partiel est défini sur l'espace des F_{Ag}. Et s'il n'y a pas de lien de précédence entre différentes F_{Ag}, il faut utiliser la première méthode pour agréger les agents en considérant que le modèle contient uniquement ces F_{Ag}, après cela nous continuons à suivre l'ordre établi.

3.3.4 Mesure de la qualité des modèles simulés

[Davis 1993] utilise la notion de cohérence (consistency) pour mesurer la qualité d'une simulation qui gère des modèles de résolutions différentes. "Consistency between a high-resolution model **M** and a low-resolution model **M'** is the comparison between the projected state of an aggregate of high-resolution entities which evolved in **M**, and the projected state of the same aggregate initially controlled by **M'**". Cette notion de cohérence faible est illustrée dans la fig. 3.9.

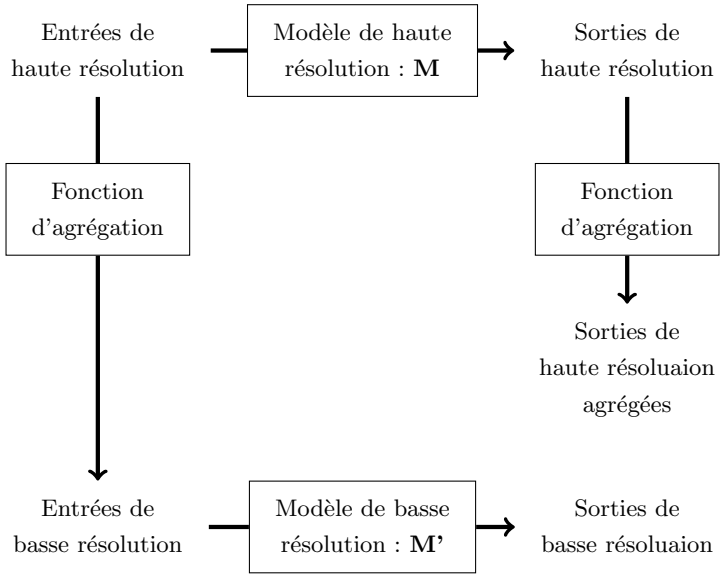

FIGURE 3.9 – Cohérence faible, selon [Davis 1993].

Il est plus intuitif de se baser sur la comparaison de l'évolution du modèle le plus détaillé au lieu du modèle agrégé parce qu'il a une plus grande résolution et contient plus d'informations significatives. Cette notion de cohérence forte est illustrée dans la fig. 3.10.

Avant de modéliser un système, il est nécessaire de déterminer les éléments significatifs de la simulation. Ces éléments peuvent être dans l'état interne (spiritAgent) ou externe (bodyAgent) des agents ou dans leur environnement. Une fois que ces éléments sont identifiés, plusieurs simulations doivent être exécutées avec les mêmes paramètres (état initial et temps d'exécution) en utilisant les niveaux les plus détaillés, porteurs de plus d'informations mais plus coûteux. À la fin de l'exécution de ces simulations un état moyen des éléments significatifs est enregistré. Le même processus est appliqué sur un modèle utilisant des changement dynamiques de LOD. Ensuite la dissimilarité est mesurée entre ces deux enregistrements pour calculer la cohérence.

3.4 Conclusion

Nous avons présenté le méta-modèle multi-agents multi-niveaux IRM4MLS. Nous avons montré comment IRM4MLS peut représenter des systèmes avec une structure et des environnements s'étalant sur plusieurs niveaux et comment il est adapté à leur simulation. De plus, nous avons présenté des outils computationnels qui permettent de simuler des systèmes complexes en faisant un compromis entre le niveau de détail de la simulation et les ressources informatiques utilisées. Nous avons également présenté un outil de mesure pour tester la qualité des simulations résultantes. Nous possédons maintenant un outil pouvant servir de base à la définition de l'environnement et la structure multi-niveaux d'un SdS. Dans le prochain chapitre nous proposons un formalisme de modélisation des SdS, qui permet de représenter leurs aspects statiques

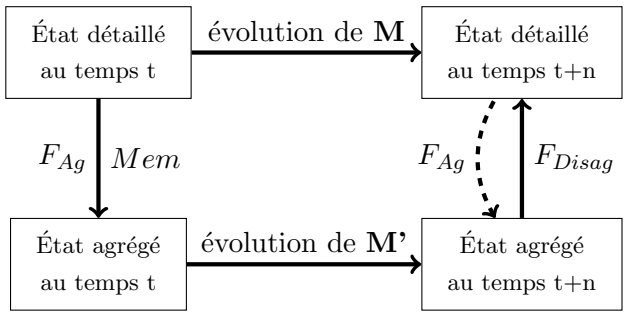

FIGURE 3.10 – Cohérence forte, selon [Davis 1993].

et dynamiques, tout en respectant leurs caractéristiques fondamentales.

Chapitre 4

Contribution à la modélisation de SdS par l'approche multi-agents

Sommaire

4.1 Introduction

Nous avons présenté le concept de SdS et proposé une définition des SdS basée sur leurs caractéristiques fondamentales. Nous avons également présenté le méta-modèle multi-agents multi-niveaux IRM4MLS. Dans ce chapitre, nous proposons un formalisme basé sur IRM4MLS pour modéliser les SdS.

Dans un premier temps, nous rappelons l'intérêt de la modélisation des SdS à travers les besoins du projet européen InTraDE. Dans un second temps, nous fournissons une représentation graphique que nous utilisons dans la suite de ce manuscrit pour représenter les différents éléments et interactions des SdS. Ensuite, nous présentons un formalisme multi-agents et multi-niveaux basé sur le méta-modèle IRM4MLS permettant de modéliser les aspects statiques et dynamiques des SdS, tout en respectant leurs caractéristiques fondamentales. Finalement, nous proposons

deux algorithmes génériques adaptés à ce formalisme et contrôlant deux mécanismes que rencontrent tous les SdS au cours de leur existence, lors leur création et lorsque leurs capacités ou leur but global évoluent.

4.1.1 Le projet InTraDE : un cadre d'étude des SdS

L'un des objectifs d'InTraDE est de fluidifier le transport de fret en milieu confiné, comme un terminal portuaire, à l'aide d'un ITS composé d'IAVs. Actuellement, dans la littérature, les IAVs sont traités comme de simples AGVs rendus plus autonomes. La technologie des véhicules autonomes guidés (AGV) qui peut être comparée aux IAVs est largement utilisée pour gérer le stockage et la manutention de biens dans les grands entrepôts. Un AGV est un véhicule autonome qui réagit immédiatement à son environnement, comme un marquage au sol. Contrairement à un AGV un IAV peut prendre des décisions complexes, collaborer avec d'autres véhicules et faire ses propres choix pour mener à bien une mission. Un autres avantage des IAVs est le fait qu'ils peuvent réagir aux imprévus et ainsi évoluer dans des environnements dynamiques au contraire des AGVs dont le comportement ne peut être modifié que par l'aménagement de l'environnement, ce qui se révèle généralement très coûteux.

Lorsqu'ils sont représentés au niveau microscopique (un IAV individuel) ou mésoscopique (un train de véhicules), les modèles traitant les IAVs sont réalistes. Cependant lorsque toute une flotte d'IAVs (niveau de représentation macroscopique) doit être considérée, aucun modèle ne permet de représenter ou contrôler cette flotte en considérant les capacités des groupes ou des IAVs seuls à raisonner sur la répartition des tâches pour atteindre un objectif global et les interactions entre les entités à différents niveaux. Or pour pouvoir superviser le fonctionnement et la conception d'un ITS formé d'IAVs nous avons besoin d'un outil qui permet de modéliser ces différents aspects et les exécuter au cours de simulation.

Les ITS considérés dans InTraDE peuvent être décrits comme des systèmes indépendants, capables de coopérer pour réaliser un but commun tout en s'adaptant à l'évolution de l'environnement : en d'auters termes un système de systèmes (SdS). Un SdS est un système complexe particulier. Un SdS est un concept organisationnel qui décrit un système, généralement de grande taille, composés de sous systèmes, appelés composants systèmes (CS), imbriqués les uns dans les autres comme des poupées gigognes, qui sont autonomes, souvent hétérogènes, qui peuvent communiquer et s'adapter de manière locale ou globale pour répondre aux changements de leur environnement.

4.1.2 Modélisation des SdS par des agents

Il manque à la définition classique des SMA (que nous avons vue au chapitre 2) des concepts, telles que des définitions organisationnelles et hiérarchiques pour définir quels CS intègrent et/ou allouent des missions à d'autres CS, pour pouvoir représenter fidèlement un SdS. De même, les notions de but global et de missions sont exprimés sans tenir compte de la structure multi-niveaux des SdS. Cela empêche les SdS de raisonner à leur propos en vue de les réaliser. Ces concepts sont cruciaux parce qu'ils sont nécessaires pour exprimer les caractéristiques d'indépendance managériale, de développement évolutif, de coopération et de coexistence.

Dans la prochaine partie, nous donnons tous les éléments statiques et dynamiques nécessaires

pour représenter un SdS et qui ne sont pas inclus dans la définition classique des SMA. Ces éléments sont une description organisationnelle des entités et un environnement multi-niveaux et deux mécanismes qui conduisent l'initialisation des SdS et l'évolution de leurs capacités. Ensuite, nous donnons plusieurs éléments pour prouver que les modèles multi-agents, créés en utilisant notre formalisme, respectent les caractéristiques fondamentales des SdS.

4.2 Représentation graphique et taxonomie

Dans cette section une représentation graphique des entités réelles d'un SdS est introduite. Pour plus de détail sur la notation des CS voir section 4.3.1. Cette représentation permet de voir aisément comment un CS est constitué par ses CS de plus bas niveau et fait la distinction entre les liens d'inclusion et de communication.

Considérons un exemple issu du projet InTraDE que nous illustrons dans la fig. 4.1. Une flotte complète d'IAVs appartenant à un port, $CS_{1,3}$, est composée de deux flottes d'IAVs attachées respectivement au quai 1, $CS_{1,2}$, et au quai 2, $CS_{2,2}$. La flotte $CS_{1,2}$ est composée d'un train de véhicules, $CS_{1,1}$, incluant les IAVs $CS_{1,0}$ et $CS_{2,0}$, et d'un IAV individuel $CS_{3,0}$. La flotte $CS_{2,2}$ est composée de trois IAVs individuels : $CS_{4,0}$, $CS_{5,0}$ et $CS_{6,0}$.

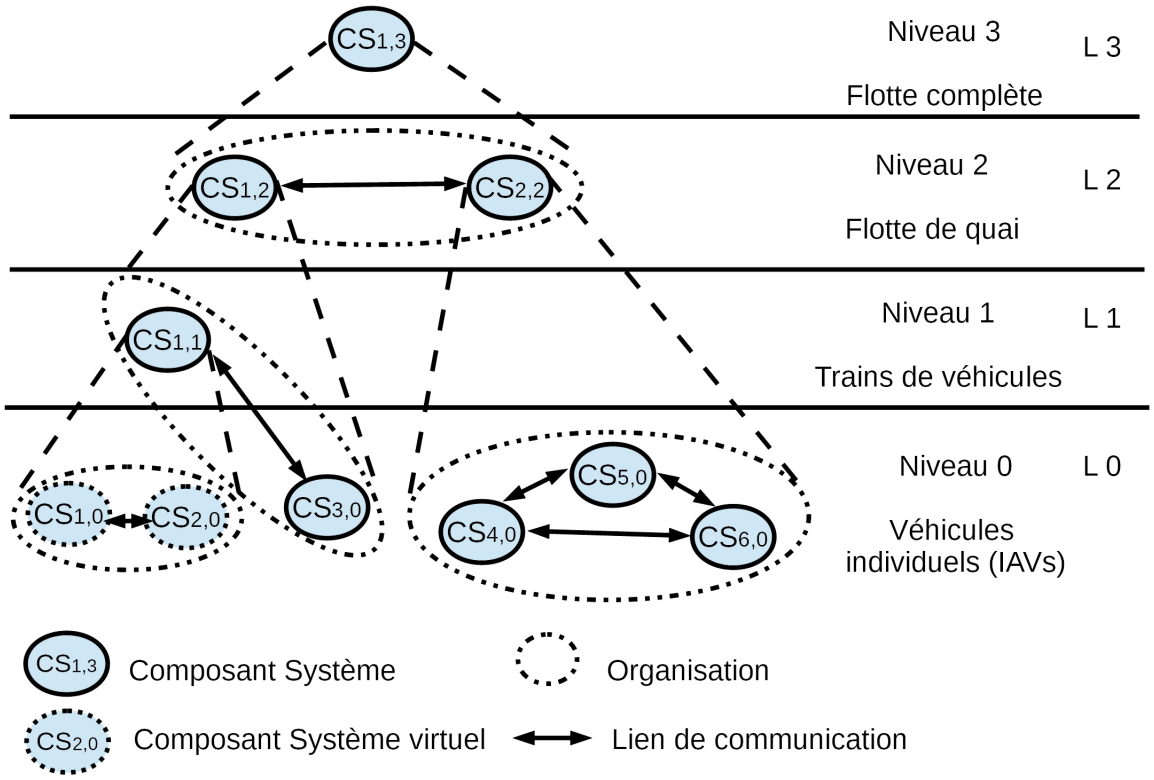

FIGURE 4.1 – Représentation graphique multi-niveaux d'un SdS.

Certains CS ou ensembles de CS peuvent violer temporairement certaines caractéristiques

des SdS expliquées plus loin. Cela ne signifie pas la fin du SdS, mais plutôt que ces CS doivent être supprimés, au moins temporairement, pour faire en sorte que le SdS reste cohérent avec ses caractéristiques. Nous appelons ces entités des **composants systèmes virtuels**. Par exemple, les véhicules $CS_{1,0}$ et $CS_{2,0}$ sont des IAVs reliés physiquement pour former un train de véhicules. Ils ont, en partie, perdu leur indépendance mais continuent à exister.

De plus un CS qui ne peut pas être divisé en un ensemble de CS tout en respectant ces caractéristiques est appelé un **composant système élémentaire** (comme $CS_{1,1}$, $CS_{3,0}$, $CS_{4,0}$, $CS_{5,0}$ et $CS_{6,0}$). De manière simple les entités physiques individuelles d'un système sont considérés comme des CS élémentaires (comme $CS_{3,0}$, $CS_{4,0}$, $CS_{5,0}$, $CS_{6,0}$ et potentiellement $CS_{1,0}$ et $CS_{2,0}$). À l'opposé des CS élémentaires, le seul CS présent au plus haut niveau, $CS_{1,3}$, qui est l'équivalent du SdS peut être appelé le **composant système total**.

Dans les prochaines sections, nous allons modéliser des CS en utilisant des agents CS en tirant partie de l'autonomie et de l'indépendance inhérente aux agents. Les agents CS sont des agents représentant des CS dans un SdS et montrant les mêmes caractéristiques que des CS.

4.3 Formalisme

4.3.1 Aspects statiques

Le concept de SdS est un concept organisationnel dans lequel les CS sont des entités qui peuvent être des individus ou des agrégations de CS dans un ou plusieurs niveaux inférieurs. Il est nécessaire que les CS non élémentaires possèdent un moyen de raisonner à propos de la distribution des missions pour atteindre leurs buts alloués. Pour cela nous représentons ces groupes grâce au modèle Agent Groupe Rôle (AGR) [Gutknecht 2001, Ferber 2004]. Les principaux concepts d'AGR sont illustrés dans la figure fig. 4.2.

D'après AGR, un agent est une entité jouant un ensemble de rôles dans différents groupes. Un agent est caractérisé par ses rôles, connaissances et capacités. Une *capacité* est la description abstraite d'une connaissance. Elle regroupe les moyens qui permettent d'accomplir une tâche. Cela représente une fonctionnalité logique pour les entités fournisseuses (possesseurs) ou demandeuses (utilisateurs). D'une manière plus prosaïque, une capacité d'agent ou de groupe est la possibilité pour cette entité d'accomplir un but. Accomplir un but pour un agent revient à fixer l'état des variables propres au niveau avec les valeurs voulues, par le moyen de ses actions.

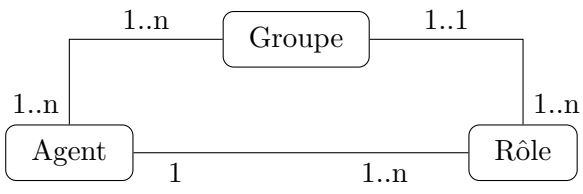

FIGURE 4.2 – Concepts Centraux d'AGR (cardinalités respectant le formalisme UML)

Un *rôle d'agent* est une instance concrète d'un rôle. Il décrit le comportement de l'agent dans un contexte défini par un groupe. Il confère un statut et des moyens, dans ce groupe, d'interagir

avec d'autres agents jouant des rôles dans ce groupe. Après l'initialisation du SdS, les groupes sont justes des définitions et un même ensemble d'agents peut instancier plusieurs groupes de manière séquentielle ou instantanée. Ces groupes possèdent des capacités déterminées par les rôles joués par les agents et leurs capacités individuelles.

Modélisation des CS

Pour désigner de manière unique et sans ambiguïté un CS dans un SdS nous utilisons le formalisme suivant : $CS_{n,l}$ désigne le n^{me} CS du système et son agent le représentant dans le niveau l. Un CS, CS_n d'un niveau l, peut être noté comme suit :

$CS_{n,l} =< CS_{n',l+1}, R_{n,l}, \mathcal{CS}_{n'',l-1}, OP, \Psi, \Lambda >$, avec :

- $CS_{n',l+1}$: un super CS directement composé de $CS_{n,l}$ et possiblement d'autres sous CS. Ce CS doit être situé dans un niveau plus haut que l.
- $R_{n,l}$: l'ensemble des rôles joués par $CS_{n,l}$, $R_{n,l} = 2^{Roles(O_{n,l})}$ avec $O_{n,l}$ l'ensemble des groupes dans lesquels $CS_{n,l}$ joue au moins un rôle.
- $\mathcal{CS}_{n'',l-1}$: l'ensemble des sous CS dont le super CS est $CS_{n,l}$. Ces CS doivent être dans un ou des niveaux plus bas que l.
- OP : l'ensemble des groupes qui prennent part à la vie et au fonctionnement des CS de $CS_{n,l}$ et contribuent à l'accomplissement des objectifs liés aux rôles $R_{n,l}$.
- $\Psi : \mathcal{CS}_{n'',l-1} \rightarrow 2^{Roles(OP)}$: une fonction qui associe un sous CS à un ensemble de rôle qu'il joue dans les groupes définis dans $CS_{n,l}$, tel que $\forall csi \in \mathcal{CS}_{n'',l-1}, \Psi(csi) \neq \emptyset$. La fonction de rôles donne l'ensemble des rôles définis dans les groupes OP.
- $\Lambda : \Sigma_l * \Sigma_l \rightarrow \{0, 1\}$: une fonction de capacité qui indique si un état but d'un niveau l est atteignable, par l'action de $CS_{n,l}$, sachant l'état actuel de l. Cette fonction dépend des capacités évolutives de $CS_{n,l}$ et aussi des capacités offertes par OP à $CS_{n,l}$ quand $CS_{n,l}$ n'est pas un CS élémentaire.

Dans le cas de l'ingénierie des SdS, il existe une hiérarchie de niveaux d'abstraction, approximation et échelles croissantes. Ces niveaux peuvent être représentés par un graphe hiérarchique, comme montré plus haut, qui indique l'imbrication des niveaux. Quand un niveau l_i est directement inclus dans un niveau l_j, les CS de l_i composent des CS de plus grande échelle dans l_j. Un CS peut appartenir directement à au plus un CS de plus haut niveau. Ainsi, dans notre cas, on considère qu'un niveau peut être directement inclus dans un niveau au maximum. Quand un CS, $CS_{i',j'}$, est un sous CS de $CS_{i,j}$ il est noté $CS_{i',j'} \in CS_{i,j}$.

Modélisation des groupes de CS

Pour décider comment former un super CS d'après l'organisation de ses sous CS et comment ces CS sont organisés pour atteindre des buts il est nécessaire de doter notre modèle d'un ensemble de spécifications de groupes définies comme suit :

$gs =_{def} < R, L_R, \mathcal{L}, C^{intra}, C^{inter}, nr, nc >$, avec :

- R : l'ensemble des rôles jouables par les agents dans le groupe créé en utilisant gs.
- $L_R : R_{gs} \rightarrow L$: une fonction qui indique le niveau d'un agent qui peut jouer un rôle donné dans gs.
- $\mathcal{L}$: indique le ou les niveaux du groupe créé en utilisant gs. Si $\mathcal{L}$ contient un seul niveau l et que l est plus haut des niveaux de $L_R(R)$ alors le groupe défini par gs doit être instancié par un agent CS situé dans l. Les agents CS jouant un ou des rôles de R sont les sous CS de ce nouvel agent CS.
- C^{intra} : une fonction de compatibilité qui indique si deux rôles sont compatibles dans le même groupe. Par défaut deux rôles ne sont pas compatibles. $\rho_a \Delta \rho_b$ dénote le fait que les agents jouant le rôle ρ_a sont autorisés à jouer le rôle ρ_b. Cette relation est réflexive et transitive.
- C^{inter} : une fonction de compatibilité qui indique si deux rôles sont compatibles dans deux groupes différents. Elle possède le même formalisme que C^{intra}.
- $nr : R_{gs} \rightarrow N * N$: une fonction qui spécifie le nombre (minimum, maximum) d'agents CS qui doivent jouer un rôle donné dans un groupe créé en utilisant gs. Par exemple, $nr_{gs}(carrier) = (1, 3)$ signifie que les groupes émanant de gs doivent posséder au moins un et au plus 3 agents jouant le rôle $carrier$.
- $nc : Rgs * C \rightarrow N * N$: une fonction qui spécifie la quantité (minimale, maximale) pour un rôle et une capacité donnée, de cette capacité qui doit être disponible auprès de chaque agent jouant ce rôle et de la totalité d'agents jouant ce rôle dans un groupe créé en utilisant gs.

Ici une capacité devrait être exprimée d'une manière fonctionnelle et si possible quantitative, i.e., non exprimée en termes de buts cibles mais en termes de résultats concrets des actions des agents. Il est à noter que nr et nc sont complémentaires car ensemble ils indiquent les quantités des capacités nécessaires au bon fonctionnement du groupe créé en utilisant gs et également la répartition de ces capacités sur les agents qui jouent un rôle dans ce groupe.

Ainsi, $nc_{gs}(carrier, carryOneContainer/hour) = (1, \infty, 3, \infty)$ signifie qu'un agent jouant le rôle $carrier$ dans un groupe émanant de gs doit au moins posséder une unité de la capacité "$CarryOneContainer/hour$" et la population totale d'agents jouant ce rôle dans ce groupe doit posséder au moins trois unités cumulatives de la capacité "$CarryOneContainer/hour$" avec dans les deux cas aucune limite maximale pour cette capacité quantitative.

Environnement multi-niveaux et décomposition de buts

IRM4MLS fournit un support de modélisation pour les buts globaux. Un but global est un état du monde à atteindre pour le système, qui peut être exprimé comme une description du système et de ses variables environnementales. Un but global peut être divisé en un ensemble de missions. Ce découpage peut être fonctionnel, géographique ou autre [Calvez 1990, Clarhaut 2009]. Parce que la modélisation d'un SdS inclut une représentation multi-niveaux, ce découpage peut se faire selon les niveaux. Ainsi les missions d'un niveau peuvent être exprimées comme des états de ce niveau, i.e., les états des CS et l'environnement de ce niveau. De plus, le but global d'un SdS peut être exprimé comme un état but du plus haut niveau L_H. Il est utilisé pour définir le but global du SdS : $Gg = \delta^{L_H}(t') =< \sigma^{L_H(t')}, \gamma^{L_H(t')} >$.

t' peut être considéré comme le temps limite pour le SdS pour atteindre Gg.

Parce que c'est le niveau le plus abstrait et le plus approximé, l'état souhaité $L_H(t')$ pour le plus haut niveau L_H peut être divisé en un ensemble d'états dans chaque niveau inférieur, correspondant à des missions locales, qui décrivent plus en détails l'accomplissement du but global.

4.3.2 Illustration des aspects statiques

Pour illustrer les aspects statiques nous modélisons le SdS présenté en section 4.2 (et représenté dans la fig. 4.3) avec notre formalisme.

$L = \{l_o, l_1, l_2, l_3\}$ avec :

- $l_3 = l_H = niveauDeToutLaFlotte$,
- $l_2 = niveauDesFlottesDeQuai$,
- $l_1 = niveauDesTrainsDeVehicules$ et
- $l_0 = niveauDesIAVsIndividuels$.

Tous les CS sont représentés par des agents CS. $\mathcal{A} = \{CS_{1,3}, CS_{1,2}, CS_{2,2}, CS_{1,1}, CS_{1,0}, CS_{2,0}, CS_{3,0}, CS_{4,0}, CS_{5,0}, CS_{6,0}\}$

Ici nous représentons seulement en détail $CS_{2,2}$, son super CS, $CS_{1,3}$, et ses sous CS $CS_{4,0}$, $CS_{5,0}$ et $CS_{6,0}$. Cette partie du SdS est représentée dans la figure fig. 4.4.

$$CS_{1,3} = \langle \emptyset, \emptyset, \{CS_{1,2}, CS_{2,2}, \}, \{g_1\}, \psi_{1,3}, \lambda_{1,3} \rangle$$

$CS_{1,3}$ étant le CS total, il n'appartient à aucun super CS et donc ne joue aucun rôle dans un groupe permettant de former un super CS. Ses sous CS sont les flottes d'IAVs attachées au quai 1 et 2, respectivement $CS_{1,2}$ et $CS_{2,2}$. Le groupe qui forme $CS_{1,3}$ et dans lequel ses sous CS jouent des rôles est g_1. $\psi_{1,3}$ indique les rôles joués par $CS_{1,2}$ et $CS_{2,2}$ dans $CS_{1,3}$. $\lambda_{1,3}$ est la fonction indiquant si $CS_{1,3}$ peut atteindre le but global dans le niveau maximal l_3. Cette fonction prend en compte les capacités de $CS_{1,3}$ et les capacités individuels de ses sous CS, $CS_{1,2}$ et $CS_{2,2}$.

$$CS_{2,2} = \langle CS_{1,3}, \{quayTransporter, quayTransmitter\}, \{CS_{4,0}, CS_{5,0}, CS_{6,0}\}, \{g_3\}, \psi_{2,2}, \lambda_{2,2} \rangle$$

$CS_{2,2}$ est un sous CS de $CS_{1,3}$, la flotte entière d'IAVs dans le port. Il joue les rôles $quayTransporter$ et $quayTransmitter$ qui consistent à gérer les communications et le transport de conteneurs d'une flotte d'IAVs rattachée à un quai. Ses sous CS sont des IAVs individuels,

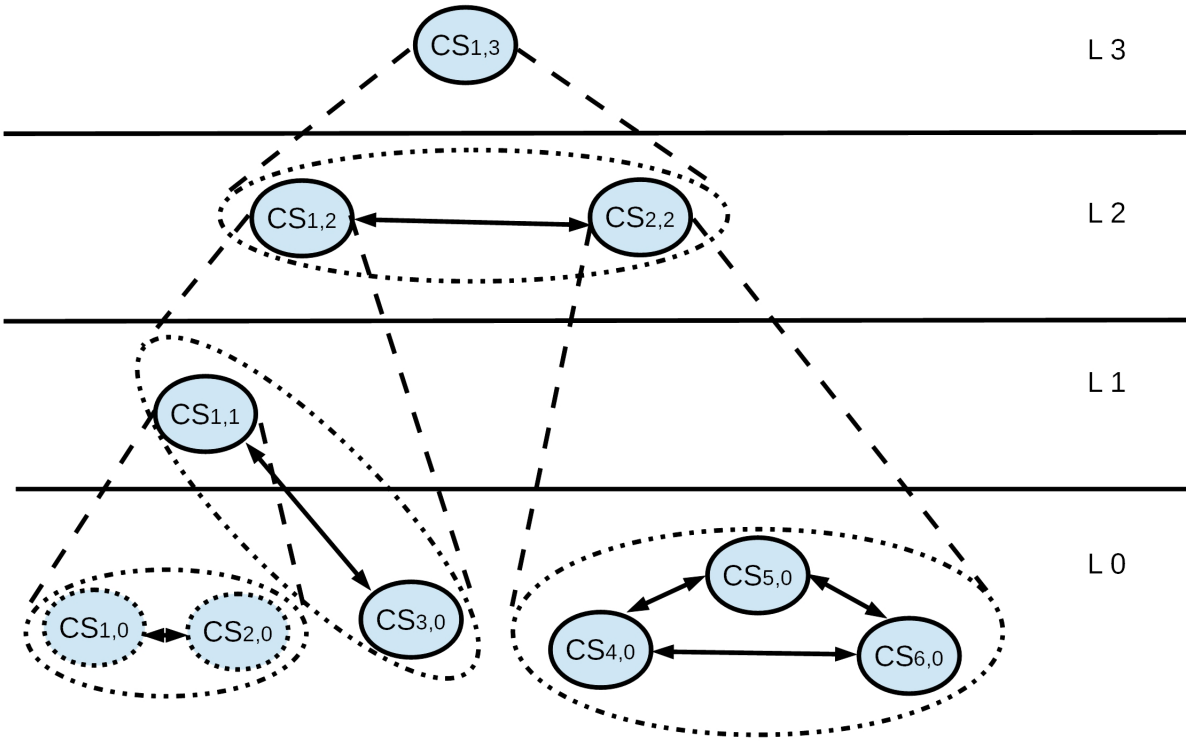

FIGURE 4.3 – Un exemple de SdS.

$CS_{4,0}$, $CS_{5,0}$ et $CS_{6,0}$. Ici $CS_{2,2}$ est constitué d'un groupe uniquement, g_3, composé de $CS_{4,0}$, $CS_{5,0}$ et $CS_{6,0}$. $\psi_{2,2}$ indique les rôles joués par $CS_{4,0}$, $CS_{5,0}$ et $CS_{6,0}$ dans $CS_{2,2}$. $\lambda_{2,2}$ est la fonction indiquant si $CS_{2,2}$ peut atteindre ses buts dans son niveau l_2. Cette fonction prend en compte les capacités de $CS_{2,2}$ et les capacités individuels de ses IAVs, $CS_{4,0}$, $CS_{5,0}$ et $CS_{6,0}$.

La spécification de groupe, g_1, peut être écrite comme suit :

$$\begin{aligned} gs_1 = \langle \{quayTransporter, quayTransmitter\}, \\ \{l_{quayTransporter} = \{l_2\}, l_{quayTransmitter} = \{l_2\}\}, \{l_3\}, \\ \emptyset, \{quayTransporter \Delta quayTransmitter\}, \\ \{nr_{quayTransporter} = (1, \infty), nr_{quayTransmitter} = (1, \infty)\}, \\ \{nc_{quayTransporter} = (1cont/hour, \infty, 1cont/hour, \infty), \\ nc_{quayTransmitter} = (2mess/sec, \infty, 2mess/sec, \infty)\}, \rangle \end{aligned}$$

Les agents CS peuvent jouer les rôles $quayTransporter$ et $quayTransmitter$ dans les groupes définis par gs_1. Ces agents CS sont dans le niveau l_2. Ces groupes produisent des agents CS dans le niveau l_3. Quand un agent CS joue un rôle dans un groupe défini par gs_1 il ne peut jouer aucun autre rôle dans ce même groupe. Cependant, s'il joue le rôle $quayTransporter$ dans ce

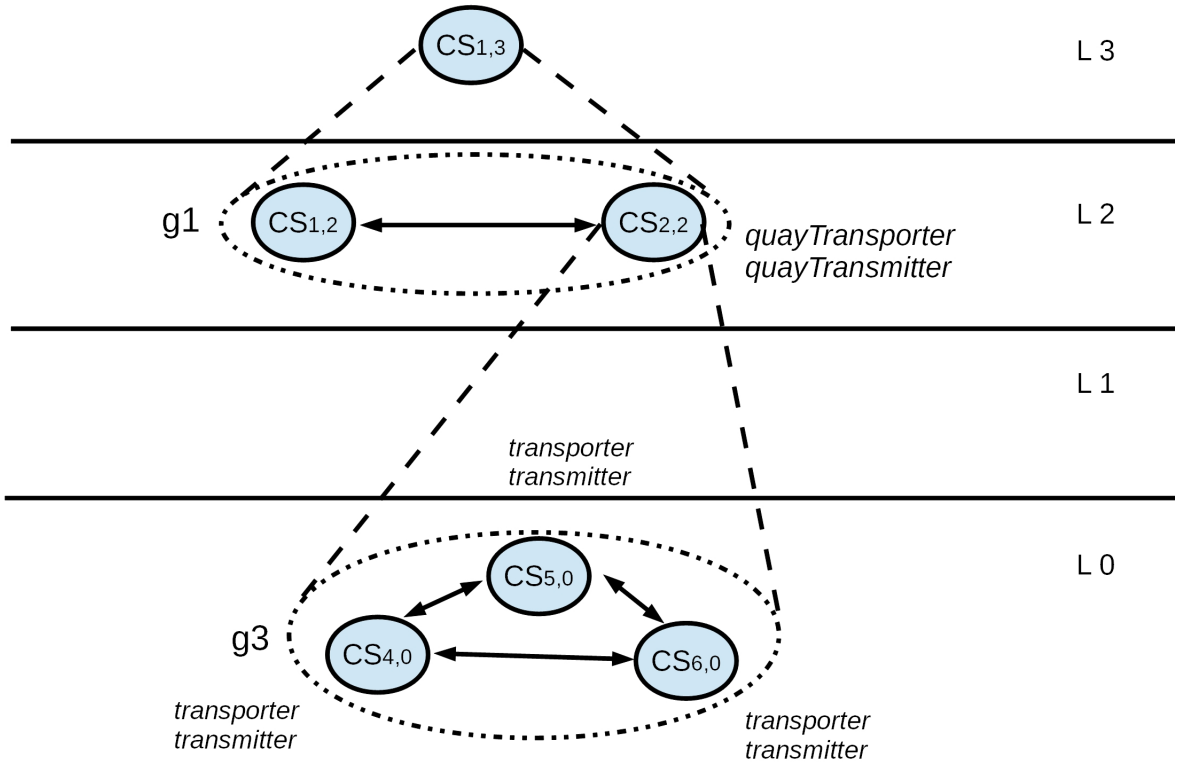

FIGURE 4.4 – $CS_{2,2}$ et ses sous CS avec leurs rôles dans leurs groupes.

groupe alors il peut jouer le rôle $quayTransmitter$ dans un autre groupe créé en utilisant gs_1 et inversement. Un groupe défini par gs_1 doit contenir, au moins, un agent CS jouant le rôle $quayTransporter$ et, au moins, un agent CS jouant le rôle $quayTransmitter$. Un agent CS jouant le rôle $quayTransporter$ doit posséder la capacité de transporter, au moins, un conteneur par heure. Un agent CS jouant le rôle $quayTransmitter$ doit posséder la capacité de transmettre, au moins, deux messages par seconde. Et la population totale d'agents jouant le rôle $quayTransmitter$ doit posséder la capacité cumulative de transmettre, au moins, deux messages par seconde.

$$CS_{4,0} = \langle CS_{2,2}, \{transporter, transmitter\}, \emptyset, \emptyset, \emptyset, \lambda_{4,0} \rangle$$
$$CS_{5,0} = \langle CS_{2,2}, \{transporter, transmitter\}, \emptyset, \emptyset, \emptyset, \lambda_{5,0} \rangle$$
$$CS_{6,0} = \langle CS_{2,2}, \{transporter, transmitter\}, \emptyset, \emptyset, \emptyset, \lambda_{6,0} \rangle$$

$CS_{4,0}$ qui est un IAV individuel est un sous CS de $CS_{2,2}$, la flotte de véhicules attachée au quai 2. Il joue les rôles *transporter* et *transmitter* qui consistent à transporter un conteneur et à communiquer avec des IAVs à distance ou l'autorité centrale du port. Comme $CS_{4,0}$ ne possède pas de sous CS, il n'existe aucun groupe formé par ces sous CS. $CS_{4,0}$ n'a pas de rôle de sous CS à définir. $\lambda_{4,0}$ est la fonction indiquant si $CS_{4,0}$ peut atteindre ses buts. Parce que $CS_{4,0}$ est un CS élémentaire, cette fonction prend seulement en compte les capacités

de l'IAV individuel instancié par $CS_{4,0}$. $CS_{5,0}$ et $CS_{6,0}$ sont définis de la même manière que $CS_{4,0}$.

La spécification de groupe g_3 peut être définie comme suit :

$$\begin{aligned} gs_3 = \langle \{transporter, transmitter\}, \\ \{l_{transporter} = \{l_0\}, l_{transmitter} = \{l_0\}\}, \{l_2\}, \\ \emptyset, \{transporter \Delta transmitter\}, \\ \{nr_{transporter} = (1, \infty), nr_{transmitter} = (1, \infty)\}, \\ \{nc_{transporter} = (1cont/hour, \infty, 1cont/hour, \infty), \\ nc_{transmitter} = (2mess/sec, \infty, 2mess/sec, \infty)\}, \rangle \end{aligned}$$

gs_3 est une spécification de groupe très similaire à gs_1 avec un changement de niveau et d'échelle.

On peut conclure cette section en signalant que dans le cas du CS total, $CS_{1,H}$, on a $CS_{n',l+1} = \emptyset$ et $R_{n,l} = \emptyset$ car le CS total ne forme aucun super CS et donc ne joue aucun rôle dans un groupe représentant un super CS. De la même manière, dans le cas des CS élémentaires n'ayant pas de sous CS virtuels (généralement les CS ayant une existence physique), on a $\mathcal{CS}_{n'',l-1} = \emptyset$, $OP = \emptyset$ et $\Psi = \emptyset$ car ces CS élémentaires ne sont formés par aucun sous CS et donc aucun CS ne joue un rôle pour former ces CS élémentaires.

4.3.3 Aspects dynamiques

Dans cette section nous présentons les mécanismes qui assurent la faisabilité du but global durant l'évolution du SdS. Le premier concerne la création de groupes dont la capacité correspond à l'accomplissement du but global et le second concerne le changement de capacité des CS. Ces mécanismes sont présentés de manière plus formelle à l'aide d'algorithmes proposés dans la section 4.4.

Il existe plusieurs articles traitant de la construction ou de la reconfiguration des SdS [Acheson 2012, Yang 2011, Khalil 2012a], certains incluant la gestion des capacités des CS [Adler 2012, Flanigan 2012], mais sans aborder le contrôle de tels systèmes.

Création des SdS

Dans notre approche nous tentons d'inciter les agents CS à adopter un structure organisationnelle appropriée pour coopérer et réaliser un but global.

À l'initialisation du SdS, en dehors du CS total et de tous les CS dotés d'une existence physique, comme la plupart des CS élémentaires, tous les CS intermédiaires doivent être instanciés. La situation est assez similaire lorsque le but global ou même le but d'un CS non élémentaire est modifié : tous ses sous CS, à l'exception de ceux sans sous CS (virtuels ou non) doivent être recréés. Ceci est fait en vue d'établir des organisations avec des capacités adaptées pour atteindre le nouveau but et ses sous buts. Pour former des organisations de CS correspondant aux buts globaux du SdS, notre modèle doit s'appuyer sur une spécification

fonctionnelle.

Les missions peuvent être vues comme des séquences d'actions des agents CS nécessaires pour atteindre un but. Les missions, dans ce modèle, dénotent le fait qu'un agent CS peut s'engager simultanément pour réaliser plusieurs buts dans la même mission. Le fait qu'une mission $\mathcal{M}x$ soit allouée au CS $CS_{n,l}$ est noté $\mathcal{M}x_{n,l}$.

Le but global est décomposé comme un graphe en forme d'arbre présenté dans la figure fig. 4.5. Chaque fois qu'un but est divisé en sous buts avec un changement d'échelle, cela montre que ces buts vont être accomplis par un CS dans un niveau et ses sous CS dans, au moins, un autre niveau. Un but g_x peut être noté avec son niveau correspondant l : $g_{x,l}$.

La figure fig. 4.6 qui suit est un graphe en forme d'arbre qui représente une possible allocation du but global au SdS présenté dans 4.2. L'agent CS $CS_{1,3}$ est engagé sur la mission $\mathcal{M}1$, $CS_{1,2}$ sur $\mathcal{M}2$, $CS_{2,2}$ sur $\mathcal{M}3$ et ainsi de suite. De cette façon, l'ensemble des missions pour la décomposition des buts est égal à $\mathcal{M} = \{\mathcal{M}1_{1,3}, \mathcal{M}2_{1,2}, \mathcal{M}3_{2,2}, \ldots\}$.

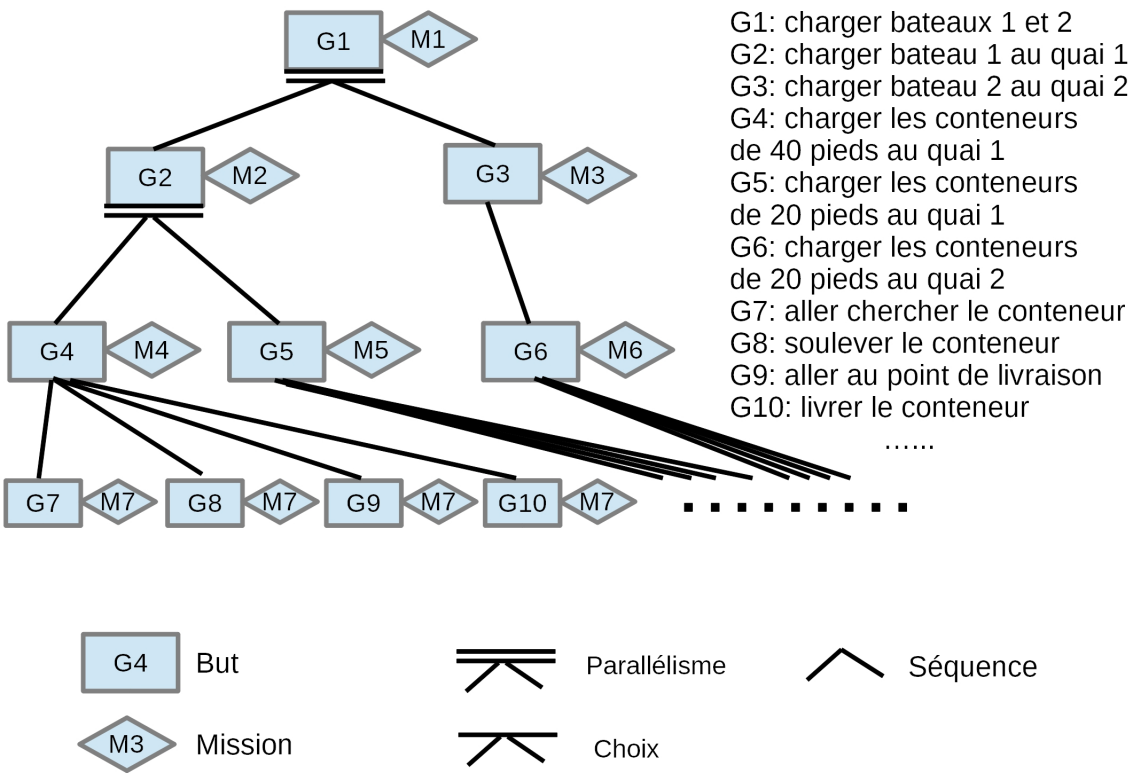

FIGURE 4.5 – La décomposition fonctionnelle du but global d'un SdS.

Il doit être noté que $CS_{1,0}$ et $CS_{2,0}$ ne doivent pas avoir de mission allouée parce qu'ils ne sont pas considérés comme des CS mais comme des CS virtuels.

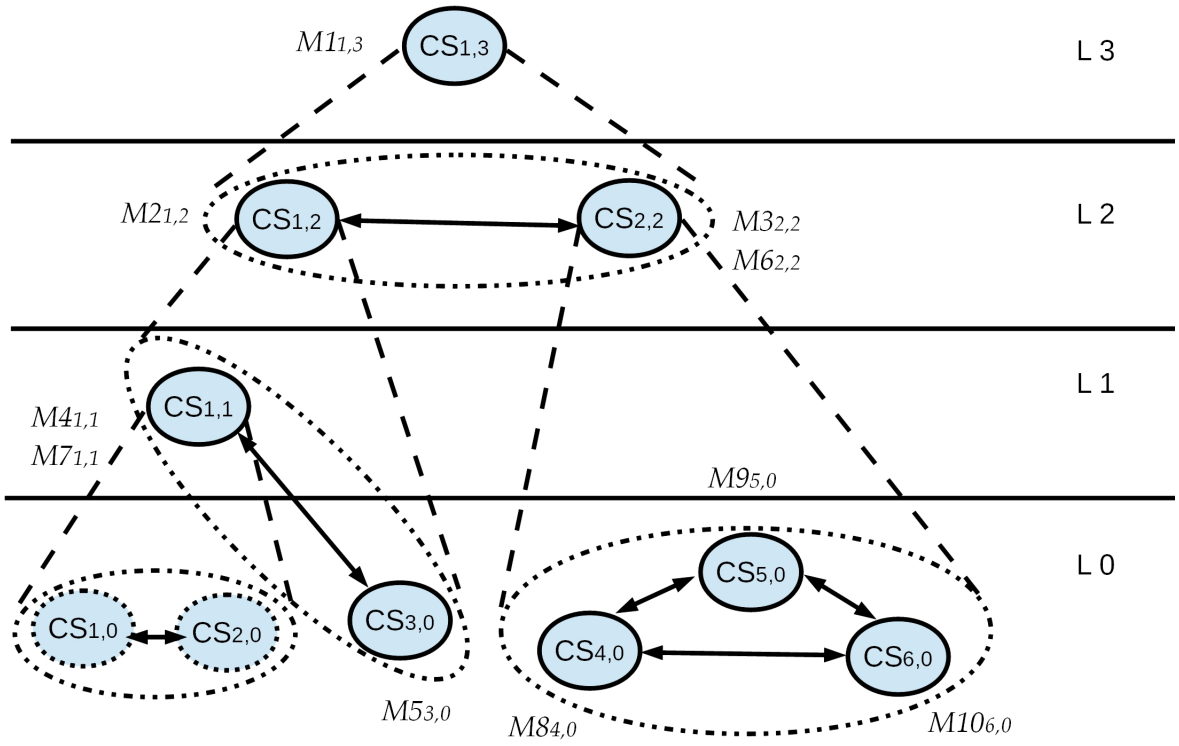

FIGURE 4.6 – Allocation partielle des missions aux CS du SdS.

Une spécification fonctionnelle adaptée à notre formalisme, notée fs, peut être définie comme suit :

$fs =< \mathcal{G}, \mathcal{M}, \mathcal{P}, mo >$, avec :

- $\mathcal{G}$: l'ensemble des buts du SdS.
- $\mathcal{M}$: l'ensemble des étiquettes associées aux missions. Les missions peuvent être allouées ($\mathcal{M}x_{n,l}$) ou non ($\mathcal{M}x$).
- $\mathcal{P}$: l'ensemble des plans globaux résultant de la décomposition des buts en arbre.
- $mo : \mathcal{M} \rightarrow 2^{\mathcal{G}}$: une fonction spécifiant pour chaque mission l'ensemble des buts attachés.

Une fois que la spécification fonctionnelle est établie, le SdS instancie les groupes correspondant aux buts en termes de capacités et d'agents disponibles, en utilisant les spécifications de groupes disponibles. Quand tous les groupes sont formés et que les agents CS élémentaires correspondent aux groupes de plus bas niveaux, tous les groupes peuvent être instanciés par des agents CS. Ensuite le SdS est entièrement formé et peut accomplir son but global.

Évolution des capacités

Comme dit auparavant, les capacités des agents CS évoluent dans le temps. À certains moments les CS peuvent perdre leurs capacités ce qui compromet leur mission et éventuellement la mission de leur super CS. Quand les capacités d'un CS sont modifiées, les capacités de son super CS sont calculées d'après les spécifications de ce CS et du groupe qui le forme. L'évolution des capacités est propagée entre les niveaux via les influences. Cela signifie que tous les agents CS avec des sous CS peuvent calculer leurs propres capacités d'après les capacités de leurs sous CS et de leur spécification de groupe. Ainsi, quand un CS perd certaines de ses capacités, il essaie d'abord de se réorganiser pour rester capable de réaliser les missions qui lui ont été allouées et si c'est impossible, il en informe son super CS qui suit le même processus et ce jusqu'à ce que l'agent CS total soit informé.

Si l'agent CS total n'est plus capable de se réorganiser, en adoptant une nouvelle structure d'organisation, il est capable d'informer le modélisateur que le SdS a échoué et qu'il n'y a aucun moyen disponible pour atteindre le but global.

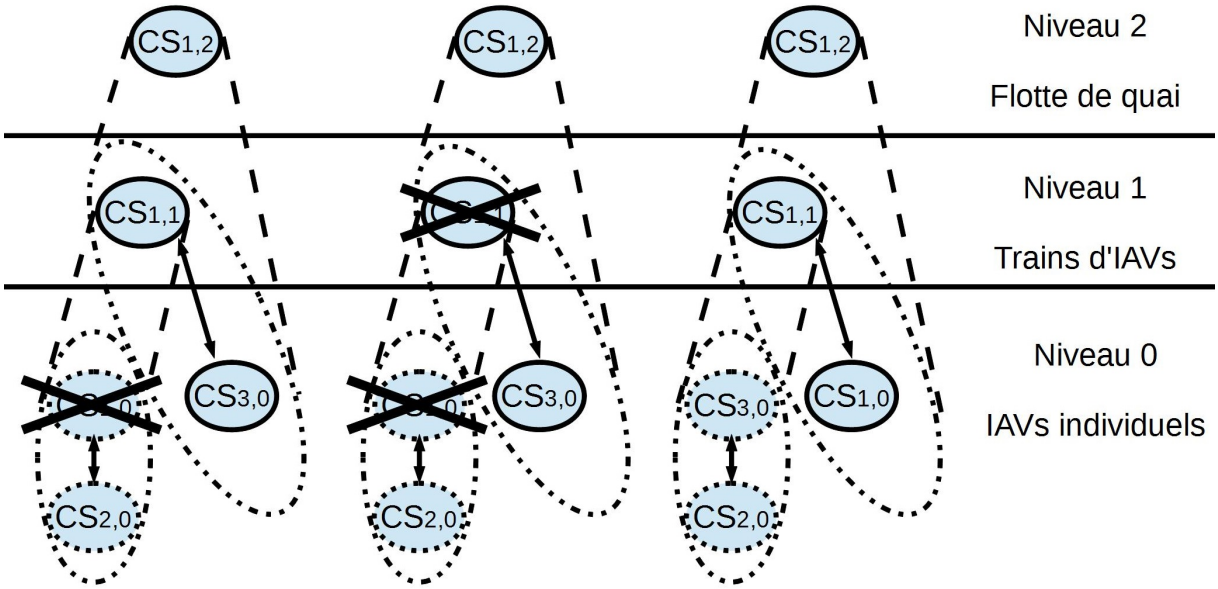

FIGURE 4.7 – Réorganisation d'un SdS guidée par l'évolution des capacités de ses CS.

Ce mécanisme peut être illustré par l'exemple suivant, représenté dans la figure fig. 4.7. L'IAV $CS_{1,0}$ a une panne et sa capacité à porter un conteneur de 40 pieds, en conjonction avec un autre IAV individuel $CS_{2,0}$, est compromise. Ainsi, la capacité du train de véhicules $CS_{1,1}$ est aussi modifiée et il ne peut pas se réorganiser avec seulement ses sous CS $CS_{1,0}$ et $CS_{2,0}$. Il ne peut plus accomplir sa mission de livrer une série de conteneurs de 40 pieds au bateau. Alors la panne est propagée à la flotte d'IAVs attachée au quai 1, $CS_{1,2}$. $CS_{1,2}$ peut se réorganiser en échangeant les IAVs $CS_{3,0}$ et $CS_{1,0}$. Ceci est fait, tout en maintenant la structure de $CS_{1,2}$.

Instanciation des CS virtuels

Dans cette section nous considérons le cas particulier de la création des CS non élémentaires dont les sous CS perdent leur statut de CS.

Dans un modèle SMA représentant un SdS, chaque CS doit être instancié par un agent CS qui respecte la définition et les caractéristiques des CS vues plus haut. En particulier, les CS élémentaires et le CS total doivent posséder leur propre agent CS instancié durant toute la simulation. Quand un CS cesse de respecter les caractéristiques d'un SdS, il cesse d'être un CS mais lui et ses sous CS interagissent avec le SdS, même si, ils ne peuvent plus être considérés comme des CS. Les agents représentant ces CS ne doivent pas être supprimés dans le modèle mais remplacés par des agents CS virtuels qui indiquent leur statut de CS potentiel.

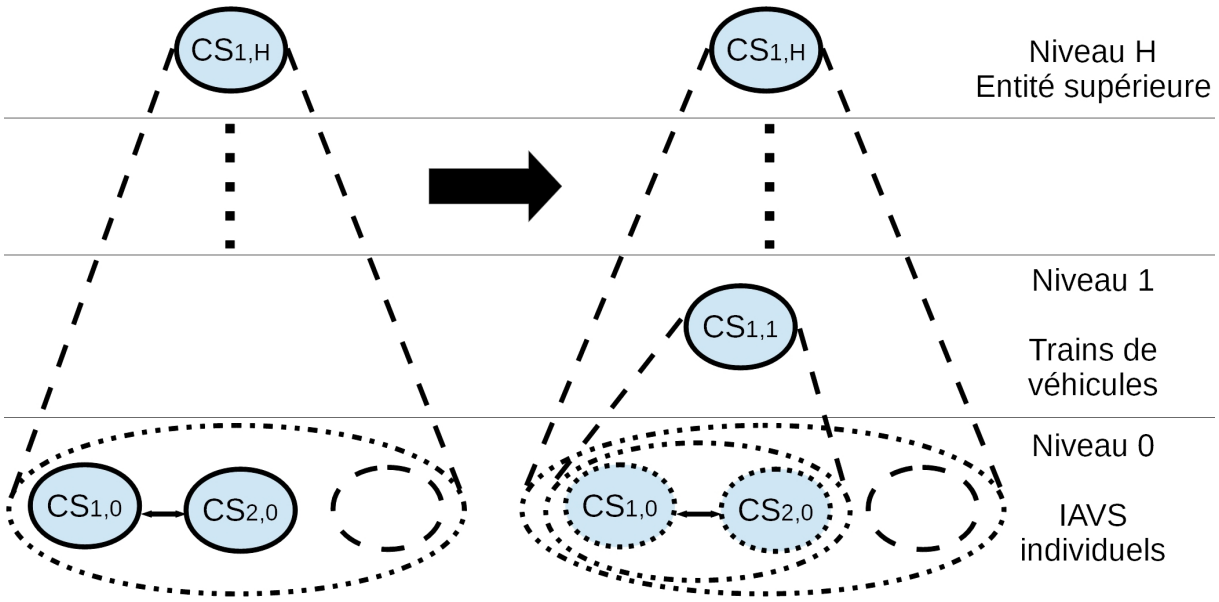

FIGURE 4.8 – Création d'un super CS par l'agrégation de sous CS en groupe(s) et mise à jour de leur statut.

Pour illustrer ce point, prenons un exemple simple, représenté dans la figure fig. 4.8. Deux agents CS, $CS_{1,0}$ et $CS_{2,0}$, représentant des IAVs individuels en tant que CS élémentaires dans le niveau l_0 et un autre agent, $CS_{1,1}$, représentant le train formé par ces deux véhicules en tant que super CS dans le niveau l_1, plus haut que l_0. Quand deux IAVs sont reliés ensemble pour porter un conteneur lourd, et l'un devient le meneur de cette tâche, $CS_{1,0}$ et $CS_{2,0}$ perdent leur statut de CS parce qu'ils violent les caractéristiques d'indépendances opérationnelle et managériale et de dispersion géographique. $CS_{1,1}$, après l'agrégation de $CS_{1,0}$ et $CS_{2,0}$, représente un CS élémentaire parce que ses sous CS ont perdu leur statut de CS. Ainsi les agents CS $CS_{1,0}$ et $CS_{2,0}$ sont remplacés par des agents CS virtuels parce qu'ils sont toujours utilisés pour déterminer l'évolution et les capacités de $CS_{1,1}$ et peuvent, de nouveau, représenter des CS non virtuels dans le futur, quand les liens d'agrégation entre $CS_{1,0}$ et $CS_{2,0}$ seront supprimés.

4.3.4 Illustration des aspects dynamiques

Pour illustrer ces aspects dynamiques nous considérons, encore une fois, le SdS présenté en section 4.2 et dont la distribution des missions a été exposée dans la fig. 4.6 avec notre formalisme.

Avant que le SdS soit construit, son modèle contient seulement le CS total au plus haut niveau, $CS_{1,3}$, et tous les CS élémentaires au plus bas niveau, $CS_{1,0}$, $CS_{2,0}$, $CS_{3,0}$, $CS_{4,0}$, $CS_{5,0}$ et $CS_{6,0}$ et donc tous les agents CS correspondants. Les autres éléments du modèle sont un ensemble de niveaux, avec les graphes hiérarchiques, d'influences et de perceptions pour appréhender les liens entre niveaux, un ensemble de définitions d'agents CS pour chaque niveau, un ensemble de spécifications de groupes correspondant à ces définitions d'agents et une spécification fonctionnelle à peu près vide. Uniquement le but global du SdS est connu et alloué au CS total. Sa spécification fonctionnelle est :

$$fs_{SoS} = \langle \{g1_3\}, \{\mathcal{M}1_{1,3}\}, \emptyset, \{\mathcal{M}1 \rightarrow \{g1_3\}\} \rangle$$

D'après fig. 4.5 et 4.6, une fois que le modèle du SdS est totalement construit, sa spécification fonctionnelle est :

$$\begin{aligned} fs_{SoS} = \langle \{&g1_3, g2_2, g3_2, g4_1, g5_0, g6_2, g7_1, g8_0, \ldots\}, \\ \{&\mathcal{M}1_{1,3}, \mathcal{M}2_{1,2}, \mathcal{M}3_{2,2}, \mathcal{M}4_{1,1}, \mathcal{M}5_{3,0}, \mathcal{M}6_{2,2}, \mathcal{M}7_{1,1}, \\ &\mathcal{M}8_{4,0}, \mathcal{M}9_{5,0}, \mathcal{M}10_{6,0} \ldots\}, \{"g1_3 = g2_2, g3_2", \\ &"g2_2 = g4_1, g5_0", "g3_2 = g6_2", \ldots\}, \{\mathcal{M}1 \rightarrow \{g1_3\}, \\ &\mathcal{M}2_{1,2} \rightarrow \{g2_2\}, \mathcal{M}3_{2,2} \rightarrow \{g3_2\}, \mathcal{M}4_{1,1} \rightarrow \{g4_1\}, \\ &\mathcal{M}5_{3,0} \rightarrow \{g5_0\}, \ldots, \mathcal{M}7_{1,1} \rightarrow \{g7_1, g8_1, g9_1, g10_1\}, \ldots\} \rangle \end{aligned}$$

4.3.5 Respect des caractéristiques fondamentales

Dans cette section, nous montrons que les caractéristiques d'un SdS dirigé sont nécessairement respectées dans les modèles à base d'agents générés en utilisant notre formalisme.

Il est important de considérer que dans les trois premières caractéristiques les agents CS $CS_{i',j'}$ et $CS_{i'',j''}$ d'un système $CS_{i,j}$ ne sont pas reliés par une relation hiérarchique directe :

$$\begin{aligned} \forall CS_{i',j'}, CS_{i'',j''} \in CS_{i,j}, \\ CS_{i',j'} \notin CS_{i'',j''} \wedge CS_{i'',j''} \notin CS_{i',j'} \end{aligned}$$

Suivent les cinq caractéristiques que tout SdS doit respecter (voir section 2.2.2) et leur traduction dans un modèle SMA représentant un SdS :

1. **Indépendance opérationnelle** : Les agents CS $CS_{i',j'}$ et $CS_{i'',j''}$ d'un système $CS_{i,j}$ sont indépendants d'une manière opérationnelle *ssi* ils sont instanciés par des agents CS et

possèdent leurs propres variables, s_a, représentant leurs ressources. L'ensemble des agents CS est noté $\mathcal{A}$.

$$\begin{aligned} \forall CS_{i',j'}, CS_{i'',j''} \in CS_{i,j}, \\ CS_{i',j'}, CS_{i'',j''} \in \mathcal{A} \wedge \\ s_{CS_{i',j'}} \neq \emptyset \wedge s_{CS_{i'',j''}} \neq \emptyset \wedge \\ s_{CS_{i',j'}} \cap s_{CS_{i'',j''}} = \emptyset \end{aligned}$$

La figure fig. 4.9 montre ainsi que $CS_{4,0}$ est indépendant opérationnellement, mais que $CS_{1,0}$ ne l'est pas, parce qu'il partage une partie de ses ressources avec $CS_{2,0}$.

2. **Indépendance managériale** : Les agents CS $CS_{i',j'}$ et $CS_{i'',j''}$ d'un système $CS_{i,j}$ sont indépendants d'une manière managériale *ssi* ils ne partagent aucune partie de leurs propres missions, $\mathcal{M}_{i',j'}$ et $\mathcal{M}_{i'',j''}$.

$$\begin{aligned} \forall CS_{i',j'}, CS_{i'',j''} \in CS_{i,j}, \\ \mathcal{M}_{i',j'} \cap \mathcal{M}_{i'',j''} = \emptyset \end{aligned}$$

Dans les SdS non spécifiques, cette caractéristique dit que pour chaque agent CS, aucun autre agent CS ne lui donne d'ordres directes et seul son agent super CS peut lui allouer des missions. Mais parce que nous considérons des SdS dirigés par une hiérarchie stricte, la notation ci-dessus est suffisante. Chaque agent CS gère ses propres missions indépendamment des autres agents CS.

Ainsi, $CS_{4,0}$ dans la fig. 4.9 est indépendant d'une manière managériale, mais $CS_{1,0}$ ne l'est pas, parce il partage ses missions avec $CS_{2,0}$.

3. **Distribution géographique** : Les agents CS $CS_{i',j'}$ et $CS_{i'',j''}$ d'un système $CS_{i,j}$ sont distribués géographiquement *ssi* leurs états physiques, respectivement $\phi_{a'}$ et $\phi_{a''}$, sont totalement distincts et non dépendants directement. Cette dernière propriété peut être formulée par le fait qu'il n'est pas possible de calculer directement tout ou partie de l'état physique d'un agent CS (en utilisant une fonction f) à partir de tout ou partie de l'état physique d'un autre agent CS.

$$\begin{aligned} \forall CS_{i',j'}, CS_{i'',j''} \in CS_{i,j}, \\ \phi_{CS_{i',j'}} \cap \phi_{CS_{i'',j''}} = \emptyset \wedge \\ \big(\forall \phi'_{CS_{i',j'}} \subset \phi_{CS_{i',j'}}, \forall \phi'_{CS_{i'',j''}} \subset \phi_{CS_{i'',j''}}, \\ \nexists f : \Phi \to \Phi, f(\phi'_{CS_{i',j'}}) = \phi'_{CS_{i'',j''}} \\ \vee f(\phi'_{CS_{i'',j''}}) = \phi'_{CS_{i',j'}}\big) \end{aligned}$$

Dans ce cas, les échanges physiques entre agents CS ne sont pas autorisés pour accomplir des missions. Seuls les échanges d'informations entre les agents CS sont autorisés.

Pratiquement, dans un modèle à base d'agents il doit être noté que si chaque agent CS est situé dans un environnement doté d'une métrique alors celle-ci permet de déterminer qu'un agent CS n'est physiquement lié à aucun autre agent CS.

La fig. 4.9 montre que $CS_{4,0}$ est géographiquement distribué, mais $CS_{1,0}$ et $CS_{2,0}$ ne le sont pas. Parce que, dans le cas où $CS_{1,0}$ est le meneur du train de véhicules $CS_{1,1}$ et $CS_{2,0}$ est un suiveur, alors il existe une fonction f, telle que $f(pos_{CS_{1,0}}) = pos_{CS_{2,0}}$, où $pos_{CS_{1,0}}$ et $pos_{CS_{2,0}}$ désignent la partie de l'état physique de ces deux agents CS qui représentent leur position.

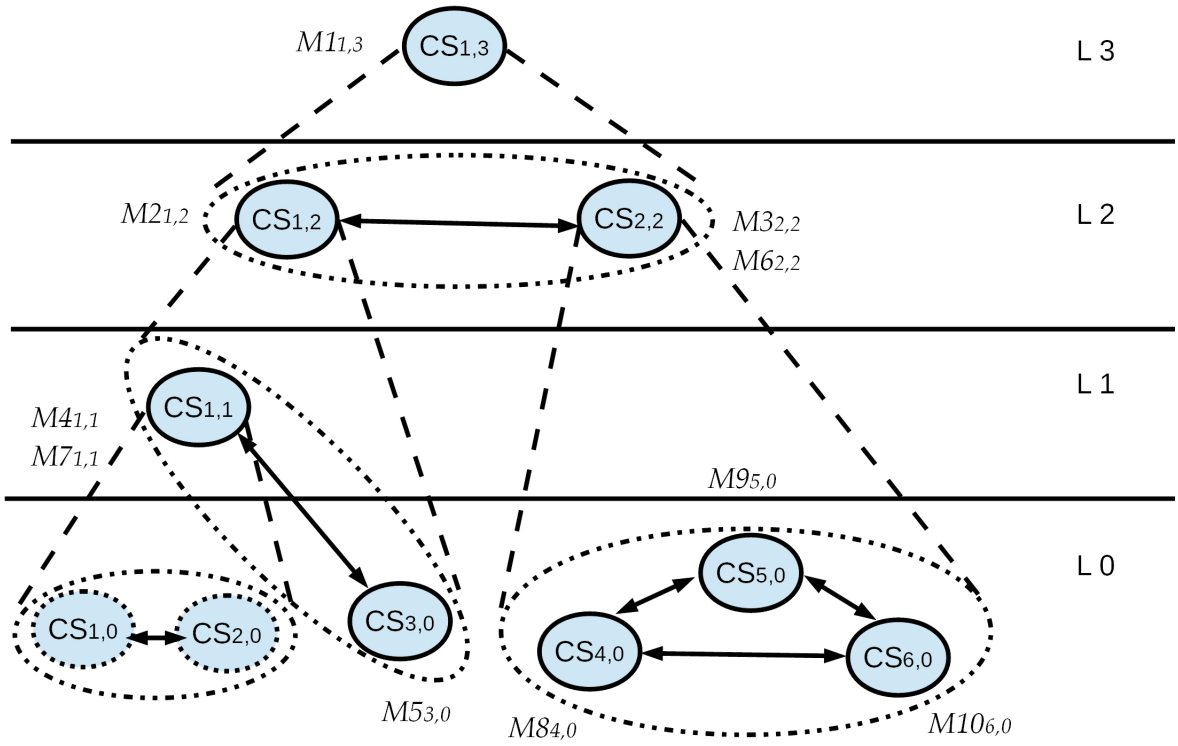

FIGURE 4.9 – Caractéristiques d'un SdS.

4. **Comportement émergent** : un ensemble d'agents CS $CS_{i',j'}$ d'un système $CS_{i,j}$ coopère *ssi* les agents CS coopèrent pour réaliser une mission globale $\mathcal{M}_{i,j}$ du système formé d'un ensemble de missions $\mathcal{M}_{i',j'}$, qu'aucun agent CS de $CS_{i,j}$ ne peut réaliser seul.

$$\forall CS_{i',j'} \in CS_{i,j},$$
$$\bigcup \mathcal{M}_{i',j'} = \mathcal{M}_{i,j}$$

Dit d'une autre façon, un modèle à base d'agents représente un SdS *ssi* les actions jointes

de plusieurs agents CS sont obligatoirement nécessaires pour atteindre le but global.

L'agent CS $CS_{2,2}$ de la fig. 4.9 ne peut pas réaliser ses missions $\mathcal{M}3_{2,2}$ et $\mathcal{M}6_{2,2}$ uniquement à travers la coopération de ses sous CS $CS_{4,0}$, $CS_{5,0}$ et $CS_{6,0}$ qui remplissent les missions $\mathcal{M}8_{4,0}$, $\mathcal{M}9_{5,0}$ et $\mathcal{M}10_{6,0}$.

5. **Développement évolutif** : Cette caractéristique est différente des autres parce qu'elle ne peut être observée que durant l'exécution du SdS. Ainsi, il s'agit d'une caractéristique émergente. En particulier, cette caractéristique s'exprime dans trois situations : a) un agent CS est ajouté au SdS, b) un agent CS du SdS est supprimé ou C) un agent CS remarque que l'une de ses missions est impossible à accomplir compte tenu de sa fonction de capacité.

   Pour permettre au SdS d'accepter de nouveaux agents CS comme dans la situation a) il est nécessaire que les nouveaux agents CS soient placés dans des groupes dont la spécification contient des rôles qui n'affectent pas les capacités de l'agent CS formé par ces groupes. Dans ce cas, l'ajout de nouveaux agents CS n'affecte pas la capacité du SdS à réaliser son but global. Pour tirer avantage de cet ajout, pour améliorer la capacité du SdS, un algorithme est nécessaire pour réorganiser le SdS de manière optimale, en incluant les nouveaux agents CS. Ce genre d'algorithme est très spécifique à l'architecture du problème du SdS, cependant un exemple d'un tel algorithme sera proposé par la suite, dans ce chapitre.

   Les situations b) et c) sont assez similaires. En effet, quand la suppression d'un agent CS n'affecte pas les capacités de son super CS, il n'affecte pas non plus la capacité du SdS dans sa globalité. Et quand il affecte la capacité de son super CS, celui-ci est dans la situation c). Dans ce cas, la perte de capacité doit être communiquée via des influences, les missions impossibles à réaliser doivent être désallouées et la réorganisation du SdS doit être effectuée comme dans l'algorithme précédent.

   Un système $CS_{i,j}$, formé par plusieurs agents CS $CS_{i',j'}$ suit un développement évolutif *ssi* lorsque la mission allouée à l'un de ses agents CS $\mathcal{M}x_{i',j'}$ est détectée comme impossible à résoudre d'après sa fonction de capacité δ et l'état actuel du niveau j' $\lambda_{i,j}$ et $\delta_{j'}(t)$, cette mission n'est plus allouée à cet agent CS. Si cette mission n'est plus allouée à aucun agent CS alors elle est rendue libre dans l'ensemble des missions $\mathcal{M}$:

$$\begin{aligned} &\forall CS_{i',j'} \in CS_{i,j}, \\ &\lambda_{i,j}(\delta_{j'}(t), \delta(\mathcal{M}x_{i',j'})) = 0 \\ &\rightarrow \mathcal{M}x_{i',j'} \notin \mathcal{M} \cup \mathcal{M}x \in \mathcal{M} \end{aligned}$$

   Une fois qu'une mission est libérée, un algorithme doit allouer cette mission à un autre agent CS pour réorganiser le SdS pour réaliser cette mission.[Khalil 2012a] a présenté des réorganisations de SdS similaires sans évoquer un algorithme formalisé.

   La fig. 4.10 montre trois étapes dans la vie d'un SdS qui illustrent cette caractéristique. Nous conservons l'exemple du SdS que nous avons détaillé depuis le début de ce chapitre. Nous nous intéressons en particulier à l'agent CS $CS_{1,2}$ qui représente la flotte d'IAVs

attachée au quai 1. La première étape montre l'état initial du SdS avec toutes les missions allouées à $CS_{1,2}$ et à ses sous CS. La seconde étape est obtenue par la suppression de $CS_{3,0}$, l'IAV individuel qui s'était engagé sur la mission $\mathcal{M}5$ correspondant au chargement des conteneurs de 20 pieds sur le quai 1. La mission $\mathcal{M}5$ doit être réallouée pour que le CS $CS_{1,2}$ puisse mener à bien sa mission $\mathcal{M}2_{1,2}$ contenant $\mathcal{M}5$.

Supposons maintenant que le seul sous CS à qui $CS_{1,2}$ peut allouer cette mission soit le train d'IAVs $CS_{1,1}$. De cette manière celui-ci va prendre en charge le transport des conteneurs de 40 pieds et ceux de 20 pieds. Ainsi la mission $\mathcal{M}5$ va être réallouée à $CS_{1,1}$ et donc devenir $\mathcal{M}5_{1,1}$ par *nécessité*. La troisième étape représente le même SdS après l'ajout de $CS_{3,0}$ et $CS_{4,0}$ au SdS et réallocation de la mission $\mathcal{M}5$ à ces nouveaux CS. Cette réallocation a été faite par *optimisation* : même si $CS_{1,1}$ pouvait prendre en charge $\mathcal{M}5$ en plus de $\mathcal{M}4$ et $\mathcal{M}7$, cette nouvelle distribution des missions permet de mieux répartir la charge de travail sur les sous CS de $CS_{1,2}$ et donc d'accélérer la réalisation de $\mathcal{M}2_{1,2}$.

Il est à noter que $CS_{3,0}$ dans les étapes une et trois n'est pas forcément le même agent CS. De même, ce n'est pas exactement la mission $\mathcal{M}5$ qui a été allouée à $CS_{3,0}$ et à $CS_{4,0}$, car si c'était le cas ces deux CS engagés sur la même mission violeraient la caractéristique d'indépendance managériale. À la place $\mathcal{M}5$ et ses buts associés ont été arbitrairement divisés en deux missions $\mathcal{M}51$ et $\mathcal{M}52$ qui consistent à effectuer une des deux moitiés de $\mathcal{M}5$. Chacune de ces missions concernent le transport de la moitié des conteneurs du quai 1.

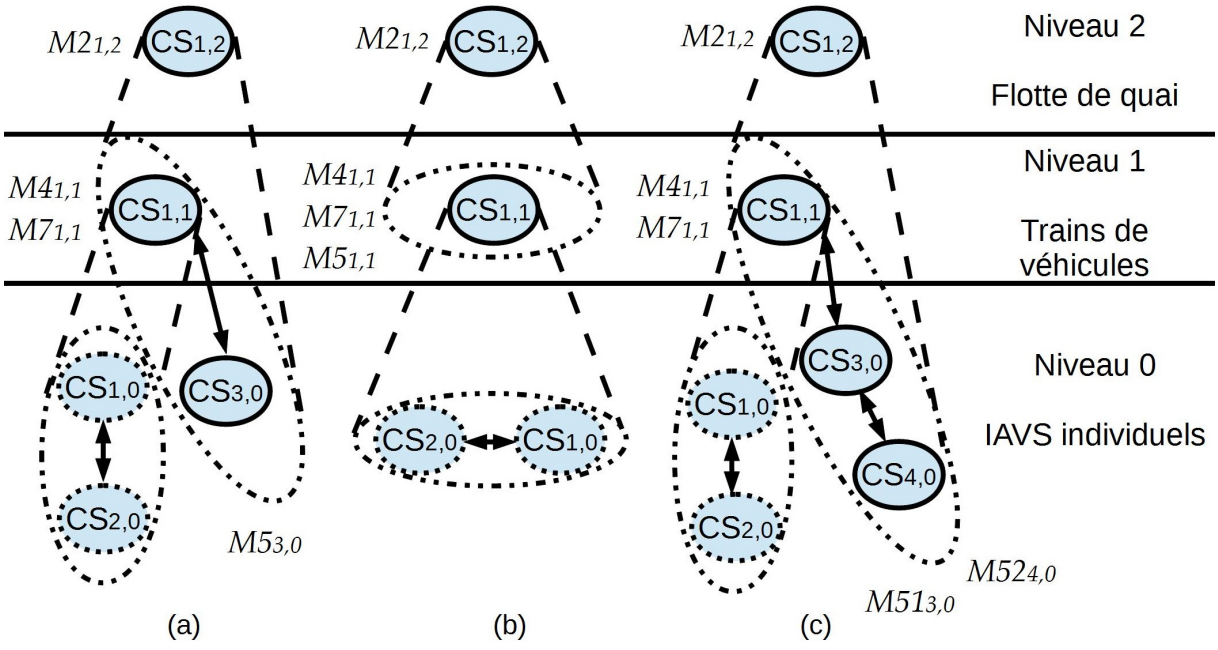

FIGURE 4.10 – Trois phases de la vie d'un SdS : a) état initial, b) suppression $CS_{3,0}$ et c) ajout de $CS_{3,0}$ et $CS_{4,0}$.

4.4 Algorithmes

Dans cette partie nous présentons de manière formelle les mécanismes qui assurent la faisabilité du but global par le SdS lors de sa création et lors de l'évolution de ces capacités, à travers deux algorithmes. Ces algorithmes visent à combler un manque dans la littérature traitant de la modélisation des SdS. Ce sont des algorithmes génériques, utilisables sur toute instance de SdS modélisée grâce aux outils que nous proposons. Cependant, ils peuvent être améliorés et ne sont pas optimisés pour accélérer le calcul d'une structure de SdS satisfaisant un but global ou pour trouver la meilleure structure de SdS pour cela. Ainsi, de nombreux éléments comme la création de l'arbre de décomposition du but global ou l'ordre dans lequel les différentes définitions d'agents CS ou de groupes de CS sont considérées restent à la charge du modélisateur.

4.4.1 Initialisation du SdS

Dans cette partie nous expliquons en détails l'algorithme guidant la création d'un SdS. Cette création revient à instancier les CS et groupes de CS et à leur allouer les missions attachées aux buts tirés de la décomposition du but global. On cherche à créer un SdS ayant la même structure que l'arbre de décomposition fonctionnelle du but global du SdS (voir section 4.3.3). L'exception à ce processus est la présence de choix dans cet arbre. On commence par "fixer" l'arbre de décomposition du but global. Pour cela on supprime tous les choix de l'arbre en sélectionnant de manière arbitraire toujours la première possibilité de choix. On fait un parcours descendant de l'arbre en partant du CS total, on instancie le groupe le représentant puis les CS qui constituent ce groupe, ensuite on fait de même avec ces CS et ce jusqu'à inclure les CS élémentaires existant dans le SdS.

La figure fig. 4.11 présente les trois étapes basiques dans la création d'un SdS : 1) étape initial un CS non élémentaire $CS_{1,2}$ possédant une mission allouée $M2$, 2) création du groupe représentant ce CS dans des niveaux inférieurs $g2$, 3) création des CS $CS_{1,1}$ et $CS_{3,0}$ qui forment ce groupe dans des niveaux où est présent $g2$ et allocation des missions attachées aux sous buts du but attachés à $M2$ à ces CS.

Nous rappelons qu'avant la création du SdS les éléments initiaux existants sont :

1. le CS total $CS_{1,H}$ qui représente le SdS dans son intégralité,
2. les CS élémentaires dotés d'une existence physique $\mathcal{CS}_0$,
3. une spécification fonctionnelle complète $fs =< \mathcal{G}, \mathcal{M}, \mathcal{P}, mo >$ dont la mission attachée au but de plus haut niveau est allouée à $CS_{1,H}$,
4. un ensemble de définition de CS possibles pour chaque niveau $\mathcal{CS}^l = \{CS^l_{def1}, CS_{def2}, CS^l_{def3}, \ldots\}$,
5. un ensemble de définitions de groupes de CS possibles pour chaque niveau $\mathcal{G}s^l = \{gs^l_{def1}, gs^l_{def2}, gs^l_{def3}, \ldots\}$.

Les définitions contenues dans $\mathcal{CS}^l$ et $\mathcal{G}s^l$ respectent le formalisme des agents CS et des groupes de CS présentés dans la section 4.3.1.

Cet algorithme se déroule selon un parcours descendant qui est répété sur chaque CS non élémentaire. Considérons ainsi un CS non élémentaire, $CS_{n,l}$ auquel une mission m est allouée. Le but B est attaché à la mission m. La première étape consiste à instancier un groupe $g_{n,l}$ qui représente l'ensemble des sous CS de $CS_{n,l}$ et jouant un rôle afin de réaliser m. Il convient

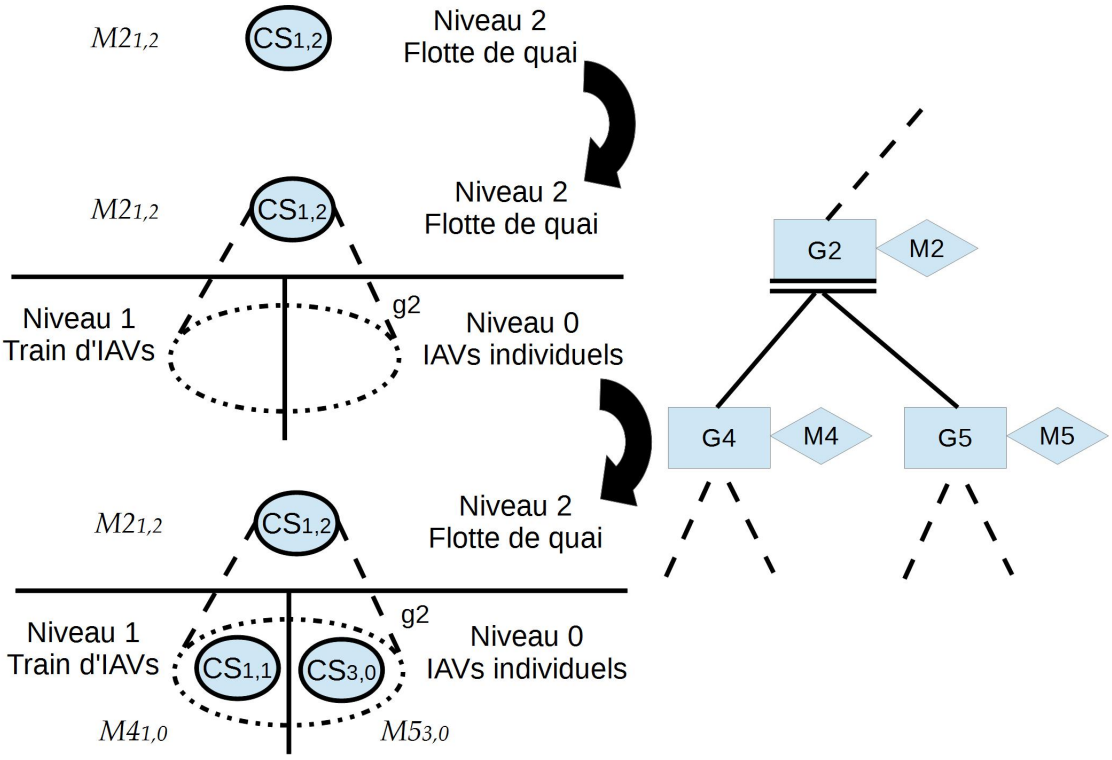

FIGURE 4.11 – Les trois étapes de base de la création d'un SdS : 1) état initial, 2) création du groupe et 3) création des sous CS et allocation des missions (à gauche) et l'arbre partiel de décomposition du but correspondant (à droite).

donc de choisir une définition de groupe qui permet de réaliser la mission m et donc de former un CS dans le niveau l, en tenant compte des capacités que peut produire le groupe instancié (calculées grâce au nombre minmum de rôles et d'agents jouant ces rôles, nr, et nc). La seconde étape consiste à créer les agents CS qui vont former le groupe $g_{n,l}$ et instancier ces CS dans leurs niveaux respectifs. Pour cela on sélectionne les définitions de CS qui permettent de créer le nombre minimum de CS nécessaire pour former $g_{n,l}$. La troisième étape consiste à allouer ces missions à ces nouveaux agents CS. Ensuite ce cycle est appliqué aux nouveaux CS auxquels on a alloué les missions qui composent m. Les trois étapes principales de cet algorithme sont illustrées dans la figure 4.11. Lors de la seconde étape, si les agents CS nécessaires pour créer le groupe sont déjà instanciés alors ils seront intégrés dans le groupe nouvellement créé au lieu de chercher à créer de nouveaux agents CS pour former ce groupe.

S'il est impossible de créer un élément lors du parcours descendant pendant la création du SdS, alors on effectue le parcours ascendant qui est décrit dans ce paragraphe. Si lors de l'étape deux il n'y a pas de définition de CS disponible qui permette de réaliser m, alors on repasse à l'étape un et on choisit une définition de groupe différente qui permet de réaliser m. Enfin, s'il n'y a pas de définition de groupe disponible répondant à ces critères alors on désalloue m de $CS_{n,l}$ et on effectue un choix différent dans l'arbre de décomposition du

but global, on sélectionne une mission alternative à m, m' et on alloue m' à $CS_{n,l}$ si c'est possible.

Si cette allocation est impossible, alors on supprime $CS_{n,l}$ et on crée un nouveau CS qui peut effectuer m', $CS_{n',l}$ et on lui alloue m'. Au cas où il n'y a pas de choix possible dans l'arbre de décomposition des buts alors on effectue le parcours ascendant décrit dans ce paragraphe en l'appliquant sur le super CS des CS que l'on est pas arrivé à instancier. Finalement, si le CS total se retrouve dans ce cas de figure alors il indique au modélisateur que le but global n'est pas atteignable étant donné la configuration du SdS. Si le problème rencontré initialement est résolu alors on continue le parcours descendant.

Pour commencer la création du SdS il faut appeler la fonction *instancier* en lui donnant comme arguments le CS total et la mission attachée au but global du SdS : $instancier(CS_{1,H}, m_H)$ Cette fonction est détaillée dans l'algorithme 3. Ce qui suit est la description des fonctions appelées dans l'algorithme 3.

$instancier(CS_{n,l}, m)$: instancie le CS $CS_{n,l}$ engagé sur la mission m, et indirectement tous les sous CS de $CS_{n,l}$ qui doivent réaliser m par leurs actions communes. Renvoie $false$ si le super CS ne peut s'instancier, $true$ sinon.

$instancierSousCS(CS_{n,l}, m)$: pour chaque sous CS présent dans le groupe , applique la fonction *instancier* à ce CS. Renvoie $true$ si tous les sous CS de $CS_{n,l}$ sont instanciés ou alors la valeur de $instancier(CS_{n,l}, m)$.

$creerGroupe(CS_{n,l}, m)$: instancie le groupe qui représente $CS_{n,l}$ pour réaliser m. Si ce groupe existe déjà alors il est détruit et un groupe alternatif est créé grâce aux définitions de groupes disponibles. Renvoie le groupe créé ou l'élément vide s'il n'y a pas de définition de groupe alternative possible.

$creerSousCS(CS_{n,l}, g_{n,l})$: instancie les sous CS de $CS_{n,l}$ nécessaires pour former le groupe $g_{n,l}$. Renvoie l'ensemble des sous CS créé ou l'élément vide s'il n'est pas possible d'instancier suffisamment de CS pour former le groupe $g_{n,l}$.

$allouerSousMissions(\mathcal{CS}_{n,l}, m)$: alloue les missions qui composent m à l'ensemble des CS $\mathcal{CS}_{n,l}$ qui forment le groupe engagé pour réaliser m.

$allouerNouvelleMission(CS_{n,l}, m)$: sélectionne une mission alternative à m, et l'alloue à $CS_{n,l}$. Renvoie cette mission ou l'élément vide s'il n'y a pas de mission alternative possible.

4.4.2 évolution des capacités du SdS

Cet algorithme est appelé lorsqu'un CS $CS_{n,l}$ constate qu'il ne peut plus résoudre une de ses missions, m, à cause de l'évolution de ses capacités. Si ce CS ne possède pas de sous CS il informe son super CS de sa défaillance sinon il tente de se réorganiser en allouant m à un autre de ses sous CS. Enfin, si le CS total ne peut pas allouer m à l'un de ses sous CS alors la structure du SdS doit être modifiée. Tous les CS et groupes en dehors du CS total et de ceux dotés d'une existence physique dans le SdS sont supprimés et toutes les missions, sauf celle attachée au but global sont désallouées. Ensuite le SdS est réinitialisé avec l'algorithme présenté précédemment. Cet algorithme est donc un parcours ascendant puis descendant du SdS.

Algorithm 3: Algorithme d'instanciation des sous CS d'un CS : $instancier(CS_{n,l}, m)$.

Input: $CS_{n,l}, m$
Output: $true/false$
$g_{n,l} \leftarrow creerGroupe(CS_{n,l}, m)$;
if $g_{n,l} \neq \emptyset$ **then**
  $\mathcal{CS}_{n,l} \leftarrow creerSousCS(CS_{n,l}, g_{n,l})$;
  if $\mathcal{CS}_{n,l} \neq \emptyset$ **then**
    $allouerSousMissions(\mathcal{CS}_{n,l}, m)$;
    return $instancierSousCS(CS_{n,l}, m)$
  else
    return $instancier(CS_{n,l}, m)$
  end
else
  $m' \leftarrow allouerNouvelleMission(CS_{n,l}, m)$;
  if $m' \neq \emptyset$ **then**
    return $instancier(CS_{n,l}, m')$
  else
    if $CS_{n,l} == CS_H$ **then**
      return false;
    else
      return $instancierCS(CS_{n'',l+1})$
    end
  end
end

Pour débuter la reconfiguration du SdS il faut appeler la fonction $reconfigurer$ en lui donnant comme arguments le CS défaillant et la mission attachée au but global du SdS : $instancier(CS_{n,l}, m_H)$ Cette fonction est détaillée dans l'algorithme 4. Ce qui suit est la description des fonctions appelées dans l'algorithme 4.

$echangerCS(CS_{n,l}, g_{n,l})$: alloue m à un des sous CS disponible de $CS_{n,l}$ avec des échanges d'autres missions si c'est nécessaire. Renvoie $true$ si l'allocation est possible, $false$ sinon.

$reconfigurer(CS_{n,l}, m)$: informe $CS_{n,l}$ que la mission m n'est plus assurée par le CS auquel elle est allouée. Renvoie $true$ si cette mission peut être allouée à un autre CS, $false$ sinon.

$initialiserSdS()$: supprime tous les CS et groupes sauf le CS total et les CS dotées d'une existence physique. Désalloue toutes les missions sauf m_H.

$instancier(CS_{n,l}, m)$: instancie le CS $CS_{n,l}$ engagé sur la mission m, et indirectement tous les sous CS de $CS_{n,l}$ qui doivent réaliser m par leurs actions communes. Renvoie $false$ si le super CS ne peut s'instancier, $true$ sinon.

Algorithm 4: Algorithme de reconfiguration du CS : $reconfigurer(CS_{n,l}, m)$.

Input: $CS_{n,l} =< CS_{n',l+1}, R_{n,l}, \mathcal{CS}_{n'',l-1}, OP, \Psi, \Lambda >, m$
Output: $true/false$

```
1  if CS_{n'',l-1} ≠ ∅ then
2      var ← echangerCS(CS_{n,l}, m);
3      if var then
4          return true;
5      else
6          if CS_{n,l} ≠ CS_{1,H} then
7              return reconfigurer(CS_{n',l+1}, m);
8          else
9              initialiserSdS();
10             return instancier(CS_{1,H}, m_H);
11         end
12     end
13 else
14     return reconfigurer(CS_{n',l+1}, m);
15 end
```

4.5 Conclusion

Dans ce chapitre, nous avons proposé un modèle multi-agents multi-niveaux générique pour représenter les SdS ainsi qu'une représentation graphique des SdS, facilement compréhensible, attachée à ce modèle. Ce modèle peut être appliqué à tout SdS et permet de les contrôler dans leur environnement évolutif affectant leurs capacités tout en respectant les caractéristiques fondamentales des SdS. Les modèles produits prennent en compte aussi bien les aspects statiques avec les CS, leurs structures organisationnelles et leurs environnements, que les aspects dynamiques d'un SdS avec les mécanismes de reconfiguration qui prennent place lors de l'initialisation ou de l'évolution des capacités du SdS. De plus, nous avons proposé des algorithmes génériques qui permettent de guider ces mécanismes d'évolution des SdS. La littérature montre que l'exécution de nombreuses simulations est un moyen efficace pour tester, valider et donc contrôler un système réel sous forme de SdS. Dans le prochain chapitre, nous allons appliquer ces concepts en modélisant et en simulant un SdS dont les CS élémentaires sont des IAVs comme dans le cas du projet européen d'InTraDE.

Chapitre 5

Implémentation

Sommaire

5.1 Introduction

Au travers de ce manuscrit, notre but est de présenter un outil de modélisation et de simulation permettant de contrôler des SdS. Notre cadre applicatif est formé par le projet européen InTraDE qui vise à gérer un ITS composé d'IAVs dans l'environnement confiné d'un terminal portuaire. Dans ce chapitre, nous allons présenter les différents outils que nous avons développé pour arriver à ce but dans une démarche incrémentale.

Nous commençons par présenter brièvement les plate-formes de simulation routière et multi-agent qui serviront de support à nos simulations. Ensuite, Nous présentons un exemple de co-simulation mettant en commun ces deux plate-formes. Puis, nous montrons en détail un module que nous avons créé pour intégrer les concepts fondamentaux d'IRM4MLS dans la plate-forme MadKit. Ce module est ensuite validé en créant et en exécutant une simulation de système biologique multi-niveaux sur MadKit. Enfin, nous présentons le module nous permettant de modéliser un SdS en se basant sur le formalisme présenté dans le chapitre 3 et utilisant le module précédent sur la plate-forme MadKit. Finalement, nous présentons une simulation d'un SdS composé d'IAVs opérant dans le cadre réaliste du port de Radicatel. Ce SdS se réorganise suite à une défaillance d'un de ses CS en suivant l'algorithme de réorganisation présenté au chapitre 4.

5.2 Présentation de la plate-forme de simulation routière : SCANeRstudio

La plate-forme de simulation routière SCANeRstudio est un produit de la société Oktal [sca]. SCANeRstudio permet de simuler l'évolution du trafic sur un réseau routier dans des conditions définies. Cette plate-forme regroupe différents modèles, comme la représentation de la dynamique des véhicules ou de l'évolution des feux de signalisation, et rassemble la gestion des données relatives à ces modèles dans un seul outil logiciel.

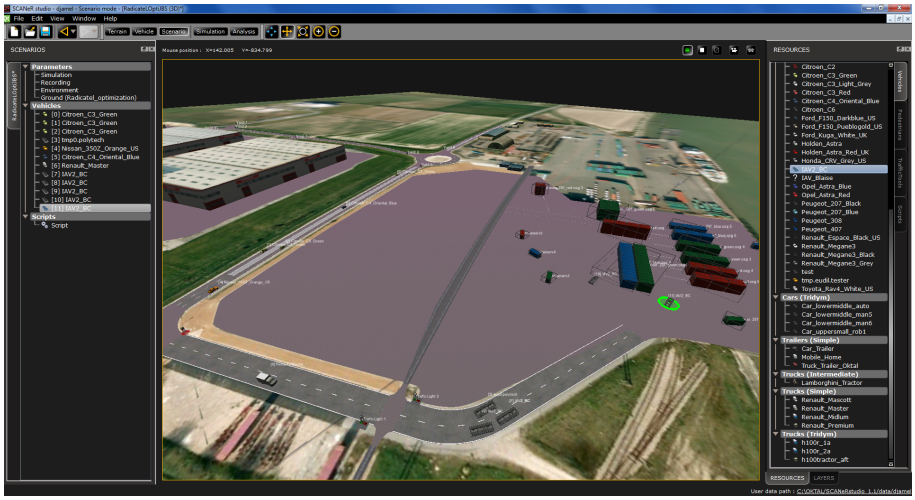

FIGURE 5.1 – Plate-forme de simulation routière SCANeRstudio.

La plate-forme SCANeRstudio contient de base un certain nombre de modules ou APIs pour gérer, lors de la simulation, la conduite et la cinématiques des véhicules, la modélisation de l'environnement ou la simulation elle-même (Fig. 5.2). Ces modules sont écrit en C++. Ils sont contenus dans un "network" qui gère l'exécution, l'ordonnancement et la transmission de messages entre modules. SCANeRstudio offre la possibilité d'intégrer des modules externes supplémentaires dédiés à une tâche spécifique comme la supervision multi-niveaux (micro,méso, macro).

FIGURE 5.2 – APIs natives de SCANeRstudio.

Ainsi, SCANeRstudio représente un outil de co-simulation, permettant de combiner des simulateurs hétérogènes (Matlab, Simulink, FlexSim, MadKit,...). Il peut aussi être utilisé comme plate-forme d'intégration de plusieurs algorithmes de commande, de contrôle, de diagnostic, d'optimisation, de supervision... Pour plus de détails sur l'utilisation de SCANeRstudio se reporter

à l'annexe A.

5.3 Présentation de la plate-forme de simulation multi-agents : MadKit

MaDKit est une plate-forme multi-agents open source modulaire, évolutive, écrite en Java [mad , Gutknecht 2001] et basée sur le modèle organisationnel Agent-Groupe-Rôle (AGR) : les agents MadKit jouent des rôles dans des groupes et ainsi créent des sociétés artificielles. Ces sociétés sont appelées communautés.

MadKit offre de nombreuses fonctionnalités pour la gestion et la simulation d'un SMA :
- Création et gestion du cycle de vie des agents artificiels.
- Communication entre agents permise par une infrastructure organisationnelle.
- Architecture des agents très hétérogène : pas de modèle d'agents prédéfini.
- Outils de simulation multi-agents.
- Capacité de créer des applications distribuées à base d'agents.

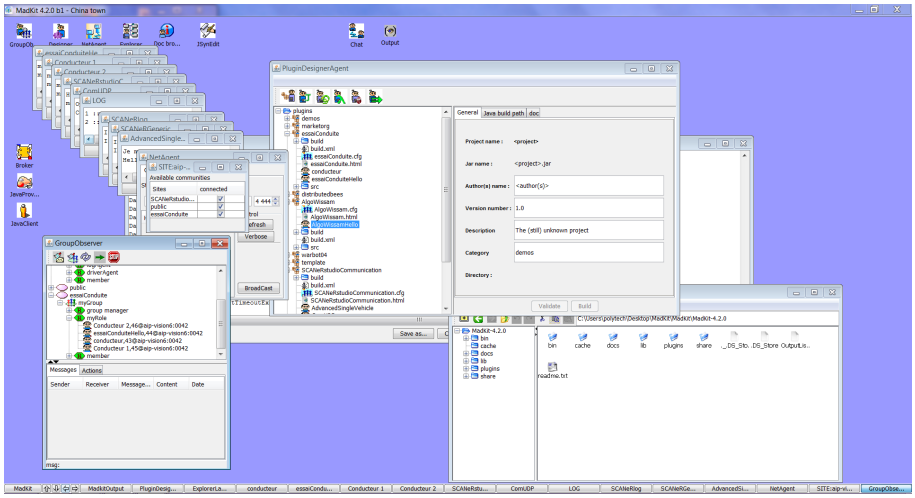

FIGURE 5.3 – Plate-forme de simulation multi-agents MadKit.

MadKit offre un environnement de développement intégré simple à utiliser et permet d'ajouter facilement de nouveaux agents ou modules appelés plugins (Fig 5.3). MadKit est une plate-forme de simulation "tout agent" dans le sens où la gestion des groupes et des rôles, la gestion du cycle de vie des agents ou la communication entre agents sont gérées par des agents de la simulation. De plus MadKit est un outil en développement permanent et doté d'une communauté active. Toutes ces raisons nous ont poussé à utiliser MadKit comme support de simulation pour nos simulations multi-agents multi-niveaux de SdS.

5.4 Exemple de co-simulation SCANeRstudio-MadKit

SCANeRstudio a été pensé à l'origine pour simuler le pilotage de véhicules en incarnant le conducteur des véhicules. Cependant il ne propose pas d'outils de pilotage automatique et évoluant dynamiquement durant la simulation tout en tenant compte du réseau routier. Or dans le cadre de cette thèse il est nécessaire de disposer d'un tel outil pour contrôler individuellement

la conduite automatique de chaque IAV qui forment l'ITS étudié dans le cadre du projet InTraDE. Au sein du projet InTraDE une API de pilotage des véhicules a été implémentée, l'API "IAV_API", cependant cette API ne tient pas compte des structures du réseau routier ce qui conduit les véhicules à rouler en dehors des routes s'ils sont pilotés uniquement grâce à cet API. L'API native "TRAFFIC" pilote des véhicules automatiquement en respectant l'infrastructure du réseau routier modélisé par l'API "ROADS". Mais la conduite produite par cet API est extrêmement simple : le véhicule roule sur la route à la vitesse maximale possible et à chaque intersection il s'engage sur une route choisie de manière aléatoire.

Nous utilisons l'API "TRAFFIC" pour faire rouler automatiquement des véhicules respectant le code de la route sur un réseau routier et nous modifions l'itinéraire enregistré dans les véhicules que nous voulons contrôler. Nous avons donc créé une API "API_Communication" qui modifie l'itinéraire de véhicules choisis en envoyant des messages via le network à l'API "ROADS". L'API "API_Communication" est divisée en deux classes. La première permet de modifier l'itinéraire de véhicules et récupère les données de ces véhicules nécessaires à leur contrôle. La seconde est un canal de communication utilisant le protocole de communication UDP permettant de communiquer ces données à des logiciels externes à SCANeRstudio et récupérant des directives de contrôle des véhicules. Le schéma fig. 5.4 présente l'utilisation de ces différents éléments lors de la co-simulation MadKit-SCANeRstudio.

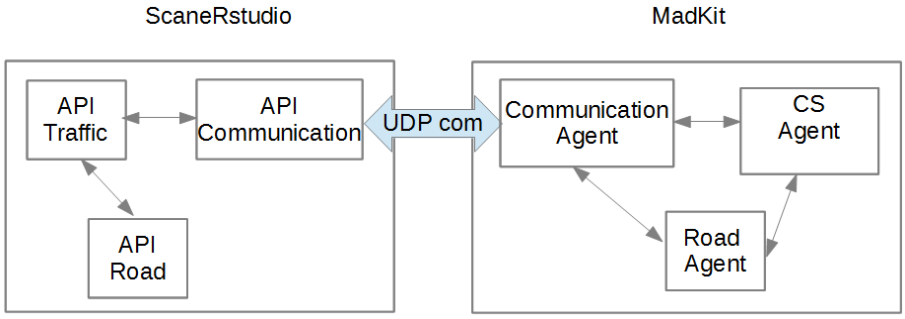

FIGURE 5.4 – Schéma des différents éléments de la co-simulation MadKit-SCANeRstudio.

Le module MadKit que nous avons créé pour mettre en place une co-simulation regroupant les plate-formes MadKit et SCANeRstudio doit comprendre les aspects qui nous manque au sein des APIs SCANeRstudio. Ces éléments sont l'enregistrement des données relatives à un réseau routier et un module de décision qui permet aux véhicules de former des décisions pour se piloter automatiquement durant la simulation en tenant compte du réseau routier. Nous avons développé deux classes d'agents implémentant ces aspects en plus d'une classe d'agent permettant d'utiliser le protocole UDP pour communiquer les décisions des agents conducteurs à des logiciels externes. Les agents conducteurs permettent de piloter "manuellement" (Fig.5.5) certains véhicules durant leur simulation ou alors en utilisant un algorithme développé dans [Khalil 2012b] qui détermine le chemin le plus court entre la position actuelle d'un véhicule et une intersection du réseau à atteindre. Cet algorithme prend en compte la distance des routes à parcourir mais également leur encombrement.

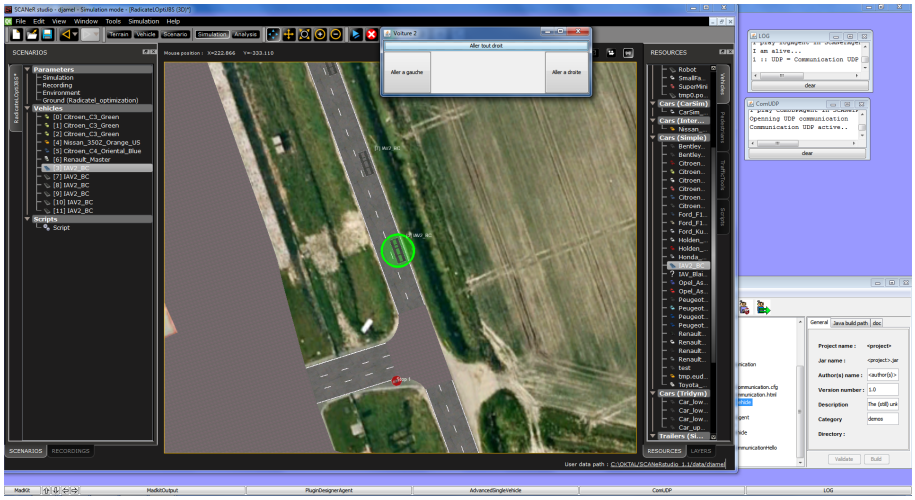

FIGURE 5.5 – co-simulation MadKit-SCANeRstudio avec “pilotage manuel” d’un véhicule.

5.5 Module Madkit pour IRM4MLS

Dans cette section nous présentons le module que nous avons développé pour intégrer les concepts multi-niveaux d’IRM4MLS dans la plate-forme de simulation multi-agents MadKit. Les principaux concepts qui doivent être modélisés sont les agents, les niveaux et les environnements. Pour que notre implémentation soit plus efficace et plus lisible, la plupart des classes abstraites possèdent une interface associée. Ainsi, par exemple, les agents qui seront instanciés grâce à ce module doivent implémenter des méthodes nécessaires contenues dans l’interface *MKAgent* et la classe abstraite *MKAbstractAgent* contient les éléments communs qui seront présents dans chaque agent d’IRM4MLS. Nous détaillons ici les différentes classes formant ce module.

MKAbstractAgent : classe abstraite héritant de la classe *Agent* de MadKit. Elle permet de représenter un agent pouvant percevoir, mémoriser et produire des influences. De plus toutes ses instances sont exécutées et simulées comme des agents MadKit jouant un ou plusieurs *Rôles* dans un ou plusieurs *Groupes* qui forment une ou plusieurs *Communautés*.

MKAbstractBehavior : classe abstraite interne à la classe *MKAbstractAgent*. Elle permet de représenter le comportement d’un agent en fonction du niveau dans lequel il agit. Elle représente la partie située de l’agent. Ainsi un agent peut disposer d’un esprit et de plusieurs corps situés dans différents niveaux qui agissent en fonction de cet esprit. Ces différents corps sont représentés par différentes instances de cette classe.

MKAbstractDynamicState : classe abstraite interne à la classe *MKAbstractLevel*. Elle représente toutes les influences qui constituent une partie de l’état d’un niveau. Ces influences sont soient produites par des agents ou par des environnements.

MKAbstractEnvironment : classe abstraite représentant les environnements d’IRM4MLS. Un environnement peut être associé à plusieurs niveaux, mais un niveau ne possède qu’un seul environnement.

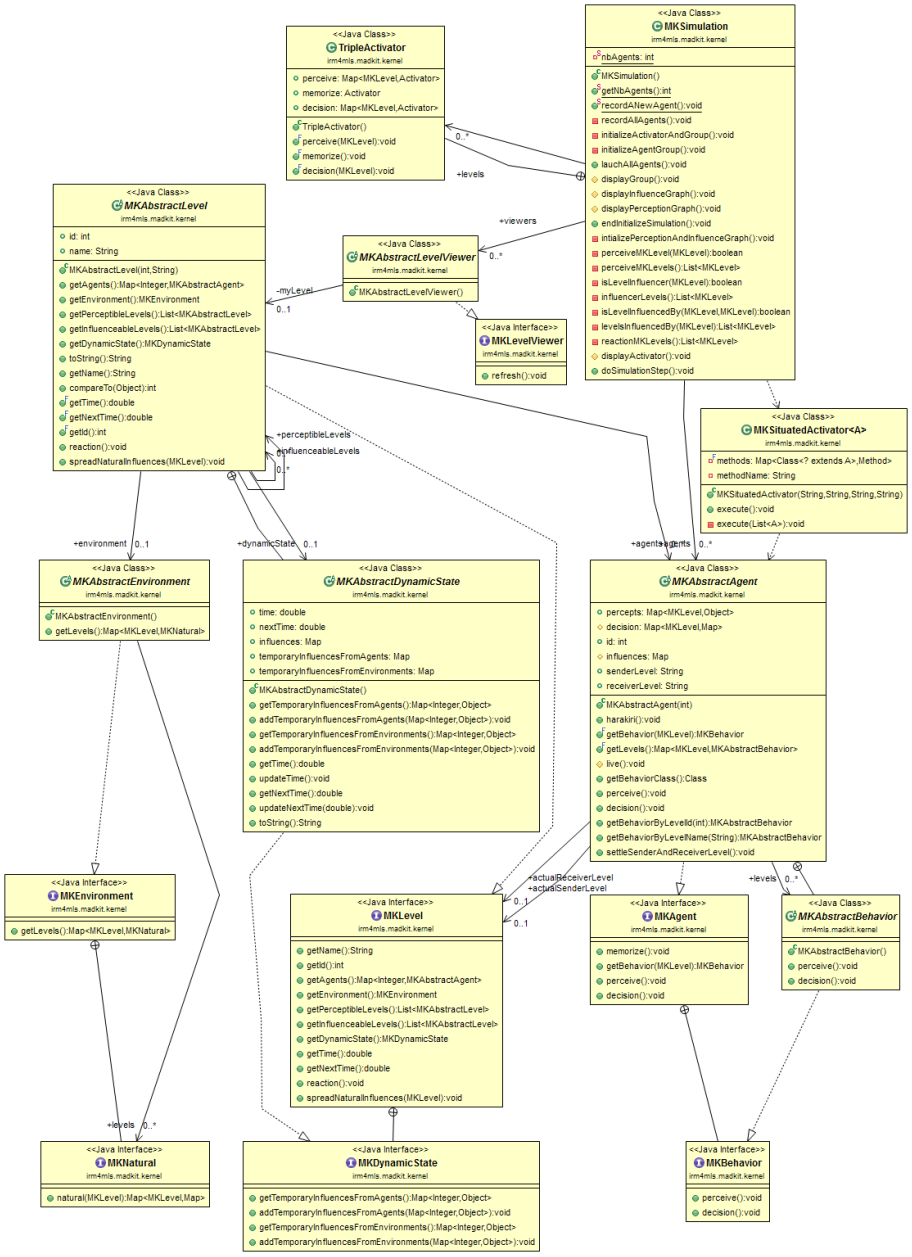

FIGURE 5.6 – Classes du package irm4mls.madkit.kernel.

MKAbstractLevel : classe abstraite représentant un niveau dans IRM4MLS. Un niveau contient des agents et possède un environnement associé. De plus un niveau est doté d'un temps actuel qui indique quand les agents qu'il contient sont activés et un prochain temps qui indique quand les agents qu'il contient seront activés.

MKAbstractLevelViewer : classe héritant de la classe *JFrame* du package *javax.swing*. Cette classe permet de visualiser les niveaux et les agents qu'ils contiennent.

MKAgent : interface implémentée par la classe *MKAbstractAgent*.

MKBehavior : interface interne à l'interface *MKAgent* implémentée par la classe *MKAbstractBehavior*.

MKDynamicState : interface interne à l'interface *MKEnvironment* implémentée par la classe *MKAbstractDynamicState*.

MKEnvironment : interface implémentée par la classe *MKAbstractEnvironment*.

MKLevel : interface implémentée par la classe *MKAbstractLevel*.

MKLevelViewer : interface implémentée par la classe *MKAbstractLevelViewer*. Sa fonction *refresh()* doit être appelé à chaque mise à jour du niveau associé pour rafraîchir son affichage.

MKNatural : interface interne à l'interface *MKEnvironment*. Cette interface permet de définir les influences naturelles que produit un environnement.

MKSimulation : classe héritant de la classe *scheduler* de MadKit qui hérite également de la classe *Agent* de MadKit. Pour lancer une simulation il suffit de créer une instance de cette classe et de l'exécuter. Elle représente le moteur de la simulation, elle gère l'ordonnancement des agents présents dans les différents niveaux. Cette classe comporte deux méthodes importantes. La méthode *activate* permet d'initialiser la simulation en intégrant les agents, les niveaux et les environnements. Cette méthode crée aussi les graphes d'influence et de réaction, fixe la temporalité des différents niveaux et instancie les activateurs pour exécuter les agents des différents niveaux. La méthode *doSimulationStep* active les agents des niveaux dont le prochain pas de temps est le plus proche. L'exécution de la simulation se fait en respectant l'algorithme qui détaille un modèle de simulation d'IRM4MLS avec des dynamiques temporelles différentes entre niveaux 2.

MKSituatedActivator Cette classe permet de créer des activateurs qui permettent d'activer certains agents d'une *Communauté*, d'un *Groupe* et jouant un *Rôle* donné. Cette classe permet d'activer la perception, la mémorisation et la production d'influences de tous les agents d'un niveau simultanément.

TripleActivator cette classe interne à la classe *MKSimulation*. Chaque instance de cette classe forme une collection d'activateurs, elle concerne un niveau et permet d'activer simultanément la mémorisation de tous les agents agents du niveau. Elle permet également d'activer la perception et la production d'influences des agents de ce niveau vers d'autres niveaux en respectant les graphes d'influence et de perception.

5.6 Exemple de simulation du module Madkit pour IRM4MLS

Un moyen de tester la validité de notre module intégrant les concepts d'IRM4MLS dans la plate-forme de simulation MadKit est de créer une application indépendante utilisant ce module et de la simuler. Nous avons créé une simulation de systèmes biologiques proies-prédateurs intégrant un aspect multi-niveaux. Chaque niveau représente un tableau dont les cases contiennent des ressources. Les différents niveaux représentent la même zone géographique mais à des échelles de temps et d'espace différentes. Plus l'échelle d'un niveau est élevé, plus son nombre de

cases représentées est petit. On veut représenter ici le fait que les prédateurs, souvent plus gros que les proies, évoluent sur un territoire plus vaste que celui de ces dernières. Au niveau le plus bas, des “herbivores” consomment des ressources (herbe) qui sont générées grâce aux influences naturelles du niveau. Aux niveaux supérieurs, la ressource de chaque case représente la quantité de proies présentes au niveau inférieur sur les cases dans la même zone géographique (Fig. 5.7). À chaque pas de temps les agents se nourrissent en consommant la ressource de leur case ou si c'est impossible se déplacent sur la case voisine dotée de la plus grande valeur de ressource.

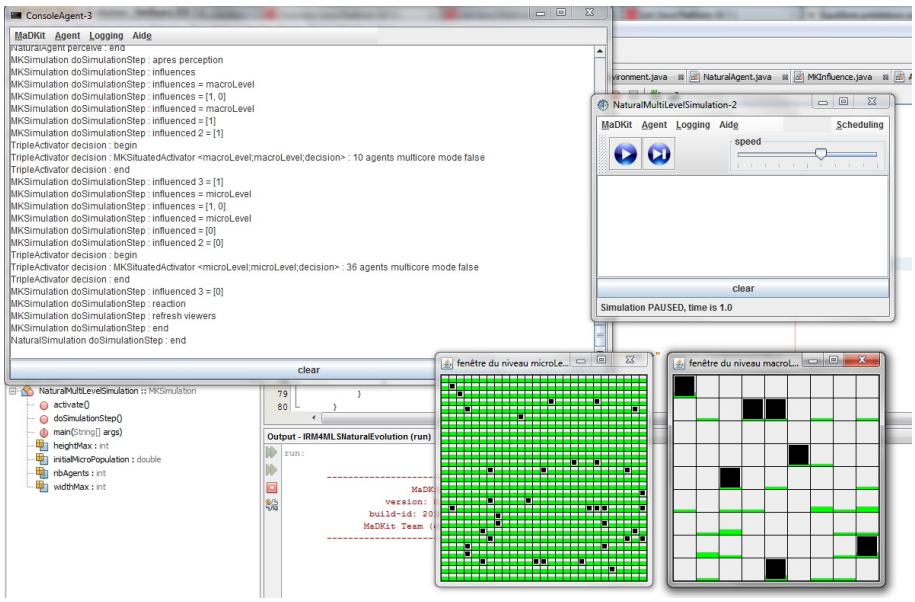

FIGURE 5.7 – Un exemple de simulation du package irm4mls.preyPredator.

Nous détaillons ici les différentes classes de ce module 5.8.

NaturalAgent : classe représentant les agents proies ou prédateurs simulés héritant de la classe *MKAbstractAgent*.

NaturalBehavior : classe interne de la classe *NaturalAgent* représentant les comportements des agents naturels héritant de la classe *MKAbstractBehavior*.

NaturalEnvironment : classe représentant l'environnement associé à chaque niveau héritant de la classe *MKAbstractEnvironment*.

NaturalGridDynamicState : classe interne de la classe *NaturalScaleLevel* représentant les différentes influences présentes dans les instances de la classe **NaturalScaleLevel** et héritant de la classe *MKAbstractDynamicState*.

NaturalMultiLevelSimulation : classe permettant d'exécuter une simulation où évoluent les agents proies et prédateurs héritant de la classe *MKSimulation*.

NaturalScaleLevel : classe représentant les niveaux support de l'évolution des agents naturels héritant de la classe *MKAbstractLevel*.

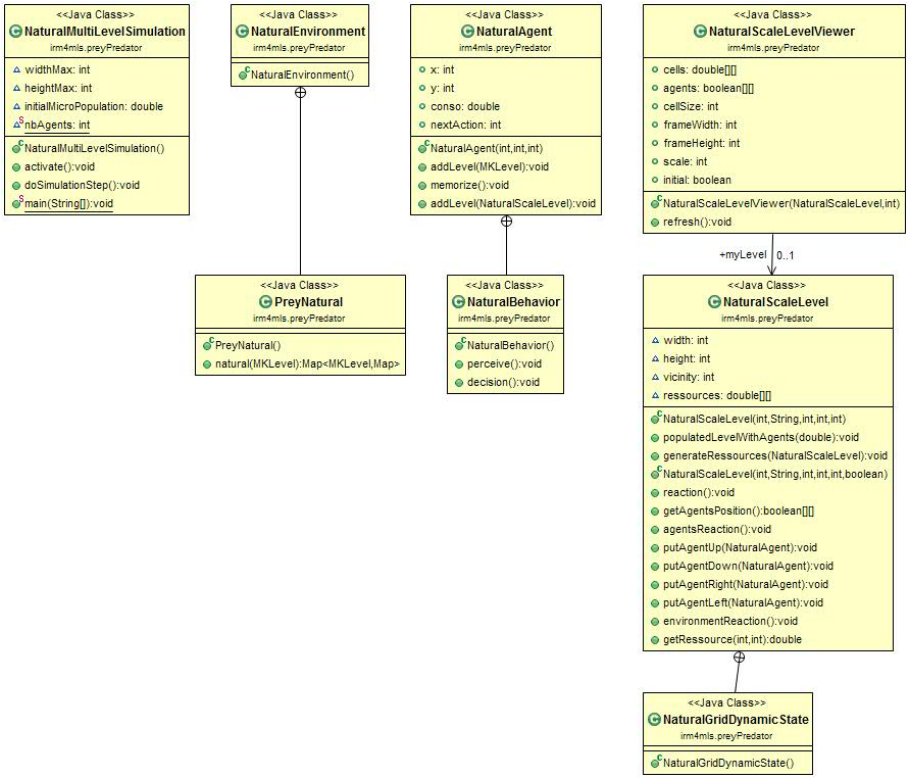

FIGURE 5.8 – Classes du package irm4mls.preyPredator.

NaturalScaleLevelViewer : classe héritant de la classe *MKAbstractLevelViewer*, permettant de visualiser chaque niveau par un tableau contenant les agents. Chaque case du tableau est remplie en proportion des ressources qu'elle contient.

PreyNatural : classe interne à la classe *NaturalEnvironment*, héritant de la classe *MKNatural* et permettant de générer les influences naturelles des différents niveaux. Les instances de cette classe font pousser l'herbe pour le niveau le plus bas et pour les autres niveaux représentent la population de proie au niveau inférieur.

Une fois que la simulation a été testée, le module intégrant les concepts d'IRM4MLS dans la plate-forme de simulation MadKit a été validé.

5.7 Module Madkit pour la représentation des SdS

Dans cette section nous présentons le module permettant de modéliser les SdS sur la plate-forme MadKit (Fig. 5.9). Ce module s'appuie sur le module MadKit présenté dans la section 5.5 intégrant les concepts fondamentaux d'IRM4MLS. Les classes de ce module représentent les différents aspects du formalisme de modélisation des SdS que nous avons proposé dans le chapitre 4.

Nous présentons ici en détails les différentes classes de ce module.

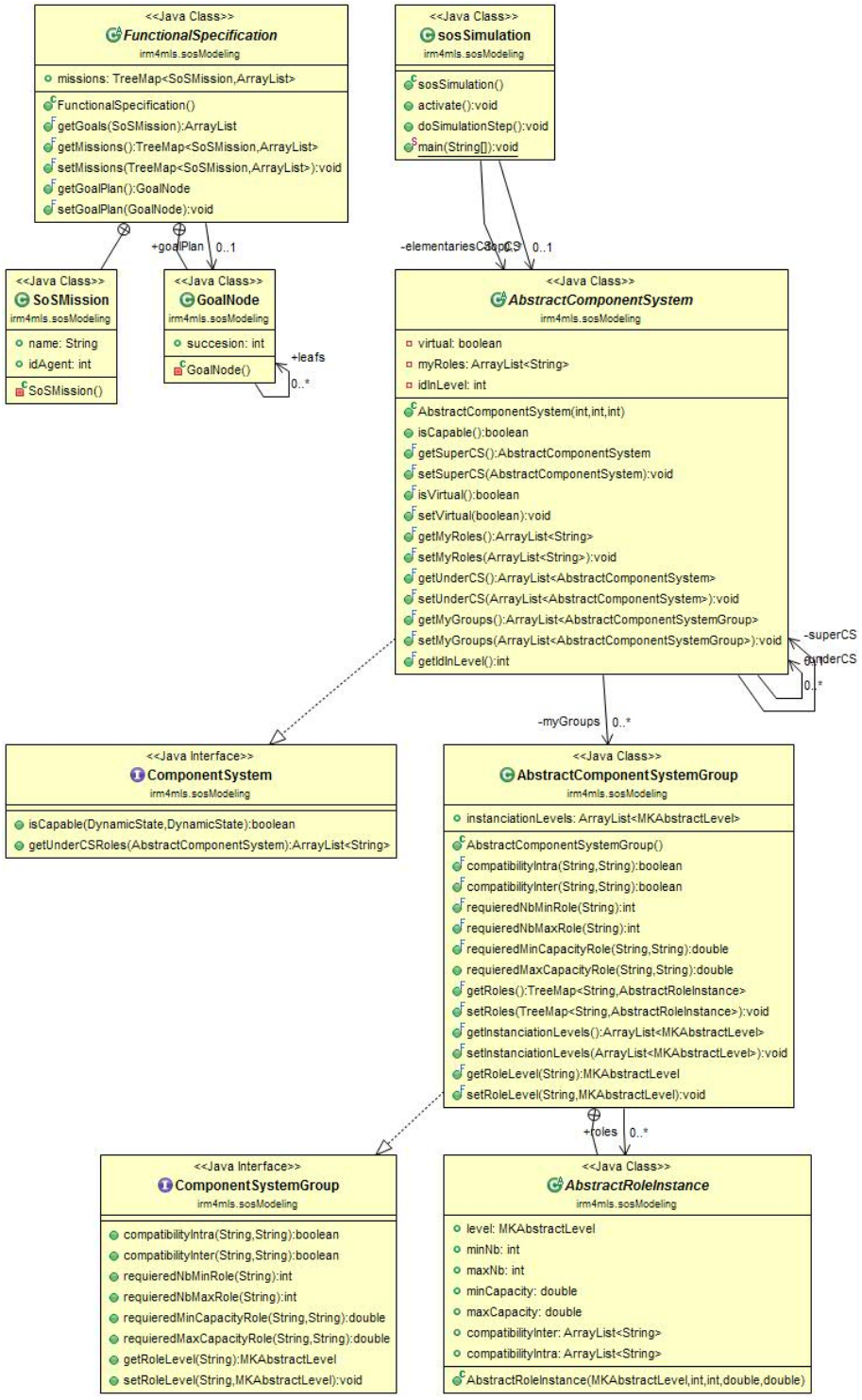

FIGURE 5.9 – Classes du package irm4mls.sosModeling.

AbstractComponentSystem : classe abstraite héritant de la classe *MKAbstractAgent* et implémentant l'interface *ComponentSystem*. Les instances de cette classe représentent les CS du SdS intégrant les différents éléments de la définition d'un CS, $CS_{n,l}$. Ces instances possèdent

un super CS, $CS_{n',l+1}$, un ensemble de rôles que le CS peut jouer, $R_{n,l}$, un ensemble de sous CS, $\mathcal{CS}_{n'',l-1}$ et un ensemble de groupe formant ce CS, OP. Ces instances indiquent également si le CS représenté est virtuel ou non.

AbstractComponentSystemGroup : classe abstraite implémentant l'interface *ComponentSystemGroup*. Les instances de cette classe représentent les groupes de CS du SdS intégrant la définition d'un groupe de CS, gs. Ces instances possèdent un ensemble de rôles jouables dans ce groupe, R et un ensemble de niveaux dans lesquels sont instanciés les CS formés par ce groupe, $\mathcal{L}$.

AbstractRoleInstance : classe abstraite interne à la classe *AbstractComponentSystemGroup*. Les instances de cette classe permettent de représenter les caractéristiques d'un rôle dans un groupe de CS. Ces instances indiquent le niveau d'un agent jouant ce rôle dans un groupe, les compatibilités entre rôles dans et en dehors de ce groupe, C^{intra} et C^{inter} et également la quantité maximale et minimale d'une capacité donnée nécessaire à l'accomplissement d'un but de ce groupe ,*nc*.

ComponentSystem : interface associée à la classe *AbstractComponentSystemGroup*. Cette interface contient les fonctions d'un CS, $CS_{n,l}$, qui indiquent l'ensemble des rôles associés à un sous CS d'un CS, Ψ et également la capacité du CS à atteindre un but Λ.

ComponentSystemGroup : interface associée à la classe *AbstractComponentSystemGroup*. Cette interface contient les fonctions d'un groupe de CS, gs, qui indiquent le niveau d'un agent jouant un rôle dans ce groupe, L_R, les compatibilités entre rôles dans et en dehors de ce groupe, C^{intra} et C^{inter}, le nombre minimum et maximum d'agents jouant un rôle donné dans ce groupe, nr et également la quantité maximale et minimale d'une capacité donnée nécessaire à l'accomplissement d'un but de ce groupe, nc.

FunctionalSpecification : classe représentant les différents éléments d'une spécification fonctionnelle fs. Les instances de cette classe contiennent un ensemble de buts du SdS, $\mathcal{G}$, un ensemble d'étiquettes associées aux missions, $\mathcal{M}$, un ensemble de plans globaux décomposant les buts en arbres de sous buts, $\mathcal{P}$, et une fonction indiquant l'ensemble des buts attachés à chaque mission, mo.

GoalNode : classe interne de la classe *FunctionalSpecification*. Les instances de cette classe représentent un but sur l'arbre de décomposition des buts. Chaque instance de cette classe possède un ensemble de sous buts.

SoSMission : classe interne de la classe *FunctionalSpecification*. Les instances de cette classe représentent les missions de la spécification fonctionnelle du SdS. Chaque instance de cette classe indique le nom de l'agent et de l'étiquette associé à cette mission.

SoSSimulation : classe héritant de la classe *MKSimulation*. Cette classe permet d'exécuter la simulation d'un SdS respectant leurs caractéristiques fondamentales et permettant de piloter leur mise en place et leur réorganisation.

5.8 Cas d'application d'InTraDE : co-simulation d'un SdS formée d'une flotte d'IAVs

Pour valider notre approche nous avons mis en place une co-simulation SCANeRstudio-MadKit de réorganisation d'un SdS. Il est moins pertinent de simuler ici la réorganisation d'un SdS plutôt que sa création telle que nous l'avons définie. Car nous créons un SdS très adapté à un but donné plutôt qu'un SdS peu spécialisé qui pourrait donner de nombreuses alternatives de structures permettant de réaliser un même but global.

Le SdS que nous utilisons pour cette simulation est celui qui a été exposé dans le chapitre 4 et qui est détaillé ci après (5.10). Au cours de la simulation l'IAV qui correspond au CS $CS_{3,0}$ va connaître une défaillance qui va annuler sa capacité à réaliser sa mission ce qui va rendre la mission de son super CS $CS_{1,2}$ impossible et ainsi la mission attachée sera elle aussi impossible à réaliser. La réorganisation qui va rendre de nouveau le but global du SdS atteignable consiste à interchanger les IAVs correspondant aux CS $CS_{3,0}$ et $CS_{4,0}$.

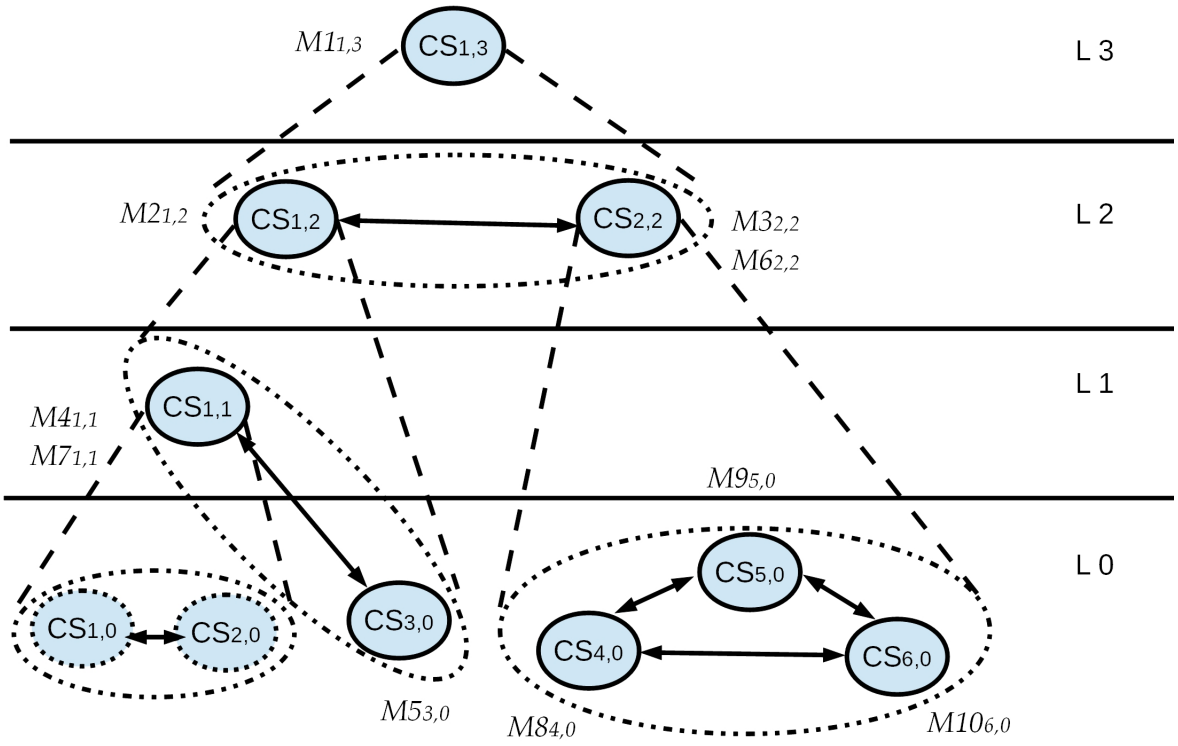

FIGURE 5.10 – SdS simulé sous d'ITS composé d'IAVs.

Dans SCANeRstudio nous utilisons les données géographiques et routières réalistes issues du port de Radicatel 5.11. Tous les CS élémentaires sont représentés par des véhicules avec un profil d'IAVs 5.12. Pour nous représenter plus clairement la réorganisation nous avons décidé arbitrairement que chaque flotte de quai, $CS_{1,2}$ et $CS_{2,2}$, opérerait sur une zone différente du

port et serait dotée d'une couleur propre (bleu pour $CS_{1,2}$ et rouge pour $CS_{2,2}$). Et l'API "API_Communication" sera activée en conjonction avec l'API "TRAFFIC" pour permettre de contrôler tous les IAVs individuels et de communiquer avec la simulation MadKit.

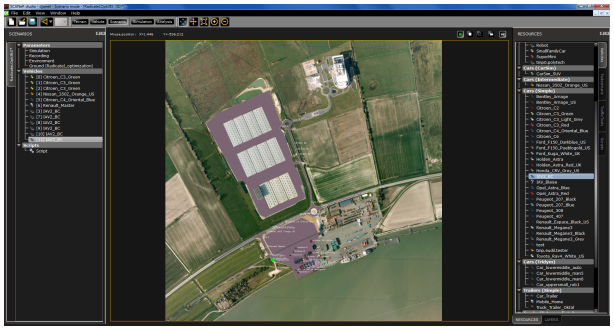

FIGURE 5.11 – Le port de Radicatel simulé dans SCANeRstudio.

FIGURE 5.12 – Vue de côté d'un IAV modélisé d'un IAV.

Au sein de MadKit le SdS sera modélisé grâce à l'instanciation des classes du module présenté dans la section 5.7. En plus de cela un agent enregistrera toutes les données des véhicules et du réseau routier et un autre agent assurera la communication avec la simulation SCANeRstudio via UDP pour envoyer des directives de pilotages des IAVs et recevoir les données physiques concernant ces IAVs.

La simulation représentant cette réorganisation au sein d'un SdS montre le changement d'appartenance des CS, $CS_{3,0}$ et $CS_{4,0}$ en leur faisant adopter la couleur de leur super CS (bleu pour $CS_{1,2}$ et rouge pour $CS_{2,2}$) et chacun des deux CS rejoint la zones où les autres sous CS de son super CS opèrent.

5.9 Conclusion

Nous avons développé un outil de co-simulation pour représenter et contrôler un SdS. D'une part, nous avons créé trois modules pour la plate-forme multi-agents MadKit. Le premier module permet de générer un moteur de simulation multi-agents multi-niveaux intégrant les concepts fondamentaux d'IRM4MLS et doté d'un algorithme qui gère l'exécution de la simulation niveau par niveau. Le second module reprenant le moteur de simulation offert par le premier permet de représenter l'aspect organisationnel et donc de simuler un SdS. Ce module s'appuie sur le formalisme présenté dans le chapitre 4 qui représente les aspects

statiques et dynamiques d'un SdS et respecte ses caractéristiques fondamentales au cours de son évolution. Le troisième module enregistre les données géographiques et les caractéristiques d'un réseau routier présent dans un fichier .rnd. Ce module permet également de communiquer avec d'autres applications grâce au protocole de communication simple UDP. Ainsi ce module intègre des données en provenance de logiciels externes, du trafic présent sur les réseaux routier et d'envoyer des ordres simples pour piloter ces véhicules dans ces logiciels.

D'autre part, nous avons développé une API pour SCANeRstudio qui permet de communiquer la position et l'état des véhicules grâce au protocole UDP. Elle permet également de piloter ces véhicules en conjonction avec l'API TRAFFIC. Nous avons mis en commun ces différents modules pour produire une co-simulation prenant place sur les plate-formes de simulation MadKit et SCANeRstudio. MadKit représente et de contrôle les aspects organisationnels des SdS tout en respectant leurs caractéristiques fondamentales. SCANeRstudio fournit des données réalistes du réseau routier et des véhicules. La simulation produite représente la reconfiguration d'un SdS sous forme d'un ITS composé d'IAVS évoluant dans l'environnement confiné et évolutif d'un terminal portuaire et faisant face à une défaillance sur l'un des IAVs. Cette simulation intègre les données géographiques réalistes du port de Radicatel et les modèles de déplacement des IAVs qui ont été validés sur des véhicules prototypes. On peut conclure que l'outil que nous avons produit permet de contrôler un SdS réel en reportant les commandes de pilotage produite dans SCANeRstudio.

De plus nous avons testé la validité des modules en testant des applications indépendantes intégrant ces modules aux différentes phases de ce processus de développement.

Chapitre 6

Conclusion et perspectives

Sommaire

6.1 Contributions

Une première contribution de cette thèse est de fournir une définition formelle des systèmes de systèmes (SdS) basée sur leurs caractéristiques fondamentales : indépendance opérationnelle, indépendance managériale, dispersion géographique des composants systèmes (CS), comportement émergent et développement évolutionnaire. Une telle définition est nécessaire et suffisante pour créer des modèles qui assurent le contrôle de tout SdS dirigé par un but et dans lequel les interactions entre composants système ne se limitent pas à l'échange d'information. À partir de cette définition nous avons proposé un formalisme générique de représentation des SdS qui permet de retranscrire à la fois leurs aspects statiques et dynamiques.

Deux éléments des SdS constituent un défi pour leur modélisation :
– les aspects organisationnels et
– plusieurs niveaux de représentation possibles.

Ces aspects nous ont poussé à condidérer les domaines des systèmes multi-agents (SMA) centrés organisation et des modèles à base d'agents multi-niveaux (ML-ABM).

Nous avons utilisé ce formalisme pour produire un modèle à base d'agents (ABM) représentant les SdS, profitant de capacités similaires chez les CS et les agents comme leur autonomie et leur capacité à décider de leur actions. Les aspects organisationnels des CS (qui peuvent être aussi bien des individus seuls que des groupes d'individus) sont gérés par le modèle Agent-Groupe-Rôle (AGR) [Ferber 1998]. Une spécification fonctionnelle permet de guider les SdS dans l'accomplissement de leur but global [Hübner 2002b]. Enfin, les aspects multi-niveaux sont modélisés en utilisant le méta-modèle multi-agents multi-niveaux IRM4MLS [Morvan 2011].

Ce modèle de représentation des SdS a été intégré à la plate-forme de simulation multi-agents MadKit [mad]. Nous avons ainsi développé un plugin pour MadKit implémentant les concepts du méta-modèle multi-niveaux IRM4MLS. Nous avons également intégré les concepts organisationnels et fonctionnels de notre formalisme, ainsi que les algorithmes de création et de réorganisation de SdS.

Nous avons proposé des outils génériques de modélisation et de simulation des SdS. Les modèles produits permettent de créer et de contrôler des SdS dirigés par un but lors de leur fonctionnement ordinaire ou en conditions détériorées. Le cadre d'application de ces travaux est le projet européen InTraDE [int] qui consiste à géré un ITS responsable de la logistique des marchandises dans des terminaux portuaires constituant un environnement évolutif.

Ces outils ont été validés en prenant comme cas d'application un système de transport intelligent (ITS) considéré dans le projet européen InTraDE. Les CS élémentaires sont les CS dotés d'une existence physique et qui ne peuvent pas être divisés sans violer les caractéristiques fondamentales des SdS, comme les véhicules intelligents autonomes (IAVs) par exemple ou les regroupements de CS dont les éléments individuels violent ces caractéristiques, comme les trains de véhicules par exemple. Les autres CS du SdS sont toutes les organisations qui regroupent des CS, comme par exemple les flottes de véhicules attachées à un quai en particulier ou le SdS dans sa globalité. Le modèle multi-agents multi-niveaux proposés permet de représenter tous ces CS en respectant les caractéristiques fondamentales des SdS.

Les résultats de ces travaux de recherche ont été valorisés par l'implémentation et l'exécution de co-simulations composées par

- une simulation 3D en temps réel fournissant des données réalistes pour représenter l'état et le comportement d'IAVs, supportée par le logiciel de simulation routière SCANeRstudio [sca]
- et par une simulation multi-agents multi-niveaux prenant en charge l'aspect décisionnel présent dans chaque CS du SdS.

Ces co-simulations ont pour application la gestion et le contrôle en ligne d'un ITS constitué d'IAVs intervenant dans un environnement portuaire, confiné et évolutif.

Nous proposons une co-simulation faisant intervenir le logiciel de simulation routière SCANeRstudio et la plate-forme multi-agents agents MadKit. Ces deux outils disposent d'un moteur de simulation ordonnançant les évènements de manière différente. Les outils de communication mis en place entre SCANeRstudio et MadKit assurent la bonne transmission des données, cependant ils génèrent des pertes d'efficacité pour le suivi de la co-simulation.

6.2 Perspectives

Les perspectives à ces travaux de recherche démarrés en 2011 concernent trois domaines différents et leur communauté : la logistique portuaire, la communauté des SdS et la communauté de la simulation multi-agents multi-niveaux.

6.2.1 Le projet européen InTraDE

L'une des continuations du projet InTraDE est d'intégrer les modèles produits par les thèmaticiens, c-à-d les logisticiens qui s'intéressent à des aspects particuliers du projet consi-

dérant l'ITS à des échelles spatio-temporelles différentes. Deux modèles à mettre en commun pourraient être le diagnostic d'un IAV qui tient compte de l'état des pièces mécaniques en temps réel (niveau microscopique) et la gestion du flux des véhicules modélisé dans une portion entière du terminal portuaire (niveau macroscopique).

SCANeRstudio présente un certain nombre de contraintes importantes pour les simulations de grande ampleur : le nombre de véhicules simulés est limité de manière arbitraire. De même sur un ordinateur disposant d'une configuration moderne on peut représenter au maximum quatre véhicules avec un modèle cinématique avancé avant que les ressources mémoires de l'ordinateur deviennent insuffisantes. Pour améliorer la simulation des SdS présents dans le projets InTraDE on peut créer un cadre de simulation routière 3D en temps réel, unifié qui intègre les aspects multi-agents multi-niveaux nécessaires aux SdS, afin de limiter les pertes d'efficacité lors de l'exécution des simulations de SdS.

6.2.2 Les systèmes de systèmes

Notre approche a vocation à être utilisée dans d'autres domaines d'application des SdS. Elle pourrait ainsi être intégrée dans la gestion de projets industriels faisant appel à l'ingénierie des SdS en intervenant durant les phases d'implémentation, d'intégration, de vérification et de validation du SdS.

Elle pourrait être également utilisée dans le domaine de la simulation militaire pour représenter des opérations militaires de grande envergure où les acteurs dotés de spécialités différentes doivent collaborer pour atteindre un but commun. Traditionnellement, ce genre de simulations n'est pas gérée de manière unifiée mais en intégrant divers outils logiciels qu'il est nécessaire d'interfacer. Comme nous proposons un cadre de simulation unifié pour gérer de nombreuses entités hétérogènes, il suffit d'encapsuler les modèles existants des acteurs dans des agents pour produire une simulation plus facilement contrôlable.

Le modèle que nous proposons pourrait aussi être utilisé pour modéliser et simuler des SdS moins contraints comme par exemple des SdS virtuels dont le seul but se borne à l'observation de l'évolution du système. Ce genre d'approche paraît particulièrement pertinent pour modéliser les systèmes complexes étudiés en biologie ou en sociologie, où l'aspect comportemental des entités est essentiel pour déterminer l'évolution du système.

6.2.3 Les simulations multi-agents multi-niveaux

Cette thèse a fait apparaître un certain nombre de perspectives liées à ce domaine qui pourraient être mises en oeuvre dans la poursuite de mes travaux :

1. La mise en place de mécanismes de couplage générique entre niveaux dotés d'échelles différentes représente un gain de temps important pour le modélisateur et augmente la réutilisabilité du modèle proposé. Pour cela, il convient de formaliser les mécanismes de détection de formation de groupes, ainsi que ceux d'agrégation et de désagrégation d'agents facilement paramétrables [Servat 1998, Navarro 2011, Navarro 2013].
2. Pour être complète, la mise en place de ces mécanismes qui font varier dynamiquement le niveau de représentation de la simulation doit s'accompagner d'outils génériques de mesure comparant la qualité de la simulation par rapport aux ressources informatiques utilisées.

De tels outils sont basés sur la notion de cohérence et doivent être paramétrables facilement en fonction du modèle simulé [Davis 1993, Navarro 2011].

Annexe A

SCANeRstudio - plate-forme de simulation routière en 3 dimensions et en temps réel

A.1 Utilisations de SCANeRstudio

La plate-forme de simulation SCANeRstudio réunit les informations pertinentes, nécessaires à la simulation de trafic routier (Fig. A.1). La structure flexible et modulaire du système permet d'ajouter des modules ou des extensions en fonction des besoins de l'utilisateur.

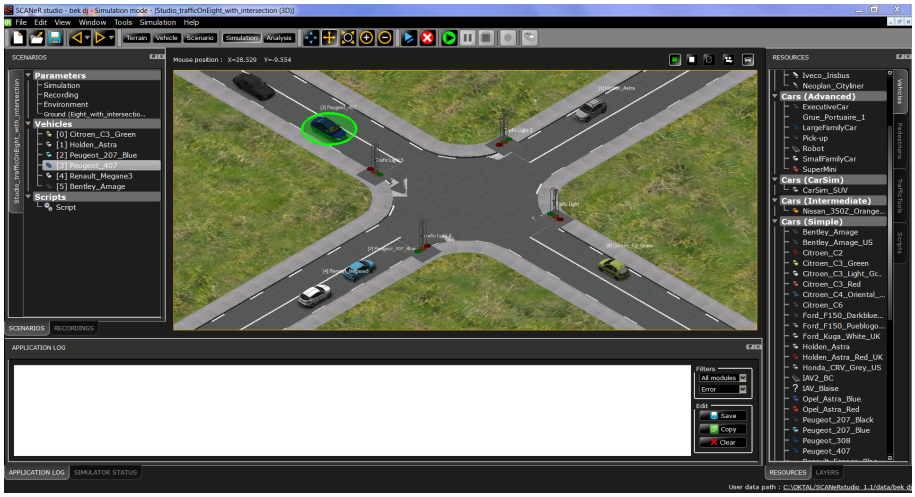

FIGURE A.1 – Plate-forme de simulation routière SCANeRstudio.

SCANeRstudio est un logiciel de simulation de trafic routier qui permet :
- la création de terrains en utilisant des données géographiques réalistes reprenant les paramètres réels et la disposition des objets dans une zone particulière,
- la modélisation de la dynamique des différents types de véhicules dépendante du terrain parcouru par le véhicule,
- la gestion du trafic routier au travers des éléments de signalisation ou des caractéristiques du réseau routier,
- l'implémentation de la régulation de trafic à travers des scripts appropriés,

– la visualisation des scènes en 3 dimensions,
– la simulation de la dynamique des véhicules et de l'évolution du trafic et de ses indicateurs.

A.2 Mode terrain

Le mode "Terrain" permet de recréer un terrain en 3 dimensions respectant les caractéristiques physiques de ce terrain (Fig. A.2). Pour cela il est possible d'intégrer des données réalistes issues de la cartographie en 3 dimensions d'un terrain pour recréer sa version virtuelle. Ce mode permet aussi de créer un réseau routier sur ce terrain en utilisant des profils de routes existants ou en générant de nouveaux profils en précisant le sens de circulation, le nombre de voie, le type de véhicules autorisé sur ces voies .... Nous pouvons également disposer de manière réaliste un certain nombres d'objets sur les terrains générés. Ces objets peuvent être neutres comme les conteneurs dans les zones portuaires, les bâtiments, les ponts ou avoir une influence sur le réseau routier comme les feux et les panneaux de signalisation.

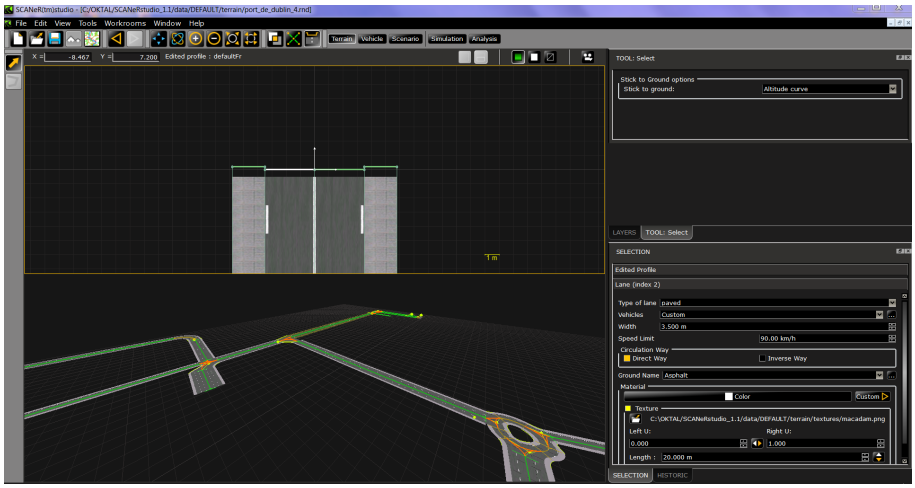

FIGURE A.2 – Mode terrain de SCANeRstudio.

A.3 Mode véhicule

Le mode "Véhicule" permet de créer des profils de véhicules ou de modifier le profil de véhicules déjà existants (Fig. A.3). Les véhicules disponibles sont non seulement des véhicules conventionnels (voitures, poids lourds), mais également des IAVs, des deux roues ou encore des piétons. Un profil de véhicule comprend les dynamiques associées à la suspension, à la pneumatique, à la motorisation ... Un profil définit également certains paramètres du véhicule comme sa position et sa vitesse initiale, sa vitesse maximale, son itinéraire, son mode de contrôle (autonome ou interactif), son comportement de suivi .... Ce mode permet aussi d'attacher une représentation en 3 dimensions à un profil donné.

A.4 Mode scénario

Le mode "Scenario" combine les éléments produits dans les modes "Terrain" et "Vehicule" à savoir placer des véhicules sur un terrain doté d'un réseau routier et d'objets (Fig. A.4).

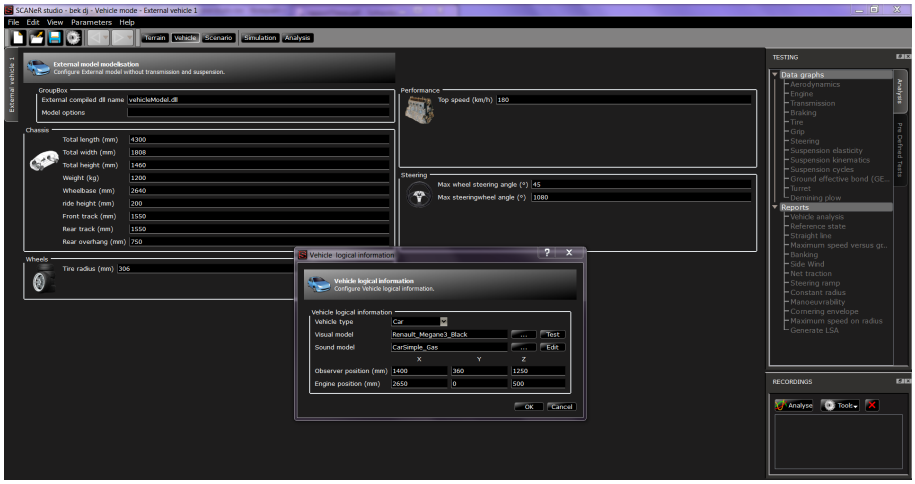

FIGURE A.3 – Mode véhicule de SCANeRstudio.

Les modèles de véhicules étant plus ou moins détaillés on peut créer des scénarios de trafic de véhicules avec différents niveaux de modélisation. Ce mode permet également de créer des scripts pour déclencher des évènements durant la simulation. Nous pouvons même placer des sources et des puits qui vont générer ou supprimer un trafic automatique de véhicule durant la simulation de ces scénarios.

FIGURE A.4 – Mode scénario de SCANeRstudio.

A.5 Mode simulation

Le mode “Simulation” permet d’exécuter la simulation d’un scénario que nous avons défini auparavant (Fig. A.5). Ce mode permet de superviser en 3 dimensions et en temps réel l’évolution des véhicules de cette simulation, avec la possibilité de changer de plan de vue (vue du dessus, vue de côté, vue du conducteur). Pour lancer une simulation il est nécessaire de démarrer un ou plusieurs modules de simulations, appelés ici Application Programming Interface (API). Par exemple l’API “TRAFFIC” gère le trafic des véhicules en mode de conduite autonome, l’API

"PHYSICS" permet de représenter la dynamique de certains objets virtuels (conteneurs) et l'API "VISUAL" assure la visualisation des éléments en 3 dimensions. Ces APIs sont inclus de base dans SCANeRstudio mais il est possible de créer ses propres APIs et les inclure dans ces simulations.

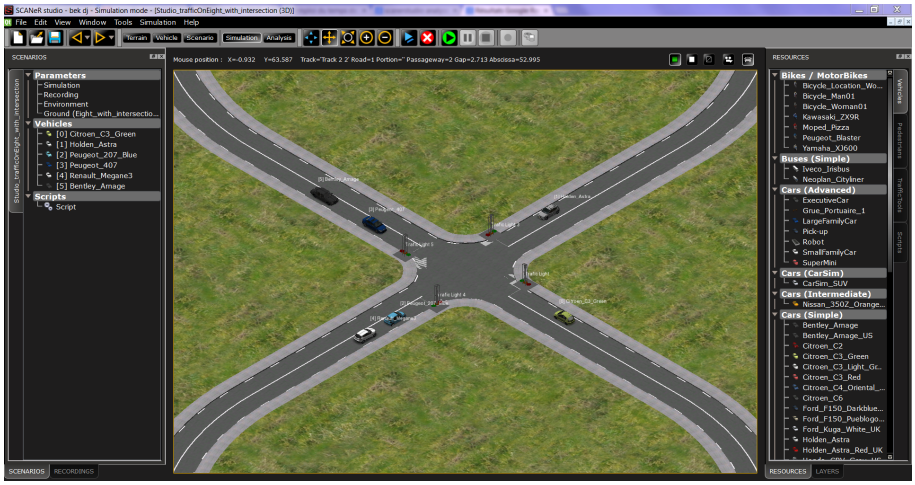

FIGURE A.5 – Mode simulation de SCANeRstudio.

A.6 Mode analyse

Durant et après la simulation, il est possible de consulter toutes les données relatives aux véhicules simulés (Fig. A.6). Ces données peuvent être affichées et analysées dans le mode "Analysis" pour générer un rendu de la dynamique d'un système de transport. Pour récupérer ces données il est nécessaire d'activer l'API "RECORDING" qui se charge d'enregistrer les données relatives à al simulation comme le feraient un ensemble de capteurs.

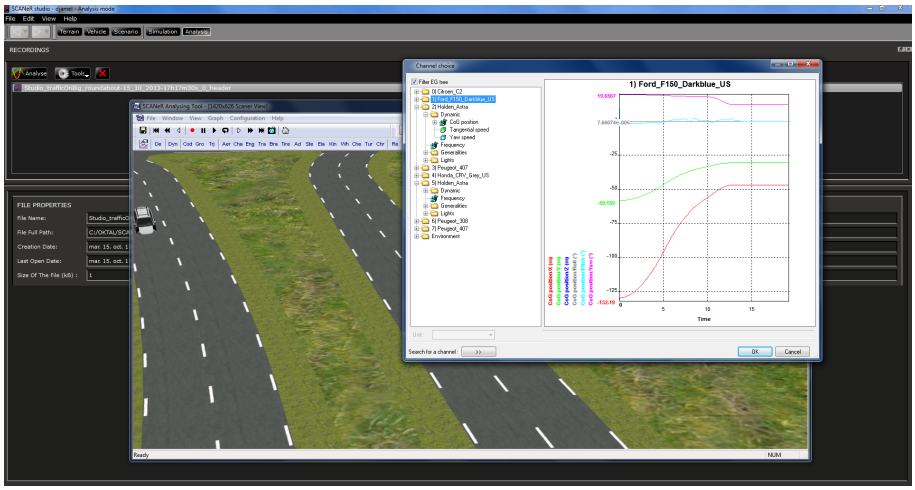

FIGURE A.6 – Mode analyse de SCANeRstudio.

Annexe B

Listes des publications scientifiques

B.1 Articles de Journaux

Jean-Baptiste Soyez, Gildas Morvan, Rochdi Merzouki and Daniel Dupont. "Agent Based Modeling of Systems of Systems". IEEE Transactions on Systems Man and Cybernetics : Systems 2013. (Soumis)

B.2 Articles de conférences internationales avec comité de lecture

Jean-Baptiste Soyez, Gildas Morvan, Rochdi Merzouki, Daniel Dupont and Philippe Kubiak. "Multi-agent Multi-level Modeling - A methodology to Simulate Complex Systems", Proc. of the 23rd European Modeling & Simulation Symposium (EMSS 2011) in Modelling Multiconference (I3M), Roma, Italy. September 2011.

Gildas Morvan, Daniel Dupont, **Jean-Baptiste Soyez** and Rochdi Merzouki, "Engineering hierachical complex systems : an agent-based approach - The case of flexible manufacturing systems". Studies in Computational Intelligence, Volume 402, p. 49-60, 2012.

Jean-Baptiste Soyez, Gildas Morvan, Daniel Dupont, and Rochdi Merzouki. "Methodology to Engineer and Validate Dynamic Multi-level Multi-agent Based Simulations". In Multi-Agent-Based Simulation XIII, Lecture Notes in Computer Science, Giardini, Francesca and Amblard, Frédéric Eds., Springer Berlin Heidelberg, p 130-142, 2013.

B.3 Articles de conférences nationales avec comité de lecture

Jean-Baptiste Soyez, Gildas Morvan, Rochdi Merzouki et Daniel Dupont. "General formalism and algorithms to insure SoS faisability". MASyCo'2013 : Modélisation Agents pour les Systèmes Complexes durant PFIA'2013 : Plate-forme IA 2013, Lille, 1-5 juillet 2013.

Bibliographie

[Acheson 2012] Paulette Acheson, Louis Pape, Cihan Dagli, Nil Kilicay-Ergin, John Columbi et Khaled Haris. *Understanding System of Systems Development Using an Agent- Based Wave Model*. Procedia Computer Science, vol. 12, no. 0, pages 21 – 30, 2012. Complex Adaptive Systems 2012.

[Ackoff 1971] R. Ackoff. *Towards a system of systems concepts*. Management Science, vol. 17(11), pages 661–672, 1971.

[Adler 2012] Charles O. Adler et Cihan H. Dagli. *Enabling Systems and the Adaptability of Complex Systems-of- Systems*. Procedia Computer Science, vol. 12, no. 0, pages 31 – 36, 2012. Complex Adaptive Systems 2012.

[Agusdinata 2008] D.B. Agusdinata et D. DeLaurentis. *Specification of system-of-systems for policymaking in the energy sector*. The integrated Assement Journal, vol. 8, no. 2, pages 1–24, 2008.

[Aniorté 2006] Philipe Aniorté, Eric Cariou et Eric Gouardères. *Modélisation de systèmes complexes distribués : l'ingénierie des modèles pour l'intégration des paradigmes "agent" et "composant"*. In Proceedings of the 2^{nd} Journée Multi-Agent et Composant (JMAC), Nîmes, 21 Mars 2006.

[Aristote 2008] Aristote. La Métaphysique. Flammarion, 2008.

[Bar-Yam 2004] Y. Bar-Yam, M. A. Allison, R. Bartdorf, H. Chen, H. Generazio, H. Singh et S. Tucker. *The Characteristics and Emerging Behaviors of Systems of Systems*. NECSI : Complex Physical, Biological and Social Systems Project, 2004. http ://necsi.edu/education/onexeek/winter05/NCESISoS.pdf.

[Bertalanffy 1968] L. Von Bertalanffy. *General System Theory*. New York, Braziller, 1968.

[Boardman 2006] John Boardman et Brian Sauser. *System of Systems the meaning of* of. In Proceeding of the 2006 IEEE/SMC International Conference on SoS Engineering, Los Angeles, CA, USA, April 2006.

[Bonneaud 2008] S. Bonneaud. *Des agents-modèles pour la modélisation et la simulation de systèmes complexes : application à l'écosystémique des pêches*. PhD thesis, Université de Bretagne occidentale, Brest, 2008.

[Boulding 1956] K. E. Boulding. General system theory—-The skeleton of science, volume 2, page 197. Manag. Sci., 1956.

[Calvez 1990] J.P. Calvez. Spécification et conception des systèmes - une méthodologie. 1990.

[Camus 2013] Benjamin Camus, Christine Bourjot et Vincent Chevrier. *Multi-level modeling as a society of interacting models*. In L. Yilmaz, editeur, SpringSim'13, ADS Symposium - Spring Simulation Multi-Conference, Agent-Directed Simulation Symposium - 2013, volume 1, pages 15–22, San Diego, États-Unis, Avril 2013. Society for Modeling & Simulation International (SCS), Curran Associates, Inc.

[Cantot 2009] P. Cantot et D. Luzeaux. Simulation et modélisation des systèmes : vers la maîtrise de la complexité. Paris, 2009.

[Chen 2011] Dan Chen, Stephen J. Turner et Wentong Cai. *Toward Fault-Tolerant HLA-based Distributed Simulations.* Simulation, vol. 8, pages 493–509, 2011.

[Clarhaut 2009] Joffrey Clarhaut. *Prise en compte des séquences de défaillances pour la conception de systèmes d'automatisation : Application au ferroutage.* PhD thesis, Université des Sciences et des Technologies de Lille, 2009.

[Contet 2011] Jean-Michel Contet, Franck Getcher, Pablo Gruer et Abderrafiaa Koukam. *Reactive Multi-agent approach to local platoon control : stability analysis and experimentations.* Intelligent Systems Technologies and Applications, 2011.

[Coyrehourcq 2011] Sebastien Rey Coyrehourcq et Clara Schmitt. *Attention ! une échelle peut en cacher une autre !* In 17th Rochebrune Conference on Echelles et modélisations multi-niveaux, 2011.

[Dauby 2011] Jason P. Dauby et Steven Upholzer. *Exploring Behavioral Dynamics in Systems of Systems.* Procedia Computer Science, vol. 6, no. 0, pages 34 – 39, 2011. Complex adaptive systems 2011.

[David 2011] Daniel David, Denis Payet et Remy Courdier. *Réification de zones urbaines émergentes dans un modèle simulant l'évolution de la population à La Réunion.* In Journées Francophones sur les Systèmes Multi-Agents (JFSMA'11), pages 63–72, Valenciennes, FRANCE, October 17-19 2011.

[Davis 1993] Paul Davis et R. Hillestad. *Families of Model that Cross Levels of Resolution : Issues for Design, Calibration and Management.* In 25th Winter Simulation Conference (WSC'93), 1993.

[DeLaurentis 2005] Daniel DeLaurentis. *Understanding Transportation as a System-of-Systems Design Problem.* In 43rd AIAA Aerospace Sciences Meeting and Exhibit. American Institute of Aeronautics and Astronautics, 10-13 January 2005. AIAA-2005-0123.

[Drogoul 1993] Alexis Drogoul. *De La Simulation Multi-Agent A La Résolution Collective de Problèmes - Une Étude De l'Émergence De Structures D'Organisation Dans Les Systèmes Multi-Agents.* PhD thesis, Université Paris VI, 1993.

[Drogoul 2003] A. Drogoul, D. Vanbergue et T. Meurisse. *Simulation Orientée Agent : où sont les agents ?* In Actes des Journées de Rochebrune, Rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels, Megève, France, 2003.

[Drogoul 2013] A. Drogoul, E. Amouroux, P. Caillou, B. Gaudou, A. Grignard, N. Marilleau, P. Taillandier, M. Vavasseur, D.-A. Vo et J.-D. Zucker. *GAMA : A spatially explicit, multi-level, agent-based modeling and simulation platform.* Advances on Practical Applications of Agents and Multi-Agents Systems, pages 271–274, 2013.

[Duhail 2013] N. Duhail. *DEVS et ses extensions pour des simulations multi-modèles ne interaction multi-échelles.* PhD thesis, Université de Rennes I - Telecom Bretagne, 2013.

[Eisner 1991] H. Eisner, J. Marciniak et R. McMillan. *Computer-aided system of (C2) engineering.* In IEEEE Int. Conf. Syst., Man Cybern., Charlottesville, VA, 1991.

[El hmam 2006] M.S. El hmam, H. Abouaissa, D. Jolly et A. Benasser. *Macro-micro simulation of traffic flow.* In Proceedings of the12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM06), pages 351–356, Saint Etienne FRANCE, 17-19 Mai 2006.

[Epstein 2006] J.M. Epstein. Generative social science : Studies in agent-based computational modeling. Princeton University Press, 2006.

[Ferber 1995] J. Ferber. Les Systèmes Multi-Agents : Vers une Intelligence Collective. InterEditions, 1995.

[Ferber 1996] J. Ferber et J-P. Müller. *Influences and Reaction : a Model of Situated Multiagent Systems.* In 2nd International Conference on Multi-agent systems (ICMAS-96), pages 72–79, 1996.

[Ferber 1998] J. Ferber et O. Gutknecht. *A meta-model for the analysis and design of organizations in multi-agent systems.* In Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98), pages 128–135, 1998.

[Ferber 1999] J. Ferber. Multi-Agent Systems : An Introduction to Distributed Artificial Intelligence. Addison-Wesley Longman Publishing Co., Inc, 1999.

[Ferber 2004] J. Ferber, O. Gutknecht et F. Michel. *From Agents to Organizations : an Organizational View of Multi-Agent Systems.* In Agent-Oriented Software Engineering (AOSE), volume 4, pages 214–230, 2004.

[Fichwick 1997] P.A. Fichwick. *Computer simulation : growth through extension.* Transactions of the Society for Computer Simulation International, vol. 1, no. 14, pages 13–23, 1997.

[Flanigan 2012] David Flanigan et Peggy Brouse. *System of Systems Requirements Capacity Allocation.* Procedia Computer Science, vol. 8, no. 0, pages 112 – 117, 2012. Conference on Systems Engineering Research.

[Fleszar 2009] Krzysztof Fleszar, Ibrahim H. Osman et Khalil S. Hindi. *A variable neighbourhood search algorithm for the open vehicle routing problem.* European Journal of Operational Research, vol. 195, no. 3, pages 803–809, 2009.

[Gaud 2007] Nicolas A. Gaud. *Systèmes multi-agent holoniques : De l'analyse à l'implantation. Méta-modèle, méthodologie, et simulation multi-niveaux.* PhD thesis, Université de Technologie de Belfort-Montbéliard, 2007.

[Gaud 2008] N. Gaud, S. Galland, F. Gechter, V. Hilaire et A. Koukam. *Holonic Multilevel Simulation of Complex Systems : Application to Real-Time Pedestrians Simulation in Virtual Urban Environment.* Simulation Modelling Practice and Theory, vol. 16, pages 1659–1676, 2008.

[Gerst 2012] M. D. Gerst, P. Wang, A. Roventini, G. Fagiolo, G. Dosi, R. B. Howorth et M. E. Borsuk. *Agent-based modeling of climate policy : An introduction to the ENGAGE multi-level model framework.* Environment Modelling & Software, vol. 1-14, 2012.

[Gezgin 2012] T. Gezgin, C. Etzien, S. Henkler et A. Rettberg. *Towards a Rigorous Modeling Formalism for Systems of Systems.* In Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2012 15th IEEE International Symposium on, pages 204 –211, april 2012.

[Gil Quijano 2009] J. Gil Quijano, G. Hutzler et T. Louail. *De la cellule biologique à la cellule urbaine : retour sur trois expériences de modélisation multi-échelles à base d'agents.* In Actes des 17èmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA'09), 2009.

[Gilbert 2007] Nigel Gilbert. *Agent-Based Models.* (Quantitative Applications In The Social Sciences), Publisher : Sage Publications, no. 153, 2007.

[Gorod 2008] A. Gorod, B. Sauser et J. Boardman. *System-of-Systems Engineering Management : A Review of Modern History and a Path Forward.* Systems Journal, IEEE, vol. 2, no. 4, pages 484 –499, dec. 2008.

[Gutknecht 2001] O. Gutknecht, J. Ferber et F. Michel. *Integrating tools and infrastructures for generic multi-agents systems.* Rapport technique, LIRMM, 161, rue Ada - Montpellier - France, 2001.

[Ham 2006] Claude Van Ham et Trevor Pearce. *The SIP-RTI : An HLA RTI Implementation Supporting Interoperability.* In Proceedings of the Tenth IEEE international Symposium on Distributed Simulation and Real-Time Application, 2006.

[Held 2008] Jason M. Held. *Systems of Systems : Principles, Performances, and Modelling.* PhD thesis, The University of Sydney, 2008.

[HLA] *1516-2010 – IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.*

[Holland 1995] J.H. Holland. Hiden Order : How Adaptations Builds Complexity. Reading, Addison-Wesley, New-York, 1995.

[Hübner 2002a] Jomi Fred Hübner, Jaime Simão et Olivier Boissier. *Moise+ : Towards structural, functional and deontic model for MAS organization.* In Proceeding of the first International joint conference on Autonomous Agents and MultiAgent Systems (AAMAS'02), page 502, Bologna, Italy, July 15-19 2002.

[Hübner 2002b] Jomi Fred Hübner, Jaime Simão et Olivier Boissier. *Spécification structurelle,fonctionnelle et déontique d'organisations dans les SMA.* In Journées Fancophones pour l'Intellignence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'02), 2002.

[Huhn 2011] M. Huhn, J.P. Muller, J. Gormer, G. Homoceanu, Nguyen-Thinh Le, L. Martin, C. Mumme, C. Schulz, N. Pinkwart et C. Muller-Schloer. *Autonomous Agents in Organized Localities Regulated by Institutions.* In Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on, pages 54 –61, 31 2011-june 3 2011.

[IEE 2005] *IEEE Standard for Application and Management of the System Engineering Process*, IEEE 1220, 2005.

[int] *http://www.intrade-nwe.eu/fr*.

[Jackson 1984] M.C. Jackson et P. Keys. *Towards a system of systems methodologies.* The Journal of the Operational Research Society, vol. 35(6), pages 473–486, 1984.

[Jacob 1974] F. Jacob. The Logic of the Living Systems. London, 1974.

[Jamshidi 2008a] Mo Jamshidi. System of Systems Engineering Innovations for the 21st Century. Hoboken : Wiley, 2008.

[Jamshidi 2008b] Mo Jamshidi. System of Systems Engineering Principles and Applications. Boca Raton, FL : Taylor & Francis, 2008.

[Keating 2003] C. Keating, R. Rogers, R. Unal, A. Sous-Poza, R. Safford, W. Peterson et G. Rabadi. *System of systems engineering.* Eng. Manage. J., vol. 15, no. 3, pages 36–45, September 2003.

[Khalil 2011a] Wissam Khalil, Rochdi Merzouki et Belkacem Ould-Bouamama. *Dynamic Model-Based for Intelligent Traffic Optimization Inside Seaport Terminals.* In the 13rd International Conference on Harbor, Maitrise & Multimodal Logistics Modelling and Simulation, 2011.

[Khalil 2011b] Wissam Khalil, Rochdi Merzouki et Belkacem Ould-Bouamama. *Modelling for optimal trajectory planning of an intelligent transportation system.* Intelligent Autonomous Vehicles, vol. 7, 2011.

[Khalil 2012a] W. Khalil, R. Merzouki, B. Ould-Bouamama et H. Haffaf. *Hypergraph Models for System of Systems Supervision Design.* Systems, Man and Cybernetics, Part A : Systems and Humans, IEEE Transactions on, vol. 42, no. 4, pages 1005 –1012, july 2012.

[Khalil 2012b] Wissam Khalil. *Contribution à la modélisation graphique de système de systèmes.* PhD thesis, École Polytechnique Universitaire de LILLE, 2012.

[Koestler 1967] A. Koestler. The Ghost in the Machine. Hutchinson, 1967.

[Kotov 1997] Vadim Kotov. *Systems of Systems as Communicating Structures.* HPL-97-124, 1997.

[Kubera 2008] Y. Kubera, P. Mathieu et S. Picault. *Interaction-oriented simulations : From theory to implementation.* In Proc. of the 18th European Conf. on Artificial Intelligence (ECAI'08), pages 383–387, 2008.

[Lar] *Dictionnaire de français Larousse.* Aux éditions Larousse. `http://www.larousse.fr/dictionnaires/francais/syst\‘eme`.

[Li 2004] Kun Li et Petros Ioannou. *Modeling of Traffic Flow of Automated Vehicles.* In IEEE Transactions on Intelligent Transportation Systems, volume 5, pages 99–113, June 2004.

[Li 2009] Jing-Quan Li, Pitu B. Mirchandani et Denis Borenstein. *Real-time vehicle rerouting problems with time windows.* European Journal of Operational Researche, vol. 194, pages 711–727, 2009.

[Luskasik 1998] S.J. Luskasik. *Systems, systems of systems, and the education of engineers.* Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, vol. 12(1), pages 55–60, 1998.

[Luzeaux 2008a] D. Luzeaux et J.R. Ruault. Ingénierie des systèmes de systèmes - méthodes et outils. 2008.

[Luzeaux 2008b] D. Luzeaux et J.R. Ruault. Systèmes de systèmes : concepts et illustrations pratiques. Paris, 2008.

[mad] *`http://www.madkit.org`*.

[Mahulkar 2009] V. Mahulkar, S. McKay, D.E. Adams et A.R. Chaturvedi. *System-of-Systems Modeling and Simulation of a Ship Environment With Wireless and Intelligent Maintenance Technologies.* Systems, Man and Cybernetics, Part A : Systems and Humans, IEEE Transactions on, vol. 39, no. 6, pages 1255 –1270, nov. 2009.

[Maier 1996] M. Maier. *Architecting Principles for Systems-of-Systems.* In proceeding of the Sixth Annual Internationnal Symposium INCOSE, INCOSE, Boston, MA, 1996.

[Maier 1998] M. Maier. *Architecting Principles for Systems-of-Systems.* In System Engineering, volume 1, pages 267–284, INCOSE, Boston, MA, 1998.

[Manthorpe 1996] W.H. Manthorpe. *The emerging joint system of system : A systems engineering challenge and opportunity for APL.* John Hopkins APL Technical Digest, vol. 17(3), pages 305–310, 1996.

[Marcenac 1998] P Marcenac et S. Giroux. *Geamas : A generic architecture for agent-oriented simulations of complex processes.* Applied Intelligence, vol. 16(7), pages 736–745, 1998.

[Mathew 2005] Reejo Mathew, Jr. James F. Leathrum, Saurav Mazumdar, Taylor Frith et Joseph Joines. *An Object-Oriented Architecture for the Simulation of Networks of Cargo Terminal Operations.* The Journal of Defense Modeling and Simulation : Methodology, Technology, vol. 2, no. 2, pages 101–116, 2005.

[Maus 2008] C. Maus, M. John, M. Rohl et A. Uhrmacher. *Hierarchical modeling for computational biology.* Formal Methods for Computational Systems Biology, vol. 5016 of Lecture Notes in Computer Science, pages 81–124, 2008.

[Michel 2003] Fabien Michel, Abdelkader Gouaïch et Jacques Ferber. *Weak Interaction and Strong Interaction in Agent Based Simulations.* Lecture Notes in Computer Science, vol. 2927, pages 43–56, 2003.

[Michel 2004] F. Michel. *Formalisme, outils et éléments méthodologiques pour la modélisation et la simulation multi-agents.* PhD thesis, Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier - Université Montpellier II, 2004.

[Michel 2007a] F. Michel. *Le modèle IRM4S. De l'utilisation des notions d'influence et de réaction pour la simulation de systèmes multi-agents.* Revue d'Intelligence Artificielle, vol. 21, pages 757–779, 2007.

[Michel 2007b] F. Michel. *The IRM4S Model : The Influence/Reaction Principle for Multi-Agent Based Simulation.* In AAMAS '07 : Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, pages 1–3, New York, NY, USA, 2007. ACM.

[Michel 2009] Fabien Michel, Jacques Ferber, Alexis Drogoul*et al.* *Multi-Agent Systems and Simulation : a Survey From the Agents Community's Perspective.* In A.M. Uhrmacher et Danny Weyns, editeurs, Multi-Agent Systems : Simulation and Applications, Computational Analysis, Synthesis, and Design of Dynamic Systems, pages 3–52. CRC Press, 2009.

[Morin 1990] Edgar Morin. Introduction à la pensée complexe. 1990.

[Morvan 2011] G. Morvan, A. Veremme et D. Dupont. *IRM4MLS : the influence reaction model for multi-level simulation.* In T. Bosse, A. Geller et C.M. Jonker, editeurs, Multi-Agent-Based Simulation XI, volume 6532 of *Lecture Notes in Artificial Intelligence*, pages 16–27. Springer, 2011.

[Morvan 2012a] Gildas Morvan. *Multi-level agent-based modeling - Bibliography.* CoRR, vol. abs/1205.0561, 2012.

[Morvan 2012b] Gildas Morvan, Daniel Dupont, Jean-Baptiste Soyez et Rochdi Merzouki. *Engineering hierarchical complex systems : an agent-based approach. The case of flexible manufacturing systems.* CoRR, vol. abs/1205.7025, 2012.

[Muller 2000] Pierre-Alain Muller et Nathalie Gaertner. Modélisation objet avec UML. Paris, 2000.

[Muller 2009] J.-P. Muller. *Towards a formal semantics of event-based multi-agents ismulations.* Multi-Agent-Based Simulations IX, vol. 5269, pages 110–126, 2009.

[Nanayakkara 2010] T. Nanayakkara, F. Sahin et M. Jamshidi. Intelligent Control Systems with an Introduction to Systems of Systems Engineering. CRC Press, 2010.

[Navarro 2011] Laurent Navarro, Fabien Flacher et Vincent Corruble. *Dynamic Level of Detail for Large Scale Agent-Based Urban Simulations.* In Tumer, Yolum, Sonenberg et Stone, editeurs, 10th Int. Conf on Autonomous Agents and Multiagent Systems (AAMAS 2011), pages 701–708, 2011.

[Navarro 2013] Laurent Navarro, Vincent Corruble, Fabien Flacher et Jean-Daniel Zucker. *A flexible approach to multi-level agent-based simulation with the mesoscopic representation.* In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, AAMAS '13, pages 159–166, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[Owens 1995] A.W.A. Owens. Dominant Battlespace Knowledge, chapitre 'The emerging U.S. system of systems'. National Defense University Press, Washington, 1995.

[Parker 2010] James M. Parker. *Applying a System of Systems Approach for Improved Transportation.* S.A.P.I.E.N.S., vol. 3, no. 2, 2010.

[Pei 2000] R. Pei. *Systems of systems integration (SoSI)-A smart way of acquiring C412WS systems.* Proceedings of the 2000 Summer Computer Simulation Conference, pages 574–579, 2000.

[Petty 2002] Mikel D. Petty. *Comparing high level architecture data distribution management specifications 1.3 and 1516.* Simulation Practice and Theory, vol. 9, pages 95–119, 2002.

[Picault 2011] Sébastien Picault et Philipe Mathieu. *An interaction-Oriented Model for Multi-Scale Simulation.* In the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11), 2011.

[Pisinger 2007] David Pisinger et Stefan Ropke. *A general heuristic for vehicle routing problem.* Computers & Operation Research, vol. 34, pages 2403–2435, 2007.

[Polacek 2004] Michael Polacek, Richard F. Hartl et Karl Doerner. *A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows.* Journal of Heuristics, vol. 10, pages 613–627, 2004.

[Railsback 2011] S.F. Railsback et V. Grimm. Agent-based and individual-based modeling : A practical introduction. Princeton University Press, 2011.

[Resnick 1994] Mitchel Resnick. Turtles, termites, and traffic jams : Explorations in massively parallel microworlds. Mit Press, 1994.

[Sage 2001] Andrew P. Sage et Christopher D. Cuppan. *On the Systems Engineering and Management of Systems of Systems and Federations of Systems.* Inf. Knowl. Syst. Manag., vol. 2, no. 4, pages 325–345, Décembre 2001.

[Sage 2007] A.P. Sage et S.M. Biemer. *Processes for System Family Architecting, Design, and Integration.* Systems Journal, IEEE, vol. 1, no. 1, pages 5 –16, sept. 2007.

[Sauser 2010] B. Sauser, J. Boardman et D. Verma. *Systomics : Toward a Biology of System of Systems.* Systems, Man and Cybernetics, Part A : Systems and Humans, IEEE Transactions on, vol. 40, no. 4, pages 803 –814, july 2010.

[sca] *http://www.scanersimulation.com/.*

[Scerri 2010] David Scerri, Alexis Drogoul, Sarah Hickmott et Lin Padgham. *An Architecture for Modular Distributed Simulation with Agent-based Models.* In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, volume 1 of *AAMAS '10*, pages 541–548, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[SdS 2004] *"SoS" and "FoS" FAQ, Office of the Under Secretary of Defense fo Acquisition, Technology, and Logistics,* `http://www.acq.osd.mil/dpap/Docs/FAQs%20--%20SoS%20&%20FoS.doc`, accessed March 4 2004.

[Seck 2012] M. Seck et H. Honig. *Multi-perspective modelling of complex phenomena.* Computational & Mathematical Organization Organization Theory, vol. 18(1), pages 128–144, 2012.

[Servat 1998] David Servat, Edith Perrier, Jean-Pierre Treuil et Alexis Drogoul. Towards Virtual Experiment Laboratories : How Multi-Agent Simulations Can Cope with Multiple Scales of Analysis and Viewpoints, pages 205–217. Springer-Verlag Berlin Heidelberg, 1998.

[Shannon 1998] R.E. Shannon. *Introduction to the art and science of simulation.* In Proceedings of the 30^{th} conference on Winter simulation, pages 7–14, 1998.

[Shenhar 1994] A. Shenhar. *A new systems engineering taxonomy.* Proceedings ot he 4th International symposium of the National Council on System Engineering, vol. 2, pages 261–276, 1994.

[Simpson 2009] J.J. Simpson et M.J. Simpson. *System of Systems Complexity Identification and Control.* In System of Systems Engineering, SoSE 2009. IEEE International Conference on, pages 1 –6, june 3 2009.

[Sloane 2007] E. Sloane, T. Way, V. Gehlot, R. Beck, J. Solderitch et E. Dziembowski. *A Hybrid Approach to Modeling SOA Systems of Systems Using CPN and MESA/Extend.* In Systems Conference, 2007 1st Annual IEEE, pages 1 –7, april 2007.

[SoS 2010] Systems Engineering Guide for Systems of Systems : Summary. Washington, D.C. : Office of the Director, Defense Research and Engineering, Director of Systems Engineering, 2010.

[Soyez 2011] Jean-Baptiste Soyez, Gildas Morvan, Rochdi Merzouki, Daniel Dupont et Philippe Kubiak. *Multi-Agent Multi-Level Modeling - A Methodology to Simulate Complex Systems.* In The 23rd European Modeling & Simulation Symposium (Simulation in Industry) (EMSS 2011), pages 241–246, September 12-15 2011.

[Soyez 2012] Jean-Baptiste Soyez, Gildas Morvan, Daniel Dupont et Rochdi Merzouki. *A Methodology to Engineer and Validate Dynamic Multi-level Multi-agent Based Simulations.* In MABS'12 - 13th International Workshop on Multi-Agent Based Simulation, Valencia, Spain, 4th-5th June 2012.

[Treuil 2008] J-P. Treuil, A. Drogoul et J-D. Zucker. Modélisation et simulation à base d'agents. Dunod, 2008.

[Uhrmacher 2007] Adelinde M. Uhrmacher, Roland Ewald, Mathias John, Carsten Maus, Matthias Jeschke et Susanne Biermann. *Combining micro and macro-modeling in DEVS for computational biology.* In Proceedings of the 39th conference on Winter simulation : 40 years ! The best is yet to come, WSC '07, pages 871–880, Piscataway, NJ, USA, 2007. IEEE Press.

[Veremme 2010] Alexandre Veremme. *Intérêts et usages de la théorie des fonctions de croyance pour les systèmes d'aide à la décision fondés sur les systèmes multi-agents. Application à l'entomologie médico-légale.* PhD thesis, Université d'Artois, 2010.

[Vo 2012a] Duc An Vo. *An operational architecture to handle multiple levels of representation in agent-based models.* PhD thesis, University of Piere et Marie Curie, 2012.

[Vo 2012b] Duc-An Vo et Alexis Drogoul and. *Multi-Level Agent-Based Modeling : a generic approach and an implementation.* In Frontiers in Artificial Intelligence and Applications„ volume 252 : Advanced Methods and Technologies for Agent and Multi-Agent Systems, pages 91–101, 2012.

[Weyns 2004] Danny Weyns et H. Van Dyke Parunak. Environments ofr Multi-Agents Systems. Springer, 2004.

[Weyns 2013] Danny Weyns et Tom Holvoet. *Model for Simultaneous Actions in Situated Multi-agent Systems*. Lectures Notes in Artificial Intelligence, vol. 2831, pages 105–118, 2013.

[wik] *http://fr.wikipedia.org/wiki/Psychologie_de_la_forme*.

[Winikoff 2011] Michael Winikoff, Hanno-Felix Wagner, Thomas Young, Stephen Cranefield, Roger Jarquin, Guannan Li, Brent martin et Rainer Unland. *Agent-Based Container Terminal Optimisation*. The Information Science Discussion Paper Series, vol. 2011/01, pages xxx–xxx, 2011.

[Wooldridge 2009] M. Wooldridge. An introduction to multiagent systems. John Wiley & Sons, 2nd revised édition, 2009.

[Yang 2011] Feng Yang, Cihan Dagli et Weiping Wanga. *Cognition Evolutionary Computation for System-of-systems Architecture Development*. Procedia Computer Science, vol. 6, no. 0, pages 40 – 45, 2011. Complex adaptive systems.

[Yilmaz 2004] Levent Yilmaz et Tuncer I. Ören. *Dynamic Model Updating in Simulation with Multimodels : A taxonomy and a Generic Agent-Based Architecture*. In Proceeding of Summer Computer Simulation Conference, July 2004.

[Yilmaz 2005] Levent Yilmaz et Tuncer I. Ören. *Discrete-Event Multimodels and their Agent-Supported Activation and Update*. In Proceeding of the Agent-Directed Simulation Symposium of the Spring Simulation Multiconference (SMC'05), pages 63–70, San Diego, CA, April 2005.

[Zeigler 2000] B.P. Zeigler, T.G. Kim et H. Praehofer. Theory of Modeling and Simulation. Academic Press, 2nd édition, 2000.

[Zhou 2011a] Bo Zhou, A. Dvoryanchikova, A. Lobov et J.L. Martinez Lastra. *Modeling System of Systems : A Generic Method Based on System Characteristics and Interface*. In Industrial Informatics (INDIN), 2011 9th IEEE International Conference, pages 361–368, July 2011.

[Zhou 2011b] Bo Zhou, A. Dvoryanchikova, A. Lobov, J. Minor et J.L. Martinez Lastra. *Application of the Generic Modelling Method for System of Systems to Manufacturing Domain*. In IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society, pages 352 –358, nov. 2011.

Conception et modélisation de systèmes de systèmes : une approche multi-agents multi-niveaux

La problématique générale de cette thèse, qui s'inscrit dans le contexte du projet européen InTraDE (Intelligent Transportation for Dynamic Environment), concerne la modélisation de systèmes de systèmes (SdS). Un SdS est un système composé d'une hiérarchie de systèmes autonomes présents à différents niveaux de représentation. Cette thèse répond au besoin d'outils de modélisation généralistes respectant les caractéristiques fondamentales des SdS, en proposant un formalisme multi-agents multi-niveaux et les algorithmes qui assurent le respect de ces caractéristiques. L'utilisation d'un modèle multi-agents permet de profiter de l'autonomie naturelle des agents et l'aspect multi-niveaux de notre modèle permet aux entités modélisées de raisonner à propos de l'organisation hiérarchique du système en leur offrant la notion explicite de niveau.

En plus de la modélisation des systèmes complexes, cette thèse aborde les problématique liées à leur simulation, en particulier, le fait que les ressources informatiques (mémoire et microprocesseur utilisés) nécessaires pour simuler avec précision de tels systèmes sont particulièrement importantes. Nous proposons ainsi une méthodologie pour tirer partie de la capacité des simulations multi-niveaux à produire un compromis entre la précision de la simulation et les ressources informatiques utilisées.

Mots-clés : modélisation et simulation multi-agents multi-niveaux, Systèmes de systèmes, approche organisationnelle, modèle Influence/Réaction, systèmes complexes, véhicules intelligents autonomes.

Conception and modeling of systems of systems : a multi-level mult-agent approach

The main problematic of this thesis, which takes place in the context of the european project InTraDE (Intelligent Transportation for Dynamic Environment), deals with the modeling of systems of systems (SoS). A SoS is a system composed of a hierarchy of autonomous systems present in several representation levels. This thesis answers the need of generic modeling tools respecting the fundamental characteristics of SoS, proposing a multi-level multi-agent formalism and algorithms wich insure their respect. The use of a multi-agent model allows to take advantage of the natural autonomy of agents and the multi-level aspect of our model permits to modeled entities to reason about the organisational hierarchy of the system, carrying the explicit notion of level.

Besides the modeling of complex systems, this thesis also deals with the problematic related to their simulations, particularly, the fact that computer resources (used memory and microchips) needed to simulate with precision such systems are truly important. We propose a methodology to benefit from the muli-level simulations capacity to produce compromise between the simulation precision and the used computer resources.

Keywords : multi-level multi-agent modeling and simulation, Systems of systems, organisational approach, Influence/Reaction model, complex systems, intelligent autonomous vehicles.