Un modèle multi-agent récursif générique pour simplifier la supervision de systèmes complexes artificiels décentralisés

Thi Thanh Ha Hoang

▶ **To cite this version:**

Thi Thanh Ha Hoang. Un modèle multi-agent récursif générique pour simplifier la supervision de systèmes complexes artificiels décentralisés. Autre [cs.OH]. Université de Grenoble, 2012. Français. NNT : 2012GRENM033 . tel-00745805

HAL Id: tel-00745805

https://theses.hal.science/tel-00745805

Submitted on 26 Oct 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministérial : 7 août 2006

Présentée par

Thi-Thanh-Ha HOANG

Thèse dirigée par **Michel OCCELLO**
et codirigée par **Thanh-Binh NGUYEN**

préparée au sein **Laboratoire de Conception et d'Intégration des Systèmes**
et de **l'Ecole Doctorale Mathématiques Sciences et Technologies de l'Information**

Un modèle multi-agent récursif générique pour simplifier la supervision de systèmes complexes artificiels décentralisés

Thèse soutenue publiquement le **12/09/2012**,
devant le jury composé de :

M. Choukri BEN YELLES professeur, Président
M. René MANDIAU professeur, Rapporteur
M. Olivier BOISSIER professeur, Rapporteur
M. Sebastien PICAULT MCF, Examinateur
M. Michel OCCELLO professeur, Directeur de thèse
M. Thanh Binh NGUYEN MCF, Co-Directeur de thèse

Remerciements

Je tiens à remercier particulièrement Monsieur Choukri BEN YELLES, Professeur à l'Université Pierre Mendes France et au LCIS, pour m'avoir fait l'honneur de présider ce jury et pour sa contribution aux éléments formels de ma thèse.

Je tiens tout d'abord à remercier Monsieur Michel OCCELLO, Professeur à l'Université Pierre Mendes France et au LCIS, Directeur de thèse. Je lui suis sincèrement reconnaissante de m'avoir dirigé, conseillé et guidé. Son aide et son soutien ont été des atouts précieux tout au long de ma thèse.

Je remercie également Monsieur René MANDIAU, Professeur à l'Université de Valenciennes et du Hainaut-Cambresis et Monsieur Olivier BOISSIER, Professeur à l'Ecole Nationale Supérieure des Mines de Saint-Etienne, d'avoir accepté d'être rapporteurs de mon travail. Merci de leur collaboration ainsi que de leurs conseils avisés et constructifs pour améliorer mon manuscrit.

Je remercie également Monsieur Sébastien PICAULT, Maître de conférences à l'Université de Lille, pour sa participation au jury.

Je remercie sincèrement Monsieur Jean-Paul JAMONT, Maître de conférences à l'Université Pierre Mendes France et à l'IUT Valence, pour ses encouragements et le regard qu'il a apporté au quotidien à ma thèse.

Je tiens à remercier particulièrement Monsieur Thanh-Binh NGUYEN, Maître de conférences à l'Université de DaNang, VietNam, qui m'a dirigé, conseillé et soutenu dans mon travail.

Je remercie Monsieur Eduador MENDES, Professeur à l'ESISAR-INPG à Valence, Madame Carole SEYVET, et Mademoiselle Jennyfer DUBERVILLE de m'avoir accueilli dans le laboratoire LCIS, tous les membres du laboratoire et plus particulièrement l'équipe COSY avec qui j'ai partagé mes inquiétudes et mes doutes.

Un grand merci à Agnès et Maurice CARRE, ma famille d'accueil, qui m'ont encouragé dans mon travail. Durant mes séjours en France et auprès d'eux, j'ai progressé dans la pratique de la langue et adopté la culture française.

Je tiens à remercier particulièrement mes collègues du département Statistique et Informatique, Ecole des Sciences Economiques. Merci pour leur disponibilité et leur écoute.

Je remercie mes parents, mes beaux-parents, ma famille pour leur soutien et pour s'être occupés de mes enfants afin de pouvoir poursuivre mes recherches et d'aller au bout de ce projet.

Merci à tous.

Table des matières

Table des figures

Liste des tableaux

Introduction

Contexte du travail

Les systèmes complexes sont de plus en plus présents dans la vie courante, de manière plus ou moins visible. Les travaux de recherche sur ces systèmes sont de plus en plus considérés comme une discipline à part entière touchant à de nombreux domaines scientifiques tels que la physique, la chimie, la biologie, l'informatique, la sociologie ou encore l'économie.

Nous nous intéressons dans ce travail aux systèmes complexes artificiels. Nous considérons que les grands systèmes logiciels à base de composants ou de services, les logiciels sociaux pour le web, les systèmes de supervision de dispositifs physiques, les systèmes d'informatique ambiante peuvent être vus comme des systèmes complexes de nature artificielle.

Notre objectif est de donner à l'utilisateur (ou éventuellement un autre système) la capacité d'observer ces systèmes de façon globale ou plus locale selon les besoins afin d'en assurer la supervision ou de prendre des décisions.

Parmi ces catégories de systèmes, se placent les réseaux de capteurs sans fils et les systèmes de services auxquels nous porterons le plus d'attention. Ces domaines peuvent nous permettre d'illustrer concrêtement l'objectif sur des situations précises comme par exemple un utilisateur qui veut interroger plus particulièrement une zone observée où se situent de très nombreux capteurs ou qui veut suivre le déroulement d'une composition de services web qui réalisent une requête sur la toile. Des problèmes se posent rapidement pour ces deux cas quand on constate que ces systèmes sont de taille importante, qu'ils sont ouverts et totalement décentralisés. Leur complexité rend leur exploration systématique et leur exploitation difficile. Il est utile de les observer en introduisant des échelles d'abstraction afin de les rendre plus intelligible. Ces systèmes complexes artificiels présentent dans leur nature certaines propriétés multi-échelles. Des descriptions multi-échelles sont donc importantes pour comprendre les systèmes complexes artificiels (en réduire la complexité) et plus loin les observer et les contrôler.

Dans notre travail, nous considèrerons le système complexe artificiel comme le phénomène réel qu'il faut observer. Les niveaux d'observation seront vus comme des phénomènes virtuels (figure 1) abstrait destinés à faciliter la supervision.

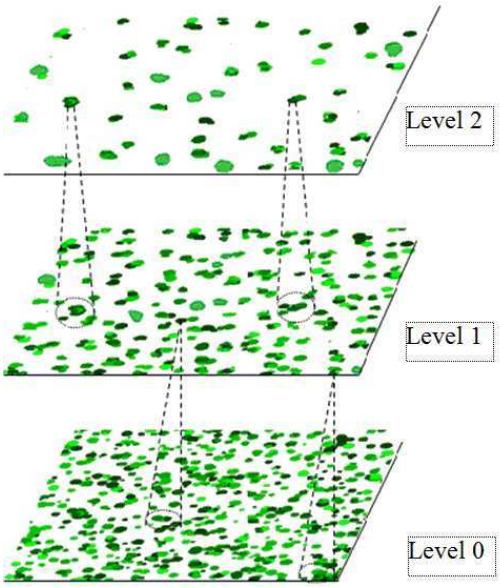


FIGURE 1 – Observation des systèmes complexes multi-niveaux d'abstraction.

Démarche suivie

Le terme échelle s'applique à trois dimensions : spatiale, d'analyse et temporelle ((**?**)). Pour chacune de ces dimensions, les systèmes peuvent être explorés à travers des échelles (ou des niveaux) pour obtenir le niveau de lecture ou de description nécessaire. Dans ce travail, nous limitons notre étude à la dimension spatiale pour observer différents niveaux abstraits des systèmes complexes.

Les questions théoriques soulevées sont très variées. Quels sont les éléments que l'on veut observer à travers des échelles ? Comment représenter des comportements collectifs qui émergent à partir des comportements individuels à travers des structures organisées ? Quelle sont les règles pour construire les niveaux ? Comment le système d'observation s'adapte quand il y a un changement du système réel ?

Modéliser les systèmes complexes artificiels à l'aide de SMA.

Pour répondre aux besoins de modélisation des systèmes complexes, l'utilisation de l'approche multi-agents est une solution appropriée. L'organisation, les composants et les interactions entre les composants dans les systèmes complexes artificiels nous mènent à utiliser l'approche AEIO (**?**) du SMA, qui est basé sur quatre axe *A* (*Agent*), *E* (*Environnement*), *I* (*Interaction*) et *O* (*Organisation*). Cette décomposition est adaptée aux systèmes complexes de par sa vision centrée agent (toute les briques sont finalement intégrées dans le (A)) et de l'expression externe du (I) et du (O).

Construire des abstractions d'un SMA.

Nous décrirons donc nos systèmes complexes artificiels comme des systèmes multi-agents (SMA). Ces SMA seront considérés au niveau d'observation le plus fin (niveau 0 ou niveau réel). Notre approche consistera à construire des niveaux d'abstraction pour ces SMA. En regroupant les agents et les objets du niveau 0 sur certains critères comme la structure d'organisation, leurs relations, leurs connaissances ou leur capacités, on peut mettre en évidence que le système contient des partitions ou phénomènes. Ces phénomènes représentent les réalités localisées correspondant à des critères précis d'observation (par exemple «tous les capteurs d'une zone», «les services web rendant un service donné», «les robots participant à la tâche X», «les membres d'un réseau social possédant une compétence spécifique»...). Si on éloigne l'observateur et que l'on voit un phénomène virtuel (composé d'individus) comme un tout, on peut alors passer à un autre niveau d'observation (appelé niveau 1). Au niveau 1, chaque phénomène virtuel représente des individus agrégés du niveau 0. Cela nous donne une nouvelle perception du système. Notamment, la complexité du système au niveau 1 est réduite, l'intelligibilité du système est donc améliorée. De la même façon que pour le niveau 1, on obtient des niveaux supérieurs (2, 3,....) en appliquant les mêmes règles de partition des phénomènes au niveau inférieur. Les phénomènes des niveaux 1, 2, 3... sont des phénomènes virtuels. Ils existent à travers leurs phénomènes agrégés. La notion d'échelle spatiale peut être assimilée ici à une échelle de densité de phénomène. Un niveau d'observation donne une vision abstraite du système. A travers les niveaux, l'intelligibilité et la complexité évoluent en sens inverse.

Mettre en avant les propriétés récursives.

Certains de ces systèmes présentent dans leur nature des propriétés récursives. Nous défendons dans nos travaux que l'intégration de l'aspect récursif dans le modèle est une façon de construire des niveaux d'observation dont les dépendances et les déductions entre les niveaux sont basées sur les propriétés récurrentes. La récursivité est aussi un moyen de résoudre les systèmes complexes en appliquant la même solution sur un système dont la taille est moins importante.

Viser la généricité.

Il va de soit que cette approche de la problématique doit se faire dans un objectif de réutilisation. Cela impose de trouver une réponse générique au problème afin qu'elle puisse être utilisée sur un SMA de base quelconque sur la classe définie.

L'objectif de la thèse

– Les travaux (**?**; **?**; **?**) menés jusqu'ici dans l'équipe ont montré que l'on peut construire un SMA récursif qui est indépendant du modèle d'agent auquel il s'applique. Le premier objectif de la thèse est de construire un *modèle de SMA récursif générique* pour la modélisation, l'observation et la supervision des systèmes complexes munis de propriétés multi-échelles. Le modèle proposé, nommé **SMA-R**, est capable de construire des niveaux d'abstraction à partir d'un SMA donné. Ce modèle s'adapte à l'ouverture, à la dynamique et aux grandes dimensions des systèmes complexes considérés.
– La classe de systèmes visés souvent en relation avec le monde physique impose de proposer une architecture vraiment (physiquement) décentralisée. Ce travail se fixe pour but de proposer une implémentation réellement distribuée sur les agents. Le deuxieme objectif de cette thèse est *une architecture décentralisée* pour le modèle SMA-R.
– Afin de rendre réutilisable l'architecture décentralisée générique du modèle SMA-R, nous proposons comme troisième objectif de construire un framework générique. Ce framework rend enfichable sur les agents du SMA réel, l'infrastructure d'agents récursifs pour construire les niveaux d'observation.
– Le dernier objectif est de *valider cette contribution sur une application* et de montrer que les *performances du framework sont acceptables au regard du service rendu*.

Organisation du manuscrit

Le manuscrit de thèse est organisé en trois parties.

La partie **Contexte** contient deux chapitres, cette partie présente le contexte de travail.

Le premier chapitre aborde les *systèmes complexes* et les systèmes multi-échelles. Dans ce chapitre, nous tentons de montrer combien les descriptions multi-échelles sont importantes pour observer les systèmes complexes et les contrôler. De l'échelle la plus fine à l'échelle la plus grosse, nous percevons des composants avec des granularités différentes. Par la suite, nous présentons l'état de l'art sur l'étude des systèmes multi-échelles et nous identifions les capacités que les modèles multi-échelles doivent satisfaire : la construction des niveaux, l'observation de chaque niveau, l'insertion ou la suppression d'un élément sur n'importe quel niveau, l'action sur n'importe quel niveau, l'interaction entre les niveaux.

Le deuxième chapitre est consacré aux *systèmes multi-agents*. Dans ce chapitre, nous abordons la définition d'agent et de système multi-agents. Nous analysons des approches multi-agents géné-

riques qui s'adaptent aux systèmes complexes multi-échelles. Nous tentons de mettre en avant les caractéristiques particulières de chaque modèle et de chaque famille dans l'objectif de construire un modèle de système multi-agent plus approprié et plus efficace pour répondre aux questions des systèmes complexes artificiels multi-échelles (multi-niveaux).

La partie **Contribution** contient trois chapitres, ce sont les contributions principales de la thèse.

Le premier chapitre de cette partie présente une tentative de *définition formelle de SMA Récursif* - une contribution originale de la thèse. Afin d'obtenir des niveaux abstraits d'un SMA donné, nous introduisons les propriétés récurrentes de quatre axes A, E, I et O comme une relation entre les éléments de deux niveaux consécutifs. Les propriétés récurrentes sont un moyen de construction de l'observation d'un SMA à différents niveaux. Une illustration de cette formalisation est donnée à la fin de ce chapitre sur le cas de la généralisation cartographique.

Suite à cette définition formelle du SMA récursif, dans le deuxième chapitre, nous proposons le *modèle SMA Récursif générique SMA-R* qui a pour objectif de modéliser et d'observer des systèmes multi-échelles. C'est la contribution centrale de la thèse. Nous mettons en évidence des propriétés récurrentes dans les cinq parties du modèle. La partie (K) connaissances nous permet de mettre en place les propriétés récurrentes de A, E et O. La partie (IR) interaction récursive nous permet d'implémenter la propriété récurrente sur le I. Nous mettons également en avant la généricité du modèle.

Le troisième chapitre concerne une *architecture générique décentralisée* du modèle SMA-R. Notre objectif est d'appliquer ce modèle à des SMA physiquement décentralisés. Nous proposons d'adopter les concepts du modèle OSI (Open System Interconnection) pour que cette architecture soit vraiment décentralisée. Grâce à cette architecture, nous construisons un framework générique réutilisable permettant d'enficher les agents récursifs sur les agents de l'application.

La partie **Application et validation** est composée de deux chapitres.

Le premier chapitre décrit l'application du modèle SMA-R et de son architecture décentralisée à un SMA pour la *supervision d'un réseau de capteurs sans fils* qui est un sujet d'étude dans notre équipe. Nous décrivons dans ce chapitre nos réalisation notamment la connexion entre SMA-R et l'outil MASH étendu à la visualisation des niveaux d'abstraction.

Le deuxième chapitre est consacré à l'*évaluation* et à la validation de notre modèle en terme de fonctionnalités, de performance, et réutilisabilité.

Enfin, le manuscrit se termine par une **conclusion** et les perspectives de ce travail.

Première partie

Contexte

Chapitre 1

Les Systèmes Complexes

Dans ce chapitre, nous donnons notre lecture de la notion de systèmes complexes par l'étude de références, reconnues dans le domaine, qui définissent leurs caractéristiques. Nous abordons ensuite la notion de multi-échelle comme une propriété particuliere de systèmes complexes qui présentent des organisations ou des structures observables à différents niveaux d'abstraction. Nous parcourons de nombreux exemples où les auteurs mettent l'accent sur les modèles multi-échelles. Ces exemples sont issus de domaines aussi divers que les géosciences, les sciences de l'environnement, la physique, la biologie, la géographie, la robotique ou les réseaux sociaux. Ce parcours a pour but d'identifier les caractéristiques indispensables que l'on doit pouvoir satisfaire pour modéliser un problème multi-échelle.

1.1 Les systèmes complexes

Les systèmes complexes sont de plus en plus présents dans la vie courante, de manière plus ou moins visible. Les travaux de recherche sur ces systèmes se révèlent de plus en plus interdisciplinaires composés de plusieurs briques scientifiques tels que la physique, la chimie, la biologie, l'informatique, et également la plupart des sciences humaines et sociales comme la sociologie ou l'économie.

Un système est un ensemble cohérent de composants en interaction qui poursuivent un but commun. Le terme *complexe* indique que plusieurs idées ou plusieurs éléments sont impliqués dans un système, qui le rendent difficile à comprendre. Ces systèmes sont constitués d'un grand nombre d'entités en interaction, on les qualifie de complexes lorsqu'un observateur ne peut prévoir leur comportement ou leur évolution par un calcul simple. Pour connaître l'évolution du système, on doit faire l'expérience, éventuellement sur un modèle réduit.
D'après (**?**), l'idée d'un système complexe est d'avoir un système régi par une dynamique liée à de

nombreuses interactions entre composants, décrits eux-mêmes selon plusieurs niveaux d'organisation, et dont la sémantique des interactions diffère d'un niveau à un autre. Les relations qui unissent les composants doivent privilégier plutôt l'aspect dynamique que l'aspect statique du système interactif.

Un *système complexe* est en général un système composé d'un grand nombre d'entités hétérogènes, parmi lesquelles les interactions locales créent plusieurs niveaux de structures et d'organisation collective. Les exemples incluent les systèmes naturels allant des biomolécules et des cellules vivantes à des systèmes sociaux humains et à l'écosphère, ainsi qu'aux systèmes sophistiqués artificiels tels que l'Internet, les réseaux électriques ou tout autre système de logiciels réparti à grande échelle.

Les systèmes complexes sont aussi vus comme des systèmes non linéaires dont les compétences issues des interactions entre les composants dépassent la somme des contributions individuelles de chaque composant (**?**).

La caractéristique essentielle d'un système complexe est donc le nombre d'entités ainsi que les relations qui s'établissent entre ces entités. Comme les entités interagissent entre elles pour atteindre un but commun, les systèmes complexes possèdent une capacité d'évolution. Certains systèmes complexes se composent de plusieurs niveaux hiérarchiques dont la complexité est croissante ou décroissante selon l'approche retenue pour les envisager.

Caractérisation d'un système complexe

Au lieu de donner une définition précise d'un système complexe, nous nous intéressons à préciser quelques caractéristiques qui nous semblent essentielles pour les identifier.

Selon la définition de Goldenfeld et Rind (**?**), les systèmes complexes ont les caractéristiques suivantes :

- les relations sont non linéaires : cela signifie qu'une petite perturbation peut provoquer un effet important (l'effet papillon), un effet proportionnel, ou même aucun effet du tout.
- les relations contiennent des boucles de rétroaction : le comportement d'un élément peut entrainer des effets sur lui-même. Ces effets peuvent être négatifs (amortissement) ou positifs (amplification).
- les systèmes complexes sont ouverts : sous des influences externes, les systèmes complexes sont souvent loin d'un état d'équilibre mais peuvent atteindre des schémas de stabilité.
- les systèmes complexes ont une mémoire : l'histoire d'un système complexe est importante. Les systèmes complexes sont des systèmes dynamiques, ils évoluent avec le temps, les états passés peuvent avoir une influence sur les états présents. Les systèmes complexes peuvent

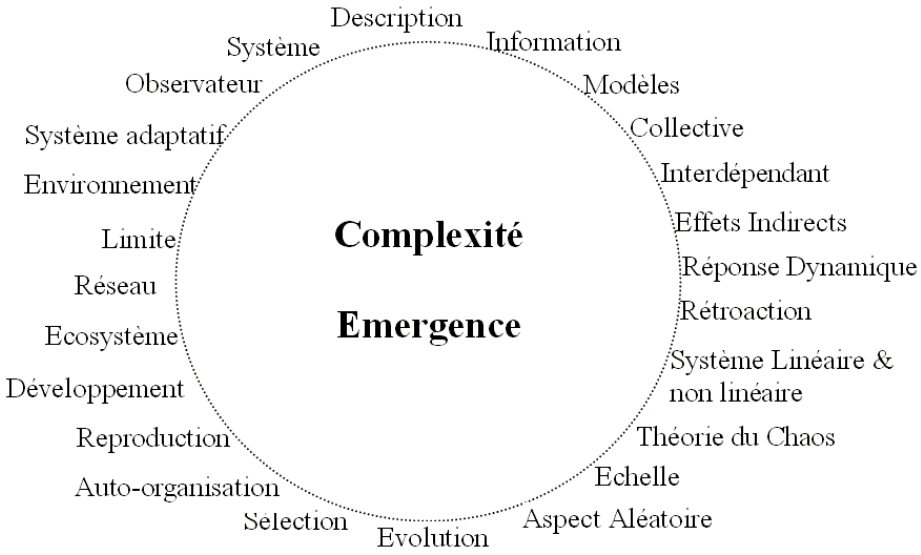


FIGURE 1.1 – Carte de concepts

même mémorisés les effets connus d'un état sur l'autre (hysteresis).

- les systèmes complexes peuvent être imbriqués : les composants d'un système complexe peuvent être eux-mêmes des systèmes complexes.
- on peut exprimer des réseaux de relations dynamiques ainsi que de règles de couplage entre les éléments d'un système complexe : on peut construire des topologies de ces relations.
- les limites des systèmes complexes sont parfois difficiles à déterminer : elles sont souvent finalement établies par l'observateur.
- les systèmes complexes peuvent produire des phénomènes émergents : si les résultats peuvent être déterministes, ils peuvent avoir des propriétés qui ne peuvent être étudiées qu'à un niveau supérieur d'observation.

Pourtant, pour donner une vision globale du terme, le NECSI [1] propose des concepts, en considérant la majorité des applications possibles dans les domaines scientifiques et professionnels (**?**). Dans son travail [2], Y. Bar-Yam met l'émergence et la complexité au coeur de la ***Carte de concepts*** (figure 1.1) et considère qu'ils jouent un rôle fondamental pour les systèmes complexes.

Afin d'étudier des systèmes complexes, il considère les points importants suivants (**?**) :

- Les descriptions multi-échelles sont nécessaires pour comprendre les systèmes complexes : la question des paramètres pertinents lors des passages d'échelles plus fines à des échelles plus grandes reste difficile.
- Les comportements des fines échelles influencent les comportements des grandes échelles : comment exprimer par exemple, que l'activité chimique contrôle le fonctionnement méca-

1. New England Complex Systems Institute (NECSI) est un établissement éducatif, indépendant et aussi un Centre de Recherche consacré à l'étude des systèmes complexes
2. http ://www.necsi.edu/guide/concepts/conceptmap.html

nique des muscles.

- La formation de patterns de comportement que l'on peut reproduire.
- Les états stables : on peut discerner de petites perturbations qui conduisent à un retour à l'équilibre, et de plus grandes perturbations qui peuvent conduire à des changements radicaux de propriétés.
- La complexité du système : comment apprécier la complexité ? Une réponse usuelle porte sur la quantité d'information nécessaire pour décrire le système. Les inférences peuvent être utilisées pour obtenir la description à partir des quantités d'informations plus petites. La complexité dépend aussi de l'échelle à laquelle le système est décrit. Pourtant, une fois qu'une échelle particulière est choisie, la complexité doit être bien définie et bornée (à un instant donné) par les informations nécessaires pour décrire le système.
- La complexité du comportement : pour décrire le comportement (actions) d'un système en réponse à son environnement, nous essayons souvent de décrire une fonction de réponse. Cependant, à moins que certaines hypothèses simplificatrices ne soient faites, la réponse à chaque environnement dépend d'une quantité d'informations qui croît exponentiellement avec la complexité de l'environnement.
- Emergence : il s'agit de la dépendance entre le tout et ses parties. Lorsqu'on étudie le système, ses parties doivent être étudiées et il est nécessaire d'étudier comment les changements dans une partie affectent les autres parties et tout le système.
- Le couplage entre les composants du système : l'auteur introduit une règle (7 ± 2) qui tente de donner une expression de l'interdépendance moyenne entre les parties d'un système.
- La relation entre les systèmes et leurs descriptions : il s'agit de la compréhension de la théorie et des simulations, de la reconnaissance des systèmes dans leurs modèles, de l'encodage et du décodage (compression), et de la complexité algorithmique.
- La composition : il s'agit de former un nouveau système complexe à partir de parties d'autres systèmes complexes. Ces parties doivent être indépendantes. Le but de la composition est de permettre une évolution rapide.
- La hiérarchie de contrôle : quand un seul composant contrôle le comportement collectif d'un système, alors le comportement collectif ne peut être plus complexe que le comportement individuel.
- La modélisation et la simulation : certaines méthodes de simulation ont été proposées pour des systèmes complexes, comme Monte-Carlo, recuit simulé, les automates cellulaires.

Les descriptions multi-échelles sont importantes pour comprendre des systèmes complexes (les rendre plus facilement lisible), et pouvoir les observer et les contrôler. De l'échelle la plus fine à l'échelle la plus grosse, nous percevons sur les niveaux des composants avec des granularités différentes. Chaque granularité associée à chaque composant se situe à une échelle quelconque. Dans

la suite de ce chapitre, nous considérons plus particulièrement ces composants et leurs relations, nous exposons notre lecture de la notion de multi-échelle, et tirons de l'analyse de travaux dans des domaines très divers quelques recommandations pour construire des systèmes répondant à une organisation multi-échelle.

1.2 Les systèmes multi-échelles

De nombreux systèmes peuvent être décrits à différentes échelles. Un nuage vu du sol a un aspect relativement homogène, rond et sans trop de discontinuité apparente. Si on se rapproche, on voit que la forme de ce nuage n'est pas régulière et que celle-ci est composée de plus petites structures, elles-mêmes irrégulières et composées de structures irrégulières encore plus fines, etc.

Il existe des systèmes complexes naturels ou artificiels qui portent dans leur nature la caractéristique multi-échelle temporelle ou spatiale. La structure multi-échelle est populairement reconnue dans toutes les disciplines scientifiques fondamentales et appliquées comme les mathématiques, la physique, la chimie, l'astronomie, la géologie, la biologie, et dans des domaines appliqués tels que les matériaux, la mécanique, l'analyse d'image, les méthodes de calcul, les sciences de l'atmosphère, le génie chimique... Et il est si important qu'un nouveau terme, les *sciences multi-échelles*, a été proposé, tout à fait indépendament dans différents domaines, et a été considéré comme un défi pour le 21e siècle (**?**; **?**). Des conférences scientifiques sur ces sujets ont été organisées ces dernières années (**?**), (**?**), (**?**), (**?**), (**?**).

Les changements d'échelle de temps ou d'espace sont étudiés dans la plupart disciplines, séparément ou conjointement. Les échelles de temps concernent particulièrement les simulations de phénomènes ou de commandes. Nous nous intéressons dans cette thèse aux travaux concernant l'échelle spatiale permettant une interprétation abstraite multi-niveau d'un phénomène ou d'un système.

Dans le cadre de la supervision de systèmes complexes artificiels la notion d'échelle spatiale sera vue comme un changement de granularité de représentation entre les niveaux. C'est dans ce sens que nous utiliserons les termes *multi-échelle* ou *multi-niveau*.

Systèmes et outils informatiques

Parmi des phénomènes du monde réel qui sont étudiés comme des systèmes complexes (**?**), l'Internet a été considéré comme un "exemple fondamental d'un système d'une grande échelle, de haute technicité, et encore très complexe". Le Web des données relie toutes les données sur le Web pour constituer un réseau global d'informations. Les auteurs proposent d'étudier le Web des données sous la forme d'un système multi-échelle. Le modèle multi-échelle proposé permet de comprendre

et de mieux observer l'évolution du Web des données au niveau détaillé ainsi qu'au niveau global. Ce modèle se compose de deux échelles. L'échelle de base est constituées par des triplets. Chaque triplet comporte un sujet, un objet et une relation reliant le sujet et l'objet. Les sujets et les objets sont des ressources sur le Web. L'échelle plus élevée est un groupe de triplets. Un groupe de triplets est créé en regroupant les triplets. Ce regroupement dépend du choix du concepteur et se base sur les contenus des ressources des triplets. Si on ajoute ou supprime des ressources ou des relations des triplets, les modifications correspondant au niveau de groupe de triplets sont aussi réalisées.

Le système multi-échelle est aussi exploité dans le domaine d'animation des images (**?**). En synthèse d'images, les scènes tridimensionnelles animées sont de plus en plus riches et détaillées (**?**). Grâce à la synthèse d'images, l'utilisateur devient, via des périphériques comme l'écran et la souris, à la fois observateur et acteur de mondes de synthèse que l'on dit virtuels. Mais, les utilisateurs veulent toujours plus d'objets, d'animations et d'interactions. Évidemment, à cause des limites corporelles et des limites intellectuelles, l'homme ne peut que percevoir certaines des caractéristiques des images, surtout le mouvement des images. Dans ce travail, les auteurs ont introduit une méthode de modélisation multi-échelles procédurale pour l'animation des images. Il utilise la technique *complexification* qui décrit un modèle par une représentation grossière et par un ensemble de fonctions qui lui ajoutent localement des détails jusqu'à satisfaire la précision requise par des critères perceptuels.

Information Géographique

Certaines activités humaines doivent être localisées dans l'espace : évaluer des distances, mesurer des surfaces, chercher les chemins. D'autres ont besoin d'informations sur le paysage, les objets et leur forme (**?**). La carte a servi (et continue à servir) de moyen de communication privilégié pour transmettre de tels messages ; c'est une vue, une représentation particulière d'un espace géographique adaptée à une utilisation précise, avec des informations différentes. Pour composer une carte, le rôle de l'échelle (rapport entre une mesure de distance sur la carte et une même mesure sur le terrain) est majeur. Elle détermine la taille de la carte, la sélection des objets et leur représentation par leur emprise ou un symbole en fonction de leur taille. Les représentations multiples sont un des problèmes clé dans le domaine des SIG (**?**). En effet, pour l'analyse spatiale et la recherche d'itinéraire, une méthode de raisonnement avec différents niveaux de détails, de résolution, de précision, est une bonne démarche (**?**). La généralisation [Ruas et Lagrange 95] de représentation géographique consiste à modifier les données, afin d'obtenir une représentation plus simple et plus abstraite. Le but de la généralisation est donc de réduire le nombre de données transmises en conservant autant que possible l'essentiel de l'information qu'elles véhiculent. Elle englobe une simplification descriptive (ou sémantique) et géométrique. Les questions posées dans la thèse de

Devogele (**?**) est de définir un processus de constitution de base de données géographiques (BDG) multi représentation issues de différentes BDG. Pour atteindre ce but, l'auteur construit deux processus : l'intégration et l'appariement de BDG classiques. Cette solution paraît le meilleur compromis possible, favorisant la réutilisation des nombreuses représentations existantes en étendant leur portée par une intégration de celles-ci. Le résultat de cette intégration n'est pas d'obtenir une seule représentation mais de permettre l'interopérabilité entre les bases et de relier les instances des différentes représentations représentant le même phénomène du monde réel.

Aménagement du territoire

Dans le domaine de recherche sur l'aménagement du territoire, (**?**) a proposé un modèle multi-échelles pour modéliser et simuler les transformations du paysage. Ce modèle est plutôt une structure de logiciel qui couple les autres modèles existants. Le terme échelle consiste en trois dimensions (**?**) : spatiale, temporelle et d'analyse. La propriété spatiale d'échelle considère la zone géographique. La dimension temporelle d'échelle considère la période d'analyse et la fréquence d'enregistrement. La dimension analytique d'échelle se réfère aux règles (par exemple, le comportement des agents) et aux techniques indirectes (par exemple, des méthodes statistiques) qui représentent le changement. Cette approche est utilisée pour construire un modèle multi-échelle en réalisant deux étapes. Tout d'abord, Il faut, pour chaque échelle, concevoir un modèle et diviser le modèle en spatial, temporel et analytique. La deuxième étape introduit l'idée de modèle couplé qui définit les liens entre les échelles. Trois types de coupleurs sont nécessaires : (a) coupleurs spatiaux, qui définissent les relations spatiales entre les échelles (par exemple, des liens parent-fils) ; (b) coupleurs analytiques, qui définissent le flux de l'information top-down et bottom-up entre les modèles ; et (c) coupleurs temporels, qui mettent en place l'exécution combinée temporelle des modèles.

Représentation, modélisation de l'environnement et des écosystemes

Déterminer les zones d'intérêt biologique pour les protéger est l'un des principaux objectifs des écologistes. Ceci peut être réalisé soit en se concentrant sur les zones soumises à des perturbations importantes, en cherchant des points de concentration de la diversité, ou des zones de priorité. À cette fin, les écologistes ont souvent recours à des modèles spatiaux de la richesse des espèces pour un large éventail d'organismes, y compris les plantes, les invertébrés ou vertébrés. L'approche de la modélisation consiste à utiliser des facteurs environnementaux comme la topographie, le climat, ou tout autre variable, comme prédicteurs du nombre d'espèces rassemblées dans une région donnée. Cependant, les mesures sont quelquefois trop rare ou trop imprécises pour permettre la modélisation pour des espèces spécifiques. L'étude menée dans (**?**) propose une approche multi-échelle pour simplifier l'analyse de la richesse en espèces. Des indicateurs de la richesse en espèces, des statistiques sur des regroupements d'espèces plus simples à estimer à partir d'échantillons de terrain que sur

des espèces spécifiques sont introduits. Dans cette approche, les modèles prédictifs indépendants ne sont pas calculés sur une base d'espèces spécifiques, mais les espèces ont été regroupées en groupes d'espèces plus grands. Dans cette étude, les auteurs évaluent l'efficacité de *valeurs de substitution* de richesse dont le calcul est basé sur l'abondance relative des groupes d'espèces. Les résultats du travail sur des peuplements de chauve-souris dans une forêt tropicale, montre que la plupart des indicateurs de remplacement sont étroitement corrélés à la richesse des espèces observées.

Gestion des ressources renouvelables

Le système de l'irrigation du Département de la Drôme a fait l'objet d'une étude pour partager la ressource en eau (**?**). Cette étude amène à un travail de coordination autour de la construction de règles collectives de gestion visant à respecter le débit de rivière imposé par le SAGE (Schéma d'Aménagement et de Gestion des Eaux) Drôme. L'eau consommée est égale au besoin global estimé des cultures d'été irriguées dans la zone, et les restrictions issues des règles sont appliquées directement à cette consommation globale. Les consommateurs de l'eau sont de plusieurs types comme : les agriculteurs, les individuels, les associations d'irrigation, l'autorité de bassin...

Ce système s'organise à plusieurs niveaux. Chaque niveau possède des règles. Le niveau global dispose des règles collectives qui sont définies par le SAGE, les indicateurs de l'état de la rivière, et du niveau individuel où s'effectue la mise en œuvre de stratégies d'irrigation. Les autres niveaux impliquant des modes de gestion où de coordination collectifs peuvent être identifiés : les règles définies au niveau du bassin imposent des contraintes au niveau des réseaux d'irrigation collectifs ; pour respecter ces contraintes, les réseaux doivent eux-mêmes définir des règles collectives qui imposent des contraintes à respecter apar leurs adhérents ; nombre de ces adhérents possèdent plusieurs types d'accès à l'eau dans leurs exploitations et doivent adapter leur irrigation en fonction des contraintes issues des différents niveaux de gestion dont ils dépendent ; enfin, les périmètres d'isolation des cultures de semence, ou des ententes de voisinages par exemple induisent des niveaux de coordination de nature plus informelle.

Biologie

Dans certains des travaux sur l'évolution des cancers, (**?**), (**?**), les auteurs décrivent le cancer comme un système dynamique complexe, d'adaptation et d'auto-organisation. Dans l'avancement de modèles de calcul base-discrets de cancer vers des applications cliniques, on fait face au dilemme de savoir comment traiter un nombre de données biomédicales croissant sans cesse qui devraient être intégrées à terme sous une forme ou une autre. Zhang (**?**) propose un système multi-échelle et multi-résolution basé sur des SMA pour la modélisation de tumeurs. Alors que le "multi-échelle" fait référence à l'emploi de la notion de facteur de croissance épidermique pour considérer les groupements de cellules, la "multi-résolution" est obtenue grâce à des algorithmes qui permettent de

classer les cellules comme actives ou inactives en clusters spatialisés.

Physique

Pour modéliser des écoulements de fluide, Bertelle (**?**) a proposé une méthodologie multi-échelle de gestion de structures auto-organisées. Les écosystèmes aquatiques sont par nature des systèmes hiérarchiques ouverts, traversés par des flux énergétiques qui les structurent. Les écoulements fluides qui régissent les courants océaniques sur l'ensemble de la planète sont la résultante d'une multitude de formations de vortex sur une large gamme d'échelles. Pour suivre l'évolution de l'écoulement du fluide, les systèmes multi-échelles sont bien exploités pour observer et pour représenter les organisations. Les auteurs ont proposé une approche de modélisation qui, en cours de simulation, détecte dynamiquement des organisations qui sont des prémisses d'une échelle supérieure émergente. Ces organisations évoluent, soit en se renforçant, soit en se déstabilisant, et éventuellement se dissipent et disparaissent. Les structures de flux détectées sont réintroduites dans la simulation en lieu et place des particules qu'elles représentent. La gestion d'évolutions grâce aux interactions est faite à la fois avec les autres structures mais aussi avec les particules.

La construction des échelles est basée sur les structures cohérentes dans le fluide. Les structures cohérentes sont des clusters de particules voisines Cette approche utilise un arbre de recouvrement de poids minimal obtenu à partir d'une triangulation de Delaunay pour détecter les clusters de particules voisines possédant des propriétés similaires. Ce travail utilise le modèle d'éco-résolution (**?**) représenté par un automate (**?**) en définissant les entrées ou les perceptions et les sorties ou les actions.

Génie des procédés

Dans un autre travail dans le domaine biologique, Zhang a déclaré que l'amélioration des souches et l'optimisation de fermentation sont deux tâches fondamentales de fermentation industrielle. L'étude multi-échelle sur l'optimisation de la fermentation industrielle a été prouvée comme une méthode efficace (**?**). Les auteurs proposent une méthodologie multi-échelle pour l'optimisation de la fermentation à faisant intervenir différentes échelles dans le processus pour l'ingénierie du réacteur.

Par l'analyse des applications et des méthodes ci-dessus, on a pu établir qu'il existe des systèmes complexes naturels ou artificiels qui portent dans leur nature la caractéristique multi-échelle temporelle ou spatiale qui permet de les voir à différents niveaux. Le multi-niveau appelé également *échelonnage* ou *scaling*, est un outil mathématique puissant pour caractériser ces structures et en déduire des propriétés à différentes échelles, au lieu de considérer seulement des propriétés relatives à une échelle. Le système multi-échelle est évidemment exploité pour la conception, la modélisation, l'observation et la simulation de ces systèmes complexes. Nous somme convaincus avec *Jinghai Li* et *Mooson Kwaukque* (**?**) que la nature multi-échelle est si importante que les structures

ne peuvent pas être correctement quantifiées sans distinguer la différence entre les échelles et les interactions entre différentes échelles. Par ailleurs, Zhang (**?**) aborde dans son review que l'analyse multi-échelle est une sorte de méthodologie, qui fournit une série de principes pour l'étude de la structure des systèmes complexes. Ses principaux principes sont :

- La description : la description de l'apparition de structures sans prêter attention au mécanisme de formation des structures et aux relations entre les différentes échelles ;
- La corrélation : la formulation des phénomènes aux échelles supérieures grâce à l'analyse des interactions aux échelles inferieures ;
- La variation : révélant la relation entre les échelles en formulant la condition de stabilité de la structure.

D'après notre analyse, les objets des modèles multi-échelles sont d'abord des systèmes complexes munis d'organisations ou de structures multi-échelles, chaque modèle multi-échelle proposé doit avoir des capacités pour :

- *La construction des niveaux* : pour appliquer un modèle multi-échelle, le système construit des niveaux de représentation, le modèle est donc capable d'assigner aux éléments du système de construire des niveaux de représentation. Très souvent, le modèle fournit des règles ou des lois de détection de clusters ou de formes pour la formation du niveau supérieur.
- *L'observation de chaque niveau* : chaque niveau du système multi-échelle peut donner une vision du système.
- *L'insertion ou la suppression d'un élément sur n'importe quel niveau* : Les systèmes complexes sont souvent des systèmes ouverts, l'apparition d'un nouvel élément et la disparation d'un élément ont lieu à n'importe quel moment. En plus, les changements d'un niveau peuvent affecter un autre niveau. Cela influence l'insertion ou la suppression d'un élément sur n'importe quel niveau. Le modèle multi-échelle doit prévenir et agir sur ces cas.
- *L'action sur n'importe quel niveau* : l'interaction avec des utilisateurs ou d'autres systèmes doit être possible sur n'importe quel niveau du système.
- *L'interaction entre les niveaux* : les niveaux doivent être capables d'interagir afin d'échanger des informations, de mettre à jour des changements, d'assurer la propagation top-down et la communication bottom-up d'informations.

Les modèles ne possèdent évidemment pas toutes ces capacités, mais ils se révèlent tous vouloir rendre accessible les systèmes complexes ou faciliter leur supervision.

Dans ce travail, nous abordons les systèmes artificiels multi-niveaux en ne considérant que l'échelle spatiale (ou plutôt une densité). Nous ne traitons pas le multi-échelle temporel. Nous mettons en relation des résultats obtenus depuis plusieurs années par notre équipe en voulant visualiser, modéliser et simuler de tels systèmes. Nous envisageons de proposer un modèle système multi-agents capable de rendre accessible ces systèmes complexes et faciliter leur supervision.

Dans le chapitre suivant, nous allons aborder les systèmes multi-agents et voir comment ils sont utilisés pour modéliser et simuler ces systèmes multi-échelles.

Chapitre 2

Les Systèmes Multi-Agents

Les systèmes multi-agents sont aujourd'hui largement acceptés comme un outil très puissant et un paradigme à part entière de modélisation des systèmes complexes. Ils sont en effet très efficaces pour gérer la nature hétérogène des composantes d'un système, pour modéliser les interactions entre ces composantes et tenter de comprendre les phénomènes émergents en cours d'évolution. Ils sont aussi une méthodologie pour réduire la complexité et observer certain des aspects des systèmes complexes.
Le domaine des Systèmes Multi-Agents (SMA) est basé sur l'étude d'agents et la définition de leurs interactions (**?**). Dans cette section nous présentons les concepts d'agent et de systèmes mutli-agents ainsi que les notions principales du domaine. Nous décrivons ensuite quelques travaux introduisant des approches multi-agents génériques dans le cadre multi-échelle. Notre objectif est de déterminer les propriétés multi-agents fondamentales pour aborder le contexte multi-échelle. Nous réalisons une analyse comparative des traits caractéristiques mis en avant dans les différentes approches. Nous rapprochons enfin ces propriétés des critères pertinents issus de l'étude sur le multi-échelle. Nous concluons sur des préconisations pour un modèle multi-agent générique pour les systèmes complexes artificiels multi-niveaux.

2.1 Les agents

Durant les dernières années de recherche dans le domaine, le terme « agent » apparaît comme un terme générique qui fait référence à différents types d'entités (**?**) :

- entités biologiques : les agents associés sont appelés agents biologiques
- robots autonomes
- logiciels informatiques et leurs composants, qu'ils soient intégrés dans des systèmes d'exploitation ou des systèmes informatiques complexes.

Considéré comme un domaine jeune, la définition d'agent dans le SMA n'est pas encore unique. Il est donc nécessaire, pour avoir une bonne vision du concept d'agent, de collationner plusieurs définitions, nous présentons trois d'entre elles :

- Jacques Ferber (**?**) définit un agent comme étant une entité physique ou virtuelle évoluant dans un environnement dont il n'a qu'une représentation partielle et sur lequel il peut agir. Il est capable de communiquer avec d'autres agents et est doté d'un comportement autonome.
  Cette définition aborde la notion d'*autonomie*, principe essentiel de la problématique agent. L'autonomie est la faculté d'avoir ou non le contrôle de son comportement sans l'intervention d'autres agents. Cette définition introduit aussi une autre notion importante qui concerne la capacité d'un agent à communiquer avec les autres.
- Pour Yves Demazeau et Antonio Rocha-Costa (**?**), un agent est une entité réelle ou virtuelle dont le comportement est autonome, qui évolue dans un environnement qu'il est capable de percevoir et sur lequel il peut agir, et qui peut interagir avec d'autres agents. Cette définition aborde la notion d'*interaction* avec les autres agents ou avec l'environnement.
- Selon Mickael Wooldridge (**?**) un agent est un système informatique qui est capable d'agir de manière autonome et flexible dans un environnement.
  La notion de *flexibilité* recouvre :
  - la réactivité : un système réactif maintient un lien constant avec son environnement et répond aux changements qui y surviennent.
  - la pro-activité : un système proactif (aussi appelé téléonomique) génère et satisfait des buts. Son comportement n'est donc pas uniquement dirigé par des événements.
  - des capacités sociales : un système social est capable d'interagir ou coopérer avec d'autres systèmes.

Grâce à ces trois définitions, en faisant une analogie, les agents sont considérés comme les grains composants du système, chargés d'une partie de la tâche globale, et communiquant avec les autres... Selon leurs propriétés et leurs capacités les agents peuvent être classés dans une des trois grandes familles : *réactifs*, *cognitifs*, *hybrides*.

Les agents *réactifs* réagissent d'une manière réflexe aux stimuli ou événements qu'ils perçoivent. Ils ne possèdent pas de moyen de mémorisation et n'ont pas de représentation explicite de leur environnement : ils fonctionnent selon un modèle stimuli/réponses pour interagir avec l'environnement dans lequel ils sont insérés. En effet, dès qu'ils perçoivent un stimulus de leur environnement, ils répondent par une action programmée. Ils sont issus d'une métaphore cellulaire ou animale.

Les agents *cognitifs* sont issus d'une analogie sociale, où leur modèle est une métaphore du modèle humain. Les agents cognitifs se réunissent en petite société et agissent en fonction de leur état mental (croyance, désir, intention) et de leurs interactions avec d'autres agents ou avec l'environne-

ment. Ces agents possèdent une représentation explicite de leur environnement, des autres agents et d'eux mêmes. Ils sont aussi dotés de capacités de raisonnement et de planification, de négociation ainsi que de communication.

Mais, dans les faits, les compétences des deux types d'agents sont souvent utilisées ensemble ; c'est-à-dire, qu'on construit un nouveau type d'agent (dit *hybride*) doté de capacités de types réactif et cognitif à la fois. Les agents *hybrides* sont considérés comme une intégration des modèles extrêmes des deux familles précédentes. En effet, ils conjuguent la rapidité de réponse des agents réactifs ainsi que les capacités de raisonnement des agents cognitifs pour assurer une meilleure coopération entre les agents et pour prendre en compte la dynamique de l'environnement dans laquelle ils évoluent. Cette famille regroupe donc des agents dont le modèle est un compromis autonomie / coopération et efficacité / complexité. Le continuum entre cognitif et réactif est d'ailleurs aujourd'hui communément réalisé. Pour illustrer cette famille, citons l'agent ASTRO (**?**) qui a été développé pour une utilisation dans les SMA soumis à des contraintes de type temps-réel.

2.2 Les systèmes multi-agents

2.2.1 Définition

Nous avons vu précédemment que les agents agissent dans des environnements qui peuvent être réels ou virtuels. Ces environnements peuvent permettre à des agents de se rencontrer et de générer des signaux de communication ou encore des interactions. Selon Ferber (**?**), un système multi-agents est un système composé des éléments suivants :

- un *environnement* c'est à dire un espace disposant généralement d'une métrique,
- un ensemble d'*agents* qui sont les entités actives du système,
- un ensemble d'*objets* situés dans l'espace, ils sont passifs, non dotés de comportement et cohabitent avec les agents,
- un ensemble de *relations* qui associent les objets entre eux,
- un ensemble d'*opérations* permettant aux agents de réaliser des *actions* sur l'environnement, c'est à dire de percevoir, de détruire, de créer, de transformer et de manipuler les objets,
- un ensemble d'*opérateurs* chargés d'opérationaliser l'application de ces opérations et de leur effet sur l'environnement (les lois de l'univers multi-agent).

Les idées dirigeant la conception d'un système multi-agents doivent être indépendantes des domaines d'application. Concevoir un système multi-agent revient à concevoir les entités autonomes pro-actives qui le composent. Les phénomènes ou les comportements sont associés au niveau individuel des agents. Chacun est spécialisé et agit de façon autonome. De ces actions individuelles, émerge la solution ou le comportement général du système, soit par une interaction grâce à la modification par les agents de l'environnement où ils s'insèrent, soit par une communication directe

entre agents par le biais d'un langage symbolique (**?**).

Les SMA peuvent avoir différentes caractéristiques auxquelles sont associées des qualificatifs :

- Un SMA *ouvert* est un système dans lequel des agents peuvent être introduits ou disparaitre. A l'inverse le qualificatif *fermé* signifie que les agents ne peuvent apparaitre ou être retirés.
- Un SMA *homogène* est composé d'agents homogènes. Des agents homogènes sont identiques du point de vue de leur modèle et de leur architecture. Un SMA est dit *Hétérogène* sinon.

Depuis leur création, les SMA font l'objet d'un véritable engouement. En effet, l'importance de l'approche SMA s'explique essentiellement par :

- le fait que les systèmes d'informations soient de plus en plus distribués, ouverts, à grande échelle et hétérogènes. Cela démultiplie tant le nombre d'interconnexions et leur complexité que l'ensemble dépasse la compréhension globale que peut en avoir un être humain.
- le fait que cette approche soit utile pour expérimenter des réflexions sociologiques et psychologiques pour ce qui concerne les relations entre les personnes.

L'approche multi-agents a une philosophie centrée sur une représentation directe des individus et des comportements. Elle permet de représenter deux niveaux : l'individu et le système. L'évolution du système au niveau supérieur doit émerger des interactions entre les individus. Cette approche est particulièrement bien adaptée à la modélisation des systèmes complexes dont le fonctionnement global émerge des interactions entre les individus.

2.2.2 L'émergence

Comme nous l'avons constaté précédemment, un SMA est un ensemble d'agents situés dans un même environnement et qui interagissent. Chaque agent agit selon des comportements programmés. Mais lorsqu'on observe une grande société d'agents et ses interactions, on constate que cette société, grâce aux interactions, produit des effets, au-delà des contributions individuelles, qui résultent des interactions entre les agents. Cette compétence de la société nous conduit à la notion d'« *émergence* » dans les systèmes multi-agents.

Pour définir l'émergence, dans (**?**) les auteurs proposent les caractéristiques des systèmes impliquant un sujet et son environnement où l'émergence est portée par les interactions entre ce sujet et son environnement :

- la première caractéristique d'un système multi-agent est qu'aucun agent ne contrôle complètement la dynamique de la population. La perception des agents est limitée et ils ignorent certaines particularités du système global. Il y a donc un extérieur relatif à chaque agent : un environnement.
- la deuxième caractéristique est que, par définition, les agents agissent et donc modifient cet environnement. Mais, les agents ne peuvent percevoir et agir que localement dans cet environnement. Autrement dit, chaque agent interprète l'environnement suivant ses moyens limités

(d'après les distinctions qu'il peut faire).

– la troisième caractéristique est que l'environnement de chaque agent peut contenir d'autres agents. Les agents partagent alors un environnement commun. Chacun considère les autres agents comme des éléments de l'environnement. Les interprétations de l'environnement par les divers agents peuvent être différentes.

Ainsi, la dynamique du système repose sur une boucle entre l'interprétation par les agents de leur environnement local, l'action des agents sur cet environnement, la nouvelle interprétation de l'environnement modifié, les nouvelles actions, etc. Quand une telle dynamique se stabilise on peut parler d'*émergence* d'une structure ou d'une fonctionnalité globale. Si le phénomène global est observable par les agents, il s'agit « d'émergence au sens fort », par contre si le phénomène global est observable seulement par un observateur extérieur, il s'agit « d'émergence au sens faible ».

L'émergence traite donc de l'apparition dynamique non programmée et irréversible de phénomènes dans un système (éventuellement multi-agents) confirmant ainsi la maxime d'Aristote « le tout est plus que la somme des parties ».

2.2.3 La conception de SMA : décomposition Voyelle (A E I O)

Les méthodes de conception de systèmes ont pour objectif de définir un ensemble d'étapes et un ensemble de règles nécessaires pour rendre le processus de construction des systèmes plus compréhensible. Le cycle de vie d'un système est constitué de deux grandes phases de développement : l'analyse et la conception.

La phase d'analyse concerne l'élaboration d'une solution détaillée mais indépendante des moyens de réalisation. Elle comprend la description de tous les processus composant le fonctionnement du système, la définition des informations utilisées, et la spécification des tâches à effectuer. La phase de conception a pour but de choisir, dans le domaine opérationnel, les modèles pour la réalisation. Elle définit les spécifications opérationnelles pour le fonctionnement du système et les moyens choisis pour la solution retenue.

Il existe de nombreuses méthodes de conception pour les systèmes multi-agents (**?**), dont la plupart sont centrées sur l'analyse des tâches des agents, comme les méthodes Gaia (**?**) et MaSE (**?**), ou sur l'organisation comme la méthode AGR (**?**).

Dans ce travail, nous adopterons la décomposition AEIO (*Voyelle* (**?**)) qui privilégie une description explicite des interactions et de l'environnement, ce qui est plus adapté pour un problème comme le nôtre.

D'un point de vue descriptif (**?**), il y a trois principes importants pour la décomposition *Voyelle* :

Principe I : Un système multi-agents se compose d'un ensemble d'agents (A), de l'environnement (E) où ils habitent, d'un ensemble d'interactions possibles (I), et d'une organisation (O) dans laquelle ils évoluent
SMA = Agent + Environnement + Interaction + Organisation

Principe II : L'intelligence collective est associée à une fonction résultant de l'émergence issue de la succession des interactions déroulées entre des agents et leur environnement. Le deuxième principe montre que la fonction collective du système multi-agents est égale à la somme des fonctionnalités (individuelle) de chaque agent additionnée à l'intelligence collective engendrée par l'émergence.
Fonction (SMA) = Σ fonction (Agent) + émergence

Principe III : Dans le principe de la Récursivité, un SMA est vu comme un agent, et chaque agent du SMA global, à tour de rôle, peut être vu comme un SMA (**?**). En effet, l'agent existe seulement à travers ses composants les plus élémentaires.
Récursion : A = A élémentaire | SMA [1]

Nous utiliserons cette approche des systèmes multi-agents qui est fondée sur quatre axes communément admis et qui sont développés ci-après.

2.2.3.1 Les agents (A)

L'axe A de l'analyse AEIO permet donc de regrouper les éléments nécessaires à la construction des agents. On définit ici les modèles des agents qui cohabitent dans le système. Il faut définir les capacités et les compétences attribuées à chaque agent. Le concept d'agent et les modèles d'agent ont été présentés en section 2.1.

2.2.3.2 L'environnement (E)

Dans cet axe, on définit l'environnement où les agents agissent et évoluent. L'environnement dépend toujours du domaine d'application du problème mais sont généralement spatialisés, c'est-à-dire dotés d'une métrique. Un environnement peut être (**?**) :

– *accessible* si un agent peut, à l'aide des primitives de perception, déterminer l'état de l'environnement et ainsi procéder, par exemple, à une action sur celui-ci. Si l'environnement est

1. Ce principe de récursion est au coeur de notre travail, même si la formulation intoduite ici par les auteurs peut être discutée. Nous approfondirons l'étude de cette notion dans le chapitre 3

inaccessible alors il faut que l'agent soit doté de moyens de mémorisation afin d'enregistrer les modifications qui sont intervenues.
- *Déterministe*, ou *non-déterministe*, selon que l'état futur de l'environnement est (ou non) fixé seulement par son état courant et les actions de l'agent.
- *Episodique* si le prochain état de l'environnement ne dépend pas des actions réalisées par les agents.
- *Statique* si l'état de l'environnement est stable (ne change pas) pendant que l'agent réfléchit. Dans le cas contraire, il sera qualifié de *dynamique*.
- *Discret* si le nombre des actions réalisables et des états de l'environnement est fini.
- *Avec ou sans adversaire relationnel* selon que les agents plongés dans cet environnement soient ou non en compétition.

Cette partie de l'analyse consiste à déterminer les éléments nécessaires à la réalisation des interactions extérieures au système comme par exemple la perception de cet environnement, les actions que l'on peut y réaliser.

2.2.3.3 Les interactions (I)

Dans cette axe, on définit le mode de communication entre deux ou plusieurs agents à travers un ensemble d'actions réciproques. La conséquence de telles actions exerce en retour une influence sur le comportement futur des agents. On distingue deux types de communication (**?**) :
- *indirecte* : par le biais de l'environnement, où les agents effectuent les émissions d'information sous forme d'actions et les réceptions d'information sous forme de perception. Ce mécanisme utilise l'environnement comme véhicule de transmission et peut être employé par des agents réactifs ou hybrides.
- *directe* : l'émission et la réception d'informations sont faites directement par les agents cognitifs. La conversation est établie entre des agents qui prennent tour à tour le rôle d'émetteur et de récepteur.

Cet axe de l'analyse AEIO permet de définir les éléments pour la structuration des interactions externes entre agents (par exemple, langage d'interaction, protocole de communication).

2.2.3.4 L'organisation (O)

Le concept d'organisation est assez vaste. On cite ici la définition de Malone (**?**) : une organisation peut être définie comme une structure de coordination (ou de prise de décision) et de communication comprenant un ensemble d'acteurs afin de réaliser un but commun. Dans le cadre de l'analyse AEIO, une organisation est soit *statique* soit *dynamique*. Dans le cas où la structure est conçue par le concepteur du système et qu'elle n'évolue pas, l'organisation est dite statique ; mais

si le système est ouvert, autrement dit, si il reçoit et élimine des agents, le type d'organisation doit être dynamique pour permettre la réorganisation de la société.

L'axe O de la méthode comprend les éléments permettant d'affecter les agents à des ensembles (des structures organisationnelles) par la détermination des rôles des agents, des dépendances entre les agents ou de leur statut dans la société.

La Figure 2.1 illustre la vision d'un SMA décomposée selon l'approche *Voyelle*.

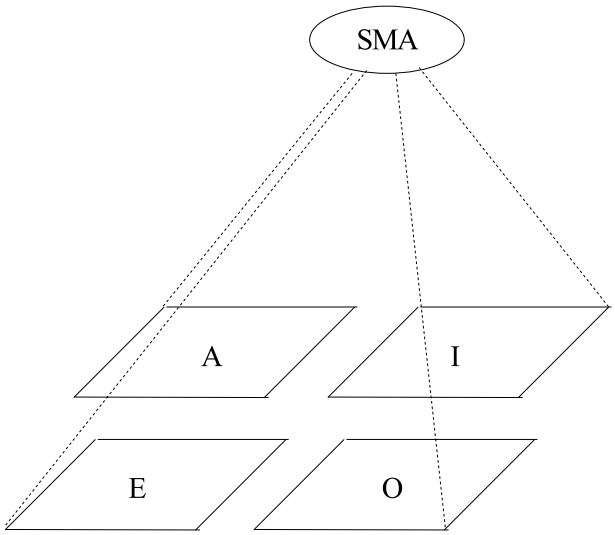


FIGURE 2.1 – Décomposition *Voyelle*

2.3 Application des SMA aux systèmes multi-échelles

L'approche SMA a déjà été étudiée, développée et utilisée pour la modélisation, la simulation, l'observation et la mise en œuvre d'un grand nombre de systèmes complexes. Certains des travaux évoqués dans la section précédente, qui portent sur des problèmes dont la nature présente des caractéristiques multi-échelles (multi-niveaux), utilisent des SMA.

Dans cette section, nous allons aborder des **approches ou architectures SMA génériques** existantes adaptées à ce contexte, en analysant des points remarquables de chaque approche. Nous tentons de décrire les axes : A (Agent), E (Environnement), I (Interaction) et O (Organisation) de ces systèmes. Mais pour mieux comprendre les modèles, nous ajoutons quelques éléments sur le contrôle de la décision, ou la dynamique. Ces descriptions nous permettront de juger de l'adaptation de ces approches aux capacités exigées pour le multi-échelle, que nous avons listées à la fin de la section 1.

Dans le contexte de recherche d'un modèle pour la modélisation de systèmes complexes multi-

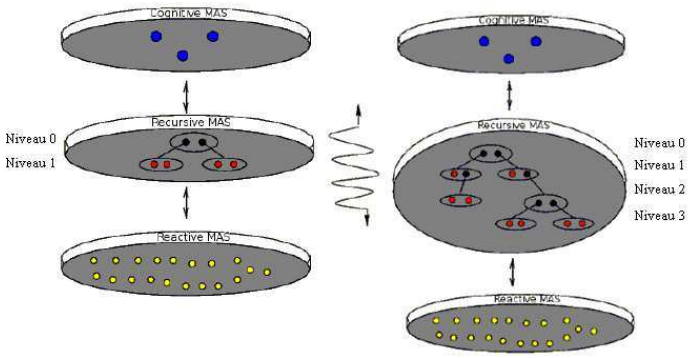


FIGURE 2.2 – La structure du modèle MMASS de Fernandes (**?**)

échelles à l'aide de l'approche Multi-Agent, Fernandes (**?**) a proposé un modèle appelé **MMASS** ("Multi-MAS System"). Ce modèle est composé de trois SMA homogènes placés chacun en trois niveaux différents du système : un MAS cognitif (SMAC) au niveau supérieur, un SMA réactif (SMAR) au niveau le plus bas et un SMA intermédiaire ou SMA récursif au niveau médian qui est chargé de faire la liaison entre les deux sociétés SMAC et SMAR (voir figure 2.2). Les couches cognitives et réactives emploient des méthodes différentes de résolution dirigées par les buts et par les données. Le cœur du modèle est la couche intermédiaire qui est responsable de fusion de ces deux types de résolutions. Pour cette couche, l'auteur a introduit un modèle d'agent appelé MORISMA. On ne se consacre, ici, qu'au modèle MORISMA qui possède une structure multi-échelle.

Les *Agent*s dans MORISMA sont à la fois *hybrides* et *récursifs*. Du côté hybride, les agents ont les capacités suffisantes pour interagir avec les deux couches inférieure et supérieure. Du côté récursif, les agents de MORISMA se présentent comme des agents *élémentaires* ou des agents *composés*. Les agents *élémentaires* se situent aux feuilles de l'architecture et gèrent un groupe d'agents réactifs à la couche inférieure. Les agents *composés* ont subi une décomposition et créent une nouvelle société des agents *élémentaires*. Ils se situent au-dessus des agents élémentaires dans l'architecture.

Concernant l'*Environnement* de MORISMA, les agents récursifs évoluent dans un environnement de *n couches* selon la définition du concepteur de système. En termes d'*Interaction*, l'auteur définit les critères de communication et de contrôle. Il y a deux types de communication : une communication interne (utilisée dans chaque couche) et une communication entre les couches. Dans SMAC, les agents communiquent en échangeant les messages. Dans SMAR, les agents s'échangent des signaux. Les agents de MORISMA utilisent des signaux et des messages pour communiquer entre eux et avec les couches cognitives et réactives. Il y a deux types de communication entre les couches : l'ordre pour le sens descendant et l'information pour le sens ascendant. Le contrôle du système est *décentralisé* pour la couche intermédiaire. Pour l'*Organisation*, le système présente une *architecture récursive*. Dans un niveau externe, chaque SMA communique directement avec un seul SMA de manière hiérarchique. Dans un niveau interne, chaque SMA possède une organisation

propre qui est définie par le concepteur du système.

SWARM (**?**) est une plateforme logicielle multi-agents pour la simulation de systèmes adaptatifs complexes. Dans SWARM, l'unité de base de la simulation est appelée un*swarm*. Une définition récursive peut être donnée pour la notion de *swarm* :

- Un *swarm* est une collection d'agents possédant un agenda des activités en cours pour ces *agents*.
- Mais un agent peut également être lui-même un *swarm*, une collection d'objets et un agenda d'actions.

Le comportement de l'agent est défini par les phénomènes émergeants des agents constituant son *swarm*. Les *swarm*s peuvent être créés et détruits en cours de simulation. SWARM peut être utilisé pour modéliser des systèmes où *plusieurs niveaux* de description dynamique émergent. Les structures *swarm* sont considérées comme des organisations récursives ou un swarm peut être formé par plusieurs agents qui sont aussi des *swarms*. Mais les simulations qui sont développées dans la plate-forme SWARM consistent à diviser le problème de l'organisation globale en un ensemble de sous-problèmes spécifiques. Une répartition des agents à exécuter est aussi réalisée conformément à la stratégie d'exécution qu'ils requièrent. Dans ce cas, *l'organisation est hiérarchique*.

AGR (**?**) est un méta-modèle pour décrire les organisations d'agents utilisant les concepts de base : *agent*, *groupe* et *rôle*. Une extension à AGR, AGRE a été développée qui prend en compte l'environnement (**?**). Le modèle n'impose aucune restriction sur l'architecture interne des agents. Un *agent* est spécifié comme une *entité* communicante qui joue un rôle actif au sein de groupes. Les *groupes* sont définis comme des agrégations d'agents en *organisations concrètes*. Chaque agent fait partie d'un ou plusieurs groupes. Le *rôle* est une représentation abstraite d'une fonction, d'un service ou de l'identification d'un agent au sein d'un groupe. Chaque agent peut manipuler de multiples rôles, et chaque rôle traité par un agent est local à un groupe. AGR permet de décrire des systèmes multi-agents avec différentes formes d'organisations telles que les orgnisations de marchés ou des organisations hiérarchiques. AGR pourrait donc être utile pour la conception de SMA multi-niveaux. Motivés par le besoin de fournir une plate-forme générique d'agents, évolutive, personnalisable, la plate forme MADKIT est construite pour la mise en œuvre du modèle AGR. Cette plate-forme fournit un ensemble d'outils et des bibliothèques. Le cœur de MADKIT est basé sur son micro-noyau qui fournit tous les services de base nécessaires aux agents : la gestion des données de l'organisation (groupe, rôle), la communication, l'organisation (concurrente ou synchrone).

ANEMONA (**?**) est une méthodologie multi-agent pour les *systèmes holoniques manufacturés* (Holonic Manufacturing Systems - HMS) où l'analyse et la conception sont basées sur la notion

holons (**?**) [1].

Dans ANEMONA, la notion d'*agent abstrait* (**?**) a été présentée comme un artefact de modélisation des entités autonomes avec des *structures récursives* [1]. ANEMONA définit un processus mixte de développement top-down et bottom-up, et fournit des guides spécifiques pour les HMS afin d'aider les concepteurs dans l'identification et la mise en œuvre des holons. Les niveaux de composition sont variables, mais définis par le concepteur. Les unites d'interaction sont des *messages* ou des *événements*.

GEAMAS (Generic Architecture for Multi-Agent Simulations) (**?**) est une architecture générique d'agents utilisée pour étudier l'émergence de comportement. C'est aussi une plateforme multi-agent qui a pour but de développer des applications de simulation. Cette approche permet de comprendre comment le comportement non-déterministe peut émerger depuis les interactions entre agents, (*concept de criticalité auto-organisée utilisée pour expliquer les phénomènes naturels*). Dans le contexte de l'*Intelligence Artificielle Distribuée*, GEAMAS est une architecture logicielle utilisant un modèle d'agent. Cette architecture est composée de trois niveaux abstraits destinés à réduire la complexité. Chaque niveau successif représente un niveau d'abstraction supérieur au précédent.

- Le premier niveau, appelé le système multi agents-SMA, gère ses agents dans le système afin d'être conforme aux spécifications globales et est inscrit dans un modèle appelé modèle de société. Ce niveau gère aussi les contraintes du système et fournit une interface de communication avec le monde extérieur ;
- Les agents au deuxième niveau sont des agents cognitifs (ou agent intermédiaire). Chaque agent est construit selon le modèle GEAMAS. Ces agents sont modifiables au cours de simulation. On peut dire qu'ils s'adaptent aux changements du système. Ce niveau doit masquer la complexité des interactions et la dynamique au niveau trois. Chaque agent du niveau joue le rôle d'une société de micro agents et il est en même temps vu comme un agent du premier niveau.
- Les Agents du troisième niveau (niveau le plus bas) sont des *agents réactifs* pour décrire les comportements aux granularités plus fines. Ils sont appelés *micro agents* ou *cellules*. Ils permettent aux agents de deuxième niveau de faire abstraction de la dynamique complexe de leurs interactions. La complexité est ensuite distribuée dans les agents réactifs, et leurs interactions locales au moment de l'exécution provoquent l'émergence de comportements globaux.

Ce modèle exprime essentiellement le rôle, l'interface et l'ensemble de l'organisation du système. Afin de faciliter la conception du système, une architecture d'agent récursive basée sur modèle

1. Nous décrirons plus en détail cette notion dans la section suivante

unique d'agent sur chacun des niveaux est introduite. Le contrôle du système est distribué au sein des agents. En ce qui concerne l'interaction, elles sont de deux types :

- L'interaction entre les agents de même niveau : les micro-agents communiquent en utilisant des signaux. L'interaction entre les agents cognitifs résulte d'une simulation, et l'on peut dire qu'elle émerge des interactions entre les agents du niveau inférieur.
- L'interaction entre les agents de différents niveaux : Ce sont des messages asynchrones. Pour tranférer à la montée et à la descente des messages entre les niveaux, GEAMAS utilise deux mécanismes la *Décomposition* et la *Recomposition*. La *Recomposition* est responsable du transfert des informations d'un niveau inferieur à un niveau supérieur. La *Décomposition* est responsable du transfert des informations d'un agent au niveau supérieur vers ses agents composants du niveau inférieur.

En ce qui concerne l'axe Organisation, les agents de GEAMAS sont organisés en une hiérarchie à trois niveaux d'abstraction, chaque niveau représente un plus haut niveau d'abstraction que son prédécesseur.

MAGIQUE (**?**) est à la base un modèle d'organisation d'agents qui propose une organisation hiérarchique. Cette structure permet de proposer un mécanisme de délégation de compétences entre agents, facilitant ainsi le développement. MAGIQUE existe sous la forme d'un framework générique pour le développement de SMA. Le concepteur de SMA peut s'appuyer sur les fonctionnalités offertes par MAGIQUE telles que les communications entre agents, la distribution de l'application, l'évolution dynamique, etc., charge à lui de développer les compétences applicatives dont il a besoin.

Dans MAGIQUE, un agent est une entité possédant un certain nombre de compétences. Ces compétences permettent à un agent de tenir un rôle dans une application multi-agents. Les compétences d'un agent peuvent évoluer dynamiquement (par échanges entre agents) au cours de l'existence de celui-ci, ce qui implique que les rôles qu'il peut jouer (et donc son statut) peuvent également évoluer au sein du SMA. Un agent est construit dynamiquement à partir d'un agent élémentaire "vide", par enrichissement/acquisition de ses compétences.

Du point de vue de l'implémentation, une compétence peut être perçue comme un composant logiciel regroupant un ensemble cohérent de fonctionnalités. Les compétences peuvent donc être développées indépendamment de tout agent et donc réutilisées dans différents contextes. Une fois ces compétences créées, la construction d'un agent MAGIQUE se fait par un simple mécanisme d'enrichissement de l'agent à construire avec ces compétences. L'objectif est de rendre plus facile l'ajout de nouvelles compétences (et donc de fonctionnalités) dans une application SMA.

Les agents sont organisés selon une *structure hiérarchique*. Les agents feuille sont appelés *"spécialistes"* et les autres *"superviseurs"*, ceux-ci doivent être capables de gérer leur équipe d'agents (la sous-hiérarchie) dont ils sont la racine. Une hiérarchie représente en fait le support par défaut

du réseau des communications entre les agents. Un lien hiérarchique représente donc l'existence d'un lien de communication entre ces agents, et lorsque deux agents d'une même structure hiérarchique communiquent, le chemin emprunté par les messages qu'ils s'échangent suit les relations d'accointances et colle, par défaut, à la hiérarchie. MAGIQUE offre aussi des liens d'accointances qui permettent des communications directes (i.e. en dehors de la hiérarchie) entre deux agents. Les agents doivent donc pouvoir communiquer avec leur supérieur hiérarchique dans les deux sens. Chaque agent possède une file d'attente des messages reçus. On peut aussi classer les interactions dans MAGIQUE en deux types : les *interactions verticales* sont des interactions entre les niveaux et les *interactions horizontales* sont des interactions entre les agents de même niveau. MAGIQUE possède une structure de contrôle hiérarchique. Les agents spécialistes possèdent leur propre contrôle, ils se réunissent en groupes et avertissent leurs superviseurs. Si nécessaire, les superviseurs peuvent également les regrouper et les informer. MAGIQUE propose donc en fait un contrôle mixte ou hybride. Les auteurs de MAGIQUE ont introduit plus récemment l'approche IODA visant à traiter moins l'"'emboitement"' des structures que des comportements (**?**).

Le modèle organisationnel holonique **SOHTCO** (Système Orienté Holon pour l'aide au Travail COopératif) (**?**) a été adapté et appliqué à la spécification d'une organisation multi-agent assistant les acteurs d'un système administratif complexe ayant un fonctionnement de type workflow. SOHTCO considère les utilisateurs humains (acteurs) et les documents que le système doit gérer comme son environnement.

L'organisation du modèle est composée de trois niveaux :

- Le premier niveau comprend les agents responsables de l'interaction avec l'utilisateur et la gestion des documents. Il contient également des agents responsables de la communication entre les différents postes de travail ;
- Le deuxième niveau est constitué par les agents responsables (Workstation responsables) des actes accomplis par les "agents d'exécution". Chacun de ces agents sont liés à un utilisateur humain ou à une procédure et contiennent les règles de la procédure ;
- Le troisième niveau est constitué par les agents chargés des processus.

L'architecture d'agent est organisée comme suit : chaque agent responsable possède sous son contrôle quatre agents d'exécution qui sont : l'agent chargé de l'interface utilisateur, l'agent chargé de l'envoi des données, l'agent chargé de la réception des données et l'agent chargé du management des données. Les quatre agents d'exécution et leurs responsables sont également appelés sous-SOHTCO. Selon la règle de *regroupement des systèmes holoniques*, chaque sous-SOHTCO peut être vu comme un seul agent. Dans SOHTCO, chaque agent détient donc des connaissances sur lui-même, ses subordonnés, son agent responsable, les agents du voisinage holonique, l'acteur auquel il est attaché et le poste de travail. En outre, chaque agent possède cinq fonctions principales :

l'initialisation, la planification, l'interaction, l'action et l'observation.

Les agents holoniques doivent obéir à des règles fixées par l'organisation et respecter les règles générales. Ils sont également libres de choisir librement leur stratégie en fonction de leur environnement. De cette façon, dans SOHTCO, chaque holon conserve un certain degré d'autonomie, mais il doit respecter certaines règles pour garantir la cohérence des activités du système. *Le contrôle reste donc centralisé.*

Pour la communication, le système SOHTCO utilise *le langage KQML*.

2.4 Des propriétés multi-agents pour un système complexe artificiel multi-échelle

Les approches examinées dans la section précédente révèlent quelques caractéristiques des modèles qui semblent discriminantes pour leur utilisation pour des systèmes multi-échelle. Il s'agit de critères comme l'architecture des agents, la dynamicité multi-niveau, les types d'interaction, la nature du contrôle, ou la structure du SMA. Nous détaillons ces caractéristiques avant d'analyser les systèmes décrits selon ces critères afin de souligner les avantages de chaque approche et de préconiser ceux à retenir pour l'élaboration d'un modèle. Ces critères sont issus du croisement de notre propre réflexion avec ceux proposés par K. Fernandes (**?**), S Rodriguez (**?**) et S. Picault (**?**) et même si ce dernier se préoccupe de simulation multi-échelle et pas de modélisation de systèmes artificiels multi-échelle.

2.4.1 Définition des critères

2.4.1.1 Architectures des agents

La notion de multi-échelle peut conduire à considérer un système à différents niveaux d'abstraction avec des descriptions différentes. Cela peut conduire à l'utilisation d'agents de modèles différents sur les différents niveaux ou même sur un même niveau. Nous devons donc compter avec l'homogénéïté ou l'hétérogénéïté des élements du SMA proposé. Ce critère se rapproche du critère d'*uniformité* de Picault et de celui d'*architecture* de Rodriguez.

2.4.1.2 Dynamicité Multi-Niveau

La notion de multi-échelle introduit celle de construction et d'observation des différents niveaux. Il est primordial de déterminer le mode de construction de ces niveaux. Sont-ils construits dynamiquement ? Peuvent-ils être modifiés dynamiquement en cours de fonctionnement ? Ce critère correspond au critère *environnement* de Fernandes, *environnement* et *mécanisme de création* de

Rodriguez et partiellement à celui de *dynamicité* de Picault.

2.4.1.3 Type d'Interaction

Il est indispensable de disposer d'une interaction entre niveaux. Il est nécessaire d'apprécier la nature de cette interaction. Il faut aussi déterminer le type d'accès que l'on aura aux différents composants des niveaux : accès direct ou accès indirect. Il faut ensuite déterminer la portée de cette interaction : accès à tous les niveaux du système ou visibilité limitée à certains niveaux. On pourra ajouter à ce niveau une indication sur le *type d'interaction avec l'environnement* : vertical (tous les niveaux peuvent interagir avec l'environnement), horizontal (seul le niveau le plus bas interagit avec l'environnement), disjoint (chaque niveau à un environnement propre) comme défini par Rodriguez.

2.4.1.4 Nature du contrôle

Le mode d'accès aux différents composants sera aussi fortement lié à la nature du contrôle du SMA multi-échelle. On distinguera entre une gestion centralisée ou décentralisée du contrôle permettant soit un accès point à point, soit un accès collectif global. Ce critère se rapproche de la notion de *mode de prise de décision* de Rodriguez.

2.4.1.5 Structure du SMA

Construire un SMA multi-échelle peut amener à construire une structure composée de systèmes multi-agents physiquement indépendants mais capables d'interagir, ou d'un seul et même SMA intégrant des capacités structurelles ou organisationnelles lui donnant des capacités de représentations abstraites. La structure arborescente est systématiquement adoptée pour les SMA multi-niveau. Le critère de structure indique la façon d'opérationaliser la notion d'*emboitement* décrite par Picault (**?**) ou de structure hiérarchique évoquée par Rodriguez (**?**). Il reprend aussi le critère d'*organisation* de Fernandes. Nous le complétons par l'appréciation de l'appartenance possible à plusieurs niveaux ou organisations par les éléments de la structure (*"appartenance multiple"* de Picault).

Réaliser l'*emboitement* de SMA peut se faire en utilisant les propriétés *hiérarchiques*, *récursives*, *holoniques* qui conduisent à exploiter une structure arborescente. Il est nécessaire de se pencher sur ces définitions pour mieux comprendre notre positionnement à venir.

2.4.1.5.1 Des propriétés hiérarchiques

Le SMA est constitué d'un ensemble d'agents. Lorsque les agents sont en relation de type maître/esclave, on parle de relation hiérarchique.

En terme fonctionnel, les agents peuvent se voir affecter des tâches qu'ils divisent en sous tâches qu'ils distribuent ensuite parmi un ensemble d'agents. Un agent maître (le superviseur) est capable

de décider quel agent possède les capacités pour effectuer une tâche. Une organisation hiérarchique s'exprime sous la forme d'un arbre dont la racine est un agent et dont les fils sont, quand il y en a, également des hiérarchies d'agents. Il s'agit donc d'une organisation multi-niveau. Les agents d'un niveau de la hiérarchie prennent des décisions pour les agents qui se trouvent au dessous. En construisant une hiérarchie de tâches, on peut traiter une quantité considérable d'informations. Si la connaissance n'est pas complète pour prendre des décisions à un niveau, alors la prise de décisions doit remonter vers les noeuds parents. L'information est échangée seulement entre des niveaux voisins. On s'aperçoit alors que beaucoup de décisions sont prises au sommet de la hiérarchie. Dans une hiérarchie multi-niveau il faut donc prendre en compte l'allocation de ressources et la planification.

En fait, la hiérarchie n'est pas forcément fonctionnelle, elle peut recouvrir des liens d'autorité, de statut, ou de structure. Dans une organisation hiérarchique multi-niveau, ce sont les liens de dépendance (en terme d'autorité, de statut, ou de structure) qui permettent la construction de l'arbre.

2.4.1.5.2 Des propriétés récursives

La littérature spécialisée n'offre que peu de références qui parlent de la récursion dans les SMA.

Occello (**?**) a défini un modèle d'agent récursif. Pour lui, un SMA est constitué par un ensemble d'agents et un ensemble d'objets de l'environnement.

$$\text{Récursif E} ::= \text{ObjetElémentaire} \mid \{\text{Récursif E}\} \tag{2.1}$$

Dans son interprétation, un objet de l'environnement peut être un objet élémentaire ou un objet récursif. Les objets élémentaires sont des objets ayant une existence physique et les objets récursifs non élémentaires sont abstraits.

De façon équivalente, on définit la partie agent, l'ensemble A :

$$\text{Récursif A} ::= \text{AgentElémentaire} \mid \{\text{Récursif A}\} \tag{2.2}$$

Un agent peut être un agent élémentaire ou un agent récursif. Les agents élémentaires sont des agents ayant une existence physique. Les agents composés sont abstraits, ils existent au travers de leurs composants.

Le concept de récursivité utilisé dans le modèle suit le principe III de l'approche Voyelle, où la récursion est comprise comme étant une organisation arborescente construite dynamiquement où seulement les agents feuilles ont une existence réelle.

Autrement dit, chaque SMA est vu comme un agent, et chaque agent du SMA global, à tour de rôle, peut être vu comme un SMA (**?**). L'agent existe seulement à travers ses composants, représentés par des agents élémentaires dans le niveau le plus bas du SMA. Les niveaux supérieurs

sont formés par des nœuds représentant les agents composés qui définissent une forme abstraite de rassemblement des agents élémentaires.

On peut trouver une autre définition d'agents récursifs dans (**?**). Dans cette définition, l'agent récursif a été défini par la notion de A-Agent ("abstract récursive agent").

Un A-Agent est un système logiciel avec une entité unique, qui est situé dans un certain environnement, qui dans son ensemble, perçoit son environnement. A partir de ces perceptions, il détermine et exécute des actions de manière autonome et flexible - réactive et proactive. Ces actions permettent à l'A-Agent d'atteindre ses objectifs et de changer son environnement. D'un point de vue structurel, un A-agent peut être un agent (entité atomique), ou il peut être un système multi-agents (avec une entité unique), composé de A-agents qui ne sont pas nécessairement homogènes.

Un A-Agent est d'un plus haut niveau d'abstraction conceptuel qu'un agent. Un A-agent peut être vu comme un SMA, une organisation, une fédération ou une institution avec la valeur ajoutée qu'il peut aussi être une composition de tous ces modèles d'abstraction.

La structure récursive d'un A-agent peut être représentée en UML par la figure 2.3.

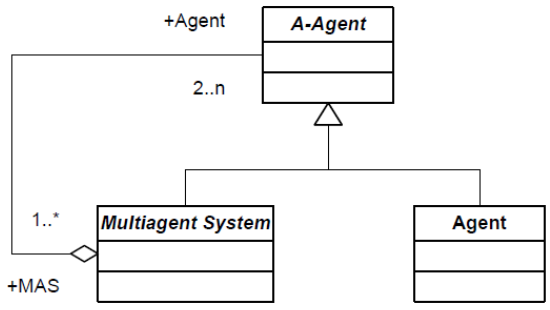


FIGURE 2.3 – La structure d'un A-Agent (**?**)

La définition récursive proposée par les auteurs d'un SMA composé de A-agents est la suivante.

$$LevelR(A) = \begin{cases} 0 & A = a,\ A\ est\ un\ agent\ a \\ max(LevelR(A_i)\} + 1 & A = \{A_1, A_2, ..., A_k\}, 1 \leq i \leq k\ est\ un\ ensemble\ d'agents\ abstraits \end{cases}$$

Pour les auteurs, cette définition exprime les points suivants :

– Un A-agent de niveau 0 est un agent.

– Un A-agent de niveau 1 est un SMA composé d'agent en interaction.

– Un A-agent de niveau $n > 1$ est un SMA composé de de A-agents de niveau de récursion $> n$.

2.4.1.5.3 Des structures holoniques

La notion de holon a été introduite par le philosophe Koestler en 1967. L'approche s'est particulièrement diffusée dans les HMS (Holonic Manufacturing Systems). Le terme "holon" naît de la combinaison des mots grec "holos", qui signifie "le tout" , et du suffixe "-on" qui désigne la notion de "partie" comme dans neutron, proton ou électron. Selon Koestler, un holon est une structure auto-similaire ou fractale qui est comme une partie d'un tout ou d'une organisation plus large, répondant strictement à trois conditions : être stable, avoir une capacité d'autonomie et être capable de coopérer :

– la stabilité signifie qu'un holon est capable de faire face et de réagir lorsqu'il est sujet à des sollicitations fortes ou à des perturbations importantes ;
– l'autonomie suggère qu'un holon est capable de s'autogérer lorsqu'il est soumis à des sollicitations, afin de réaliser ses objectifs personnels ;
– la capacité de coopérer signifie que les holons sont en mesure de coexister avec d'autres holons ou d'autres couches de holons, et sont capables de travailler sur des objectifs et dans des projets communs.

Les organismes ou les systèmes complexes comme les organismes biologiques ou les sociétés doivent être considérés comme un ensemble multi-échelle d'éléments ou de sous-systèmes intermédiaires qui sont stables et organisés selon une structure hiérarchique. Chaque holon connaît ses subordonnés et connait le holon duquel il dépend. Toutefois, la caractéristique importante dans cette *hiérarchie distribuée* est le fait que les sous-systèmes coopèrent sous forme de *hiérarchies* dynamiques. Le système se réorganise périodiquement pour atteindre un objectif global ou à la demande lorsque l'objectif global à accomplir change.

L'auteur a proposé non seulement les trois règles de base ci-dessus, mais également 65 règles décrivant les notions de dualité coopération-autonomie, de communication et d'architecture. Cela peut permettre de mettre en évidence une condition nécessaire de viabilité et de pérennité des systèmes sociaux. En annexe de son livre (**?**), Koestler a introduit des règles concernant la définition du système holonique qu'il appelle *système hiérarchique* ouvert ou encore *holarchie*. La définition récursive de holon permet de décrire naturellement des systèmes de nature hiérarchique.

Ce concept a été récemment adopté par la communauté de l'intelligence artificielle distribuée sous le nom de systèmes multi-agents holoniques par (**?**).

Nous pouvons évoquer la définition de Michael Schillo et Klaus Fischer (**?**).

Soit A_t l'ensemble des agents du SMA, $\mathcal{H}_t$ l'ensemble de tous les holons au temps t. $\mathcal{H}_t$ est défini récursivement :

– pour chaque $a \in A_t, h = (\{a\}, \{a\}, \oslash) \in \mathcal{H}_t$, c'est à dire chaque agent instancié constitue un

holon atomique, et

– $h = (Head, Subholons, C) \in \mathcal{H}$, où *Subholons* est un ensemble non-vide de holons qui constituent *h*, $Head \subseteq Subholons$ est l'ensemble non-vide des holons qui représentent le holon et qui sont responsables de la coordination des actions à l'intérieur du holon. Les relations entre holons à l'intérieur d'un holon sont représentées sous la forme d'engagements en les éléments (*Commitments*). $C \subseteq Commitments$ est le sous ensemble des relations qui spécifient la structure organisationnelle. Tous les *holons* $h' \in Subholons$ doivent respecter C au moment t où ils rejoignent un holon *h*.

Tout holon h' est autorisé à s'engager dans plusieurs holons différents dans le même temps, aussi longtemps que cela ne contredit pas l'ensemble des engagements des *superholons*.

Dans un holon, certains holons sont responsables du holon dont ils sont la tête (head), le reste du *subholon* est passif. *Un holon peut participer à plusieurs holons en même temps.*

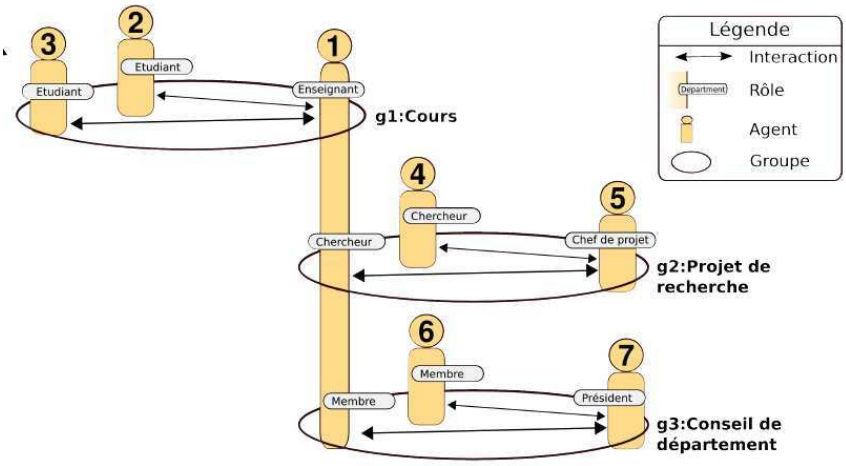


FIGURE 2.4 – Un holon peut particper à plusieurs holons en même temps (**?**)

La figure 2.4 illustre un système holonique. Un agent Enseignant-Chercheur joue simultanément plusieurs rôles dans des groupes différents. L'agent A_1 joue ainsi le rôle d'enseignant dans le groupe *g1*, le rôle de chercheur dans le groupe *g2* et le rôle de membre dans le groupe g3.

2.4.1.5.4 En résumé

Après avoir abordé ci-dessus les concepts hiérarchiques, holoniques et récursifs, nous pouvons nous pencher sur ce qui rapproche ou différencie ces notions, même si on pourrait comme (**?**) rassembler l'ensemble des systèmes multi-agents détaillés précédemment sous le nom de SMA holoniques.

Il est clair que la notion de hiérarchie en tant qu'organisation est l'expression la plus naturelle pour exprimer une arborescence mais elle ne réalise pas vraiment un emboitement et ne porte pas

réellement sur la structure du SMA mais sur les liens entre agents. Il apparait que la notion d'emboitement est mieux réalisée par une nature récursive.

Les holons proposent d'opérationaliser les emboitements et leur dynamique recursive en fixant certains choix architecturaux allant de la fédération à la fusion d'agents (**?**).

L'approche réalise sa propre lecture de la notion de récursion et fixe un certain nombre de contraintes en fonction notamment des contraintes de son domaine d'origine (HMS). L'option architecturale la plus souvent choisie pour implémenter des holons reste le groupe managé où un agent holonique possède des sub-holons dont un agent-tête décide de la coordination des actions à l'intérieur du holon.

D'un point de vue plus conceptuel, un agent holonique peut participer à plusieurs holons en même temps. Un système peut être récursif sans pour autant reprendre à son compte la multi-appartenance. De même, une interprétation possible de la récursion laisse envisager un processus reproductible à l'infini, avec similarité des modèles et un nombre de niveaux non fixé évoluant dynamiquement. Les systèmes holoniques ont plutôt vocation à voir leur niveau défini par le concepteur.

En fait, nous pouvons considérer qu'un système holonique est récursif. Les libertés d'interprétation de certains aspects de la notion de récursion dans les systèmes holoniques ainsi que les artefacts de mise en oeuvre nous poussent à penser qu'un système multi-agent récursif n'est pas forcément holonique.

2.4.2 Synthèse de l'analyse suivant les critères

Nous avons introduit dans la première partie de cette section les critères qui nous paraissent les plus discriminants pour évaluer les approches SMA pour le multi-niveau. Le tableau 2.1 résume la synthèse des résultats de l'analyse des approches retenues à travers les critères défendus jusqu'ici. Nous allons maintenant analyser ces approches selons ces critères pour déterminer les propriétés multi-agents les plus pertinentes à retenir en vue de satisfaire aux mieux les besoins pour les systèmes complexes multi-échelles (multi-niveaux) établis dans le chapitre 1.

Tableau 2.1 – Résumé des analyses des SMA retenus

Système	Types d'Agent	Dynamique multi-niveau	Communication / Interaction avec l'environnement	Type de contrôle	Structure du SMA
SWARM	Agents élémentaires, swarm	n couches, prédéfini	Message, vertical	Décentralisé	Hiérarchique
AGR	Rôles hétérogènes	n couches, statique	Message, vertical	Décentralisé	Hiérarchique
ANEMONA	Agents abstraits	n couches, dynamique	Message/événement, horizontal	Centralisé	Holonique
GEAMAS	Agent, groupe, société	3 couches, prédéfini	Message/signal, horizontal	Décentralisé	Récursive
MAGIQUE	Spécialistes, superviseurs	n couches, dynamique	Message, horizontal	Hybride	Hiérarchique
SOHTCO	Holons Sub-SOHTCO (4 types d'agents)	n couches, prédéfini	Message, horizontal	Centralisé	Holonique
MMASS	Agents récursifs	n couches, dynamique	Message/signal, horizontal	Centralisé	Récursive

Quelques commentaires s'imposent sur chacune des dimensions de l'analyse.

2.4.2.1 Architectures d'agent

D'une façon générale, parmi ces architectures, celles qui souhaitent représenter un phénomène ou un système à plusieurs niveaux d'abstraction de façon homogène (telles SWARM, ANEMONA, MAGIQUE, MMASS ou SOHTCO) utilisent des agents à l'architecture homogène dans ces niveaux. Il est à remarquer cependant que le phénomène réel lui-même constituant le niveau élémentaire peut être modélisé par des agents de nature différente. Seul GEAMAS propose un système d'agents de trois types différents localisés sur trois niveaux différents.

2.4.2.2 Dynamicité des Niveaux

Les architectures qui visent à construire les abstractions de façon dynamique s'adaptant à des problèmes différents proposent une construction dynamique de niveaux en nombre variable. Les systèmes ou approches retenus dans notre description sont souvent dans ce cas là. Les architectures holoniques, par construction (visée fonctionnelle), visent plus la réorganisation que la création dynamique de niveaux par rapport à un problème donné. GEAMAS ne propose que 3 niveaux et ne permet pas la réorganisation dynamique. Dans AGR, la structure organisationnelle est construite (dans les cas courant) par le concepteur.

2.4.2.3 Communication / Interaction avec l'Environnement

Les interactions sont possibles dans tous ces systèmes tout au long de la structure arborescente qu'ils construisent et très exceptionnellement de façon transversale. Le plus souvent les communications directes ne traversent pas plus d'un niveau de profondeur. MAGIQUE permet la constitution d'un treillis de communication en créant des accointances de communication entre des noeuds de différents niveaux. GEAMAS ne possède pas réellement d'interaction entre niveaux mais met en oeuvre une observation continue des modifications qui surviennent sur les niveaux (communication indirecte). L'interaction avec l'environnement se fait généralement à travers la couche de plus bas niveau. Seul SWARM permet une communication avec un même environnement sur chacun des niveaux.

2.4.2.4 Nature du contrôle

Les approches sont très variées sur ce critère : centralisées, décentralisées, hybrides. Un modèle hirerachique comme MAGIQUE autorise la commutation entre différents types de contrôles. Les modèles holoniques étudiés ne s'avèrent pas totalement décentralisés car le contrôle de l'interaction interne repose le plus souvent sur des agent-têtes (SOHTCO, ANEMONA).

2.4.2.5 Structure du SMA

L'emboitement est la solution qui représente de la façon la plus naturelle l'encapsulation des niveaux. L'approche récursive (holonique ou non) le réalise dans les meilleures conditions. Elle est la mieux profilée pour la réduction de complexité par une construction dynamique de niveaux adaptables en cours de fonctionnement. Toutes les approches décrites (hors MAGIQUE et AGR) développent plus ou moins complètement une approche récursive. Le modèle hiérarchique (MAGIQUE) est en effet moins souple quand le système veut ajouter un nouvel élément car il doit pour cela avertir tous les agents verticaux et horizontaux.

2.5 Conclusion

Les modèles analysés dans cette section apportent différentes contributions sur la réduction de la complexité du système, les interactions verticales et horizontales, le contrôle de la décision, la dynamique du modèle, la capacité d'interagir avec le système. Ils répondent de cette façon à certaines des exigences des systèmes multi-échelles établies dans le chapitre précédent :

- *Un SMA multi-niveau avec une capacité de construction dynamique d'un nombre variable de niveaux sera mieux adapté.* Utiliser une approche multi-échelle impose de savoir construire des niveaux dans le SMA. La réduction de complexité est obtenue par abstraction du système. La lecture de cette abstraction dépend des conditions initiales du problème traité et de la nature de l'observation appliquée. Le nombre de couche à construire n'est pas fixé à l'avance, il est variable. Il évolue aussi de façon dynamique selon les règles d'observation fixées.
- *Il est indispensable de disposer d'un SMA à structure adaptative constitué d'agents possédant la même structure.* L'insertion ou la suppression d'un agent doivent être autorisées à n'importe quel niveau. Dans notre contexte de système complexe artificiel, la réelle apparition ou disparition d'agent n'aura réellement lieu que sur le niveau physique élémentaire. Cependant cette perturbation entrainera une reconfiguration des couches abstraites. De même, il semble difficile dans ce contexte d'envisager de créer ou supprimer d'une même façon des agents hétérogènes.
- *Une architecture du SMA totalement décentralisée sera requise*. Agir sur n'importe quel constituant de façon indépendante doit être possible. Pour cela chaque agent sur n'importe lequel des niveaux doit être adressable indépendamment et capable de réagir aux sollicitations. Dans un contexte de systèmes artificiels cela implique que chacune des entités devra disposer de son propre flôt de contrôle.
- *L'interaction avec l'environnement ne peut se faire qu'au niveau le plus bas*. Dans l'optique de la supervision d'un système physique, la réalité physique est décrite par des niveaux de représentation abstraite.

– *On prèfèrera une interaction par messages*. L'interaction entre les niveaux est indispensable à la fois pour réaliser les transformations d'information entre les niveaux d'abstraction, mais aussi pour gérer la structure du système. Dans l'hypothèse d'un SMA au contrôle décentralisé voué à une distribution physique comme on peut l'imaginer dans un contexte de SMA artificiel, il est difficile d'envisager une communication indirecte entre agents des différents niveaux.

Dans ce chapitre, nous avons abordé la définition d'agent, de système multi-agents. Nous avons aussi analysé des systèmes multi-agents génériques qui s'adaptent aux systèmes complexes multi-échelles.

Les caractéristiques particulières de chaque modèle et chaque famille nous emmènent vers la définition d'un modèle d'agent plus adapté et efficace pour répondre aux questions des systèmes complexes artificiels multi-échelles (multi-niveaux).

Nous avons fait émerger les critères que nous défendons pour ce modèle :

– une architecture récursive,
– une création et une adaptation dynamique et ouverte des niveaux en fonction du problème
– une architecture décentralisée
– une interaction inter-niveau par message
– une interaction avec l'environnement par l'intermédiaire du niveau le plus bas

Le chapitre suivant se propose d'approfondir la notion de récursion dans les SMA afin de jeter les bases théoriques de notre modèle.

Deuxième partie

Contribution

Chapitre 3

La récursivité dans les SMA

Comme explicité précédement, les systèmes multi-agents offrent un outil puissant pour résoudre ou simuler les problèmes complexes. Certains de ces problèmes présentent des exigences multi-échelles et évoluent dans des environnements structurés qui possèdent des propriétés hiérarchiques. Nous défendons l'idée que la récursivité est une manière efficace de satisfaire les exigences et de réduire la complexité de conception du système. Dans ce chapitre, nous présentons d'abord la notion de récursion des problèmes pour rechercher sur quels objets et comment on peut exprimer les propriétés de récurrence dans les SMA. Dans la suite, nous proposons une expression multi-niveau des SMA. L'expression des niveaux d'abstraction est réalisée grâce à la formulation d'un processus récursif de décomposition. Nous décrivons un mécanisme original générique permettant d'établir une architecture générique de SMA multi-niveau.

3.1 La récursion des problèmes

La récursion peut être utilisée pour décrire des structures, des processus et entraine un mode particulier de conception.

3.1.1 Récursion de structure

Une connaissance peut être modélisée comme un ensemble de données. Les éléments d'un ensemble peuvent être définis de plusieurs manières, en particulier :

- en listant tous les éléments de l'ensemble (définition explicite).
- en décrivant les propriétés des éléments (définition implicite).

La définition récursive d'un ensemble comprend deux parties :

- la base : description d'éléments de base ;
- la récurrence : description des règles pour créer les nouveaux éléments à partir des éléments définis ;

Dans la plupart des problèmes, les connaissances peuvent être vues comme une composition d'éléments. Selon Occello (**?**), l'approche récursive peut nous mener à adresser à différents niveaux d'abstraction les phénomènes ou problèmes modélisés et à apporter des améliorations par rapport à :

- la performance du système en réduisant la taille de l'espace des solutions dans chaque niveau ;
- l'efficacité de la représentation complexe de données ;
- l'adaptation dynamique du système en permettant la réorganisation des structures de données de différentes manières en cours de processus.

3.1.2 Récursion de processus

Supposons que le processus *P* traite une information *D* de type quelconque avec une taille quelconque. *P* dépend donc de l'ensemble de données *D*. Une approche récursive de *P* consiste à décrire le processus *P* dépendant de données *D* en rappelant ce même processus *P* sur d'autres données plus «simples» issues du traitement de *D* (**?**).

Cette approche récursive peut être utilisée pour améliorer la phase d'analyse et la conception du SMA en réduisant la complexité de connaissance, de processus et de structure concernée dans le système. On peut appliquer la récursion dans le SMA selon deux méthodes :

- construction des processus et des connaissances utilisées par les comportements d'agents ;
- construction des architectures adéquates du SMA.

Une analyse récursive pour les SMA est utile si on veut résoudre un problème récursif. Cette analyse a l'intention de fournir un haut niveau de spécification pour les SMA à partir d'un problème de spécification. Un des avantages de l'analyse récursive est qu'elle conduit à spécifier un ensemble de composants élémentaires et des règles d'assemblage. La phase d'analyse adopte souvent une approche top-down. L'analyse récursive nous permet de choisir une conception qui suit une approche bottom-up.

3.2 Une proposition pour la modélisation d'un SMA récursif

Le concept de récursion utilisé dans notre travail suit le principe III de l'approche de *Décomposition AEIO* (**?**) (illustrée en Figure 3.1), où la récursion est comprise comme étant une organisation arborescente construite dynamiquement où seuls les agents feuilles ont une existence réelle. Autrement dit, chaque SMA est vu comme un agent, et chaque agent du SMA global, à son tour, peut être vu comme un SMA (**?**). L'agent existe seulement à travers ses composants, représentés par des agents élémentaires dans le niveau le plus bas du SMA. Les niveaux supérieurs sont formés par des nœuds représentant les agents composés qui définissent une forme abstraite de rassemblement des agents élémentaires.

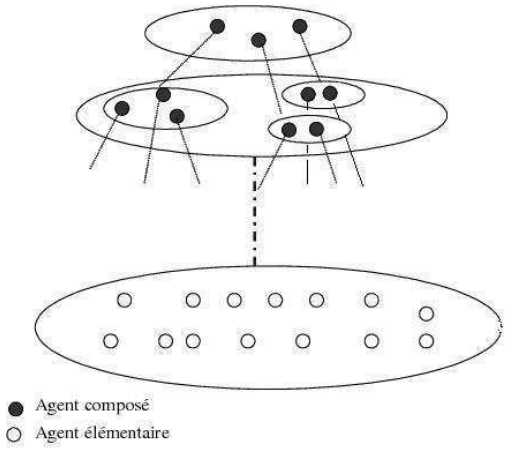


FIGURE 3.1 – Décomposition SMA récursive

On analyse maintenant le SMA selon la décomposition AEIO.

3.2.1 Une expression des composantes d'un SMA selon une décomposition AEIO

3.2.1.1 A (Agents) et E (Environnement)

On suppose que U est l'univers de la société qui est constitué par un ensemble d'agents (A) et un ensemble d'objets de l'environnement (E), on a alors :

$$U = A \cup E$$

Notation et Convention :
On notera :
$a, b, ...$ des agents,
$e, f, ...$ des objets de l'environnement,
$u, v, ...$ des éléments de l'univers.
avec des indices éventuels.

3.2.1.2 I (Interactions)

Les interactions recouvrent tous les échanges entre les agents et entre les agents et leur environnement. Selon Ferber (**?**), il y a deux types d'interactions : les interactions entre les agents et l'environnement et les interactions entre les agents via l'environnement. Dans une autre définition, Occello (**?**) voit les trois types d'interaction :

- perception : action de l'environnement vers les agents ;
- action : action de l'agent vers l'environnement ;
- communication : échange de messages entre les agents.

On peut croiser ces interprétations pour uniformiser une définition de la capacité d'interaction de base.

Définition 3.2.1.1.

Une interaction entre 2 éléments u et v de l'univers sera exprimée par :

$$interaction =< u,\ v,\ c(u,v) >$$

où u est l'*émetteur*, v le *récepteur* et c le contenu.

Le contenu c exprime la catégorie de l'interaction : *perception*, *action*, *communication*, la connaissance échangée entre l'émetteur et le récepteur sous la forme d'une proposition ainsi que les informations classiques sur les modalités d'échange dans le cas des messages. c est en fait un contexte appliqué à l'émetteur et au récepteur, $c \in C$ l'ensemble des contenus.

Dans cette expression, la *perception* est considérée comme une interaction que l'environnement envoie à un agent. L'*action* peut être définie comme une interaction entre un agent et son environnement. Ces deux types sont suffisants pour modéliser les interactions d'agents réactifs car ces agents communiquent uniquement via leur environnement. La *communication* est introduite pour modéliser les échanges de messages entre les agents (émission , réception).

Ceci nous permet de modéliser les interactions :

– perception d'un objet e de l'environnement par un agent a :

$$Perception(a,e) =< e,a,c(e,a) > \ avec\ e \in E,\ a \in A,\ c \in C$$

– action d'un agent a sur un objet e de l'environnement :

$$Action(a,e) =< a,e,c(a,e) > \ avec\ e \in E,\ a \in A,\ c \in C$$

– envoi d'un message d'un agent a_i vers un agent a_j :

$$Message(a_i,a_j) =< a_i,a_j,c(a_i,a_j) > \ avec\ a_i,a_j \in A\ ,\ c \in C$$

Définition 3.2.1.2. *L'ensemble des interactions pour un agent a est défini comme :*

$$I_a = \{< u,v,c(u,v) > | u \in U,\ v \in U,\ c \in C\ avec\ u = a\ ou\ v = a\}$$

3.2.1.3 O (Organisation)

L'organisation est modélisée comme un ensemble de relations entre les agents. On utilise la définition d'organisation définie dans (**?**). Il y a trois types de relations : connaissance (un agent connaît un autre agent), communication (un agent communique avec un autre agent), et subordination (un agent est subordonné à un autre agent).

L'organisation peut être décrite par l'ensemble des relations existant pour l'ensemble des agents de A.

Définition 3.2.1.3.

Une relation d'organisation entre 2 agents a_1 et a_2 sera exprimée par :

$$relation = < a_1, a_2, nt(a_1, a_2) >$$

La nature *nt* exprime le type de la relation : *connaissance, communication, subordination,* ainsi que son contexte c'est à dire le thème sur lequel porte la relation. $nt \in N$ l'ensemble des natures.

Définition 3.2.1.4.

L'ensemble des relations impliquant un agent peut être représenté par :

$$O_a = \{ < a_1, a_2, nt(a_1, a_2) > \mid a_1,\ a_2 \in A,\ nt \in N \text{ avec } a_1 = a \text{ ou } a_2 = a \}$$

Remarque :

Un ensemble de relations de même nature constitue une structure organisationnelle.

3.2.1.4 Le SMA

Un système multi-agent sera donc décrit par le n-uplet $< A, E, I, O >$

avec I ensemble des interactions et O ensemble des relations d'organisation des agents.

3.2.2 Une vision de l'abstraction d'un SMA

L'hypothèse de travail principale de notre approche est que nous disposons d'un SMA donné, dont on souhaite avoir une représentation multi-niveau.

Il peut être décrit selon les définitions énoncées dans la section 3.2.1 par le n-uplet $< A, E, I, O >$ avec $U = A \cup E$.

Nous pouvons construire des abstractions de ce SMA par une transformation récursive afin d'obtenir sa représentation multi-niveau :

- le SMA donné sera assimilé à notre niveau de base que nous noterons SMA^0 caractérisé par le n-uplet $< A^0, E^0, I^0, O^0 >$;
- nous obtiendrons alors des représentations de niveau n , $n \geq 1$. Une représentation abstraite de niveau n du *SMA* sera notée SMA^n.

L'abstraction produit un SMA au modèle similaire dont les éléments sont obtenus à partir du regroupement de ses agents et de ses objets de l'environnement. Ce regroupement est réalisé par l'application de règles de regroupement, qu'on appellera par abus de langage la règle R).

Chaque élément ne peut appartenir qu'à un seul groupe. A chaque groupe d'agents ou d'objets du SMA est associé un nouvel agent ou objet unique.

3.2.3 Définition des transformations pour les Agents et l'Environnement

On peut considérer d'une même façon les agents et objets de l'environnement qui constituent les éléments de l'univers.

Notation :
En complément des notations définies en section 3.2.1.1, on utilisera un exposant pour préciser le niveau d'abstraction i : a^i, e^i, u^i, a^i_j, e^i_j, u^i_j.

A tout élément u^n d'un univers de niveau d'abstraction n, la partition déterminée par la règle R permet d'associer le groupe $G_R(u^n)$ auquel appartient cet élément puis d'associer un nouvel élément u^{n+1} sur le niveau d'abstraction $n+1$.

$$u^n \overset{R}{\longmapsto} G_R(u^n) \longmapsto u^{n+1}$$

Définition 3.2.3.1.
On appellera PU la fonction de transformation définie comme :

$$PU(u^n) = u^{n+1} \textit{ tel que } G_R(u^n) \longmapsto u^{n+1}$$

Remarque :
On remarquera que la transformation PU est dépendante du contexte applicatif.

Propriétés :
On peut exprimer les propriétés suivantes :

$$\forall u \in G_R(u^n)\ PU(u) = u^{n+1}$$

$$PU^{-1}(u^{n+1}) = g \text{ tel que } g \longmapsto u^{n+1}$$

PU et PU^{-1} expriment à la fois les PA et PE :

$$PA = PU|_A \textit{ et } PE = PU|_E$$

3.2.3.1 Interactions et Organisations

Les interactions et organisations associées à un univers sont induites par l'abstraction de cet univers obtenu à l'aide des règles R.

L'abstraction d'une interaction ou d'une organisation peut s'exprimer comme la transformation de cette interaction dans l'univers transformé correspondant. Ce passage d'un niveau d'abstraction n au suivant est pris en charge par les opérateurs PA en ce qui concerne les agents et PE en ce qui concerne les objets de l'environnement.

L'objectif est maintenant d'exprimer les opérateurs de transformation *PI* et *PO* afin de prendre en charge les dimensions *I* et *O*. Afin de conserver la cohérence de notre SMA, il suffira de considérer les interactions et relations induites par l'opérateur *PU*.

Définition 3.2.3.2.

On peut définir PI comme :

$$PI(< u^n, v^n, c(u^n, v^n) >) = < u^{n+1}, v^{n+1}, c(u^{n+1}, v^{n+1}) >$$
$$tel\ que \begin{cases} u^{n+1} = PU(u^n) \\ v^{n+1} = PU(v^n) \end{cases}$$

Propriétés :

$$PI^{-1}(< u^{n+1}, v^{n+1}, c(u^{n+1}, v^{n+1}) >) = \{< u^n, v^n, c(u^n, v^n) > \mid u^n \in PU^{-1}(u^{n+1})\ et\ v^n \in PU^{-1}(v^{n+1})\}$$

Remarques :

- L'*action* d'un agent a^n sur un objet de l'environnement du niveau *n* est la composée de toutes les actions des agents de a^{n-1} antécédents de a^n.
- La *perception* d'un objet de l'environnement e^n par un agent a^n correspond à l'ensemble des perceptions des objets antécédents de e^n par les antécédents de a^n. La liste des agents a_k^{n-1} antécédents de a_n^i est obtenue grâce à *PA*. Vue sous un autre angle, la transformation *PI* peut être considérée dans ce cas comme une règle de fusion des perceptions.
- L'envoi d'un message par a_i^n vers a_j^n sera transformé en un ensemble de messages envoyés depuis les antécédents de a_i^n vers les antécédents de a_j^n.
  Un message reçu par a_i^n de a_j^n sera issu de la transformation d'un ensemble de messages reçus depuis les antécédents de a_i^n depuis les antécédents de a_j^n.

Organisation :

Définition 3.2.3.3.

On peut définir PO comme :

$$PO(< u^n, v^n, nt(u^n, v^n) >) =< u^{n+1},\ v^{n+1},\ nt(u^{n+1}, v^{n+1}) >$$
$$tel\ que \begin{cases} u^{n+1} = PU(u^n) \\ v^{n+1} = PU(v^n) \end{cases}$$

Propriétés :

On peut définir PO^{-1} comme :

$$PO^{-1}(< u^{n+1},\ v^{n+1},\ nt(u^{n+1}, v^{n+1}) >) = \{< u^n,\ v^n,\ nt(u^n,\ v^n) > \mid u^n \in PU^{-1}(u^{n+1})\ et\ v^n \in PU^{-1}(v^{n+1})\}$$

Une relation entre deux agents d'un niveau n sera transformée en un ensemble de relations entre les antécédents de a_i^n et de a_j^n au niveau $n-1$. La réciproque de la transformation *PO* donne une traduction de l'organisation d'un niveau n au niveau $n-1$.

3.2.3.2 La dynamique de la propriété récurrente

Le passage d'un niveau à l'autre et quel que soit le niveau se fait donc par application des transformations *PA, PE, PI* et *PO*. Ces transformations mettent en oeuvre les règles qui fixent les modalités de passage d'un niveau vers l'autre.

L'appel dynamique de ces transformations permet d'augmenter l'adaptativité et la flexibilité des structures en cours de processus. Cette approche est particulièrement utile si le nombre de décompositions récursives est variable et n'est pas défini au préalable. Le développement d'un niveau de récursion peut être créé dynamiquement d'après l'état courant du processus. Cela peut être réalisé par l'application intelligente et dynamique de l'ensemble des transformations récurrentes définies pendant la phase de conception et qui sont dépendantes de l'application. Le nombre de niveaux maximum dépend du problème, c'est à dire du choix de G_R, la méthode de construction des groupes.

3.3 Illustration : le cas de la généralisation cartographique

Nous reprenons cet exemple issu du projet AGENT (**?**) traité dans (**?**). La généralisation cartographique consiste à transformer une information géographique qui correspond à un certain niveau de représentation, en une information généralisée qui correspond à un autre niveau de représentation (plus abstrait ou schématique).

Pour un utilisateur, la visualisation d'une carte est fondamentale pour faire une exploration et une interprétation des données. Le processus de généralisation est utilisé en fonction d'une échelle, laquelle change en fonction de l'application : le climat, les routes, les rivières, l'agriculture, les plans de villes, les bois, ...

Le processus de généralisation cartographique utilise un lot de données, lesquelles peuvent être visualisées depuis différents niveau d'abstraction. Il est possible de modéliser le problème de généralisation à l'aide d'une architecture de SMA.

Nous présenterons par la suite le SMA développé en appliquant l'approche AEIO, à l'aide d'un modèle récursif des quatre éléments A E I O, ainsi qu'une ébauche de sa modélisation selon les définitions de formalisation que l'on a proposé dans la section précédente.

3.4 La description du Problème

Le problème à résoudre consiste à la généralisation des routes, à partir d'une base de données en entrée constituée d'objets géographiques.

Un objet géographique peut être un *ponctuel*, *linéaire* ou *surfacique*. Un objet *ponctuel* est défini par un nœud, un objet *linéaire* est défini par un nœud initial et un nœud final et un objet *surfacique* est spécifié par un ou plusieurs objets *linéaires*. Dans notre application, les données entrées sont des objets routes (objets linéaires)et objets bâtiments (objets surfaciques).

La généralisation des routes consiste à faire ressortir des routes dans carte, c'est-à-dire augmenter la taille des routes demandées. Cela provoque un déplacement ou une réduction des bâtiments qui sont liés aux routes. Le résultat du processus de généralisation cartographique est présenté à l'utilisateur sous la forme d'une carte. On voit quelque fois la généralisation locale des objets géographiques échouer car les objets modifiés ne s'adaptent pas aux objets voisins. Il faut donc une vision dans un niveau supérieur pour arriver à gérer toutes les informations nécessaires pour une bonne généralisation. C'est pourquoi l'utilisation de la récursion pour l'application semble utile.

Cet exemple n'est bien sûr qu'une illustration qui ne prétend pas être réaliste pour le domaine. De très nombreux problèmes difficiles posés par la généralisation (le partitionnement, les limites...) sont ici abordés de façon quelquefois simpliste voire même occultés.

3.4.1 La décomposition du SMA

Par une analyse du problème, il est possible d'identifier la décomposition en E, puis en des éléments A, I et O. Pour simplifier, le SMA de base (donné) sera assimilé au SMA élémentaire de la vision abstraite. En fait, ils sont différents mais l'un représente intégralement l'autre.

3.4.1.1 Le SMA de base

Nous définissons ici SMA^0 en définissant le n-uplet $< A^0, E^0, I^0, O^0 >$.

L'environnement du SMA de base.

L'ensemble des objets géographiques constitue l'environnement du SMA. Il est constitué des *bâtiments* et des *routes*. Les routes sont réparties en catégories selon leur importance.

On nommera *ObGéo* un objet géographique (bâtiment ou route).

E^0 = {ObGéo| ObGéo est un objet-bâtiment ou ObGéo est un objet-route }

Les agents du SMA de base.

A chaque batiment est associé un agent. L'ensemble des agents *A* est formé de tous les agents qui sont associés aux bâtiments.

A^0 = {AgB| AgB est un agent et AgB est associé à un bâtiment}

Un agent est associé à un objet bâtiment.

Les interactions du SMA de base

Dans notre exemple, les interactions ont lieu uniquement entre agents. A ce niveau, elles sont modélisées sous la forme de forces contraignant les agents à se déplacer dans l'environnement.

Une interaction entre 2 agents *a* et *b* de l'univers sera exprimée par :

$$< a,\ b,\ f(a,b) >$$

où *a* est l'*Emetteur*, *b* le *Récepteur* et f la force exercée par l'agent émetteur sur l'agent récepteur.

L'organisation du SMA de base.

Dans (**?**), les agents sont organisés selon deux types de groupes :

(a) *Groupe naturel* : il regroupe tous les agents qui occupent un même point géographique.

(b) *Groupe Artificiel* : c'est l'ensemble des agents qui sont contraints par une topologie commune (par exemple : un ensemble de bâtiments alignés).

Nous ajoutons à ces groupes la notion de *Groupe d'adjacents*, groupe de bâtiments dont la distance de séparation est inférieure à une constante *k*. *k* dépend de la catégorie des routes et du zoom de la carte.

Nous pouvons les exprimer selon le format proposé dans la section précédente. L'appartenance de 2 agents *a* et *b* à un même groupe naturel (le Groupe naturel des objets positionnés en (10,10)) sera exprimée par :

$$< a, b, Groupe\_Naturel\_10\_10 >$$

On peut faire de même avec les groupes artificiels ou les groupes d'adjacents.

3.4.2 Définition des règles de transformation

Pour chaque élément, on définit aussi la propriété récurrente pour sa transformation entre les niveaux.

Transformation pour l'environnement

Nous pouvons définir une règle R de partitionnement qui permet d'associer tout élément e^n de l'environnement de niveau d'abstraction n à un groupe, qui permet de construire un nouvel élément e^{n+1} sur le niveau d'abstraction $n+1$.

Cette règle de transformation permet d'exprimer *PE* comme un partionnement de l'environnement par rapport aux catégories de routes. Par exemple, un objet de l'environnement du niveau 1 peut être considéré comme un *Bloc* qui contient un ensemble de routes et de bâtiments du niveau 0 inscrits dans une zone définie par des routes (rues) de catégorie 1 ou une limite de carte. Cette règle peut être appliquée récursivement sur les *Blocs* pour definir des *Quartiers*, *Districts*, *Villes*....

Transformation pour les agents

Nous pouvons définir une règle R de partitionnement qui permet d'associer tout élément a^n de l'environnement de niveau d'abstraction n à un groupe, qui permet de construire un nouvel élément a^{n+1} sur le niveau d'abstraction $n+1$.

Nous pouvons choisir par exemple la règle de la façon suivante : les agents appartenant à un même groupe adjacent à un niveau n et représentant des objet géographiques situés dans un même bloc (une même partition de l'environnement) dans ce même niveau n deviennent un seul est même agent abstrait de niveau $n+1$.

La propriété récurrente *PA* qui génère les agents abstraits de niveau supérieur est basée sur cette règle. Cette propriété consiste en pratique à agréger des batiments très proches et non séparés par des routes de l'importance considérée au niveau courant. Elle doit aussi prévoir de calculer les nouvelles positions et dimensions du nouveau batiment.

Transformation pour l'interaction

La propriété récurrents *PI* est définie comme une transformation d'une interaction du niveau n à partir d'un ensemble d'interactions du niveau $n-1$. Pour trouver la traduction d'une force d'un agent a_i^n sur l'agent a_j^n. On fera le calcul de la force résultante des forces des agents de $n-1$ agrégés en l'agent a_i^n sur les agents de $n-1$ agrégés en l'agent a_j^n selon la transformation *PA*. A l'inverse, *PI* doit expliciter la force exercée sur un agent a^n au niveau n en un ensemble de force sur tous les agents de $n-1$ agrégés en a^n donnés par $PA^{-1}(a^n)$.

3.4.2.1 Transformation pour l'organisation

La transformation *PO* est définie par la transposition entre les niveaux des règles d'appartenance aux trois types de groupes.

PO consiste, par exemple, dans notre cas en l'identification sur le niveau $n+1$ des agents a^{n+1} qui rejoignent le groupe naturel <Groupe_Naturel_10_10> (le Groupe naturel des objets positionnés en (10,10)) défini au niveau *n*.

Cet exemple très incomplet donne une idée de l'interprétation pratique de la formalisation précédente. Au chapitre suivant, nous présentons un modèle d'architecture d'agent récursif générique fondé sur cette vision de l'abstraction d'un SMA mise en oeuvre à l'aide de la récursivité. Ce modèle répond à la fois aux questions de modélisation et observation des systèmes multi-échelles et satisfait à la décomposition AEIO définie dans la section 3.2.1.

Chapitre 4

Le modèle SMA Récursif Générique - SMA-R

L'observation et la supervision des systèmes complexes artificiels multi-échelles lance des défis liés à leur dynamique, à leur ouverture et à leur hétérogénéïté.

Malgré la limitation de l'étude à la dimension spatiale vue sous l'angle de la densité de la population d'agents, les questions théoriques soulevées restent nombreuses et très variées : Quels sont les éléments que l'on veut observer à travers des échelles ? Quelles sont les règles pour partitionner les individus de chaque niveau ? Comment représenter des comportements collectifs qui émergent à partir des comportements individuels à travers des structures organisées ?

Selon l'intention de l'utilisateur, tous les éléments du système ou seulement certains sont observés. On conserve une même vision de certains éléments sur tous les niveaux d'observation. Mais certains éléments sont regroupés pour donner une vision plus abstraite et plus synthétique au niveau supérieur. Les règles de discrimination pour partitionner un niveau sont déterminantes. Elles sont basées sur la structure organisationnelle des éléments, sur les connaissances ou les capacités des éléments, ou sur les deux. Les règles de partition dépendent de la représentation des comportements collectifs. Le concepteur qui définit les comportements collectifs doit aussi définir les règles de partitionnement, de construction des groupes. L'adaptation de l'observation est effectuée en observant les changements d'un niveau inférieur. La suppression d'un agent abstrait sur un niveau, l'insertion d'une nouvelle abstraction ou l'évolution d'une abstraction ont lieu lorsqu'un changement se produit sur le système réel.

Nous voulons construire un modèle conceptuel qui répond aux exigences que nous avons établies au chapitre 2.

Nous nous appuyons sur la vision de l'abstraction d'un SMA introduite au chapitre précédent

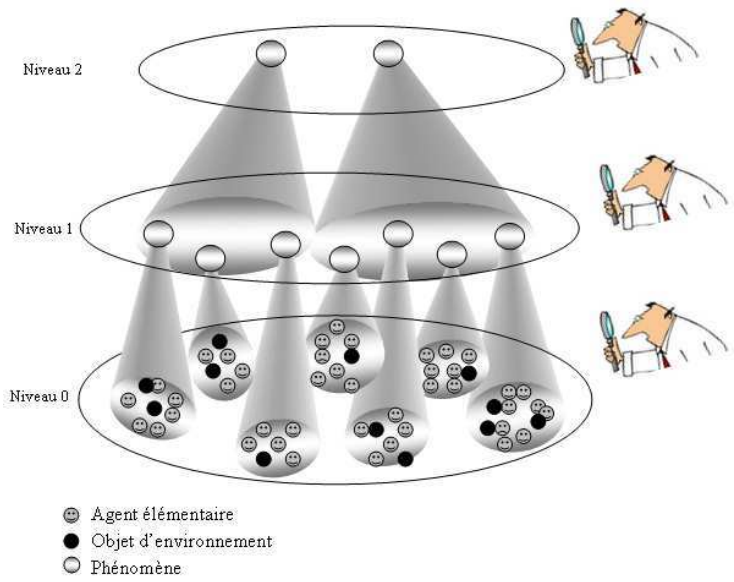


FIGURE 4.1 – Observation d'un système complexe à plusieurs niveaux d'abstraction

et les propriétés récurrentes pour construire ce modèle et sa dynamique. Les propriétés récurrentes permettent de passer d'un niveau à l'autre à la fois en termes de comportement et de structure. Nous proposons un modèle multi-agent récursif basé sur cette idée pour l'observation multi-échelle de systèmes complexes artificiels à grande échelle dans la section suivante. Avant de présenter le modèle SMA récursif générique SMA-R, nous voulons introduire deux modèles développés auparavant dans notre équipe dont nous héritons pour ce travail : les modèles MORISMA (**?**) et MIRAGE (**?**) sont les premières briques pour construire notre modèle.

4.1 Des modèles de base

4.1.1 Le modèle MMASS-MORISMA

Le modèle MMASS (Multi MAS System), proposé par Fernandes (**?**), se compose de trois SMA homogènes : un MAS Cognitif (SMAC) au niveau supérieur, un SMA réactif (SMAR) au niveau le plus bas et un *SMA récursif* ou *SMA intermédiaire* au niveau médian qui est chargé d'assurer l'intégration des deux sociétés SMAC et SMAR (voir Figure 4.2).

On peut considérer que ce travail aborde déjà la notion de passage entre des échelles différentes en termes de granularité, de mode d'interaction, de densité, même s'il ne s'agit que de deux niveaux. La structure de cette couche intermédiaire peut donc nous donner des éléments pour construire un modèle multi-niveau.

La *couche intermédiaire* propose un modèle d'agent appelé **MORISMA** (**MO**dèle **R**écursif pour l'**I**ntégration de **SMA**) mettant en oeuvre une dynamique récursive (**?**). MORISMA construit

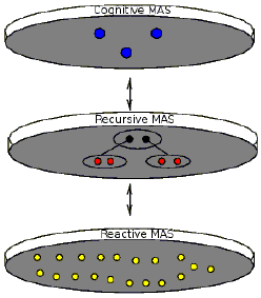


FIGURE 4.2 – La structure du modèle MMASS (**?**)

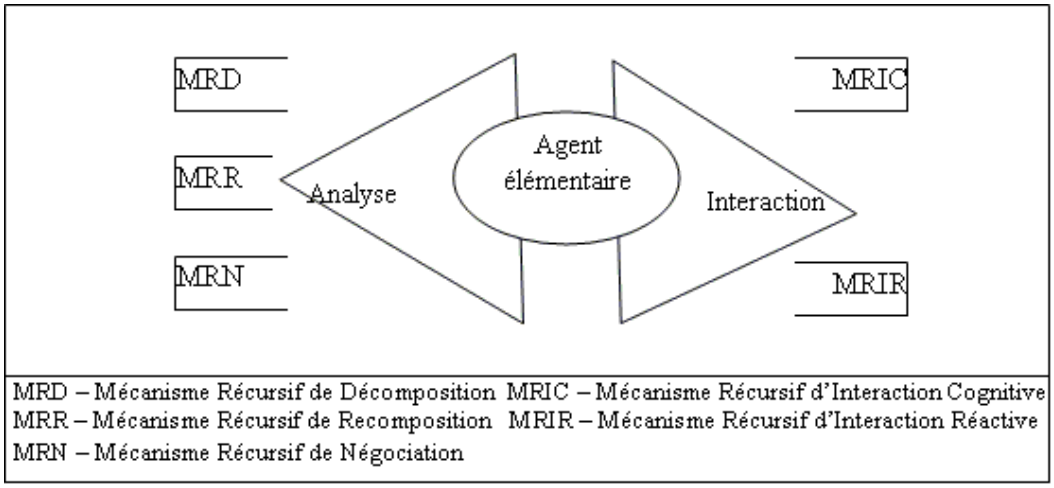


FIGURE 4.3 – L'architecture d'agent MORISMA

dynamiquement des niveaux dans la structure de la couche intermédiaire. La dynamique du modèle MORISMA est réalisée par des fonctions et des mécanismes attribués à chaque agent.

Le modèle de l'agent MORISMA est à la fois hybride (mélange de parties cognitives et réactives) et récursif. Les agents MORISMA peuvent se présenter comme des agents élémentaires ou des agents composés. Les agents élémentaires constituent les feuilles de l'architecture récursive et possèdent une liaison hiérarchique avec le SMAR. Chaque agent élémentaire contrôle un groupe d'agents réactifs du SMAR. Les agents composés sont des agents qui ont subi une décomposition et ont créé une nouvelle société d'agents récursifs élémentaires auxquels ils ont délégué le contrôle des agents du SMAR dont ils étaient responsables. L'architecture des agents MORISMA est présentée dans la Figure 4.3.

L'évolution de la structure d'agents récursifs est assurée par **les mécanismes des agents MORISMA** :

– **Le mécanisme de décomposition**

La *décomposition* est la capacité d'un agent MORISMA à construire un ensemble d'agents

et à leur affecter le contrôle des agents réactifs dont il a la charge.

L'agent déclenche ce mécanisme en cours de fonctionnement au niveau élémentaire de la couche récursive s'il s'évalue comme trop complexe.

La décomposition crée dynamiquement un groupe d'agents dans le niveau inférieur, en répartissant le groupe d'agents réactifs qu'il gérait et en diminuant ainsi la complexité de sa tâche originelle. L'ancien agent élémentaire devient un agent composé. Les nouveaux agents élémentaires s'occupent de contrôler les agents réactifs que gérait l'agent devenu composé.

– **Le mécanisme de recomposition**

La *recomposition* consiste à fusionner un groupe d'agents élémentaires.

Seul un agent composé peut déclencher ce mécanisme. La *recomposition* a lieu si la société créée auparavant par l'agent élémentaire est satisfaite, ou en cas de négociation non réussie, ou encore s'il existe un agent dans la société devenu instable suite à une perturbation interne.

Ce mécanisme a pour effet de supprimer un groupe d'agents élémentaires. L'agent composé qui gère le groupe supprimé devient un agent élémentaire. Il doit contrôler les agents réactifs anciennement rattachés à tous les agents élémentaires supprimés.

– **Le mécanisme de négociation**

La *négociation* a pour objectif de mettre fin aux perturbations touchant les agents du groupe d'un agent composé afin que celui-ci puisse converger (devenir satisfait).

Le mécanisme est lancé par l'agent composé dans le cas où il constate que la société qui le compose est perturbée.

Le succès de la négociation entraine la levée de l'état de perturabation, son échec entraine le lancement d'une réorganisation (décomposition ou recomposition).

La dynamique de la structure récursive est assurée par **les fonctions d'agent du modèle MORISMA**. Un agent MORISMA possède un état courant qui détermine sa perception de l'état de l'organisation récursive. L'agent possède des fonctions d'auto-évaluation pour mesurer ces états. Les changement d'états sont les moteurs de la dynamique du SMA récursif.

Les principales fonctions sont :

– $F_{Complexité}$ (F_C) : Fonction qui mesure le degré de complexité qui détermine si l'agent doit se décomposer ou se recomposer, selon les principes adoptés par le concepteur du système.
– $F_{Satisfaction}$ (F_S) : Fonction de mesure du niveau de satisfaction de l'agent par rapport à son objectif.
– $F_{Perturbation}$ (F_S) : Fonction de mesure du degré de perturbation du comportement de l'agent

par d'autres agents.

- $F_{Reflexion}$ (F_R) : Fonction de mesure du degré de réflexion sur son comportement.
- $F_{StabilitéAgent}$ (F_{SA}) : Fonction de stabilité qui calcule l'état interne de l'agent élémentaire.
- $F_{StabilitéSociété}$ (F_{SS}) : Fonction de stabilité de l'agent composé, qui calcule l'état de satisfaction de la société qu'il gère.

4.1.2 Le modèle MIRAGE

L'objectif du travail présenté dans (**?**) était d'envisager de construire un modèle récursif générique SMA à partir d'un modèle indépendant d'agents applicatifs. Ce modèle adopte une décomposition AEIO. Il est construit grâce à l'analyse de la récursivité sur les quatre axes A, E, I et O présentés dans la section 3.2. Concernant l'agent, le modèle doit donc être capable d'intégrer à la structure du modèle récursif introduit dans le chapitre précédent :

- la structure d'un Agent/SMA récursif
- la fonction d'interaction récursive.

La structure de l'environnement est intégrée dans celle du SMA. L'organisation du SMA est récursive, elle est traitée comme une organisation ordinaire.

La dynamique du système dépend évidemment de la propriété récurrente de la même façon qu'elle est définie dans le chapitre 3. On peut considérer le mécanisme de récursivité comme un moteur, externe et encapsulé dans les agents, qui observe et décide quand la récursivité est déclenchée ou pas. Le moteur est appelé **MIRAGE** (**M**anager of **I**nteraction for a **R**ecursive **AGE**nt) (**?**). Ce moteur joue le rôle d'un agent ordinaire et constitue l'interface d'un SMA qui perçoit les événements des autres agents.

Un agent MIRAGE encapsule un agent applicatif ou un groupe d'agents MIRAGE.

Le modèle récursif générique de l'agent MIRAGE se compose de 4 modules :

- l'*observateur* : évalue la situation de l'agent élémentaire ou de sa société. Cette fonction décide si une composition ou une recomposition est nécessaire ;
- le manager de la *recomposition* récursive : met en oeuvre le mécanisme d'ouverture d'un niveau de récursion ;
- le manager de la *décomposition* récursive : met en oeuvre le mécanisme de fermeture d'un niveau de récursion ;
- le manager de l'*interaction récursive* : met en oeuvre la fonction d'Interaction qui implémente les règles de gestion d'action, de perception ou d'échange de message à travers les niveaux de récursion selon la définition reprise dans le chapitre précédent.

L'architecture MIRAGE décrite sur la Figure 4.4 répond aux exigences de la modélisation des systèmes multi-échelles. Elle améliore les points faibles du modèle MORISMA tels que l'insertion des nouveaux éléments ou la généricité.

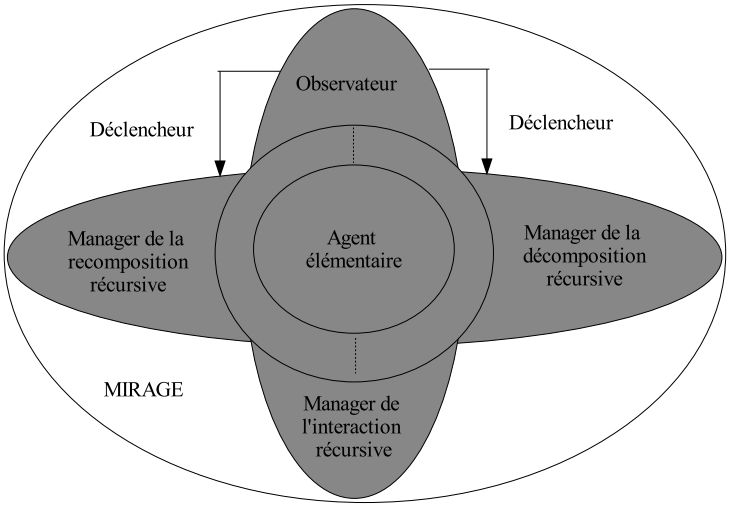


FIGURE 4.4 – L'architecture de MIRAGE

4.1.3 Discussion

Le modèle MMASS répond à des questions de modélisation de systèmes complexes multi-échelles par son souci d'assurer une interopérabilité entre deux systèmes multi-agents qui introduisent des niveaux de lecture différents en terme de granularité, de capacités de comportement, de mode d'interaction. Cette interopérabilité est réalisée à l'aide d'une couche multi-agent (MORISMA) qui utilise une dynamique récursive. Une première lacune de ce modèle est son incapacité à traiter le problème de l'ouverture. Les mécanismes récursifs mis en oeuvre sont tout à fait pertinents pour nos besoins. Cependant, nous sommes ici dans le cas d'une décomposition récursive de tâches ("divide and conquer") dans l'optique du contrôle de résolution et non dans un processus de construction de niveaux d'abstraction. Cela constitue le second handicap de MORISMA, car l'application des mécanismes est inversée par rapport à nos objectifs.

Le modèle MIRAGE, même s'il ne possède qu'une couche, a aussi deux mécanismes *Décomposition* et *Recomposition* pour la fermeture ou l'ouverture d'un niveau de récursivité. Le modèle permet aussi de superposer les agents récursifs sur chaque agent applicatif. Les agents applicatifs sont attachés au niveau élémentaire. Les agents récursifs possèdent des propriétés récursives pour transformer les quatre éléments de l'approche AEIO entre les niveaux. Cependant, ce modèle reste un modèle uniquement abstrait. Par exemple, la circulation des messages entre les couches n'est pas abordée.

Concernant la généricité du modèle et son adaptation à l'encapsulation d'agents de modèles variés. MORISMA est conçu pour gérer des groupes d'agents réactifs au modèle prédéfini. L'idée est d'exploiter les mécanismes, de les approfondir et de les rendre indépendants du modèle d'agent intégré. MIRAGE a adopté cette approche et avancé dans ce sens mais n'a pas proposé de solution

opérationnelle.

Nous situant dans la suite des travaux sur MORISMA et MIRAGE, nous proposons le modèle d'architecture d'agents récursifs SMA-R. Il reprend les mécanismes de MORISMA et l'organisation interne de MIRAGE. SMA-R répond à la fois :

- à l'idée principale du concept de récursivité où la récursion est comprise comme étant une organisation arborescente construite dynamiquement, où seuls les agents feuilles ont une réalité applicative ;
- aux questions posées par le problème de l'observation d'un système complexe de grande échelle ;
- et à la définition de la récursivité présentée dans le chapitre précédent (chapitre 2).

4.2 Modèle SMA Récursif Générique - SMA-R

Nous décrivons maintenant SMA-R, notre modèle SMA récursif générique.

4.2.1 Le modèle des agents - leurs compétences

Nous commençons par détailler les types d'agents du modèle et leurs capacités.

Les agents applicatifs

SMA-R s'applique sur un SMA donné, que l'on veut observer selon des règles d'abstraction, sur lequel on construit une hiérarchie de niveaux. Ce SMA est appelé le système applicatif.

On appelle **Agent applicatif** un agent de l'application qui applique le modèle SMA-R. On peut aussi dire "agent du système". Ces agents qui ont leurs propres compétences, connaissances et capacités sont organisés selon leur propre modèle d'agent et de société. Le modèle SMA-R ne modifie en rien le SMA applicatif. Il interagit simplement avec lui.

Les agents récursifs

Le SMA-R est composé d'agents récursifs.

Un **agent récursif** peut être *élémentaire* (il est alors associé à un agent applicatif) ou *composé* (ou abstrait, il représente alors un groupe au niveau inférieur) :

- Le **niveau élémentaire** d'un *agent récursif* est associé à un *agent applicatif* qu'il encapsule ou avec qui il est interfacé. Les agents récursifs de niveau élémentaire sont les seuls qui peuvent interagir directement avec les agents applicatifs. Chaque agent du niveau élémentaire encapsule un seul agent applicatif. Il y a un seul agent du niveau élémentaire par agent applicatif. Les agents récursifs sont capables de construire une abstraction au niveau supérieur. Du

point de vue du niveau supérieur, de chaque SMA du niveau inférieur émerge des abstractions qui peuvent être vues comme des agents composés (ou virtuels). Chaque agent composé d'un niveau supérieur a les mêmes capacités et compétences qu'un agent du niveau élémentaire. Il peut également traduire les messages circulant verticalement. À son tour, en observant la société de son niveau, il peut s'agrèger pour construire une abstraction au niveau supérieur. Cette construction est basée sur des conditions dynamiques du système de son propre niveau. Les agents récursifs doivent alors évidemment être dotés des mécanismes pour la construction et la réduction des phénomènes présentés à la fin de cette section.

– Sur un **niveau quelconque**, un *agent récursif* est appelé *agent composé*, il représente un groupe d'agents récursifs du niveau inférieur. Chaque agent possède deux mécanismes de *composition* et de *réduction* pour construire ou détruire des niveaux d'abstraction dont le niveau le plus bas contient des agents élémentaires du système. Les niveaux supérieurs, qui sont des niveaux abstraits pour l'observation du système, sont constitués d'agents récursifs composés (ou agents abstraits). Les agents récursifs possèdent aussi une fonction d'*observation* qui examine les conditions pour déclencher les deux mécanismes *composition* et *réduction*.

L'observation se base sur l'analyse des états des agents, ces états sont évalués par des **fonctions dynamiques**. Une fonction dynamique est une fonction d'auto-évaluation. Les fonctions dynamiques sont présentées dans la section 4.2.1. Ces fonctions peuvent-être dépendantes des agents, des objets de l'environnement, de l'organisation ou des interactions dans la société.

L'agent récursif possède une partie *Connaissances* qui contient les connaissances et les capacités d'un agent applicatif. Cette partie contient de plus des informations du contexte de récursion.

En ce qui concerne les **interactions**, chaque agent récursif doit comporter un module d'interaction pour interagir dans sa société. On peut dire que ce sont des *interactions horizontales*. L'agent récursif possède des capacités de traduction des interactions au niveau supérieur ou réciproquement. Ce sont des *interactions verticales* (voir Figure 4.5). Une fonction d'*interaction récursive* prend en charge de recevoir, envoyer, transformer et transférer les messages.

Le modèle SMA-R répond aux exigences d'un système récursif définies au chapitre **??**. Il faut construire tout d'abord le niveau 0 qui est la granularité la plus fine du système (niveau élémentaire). Chaque agent du niveau élémentaire encapsule un agent applicatif. Si un agent applicatif apparaît en cours de fonctionnement un agent élémentaire est créé et lui est associé. Le concepteur doit fournir les conditions pour construire les niveaux virtuels supérieurs. Le concepteur précise les propriétés récurrentes de construction et de passage entre les niveaux. Les 4 propriétés *PA*, *PE*, *PO* et *PI* sont intégrées dans la structure des agents récursifs. *PA* et *PE* définissent comment construire une abstraction sur un niveau à partir des données des niveaux inférieurs. *PI* définit comment une

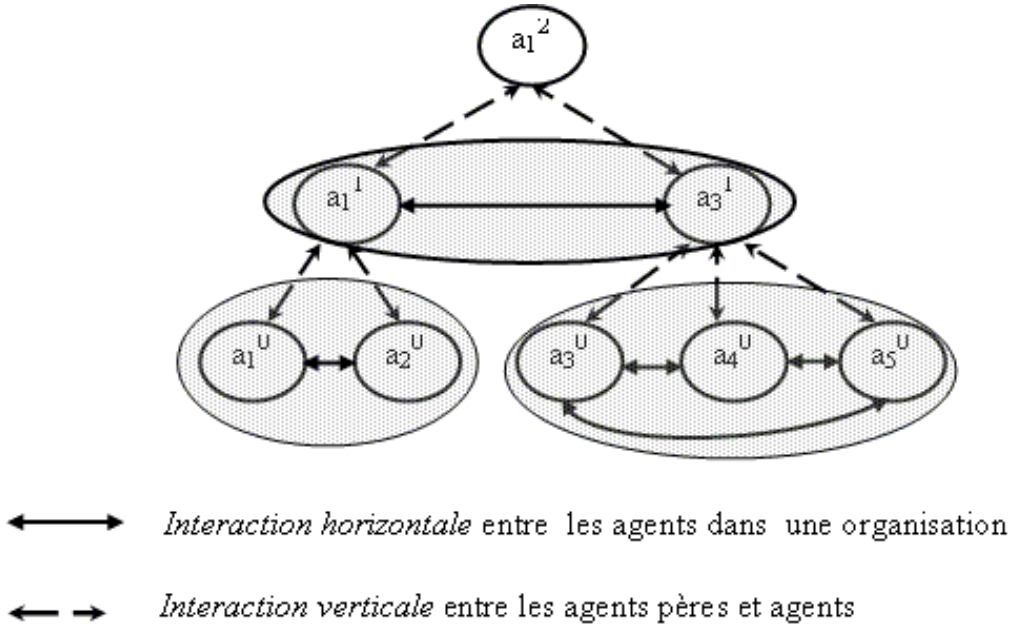


FIGURE 4.5 – Interaction horizontale et verticale

interaction d'un agent abstrait du niveau supérieur dépend des interactions du niveau inférieur. La dynamique récursive des niveaux est réalisée grâce aux fonctions dynamiques et aux mécanismes récursifs.

Les fonctions dynamiques du modèle

Les agents récursifs observent l'organisation, les agents du voisinage, l'environnement, leur état interne et obtiennent des connaissances sur le groupe observé dans leur niveau. Ces connaissances dépendent des critères d'observation fixés par le concepteur. Ces critères peuvent traduire des comportements ou des états des structures organisationnelles des agents. Ils peuvent traduire aussi des situations constatées sur un groupe associé à un agent composé. En se basant sur les critères d'évaluation des travaux de Fernandes (**?**) nous considérons les critères qui sont également conformes à notre problème :

- l'état de *complexité* de l'agent : indique si l'agent satisfait à la condition de changement de niveau d'abstraction spécifiée par le concepteur,
- l'état de *stabilité* de l'agent : indique si l'état de complexité n'est pas modifié depuis un temps donné appelé période de sureté,
- l'état de *stabilité du groupe* qu'il représente : indique si l'ensemble des agents composant un agent composé est stable,
- l'état de *satisfaction* : indique si l'agent a réussi à atteindre son objectif.

Les fonctions dynamiques sont détaillées comme suit [1] :

1. F_C : complexité de l'agent. Cette fonction détermine si l'agent doit se composer avec d'autres ou se réduire, selon les principes fixés pour la propriété récurrente *PA* par le concepteur du système.
   - Initialement $F_C = 0$;
   - S'il est en capacité de générer un groupe, alors $F_{Complexité} = 1$;
2. F_{SA} : stabilité de l'agent. Cette fonction vérifie le changement d'état interne de l'agent. Le résultat de la fonction est :
   - Initialement F_{SA}=0 ;
   - L'agent devient instable F_{SA}= 0 durant la période de *sûreté*, s'il y a des :
     - changements de l'agent,
     - changements de ses voisins,
     - changements de l'environnement.
   - S'il n'y a aucun de changement durant la période de *sûreté*, l'agent est stable, alors F_{SA}=1. Le concepteur doit définir les états stables et instables selon les applications.
3. F_S : satisfaction de l'agent. Cette fonction mesure le niveau de satisfaction de l'agent par rapport à son objectif.
   - Initialement F_S=0 ;
   - Lorsque l'objectif choisi est satisfait, alors F_S=1.
4. F_{SS} : stabilité de la société. Cette fonction concerne la société gérée par l'agent. La valeur de cet état est issue de l'observation des états internes de tous les agents de la société agrégée. Sa valeur est calculée comme suit :

   $$F_{SS}(a) = \text{f}(F_{SA}(a_1), , ..., F_{SA}(a_n)) \times F_{SA}(a) = Min(F_{SA}(a_i)) \times F_{SA}(a)$$

   où $a_1, ..., a_n$ sont des agents agrégés du niveau inférieur. Sous cette forme, tout agent composant instable rend obligatoirement l'agent abstrait instable. Le changement de fonction *Min* par une fonction plus flexible f(F_{SAi}), par contre plus complexe, permet de tolérer, une instabilité partielle de l'agent dans la société.

   Les valeurs de ces fonctions peuvent varier de 0 à 1. Cependant, le concepteur peut modifier ces valeurs pour affiner le comportement de son système.

Toutes les fonctions ne sont pas systématiquement obligatoires.

La dynamique du modèle

Pour présenter la dynamique de notre organisation, nous définissons des mécanismes internes aux

1. On confond souvent dans notre discours l'état et sa fonction d'évaluation

agents qui sont responsables de la création et de la suppression automatique des phénomènes dans les niveaux d'abstraction : fonction de *Composition* et fonction de *Réduction*. En cours d'exécution, l'état des agents peut être modifié sur l'ensemble des niveaux d'abstraction. La modification des états peut entrainer dynamiquement la création ou la suppression d'agents abstraits sur leur seule décision. La structure récursive évolue donc de façon dynamique au cours de la résolution. Les agents du niveau élémentaire ne se réduisent jamais puisqu'ils sont attachés à des agents applicatifs. Les phénomènes au niveau supérieur à 0 sont les seuls qui peuvent se réduire en réalisant le processus *Réduction* :

- **Mécanisme de Composition** :
  Il est chargée de créer dynamiquement un agent composé au niveau supérieur. Ce mécanisme est déclenché par les mécanismes d'*Observation* d'un agent récursif. Quand les fonctions dynamiques de l'*Observation* mettent en évidence certaines valeurs pour les états de complexité, de stabilité de la société, de stabilité locale, de satisfaction ..., l'agent déclenche le mécanisme *Composition* et invite les autres agents à construire un nouveau phénomène au niveau supérieur. Ce nouveau phénomène représentera une nouvelle société contenant les agents qui acceptent l'invitation et celui qui a déclenché le processus *Composition*. Il compose également des objets de l'environnement qui ont des relations avec ces agents ou phénomènes selon la propriété de récursion *PE* définie par le concepteur.
- **Mécanisme de Réduction** :
  Ce mécanisme est utilisé si les abstractions doivent être détruites car remis en question par des changements du niveau élémentaire. Les agents du niveau élémentaire ayant pris en charge les compositions d'agents, sont aussi ceux qui provoquent la destruction de leur agent père. Si leur condition de *Réduction* est satisfaite, ils envoient à leur agent père une demande de *Réduction*. L'agent père avertit à son tour son agent père de sa disparation et se détruit.

4.2.2 L'architecture d'agent récursif générique

Chaque agent récursif s'organise en 5 parties (voir Figure 4.6). Les propriétés *PA*, *PE*, *PI* et *PO* sont exprimées dans ces différents modules. Les fonctions et les paramètres sont définis par le concepteur. La partie *K* est totalement dépendante de l'application. Le *R* et le *C* sont des mécanismes indépendants de l'application mais utilisent des règles de passage dépendant du problème. Le *O* et le *IR* ne sont liées que partiellement à l'application. L'ensemble de ces mécanismes permet de construire la récurrence sur l'organisation.Les 5 modules sont détaillés dans le tableau 4.1.

Au niveau élémentaire les agents récursifs encapsulent ou interfacent les agents applicatifs et sont capables d'interagir par leur biais avec le système réel. Les agents composés se chargent de transmettre les messages à leurs agents fils et à leur agent parent ou interagissent dans leur société

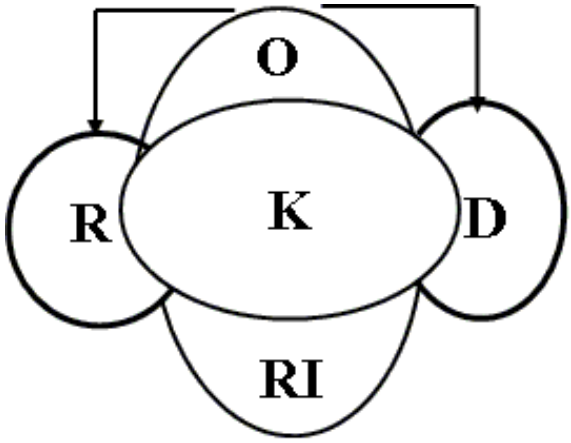


FIGURE 4.6 – Architecture d'agent récursif générique

Partie	Contenu	Commentaire	Propriété récursive
K	Connaissances, Capacités pour la construction des abstractions, Fonctions dynamiques	Partie liée à l'application	PA, PE, PO
	Niveau d'abstraction de l'agent, État de récursion Liste d'agents agrégés, Liste des objets agrégés	Connaissance du contexte récursif : Connaissance du contexte récursif	
O	Observation des états dans *K* pour appeler des fonctions *C* ou *R* Appel Composition(), Appel Réduction()	Partie générique : Observation et Auto-évaluation des fonctions dynamiques Partie liée à l'application : Observation des états (dans la partie *K*) pour déclencher les *C* ou *R*	
C	Composition()	Partie générique : Création d'un nouvel agent à partir d'une liste des agents agrégés du niveau inférieur	
R	Réduction()	Partie générique : Notification à l'agent père de sa disparition et suppression de lui-même. La taille de structure de la récursivité est réduite.	
IR	Envoi, réception, analyse, traitement, transformation, transfert des messages. Mise à jour de *K*	Partie générique : Interaction du mécanisme récursif Partie liée à l'application : interaction applicative	*PI*

Tableau 4.1 – Tableau du détail des cinq parties de l'agent générique

virtuelle. Ils héritent aussi des informations communes de leur sous-société. Autrement dit, ils jouent un rôle de représentant de tous les agents qui les composent :

1. Les *connaissances et capacités* (*K*) : Les connaissances et les capacités dépendent de l'application. Elles décrivent les propriétés de transformation récurrentes *PA*, *PE* et *PO* utilisées pour induire les abstractions, des agents, des objets de l'environnement et de l'organisation. Elles décrivent, en outre, des informations de contexte sur la dynamique récursive, calculées par des fonctions dynamiques comme :
   – État de complexité : évalué par F_C, détermine si l'agent satisfait à une règle de regroupement et s'il doit demander la création d'une abstraction,
   – État de satisfaction : évalué par F_S, détermine si l'agent a satisfait son objectif,
   – État de stabilité de l'agent (lui même) : F_{SA}, détermine si l'agent est dans un contexte qui n'a pas changé depuis un certain temps (la *période de sureté*),
   – État de stabilité de la société qu'il gère : évalué par F_{SS}, détermine si l'ensemble d'agents représentés par l'agent courant est lui même stable.
2. L'*observation*(*O*) : Il se charge d'évaluer les états du *K* à l'aide des fonctions dynamiques afin de déclencher si nécessaire les deux mécanismes de *Composition* et *Réduction*,
3. Le mécanisme de Composition (*C*) : Les niveaux d'abstraction sont construits grâce à ce mécanisme. Les agents récursifs dans le niveau inférieur créent le niveau supérieur s'ils satisfont la condition de composition. La mesure de la satisfaction est une implantation des règles données par le concepteur.

   Nous considérons le SMA constituant le niveau 0 de la Figure 4.7. Le mécanisme d'observation de l'agent a_2^0 le conduit à déclencher une composition et à inviter les agents a_1^0 et a_3^0 à se composer avec lui. Après avoir reçu l'acceptation de a_1^0 et le refus de a_3^0, le résultat est alors :
   – un nouvel agent a_1^1 est créé comme une société contenant deux agents a_1^0 et a_2^0
   – a_1^0 et a_2^0 ont un nouvel agent père a_1^1 (voir Figure 4.7).

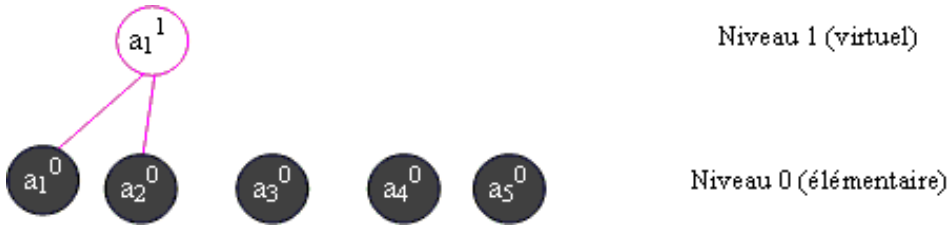


FIGURE 4.7 – La composition de deux agents a_1^0 et a_2^0 en agent composé a_1^1

4. Le *mécanisme de Réduction* (*R*) : Ce mécanisme réalise le processus de réduction du niveau d'abstraction. Considérons le SMA de la Figure 4.8. Supposons que l'agent a_1^0 ait disparu, le mécanisme d'observation de l'agent a_2^0 avertit son agent père a_1^1. L'agent a_1^1 avertit son agent père a_2^2 de sa *disparition* et se détruit lui-même. L'agent a_2^2 supprime l'agent a_1^1 de sa liste

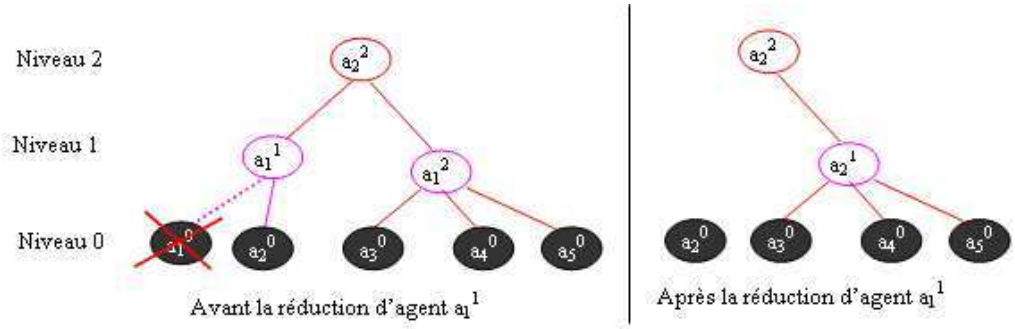


FIGURE 4.8 – L'agent a_1^0 a disparu, cela provoque la réduction de l'agent a_1^1

d'agents agrégés. Il est a noter que dans ce cas l'agent a_2^0 restant seul doit chercher un autre groupe d'appartenance, sous peine de disparaitre de la vision abstraite.

5. L'*interaction récursive* (*IR*) : Cette fonction prend en charge d'envoyer, de recevoir, d'analyser, de transformer et de transférer les messages. Dans cette fonction, le concepteur doit définir la propriété récursive *PI*. On doit traiter deux types d'interaction :
   – L'*interaction système* qui concerne les messages de construction et gestion des niveaux,
   – L'*interaction applicative* qui reproduit au moins partiellement celle de l'application de base.

   Les agents récursifs mettent en œuvre des interactions spécifiques à la construction des niveaux d'abstraction. Les agents récursifs utilisent également des interactions de même nature que celles de l'application de base pour interagir dans la société du niveau auquel ils appartiennent. Au niveau élémentaire, un troisième type d'interaction concerne celles avec les agents applicatifs.

4.3 Conclusion

Dans ce chapitre, nous avons présenté le modèle *SMA-R* générique qui a pour objectif de simplifier l'observation des systèmes multi-échelles et de superviser leur évolution à travers une organisation flexible.

SMA-R peut construire une structure arborescente récursive d'agents dont les nœuds sont des agents récursifs. Les feuilles sont des agents de niveau élémentaire liés aux SMA applicatifs. Les agents récursifs composés sont des agents virtuels qui portent une abstraction. SMA-R est indépendant du SMA applicatif. Même s'il apparait centré sur l'agent récursif, SMA-R réalise bien un SMA récursif pour chacune de ses composantes *AEIO*.

Les principes du modèle SMA-R correspondent bien aux préconisations pour un SMA multi-niveau établies au chapitre 2 et répondent donc ainsi aux exigences établies pour les systèmes mutiéchelles :

- SMA-R propose une architecture récursive : les agents de SMA-R respectent les exigences des règles de récursion détaillées au chapitre 3, qui sont intégrées dans les différents mécanismes du modèle,
- SMA-R est capable de générer plusieurs niveaux d'abstraction comme des niveaux d'observation du système de base : le modèle propose de représenter le SMA applicatif sur un niveau élémentaire. Dans les niveaux supérieurs, les agents représentent des abstractions,
- SMA-R propose une interaction inter-niveau par message : le modèle précise les modalités d'échange de messages entre niveaux, mais aussi les interactions entre agents virtuels de chaque niveau reproduisant le processus abstrait,
- SMA-R propose une interaction avec l'environnement par l'intermédiaire du niveau le plus bas : le modèle est indépendant du SMA applicatif. Il est capable d'intégrer les agents applicatifs pour construire les agents récursifs des niveaux d'abstraction.

Le modèle propose bien-sûr des comportements décentralisés mais pour répondre aux exigences ceux-ci doivent être physiquement décentralisés. Nous nous consacrons, dans le chapitre suivant, à la présentation de l'opérationalisation de ce modèle en une architecture véritablement décentralisée.

Chapitre 5

Architecture décentralisée du SMA Récursif

Le modèle SMA-R présenté dans le chapitre précédent est un modèle conceptuel. Ce modèle nous permet d'observer les systèmes complexes à grande échelle. Les systèmes complexes artificiels que nous visons, sont souvent des systèmes physiquement décentralisés. Notre objectif est de rendre ce modèle applicable à des SMA physiquement décentralisés. Il est donc indispensable de disposer d'une architecture décentralisée qui opérationnalise notre modèle de SMA récursif générique. Nous présentons dans ce chapitre l'architecture décentralisée de SMA-R.

Nous proposons d'adopter les concepts du modèle OSI (Open System Interconnection) (**?**) pour que cette architecture soit vraiment décentralisée. La norme OSI met, en effet, en avant les capacités que nous cherchons : multi-niveau, encapsulation des messages, échange de messages virtualisés et physiques... Le concept OSI repose sur la notion de couche de service. Chaque couche fournit un niveau d'abstraction, en cachant les détails des fonctions des couches inférieures et en offrant des services aux couches supérieures. Les couches d'un même niveau ont leurs propres protocoles de communication qui servent à véhiculer leurs données (PDU - Protocol Data Unit). Cependant, les données ne sont réellement transférées que via la couche physique. Nous désirons construire un système d'observation dont chaque niveau donne une vision abstraite du système. Chaque niveau supérieur donne une vision plus abstraite en cachant les détails des niveaux inférieurs. Les principes sont identiques dans le modèle OSI. Dans notre modèle, les agents de la couche élémentaire sont les seuls pouvant réellement et physiquement être reliés entre eux. Les couches supérieures doivent utiliser les couches inférieures pour envoyer leurs données (encapsulation des messages). Les primitives sont les mêmes pour tous les niveaux (propriété récursive). Le nombre de niveaux est multiple et variable en fonction de la taille de l'application et des propriétés récurrentes données par concepteur. Les messages transmis entre les niveaux sont transformés pour s'adapter au protocole

du niveau inférieur (mécanisme d'encapsulation des PDU). Pour détailler, nous présentons par la suite les principes de l'architecture décentralisée, l'échange des messages entre les agents récursifs et l'architecture du framework d'agent récursif.

5.1 Les principes de l'architecture décentralisée

5.1.1 Le modèle OSI et l'échange de données entre couches

Le modèle de référence OSI (Open Systems Interconnection) a été créé par l'Organisation internationale de normalisation (ISO) en 1978 (**?**). La motivation de ce travail a été de réduire la complexité des logiciels réseaux et d'aller vers la normalisation des différents protocoles existants. Malgré son âge, ce modèle est toujours très utilisé comme un outil d'analyse et un outil pédagogique pour expliquer les étapes importantes impliquées dans un système de communication.

Ce modèle est présenté comme une architecture en couches dans laquelle chaque couche représente un niveau d'abstraction dans la communication qui effectue un ensemble de fonctions. La mise en œuvre d'une couche ne doit pas affecter ses couches adjacentes. Ce modèle est construit sur 7 couches (Figure 5.1) :

- la couche Physique - 1 : cette couche s'occupe de la transmission de symboles sur le médium. Elle prend donc en charge la transformation d'un signal binaire en un signal compatible avec le matériel donné (fibre optique, HF etc.) et réciproquement. L'unité d'information est ici le bit ;
- la couche Liaison - 2 : cette couche assure une communication fiable entre les hôtes directement connectés. Elle prend donc en charge le partage du médium et la détection d'erreurs de transmission des données. L'unité d'information est ici la trame (séquence de bits) ;
- la couche Réseau - 3 : Cette couche s'occupe de l'acheminement des informations au travers du réseau pour des systèmes non directement physiquement connectés. C'est ici en effet que l'interconnexion de réseaux différents est rendue possible et donc que la notion de routage intervient. L'unité d'information est ici le paquet ;
- la couche Transport - 4 : Cette couche assure un transport d'information de bout en bout (d'application à application et non pas de machine à machine) entre deux systèmes d'extrémité et ce indépendamment des caractéristiques du réseau utilisé ;
- La couche Session - 5 : Cette couche fournit à la couche supérieure des outils plus souples que ceux de la couche transport pour la communication d'informations, en introduisant la notion de session. Ainsi des outils de synchronisation et de gestion du dialogue sont fournis ;
- La couche Présentation - 6 : Cette couche s'intéresse à la syntaxe des messages échangés (conversion de code, chiffrement éventuel, compression).
- la couche Application - 7 : Cette couche regroupe les mécanismes utiles aux applications

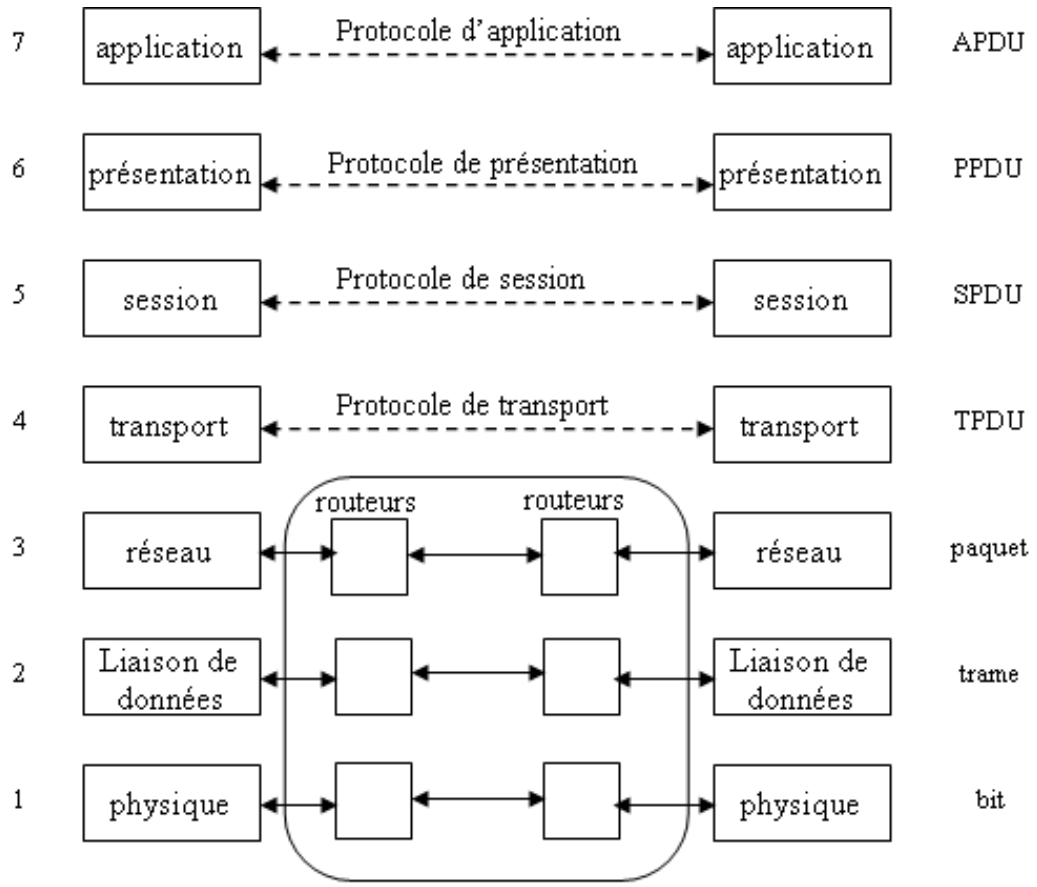


FIGURE 5.1 – Le modèle de référence OSI

réparties. Elle s'intéresse aussi à la sémantique des informations échangées (transfert de fichiers, messagerie, "remote job",...). On retrouve dans cette couche l'ensemble des services réseaux tels que le de transfert de fichiers (FTP), de messagerie (SMTP), de documentation hypertexte (HTTP), etc.

Quand un logiciel utilise la pile de communication pour transmettre des données au logiciel d'un autre hôte, la donnée traverse l'ensemble des couches chez l'émetteur (couche application → couche physique) et de chez le destinataire (couche physique → couche application). A chaque fois la donnée va être encapsulée (ou désencapsulée) c'est-à-dire que chaque couche rajoute une entête et éventuellement une en-queue (ou la retire). Ce principe est illustré en figure 5.2.

En effet, une couche N veut transmettre une donnée (une N—PDU) à sa couche homologue (couche N coté destinataire) elle va pour cela solliciter un service de sa couche adjacente (la couche N-1). Cette (N—PDU) transmise est l'unité de donnée de service (N-1—SDU) qui va être traitée par la couche N-1. La couche (N-1) va rajouter de l'information (N-1—PCI) qui lui sera utile pour le traitement par les couches (N-1) des équipements intermédiaires. Ces informations sont généralement l'adresse de l'émetteur et l'adresse du destinataire, l'identifiant du protocole encapsulé etc.

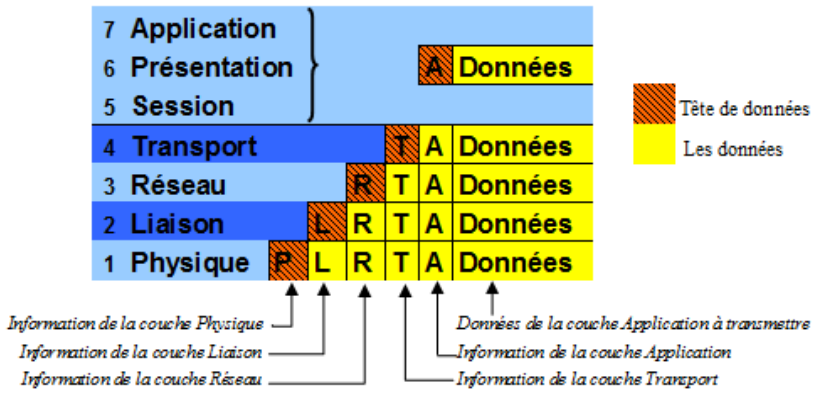
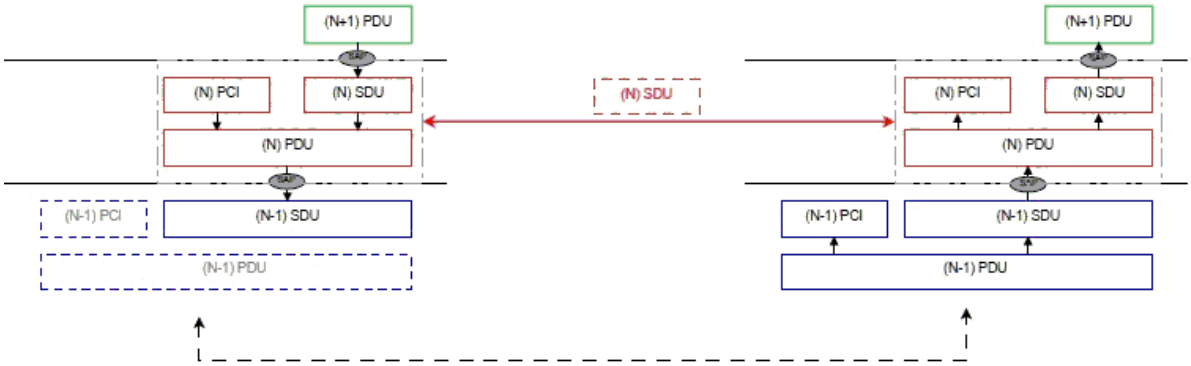


FIGURE 5.2 – L'échange de données entre les couches du modèle OSI



PDU : Protocol Data Unit — Unité d'information échangée entre couches homologues
SDU : Service Data Unit — Unité d'information échangée entre couches adjacentes
PCI : Protocol Control Information — Informations ajoutées aux données pour le fonctionnement du protocole

FIGURE 5.3 – Les changements de données entre les couches du modèle de référence OSI

La (N-1—PCI) et la (N-1—SDU) forment la (N-1—PDU). Le moyen par lequel les utilisateurs de services et les fournisseurs de services peuvent interagir est une interface appelée **SAP** (Service Access Point).

5.1.2 Le principe de l'architecture SMA-Récursif

5.1.2.1 L'architecture externe d'agent récursif

Au lancement du système, les agents du niveau élémentaire encapsulent les agents applicatifs. Ils sont équipés des règles et des mécanismes définis au chapitre 4.2. Ils construisent les niveaux abstraits. Un niveau abstrait contient des agents composés (ou abstraits) qui sont donc des agents agrégés du niveau inférieur. Le niveau le plus bas est le niveau 0. Avant de détailler l'architecture, nous présentons quelques définitions en complément des définitions d'*agent récursif élémentaire* et

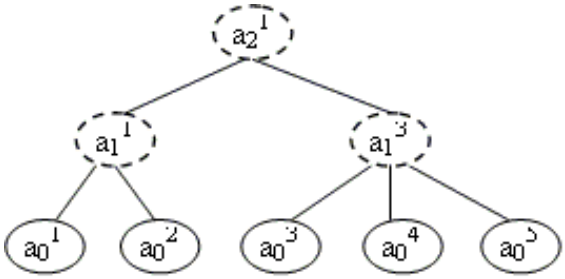


FIGURE 5.4 – Construction des niveaux abstraits

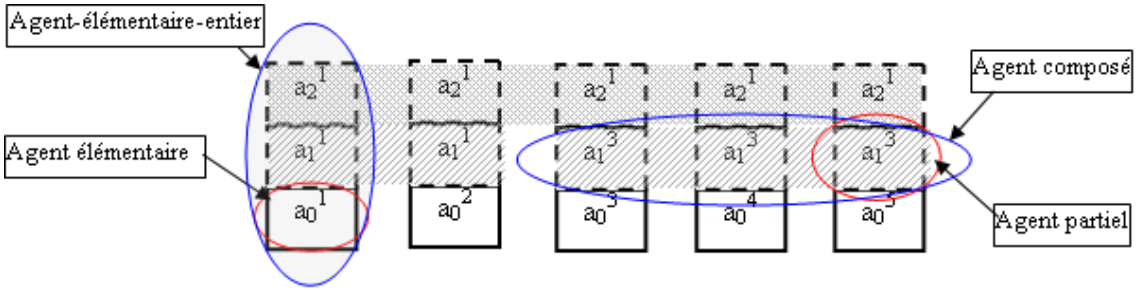


FIGURE 5.5 – Les agents-partiels et les agent-élémentaire-entiers

d'*agent récursif composé* du chapitre 4 [1].

Une structure **agent-partiel** est la reproduction des données d'un agent composé empilée sur un agent agrégé. Tous les agents agrégés au sein d'un même agent composé (créé sur le niveau supérieur) s'attachent une reproduction partielle (ou réplication partielle) de cet agent composé (ou "**agent-partiel**").

Quand des agents agrégés se composent, ils créent un nouvel agent composé conservant les données communes de tous les agents agrégés. Il est considéré comme un agent du niveau supérieur qui représente et donne une vision abstraite de ses agents agrégés. Comme les agents sont distribués, chaque agent agrégé crée son *agent-partiel* de type "agent récursif" et s'attache cette nouvelle structure (figure 5.5).

Un agent composé n'existe donc que de façon décentralisée. L'agent composé s'incarne en un *rassemblement d'agents-partiels* et joue le rôle d'un agent unique. Toutes les structures *agents-partiels* sont identiques pour un même agent composé qu'il représente. Ils portent les mêmes informations, données, capacités... Ces agents-partiels doivent effectuer une mise à jour des données. Quand un *agent-partiel* reçoit un message, il avertit les autres *agents-partiels* composant le même *agent composé* (la société abstraite). Les *agents-partiels* peuvent être dotés de capacités avancées d'interaction pour optimiser leurs échanges d'information.

Les agents composés n'ont pas d'existence centralisée, ils existent à travers des agents agré-

1. Pour simplifier, on omettra volontairement le terme *récursif* dans la suite

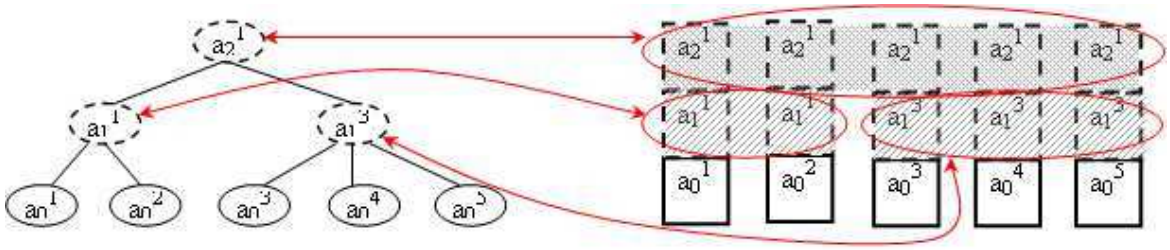


FIGURE 5.6 – L'architecture d'agent récursif pour le SMA de cinq agents

gés. Seuls les *agents élémentaires* sont tous différents. Chaque agent élémentaire porte des *agents-partiels*.

Un **agent-élémentaire-entier** est un *agent élémentaire* muni de tous ses **agents-partiels** (voir figure 5.5).

Les agents récursifs composés ou élémentaires continuent à construire les niveaux supérieurs selon les propriétés récurrentes. Dans la figure 5.4, page 75, cinq agents élémentaires au niveau 0 construisent deux niveaux abstraits.

La mémoire des agents élémentaires est organisée en étage. Chaque étage stocke un des agents composés de la hiérarchie d'appartenance triés par niveau d'abstraction. Le niveau le plus abstrait (le plus élevé) figure à l'étage le plus haut (voir la figure 5.5). Les agents composés ne peuvent communiquer qu'avec les agents composés adjacents. Ils ne peuvent pas sauter les étages.

Un agent élémentaire, qui encapsule un agent applicatif, se situant au niveau 0, joue le rôle de la "couche physique" du modèle OSI dans le sens où c'est lui qui utilisera le médium de communication pour faire transiter les messages. Ses agents composés (niveaux supérieurs) interagiront en utilisant des primitives de services comme dans le modèle OSI.

Le principe de passage des messages entre les niveaux dans l'architecture SMA-R est le même que celui du modèle OSI (voir la section 5.1.1). Chaque fois qu'un message est émis, il doit "descendre" en ajoutant un entête de chaque niveau jusqu'au niveau 0. L'agent élémentaire envoie le message en empruntant le canal de communication des agents applicatifs. Une fois que le message arrive à l'agent élémentaire qui porte l'agent destinataire (composé ou élémentaire), le message est "remonté" pour arriver au destinataire. Les entêtes sont alors supprimés niveau par niveau jusqu'au niveau destinataire.

5.1.2.2 La structure et la circulation des messages entre les couches

Conformément à la définition de l'Interaction dans la section 3.2.1 le message contient une entête et un contenu.

Considérons un agent de niveau *n*. L'entête du message qu'il va généré contient :

– l'adresse de niveau *n* permettant d'identifier l'agent émetteur,

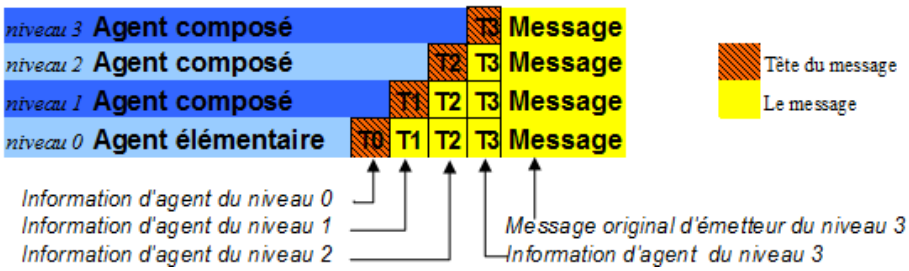


FIGURE 5.7 – Modification du contenu des messages dans le modèle SMA-Récursif

- l'adresse de niveau *n* permettant d'identifier l'agent destinataire,
- un identifiant du type de message (que nous verrons en détail en section 5.2.3,
- des options (informations utiles au niveau *n* pour le traitement du message et qui est donc fonction du type de message)

Le contenu de ce message de niveau *n* sera :

- soit un message de niveau $n+1$,
- soit une donnée de niveau *n*.

5.1.3 Le traitement des messages des agents récursifs

Avant d'aborder la réception et l'émission des message, nous définissons les notions de *message applicatif* et de *message système*.

Un **message applicatif** est un message défini dans l'application indépendamment du modèle SMA-R.

Un **message système** est un message propre à la mécanique de l'architecture SMA-R. Les messages systèmes sont des messages de routage, d'établissement des niveaux abstraits ou d'échange entre les agents composés. Ils sont indépendants de l'application.

La partie IR (Interaction Récursive) du modèle (voir la section 4.1) prend en charge l'envoi, la réception, l'analyse, le traitement, la transformation et le transfert des messages. Particulièrement, les agents élémentaires doivent posséder une interface de communication avec les agents applicatifs. Les agents élémentaires transforment les messages des agents composés du niveau supérieur en messages compatibles avec le canal de communication applicatif.

5.1.3.1 La réception d'un message par un agent élémentaire

La procédure de réception des messages d'un agent élémentaire est illustrée dans la Figure 5.8. Cette procédure de réception est décrite dans l'algorithme 1 :

Algorithme 1 Procédure de réception d'un message par un agent élémentaire

1: *Analyse du message dans la liste des message reçus* : si le message a déjà été reçu, il est rejeté, sinon on passe à l'étape 2 ;
2: *Analyse de l'adresse du destinataire* : si un de ses agents composés (qui sont aux niveaux supérieurs) ou lui même est destinataire, il reçoit le message (dans le cas inverse, voir la section 5.1.3.3)
3: *Déencapsulation du message* : le message devient un message du niveau 0 (il était un message compatible avec le canal de communication des agents applicatifs) ;
4: *Transformation de message* : le message est transformé et devient un nouveau message en appliquant la propriété récurrente *PI* du système ;
5: *Interprétation du type de message* : s'il est un message du type d'application, il est transféré à l'agent applicatif, sinon, l'agent élémentaire traite le message.

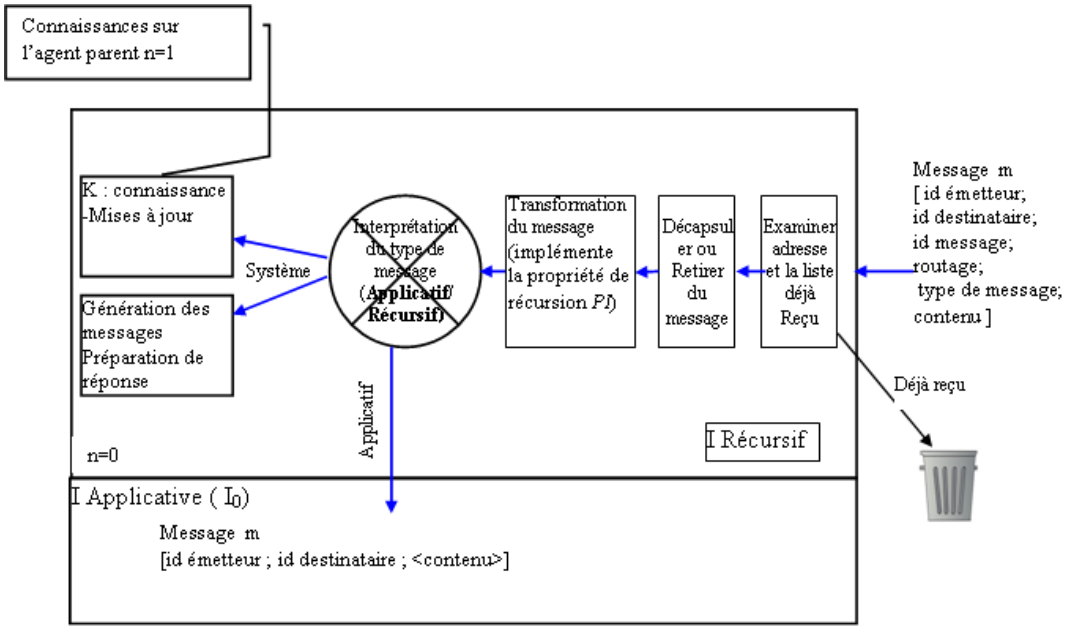


FIGURE 5.8 – Procédure de réception d'un message par un agent élémentaire

5.1.3.2 L'émission d'un message par un agent élémentaire

Les agents élémentaires envoient les messages lorsque leur agent applicatif envoie un message applicatif ou qu'ils veulent envoyer un message du type système. Le message est d'abord transformé en appliquant la propriété récurrente *PI*. Il est ensuite encapsulé dans un nouvel entête contenant ces informations. La phase d'interprétation de l'adresse pour acheminer le message termine ce processus (voir Figure 5.9).

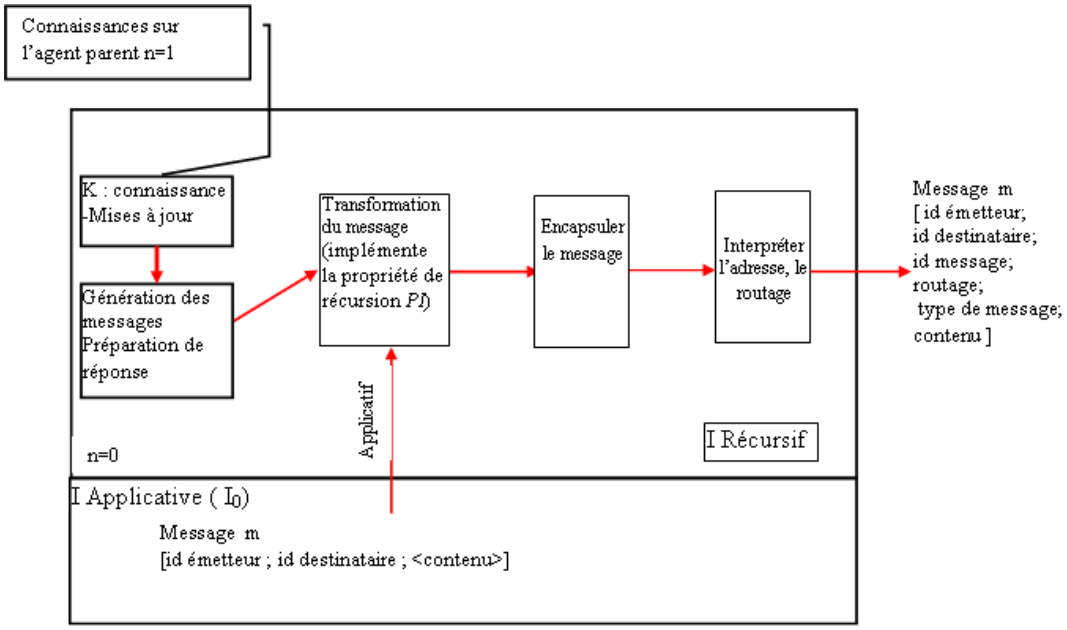


FIGURE 5.9 – Procédure d'émission d'un message par un agent élémentaire

5.1.3.3 Le transfert d'un message par un agent élémentaire

Dans les systèmes distribués, en particulier, les systèmes embarqués autonomes en réseau, outre leur travail, les agents applicatifs prennent aussi en charge de transférer les messages pour que les messages arrivent aux destinataires. Dans ce cas, les agents élémentaires réalisent le transfert des messages comme dans la Figure 5.10.

La procédure de transfert des messages d'un agent élémentaire est décrite dans l'algorithme 2 :

Algorithme 2 Procédure du transfert d'un message par un agent élémentaire

1: *Analyse du message dans la liste des message reçus* : si le message a été reçu auparavant, il est rejeté, sinon on passe à l'étape 2 ;
2: *Analyse de l'adresse de destinataire* : s'il est dans le chemin d'acheminement du message, il est transféré au suivant.

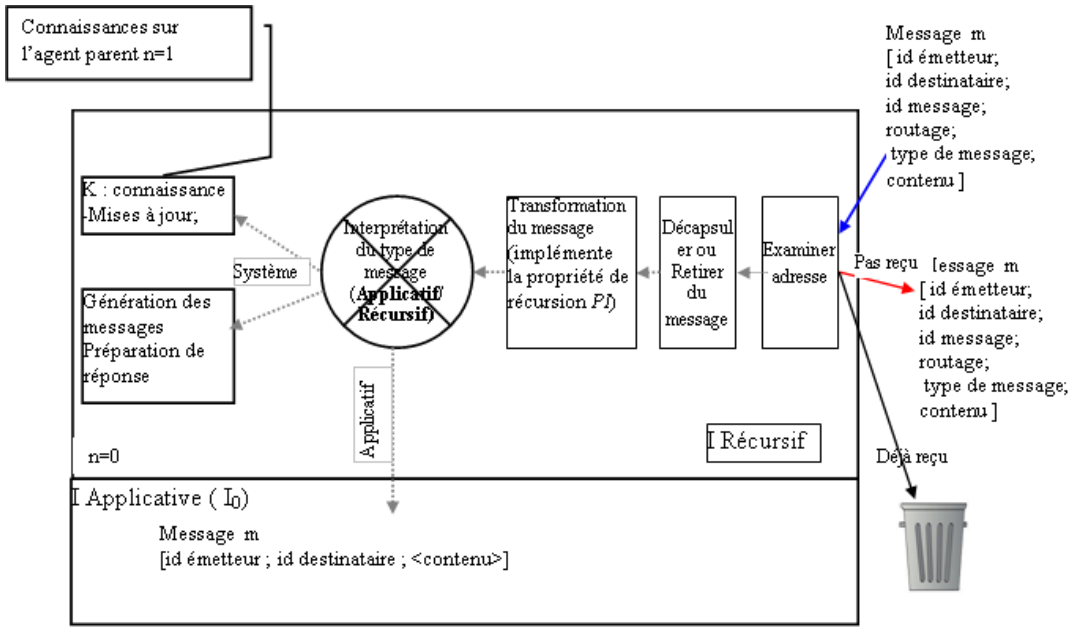


FIGURE 5.10 – Procédure de transfert d'un message par un agent élémentaire

5.1.3.4 L'émission d'un message par un agent composé

La procédure d'émission par un agent composé est présentée dans la Figure 5.11. A priori, l'émission est forcément déclenchée, soit par action de l'utilisateur, soit par sa partie IR. Le message est d'abord transformé suivant la propriété récurrente *PI*. Il est ensuite encapsulé et un nouvel entête contenant ces informations est ajoutée. Le message est descendu à l'agent fils du niveau inférieur. L'agent fils au niveau inférieur (dans cet exemple, le niveau courant est 0) examine l'adresse et désencapsule le message. Si il est destinataire du message, il le reçoit et le traite (voir figure 5.8). Sinon, il encapsule le message, cherche un routage, met à jour la liste des messages reçus et renvoie ce message.

5.1.3.5 La réception d'un message par un agent composé

La réception d'un agent composé est toujours "remontée" par sont agent-partiels fils du niveau inférieur. La procédure de réception d'un agent composé est illustrée dans la Figure 5.12 et expliquée dans l'algorithme 3, page 81.

5.2 Architecture décentralisée d'agent récursif

Dans la section précédente, on a présenté le principe de l'architecture des agents récursifs.

Dans cette section, nous présentons la structure récursive multi-agent générique. Nous voulons fournir un framework qui permet d'appliquer le modèle récursif sur n'importe quelle application

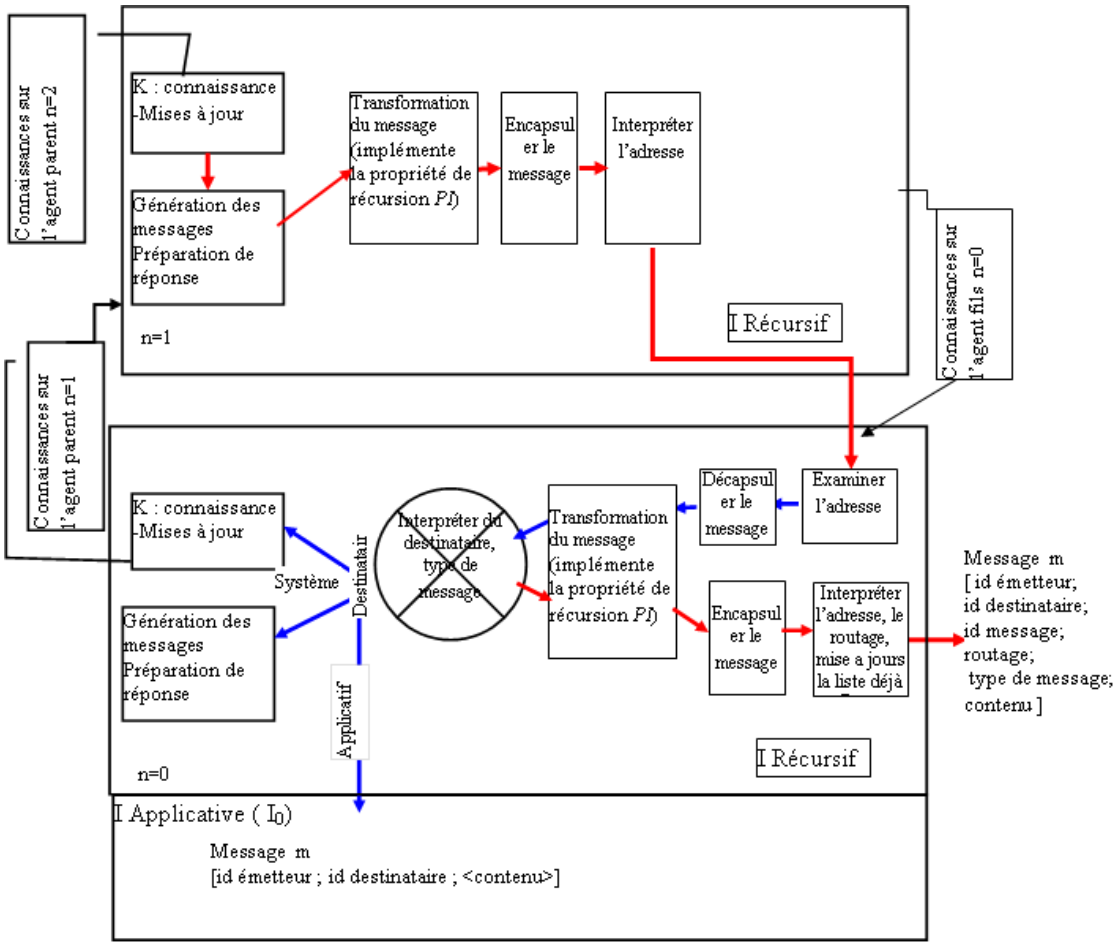


FIGURE 5.11 – Procédure d'émission d'un message par un agent composé

Algorithme 3 Procédure de réception d'un message par un agent composé

1: *Réception du message par l'agent élémentaire* : si le message est déjà reçu auparavant, il est rejeté, sinon on passe à l'étape 2 ;
2: *Analyse de l'adresse du destinataire* : si un de ses agents composés (qui sont aux niveaux supérieurs) ou lui-même est destinataire ;
3: *Désencapsulation du message* : le message devient un message du niveau 0 ;
4: *Transformation de message* : le message est transformé et devient un nouveau message en appliquant la propriété récurrente *PI* du système ;
5: *Passage du message à son agent père au niveau supérieur* (parce que le message est à destination des agents composés qui se situent aux niveaux supérieurs).

réelle.

5.2.1 Une structure récursive d'agents SMA-R

Cette architecture est présentée par des diagrammes UML. Le diagramme de classes de la Figure 5.13 décrit les principaux éléments du diagramme d'architecture. Le diagramme d'états des

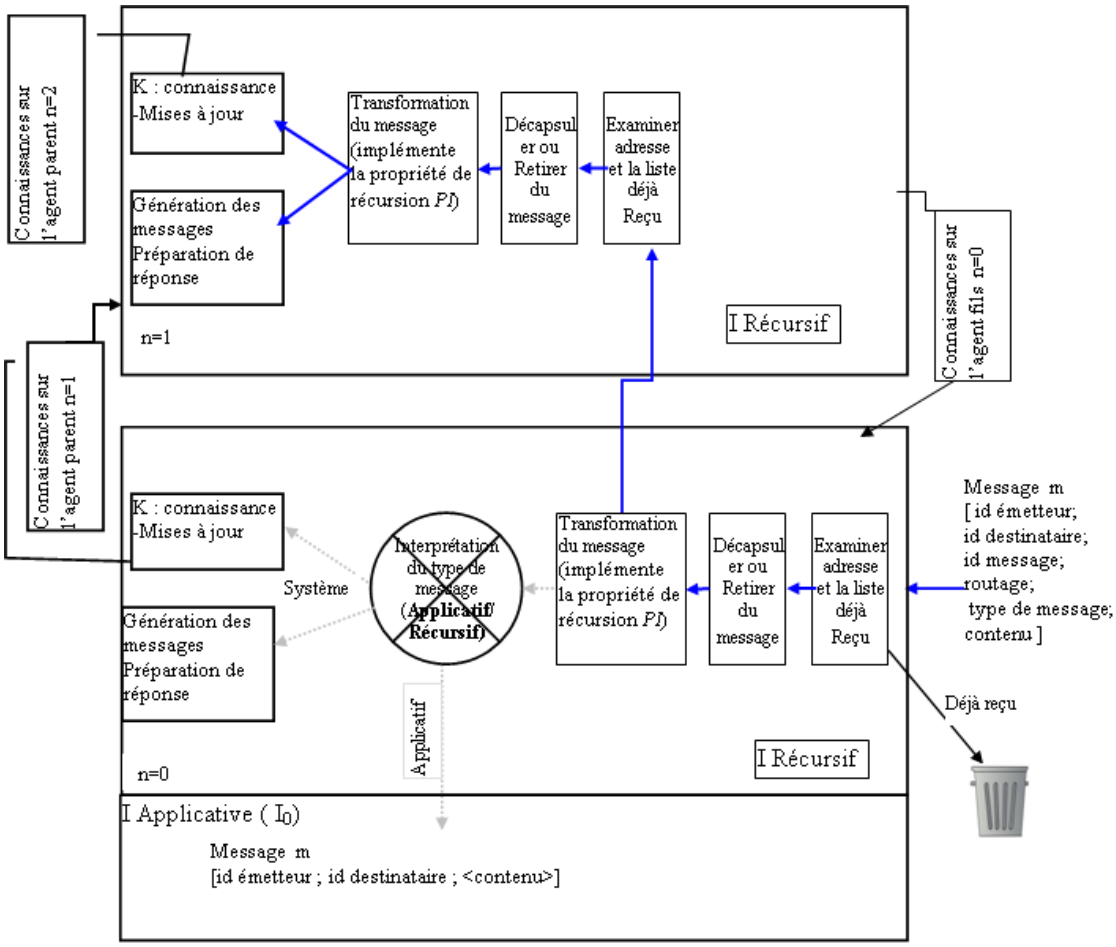


FIGURE 5.12 – Procédure de la réception d'un message par un agent composé

Figures 5.14 et 5.15 décrivent le processus de *Composition* et de *Réduction*.

Dans le schéma (Figure 5.13), la classe principale est *RecursiveAgent*. Chaque instance de la classe *RecursiveAgent* est associée à un agent applicatif (classe *ApplicativeAgent*). La classe *RecursiveAgent* a deux associations réflexives : l'association *next* pointe vers l'agent père dans la couche supérieure et l'association *previous* pointe vers l'agent fils dans la couche inférieure. La classe *ApplicativeAgent* représente le lien avec les agents de l'application réelle, ce peut être un interfaçage ou un héritage. La classe *DynamiqueFunction* contient les *Fonctions dynamiques*, qui calculent des fonctions dynamiques des agents afin de déclencher les deux fonctions *Composition* et *Réduction*. La classe *SystemMessage* contient des messages pour gérer la recursivité des agents.

Reprenons les parties composantes du modèle SMA-R (section 4.2) :

– la partie *K* contient une identification, un niveau de récursion, un lien vers un *agent applicatif*, des *fonctions dynamiques* et une liste d'agents agrégés,
– la partie *O* est implémentée dans la méthode *recursive_Observation()*. Cette méthode appelle

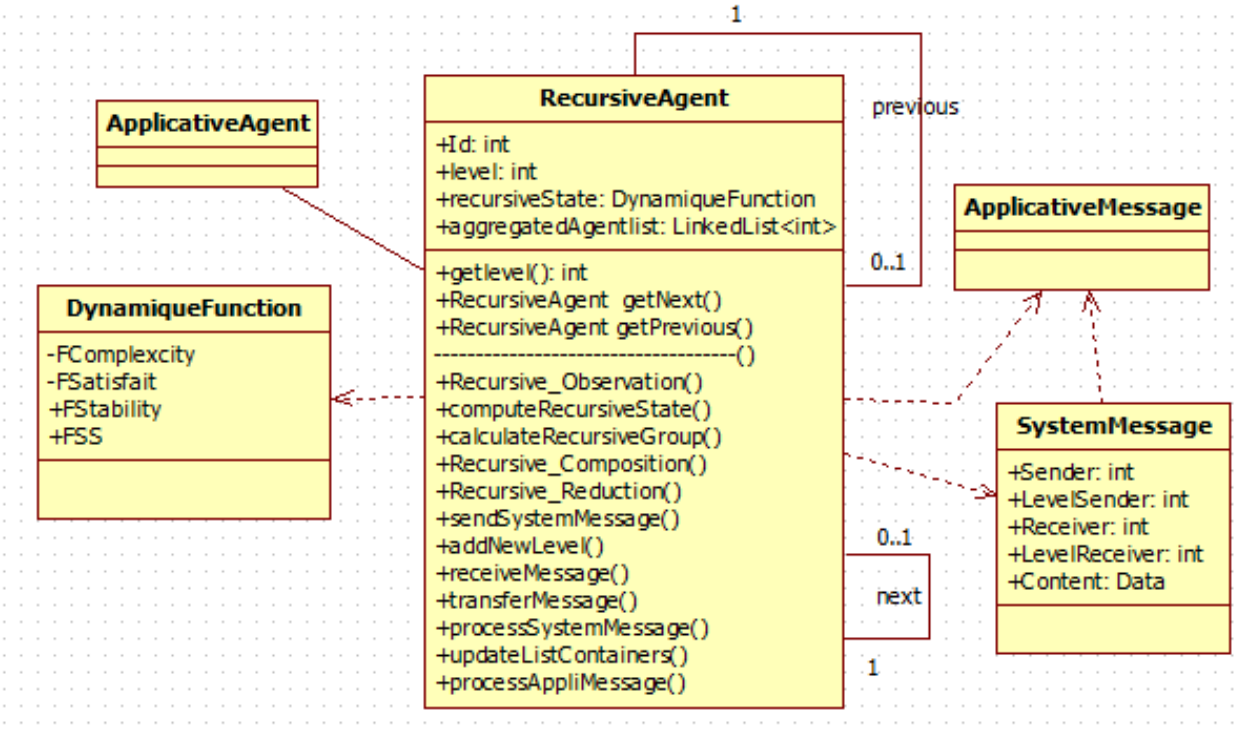


FIGURE 5.13 – Diagramme de la classes *RecursiveAgent*

naturellement la méthode *computeRecursiveState()* pour calculer les *fonctions dynamiques* et à partir de cela, elle évalue les états de récursion. Suivant les états de récursion, les méthode *recursive_Composition* ou *recursive_Reduction* sont appelées,

– les parties *C* et *R* sont évidement implémentées dans les méthodes *recursive_Composition* et *recursive_Reduction*,

– la partie *IR* est assurée par les méthodes : *sendMessage()*, *receiveMessage()*, *transfertMessage()*, *processSystemMessage()* et *processApplicationMessage()*. Les algorithmes de réception et d'émission des messages présentés dans la section 5.1.3 sont implémentés dans la méthode *processSystemMessage()*.

5.2.2 Le comportement du niveau

Les diagrammes d'états proposés dans la Figure 5.14 et dans la Figure 5.15 détaillent la dynamique de récursion. Au début, il y a trois processus réalisés en parallèle :

1. Le premier calcule régulièrement les fonctions dynamiques et les états de récursion ;
2. Le second observe les états de récursion et les fonctions dynamiques d'agents et déclenche les deux fonctions *Réduction* et *Composition*.

   Lorsque la fonction *Composition* est déclenchée (voir la Figure 5.14) :

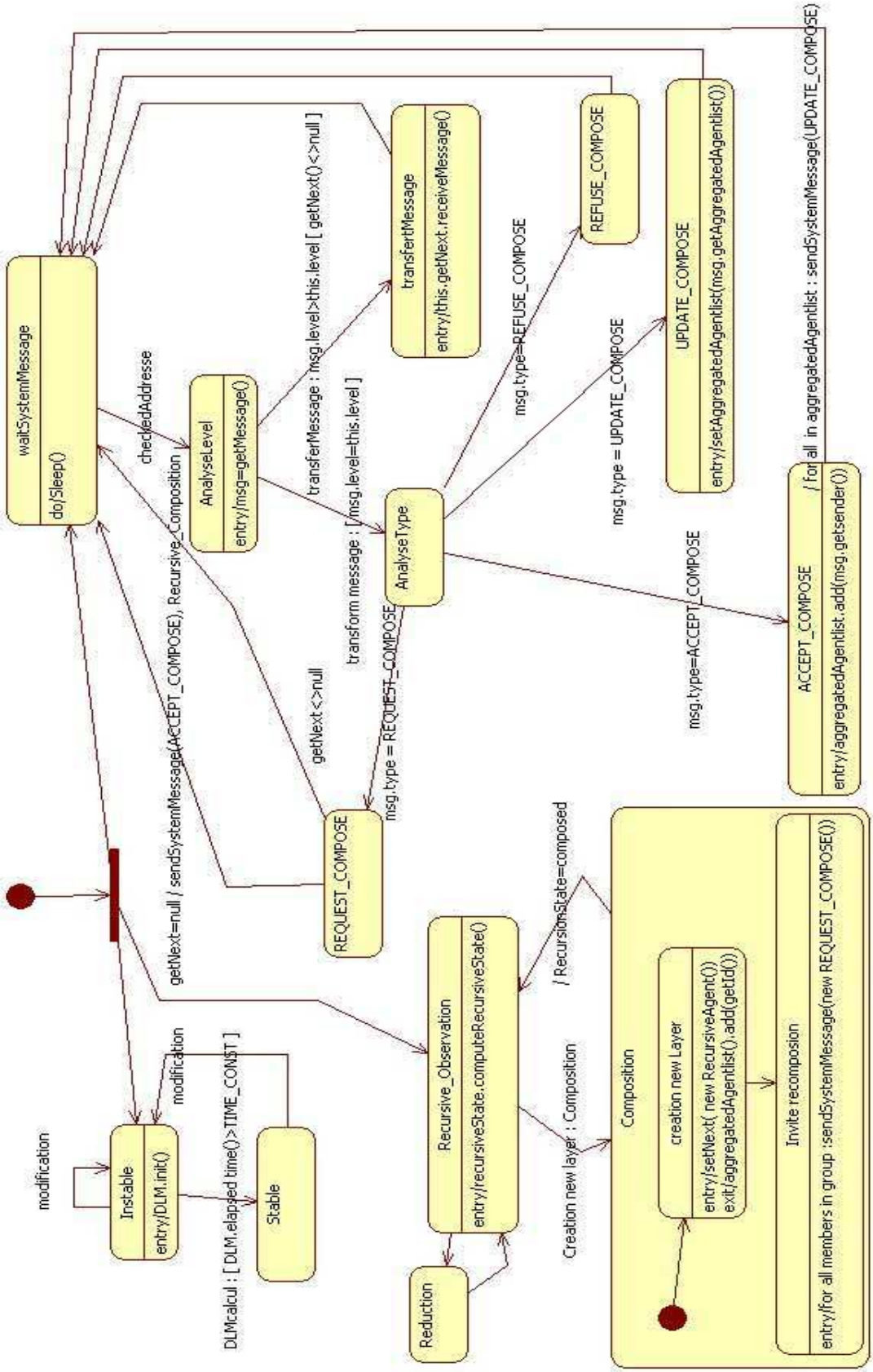


FIGURE 5.14 – Diagramme d'états de la composition d'un *Agent récursif*

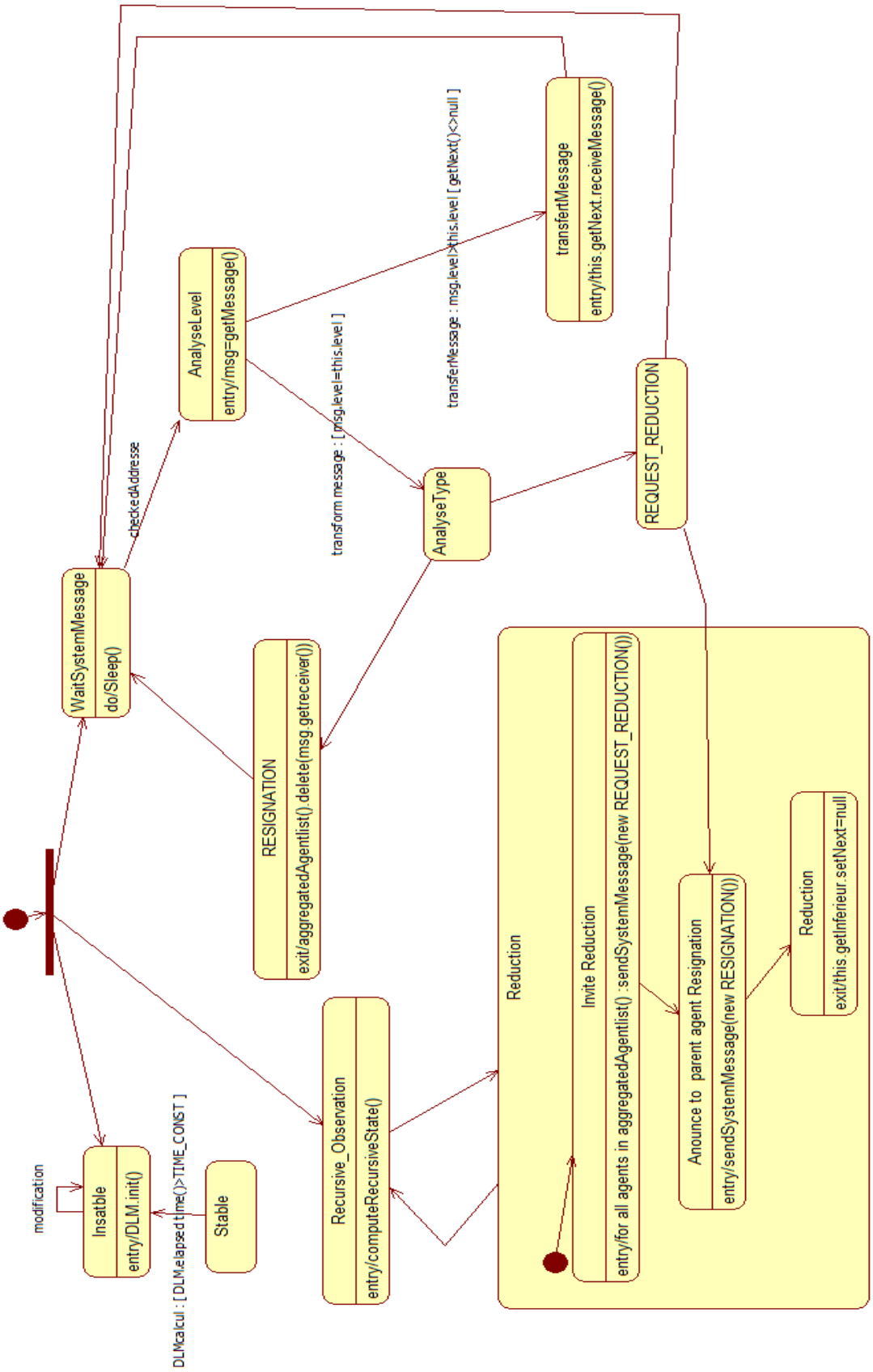


FIGURE 5.15 – Diagramme d'états de la réduction d'un *Agent récursif*

(a) L'agent crée son agent père (dans le niveau supérieur). Ensuite, l'agent lui-même s'ajoute à la liste des agents agrégés ;

(b) L'agent envoie un message de l'invitation de composition *REQUEST_COMPOSE* à d'autres agents dans son groupe ;

(c) Lorsqu'un agent reçoit un message *REQUEST_COMPOSE*, s'il accepte l'invitation de composition, il crée son agent père (dans la couche supérieure) avec les paramètres reçus dans le message *REQUEST_COMPOSE* et renvoie un message *ACCEPT_COMPOSE*, sinon il renvoie un message *REFUSE_COMPOSE* ;

(d) Si un agent reçoit une réponse *ACCEPT_COMPOSE*, il ajoute alors l'expéditeur du message *ACCEPT_COMPOSE* à la liste des agents agrégés et envoie à tous les agents dans la liste message *UPDATE_COMPOSE* ;

(e) Lorsqu'un agent reçoit un message *UPDATE_COMPOSE*, il doit mettre à jour l'information concernant la liste des agents agrégés et l'information de l'agent père pour que l'agent père de tous les agents agrégés soit identique.

Si la fonction *Réduction* est déclenchée (voir la Figure 5.15) :

(a) L'agent envoie un message *REQUEST_REDUCTION* à son agent père pour lui demander de se détruire ;

(b) L'agents père reçoit le message *REQUEST_REDUCTION*, s'il a un agent père, il envoie à son agent père un message *RESIGNATION* pour l'avertir son départ et se détruit lui même. S'il n'a pas d'agent père, il se détruit tout seul sans envoyer le message *RESIGNATION* ;

(c) L'agent père reçoit le message *RESIGNATION*, il supprime l'émetteur du message dans la liste d'agent agrégés.

3. Le troisième processus attend les messages provenant d'autres agents et les traite.

Le traitement des messages est décrit dans la partie *IR* de la section 5.

5.2.3 Les messages récursifs

Les agents récursifs communiquent en échangeant des messages afin de réaliser :
– la construction des niveaux d'abstraction ;
– l'insertion ou la suppression d'un agent ou un objet à n'importe quel niveau ;
– une action sur n'importe quel niveau (pour une interaction externe) ;
– l'interaction entre les niveaux.

Nous avons décidé de créer notre propre protocole d'interaction entre les agents. Les messages respectent la structure des définitions formelles du le chapitre 3 et de la section 5.1.2.2. Nous avons créé 14 types de messages présentés dans la Table 5.1.

N°	**Message**	**Commentaire**
1	APPLImsg_REQUEST	Demander aux agents fils d'envoyer un message applicatif
2	ApplicativeMessage	Encapsuler et transporter un message applicatif
3	RECURSIVE_STATE_REQUEST	Demander les états récursifs contenant les fonctions dynamiques
4	RECURSIVE_STATE_REPLY	Répondre avec les états récursifs
5	RECURSIVE_STATE_UPDATE	Envoyer la mise-à-jour des états récursifs
6	REQUEST_COMPOSE	Inviter à la Composition
7	ACCEPT_COMPOSE	Accepter l'invitation de Composition
8	UPDATE_COMPOSE	Mettre à jour la Composition quand il reçoit un message ACCEPT_COMPOSE
9	REFUSE_COMPOSE	Refuser la Composition
10	REQUEST_REDUCTION	Demander à l'agent père une réduction
11	RESIGNATION	Avertir l'agent père de sa disparition
12	ROUTE_REQUEST	Demander un chemin
13	ROUTE_REPLY	Répondre un chemin

Tableau 5.1 – Les messages récursifs

Nous interprétons les types de ces message récursifs ci-dessous :

- Le message *APPLImsg_REQUEST* est un message qui permet à un agent composé de demander à ses agents fils d'envoyer un message applicatif. Le message *APPLImsg_REQUEST* est propagé à tous les agents élémentaires de l'agent composé. Ces agents élémentaires envoient ensuite un message *ApplicativeMessage* dont la contenu est le message applicatif *msg*1.
- Le message *ApplicativeMessage* est un message récursif particulier. Il contient un message applicatif. Il est créé quand l'agent veut envoyer un message applicatif. L'agent récursif l'encapsule, le transforme et le transporte.
- Le message RECURSIVE_STATE_REQUEST est un message qui permet à un agent de demander aux agents récursifs leurs états récursifs contenant les fonctions dynamiques. Ce message est émis quand un agent composé veut connaitre les états récursifs de ses agents fils.
- Le message RECURSIVE_STATE_REPLY permet à un agent de répondre au message RECURSIVE_STATE_REQUEST. Ce message contient des états récursifs de l'agent.
- Le message RECURSIVE_STATE_UPDATE permet à l'agent d'envoyer ses états récursifs à son agent père pour avertir le changement de ses états. Ce message contient donc les états récursifs.

- Le message REQUEST_COMPOSE permet à un agent d'envoyer une invitation de composition (création une nouvelle couche) à tous les agents dans son groupe. Ce message contient les informations de composition comme l'identification et le niveau. Quand un agent reçoit ce message, selon ses états, il peut décider de se composer avec l'émetteur du message ou de ne pas se composer.
- Le message ACCEPT_COMPOSE permet à un agent de répondre au message REQUEST-_COMPOSE s'il accepte l'invitation. Le message contient son identification et son niveau.
- Le message UPDATE_COMPOSE permet à un agent d'avertir à tous les agents ayant accepté la composition de la liste des agents d'acceptation (la liste d'agents agrégés). Ce message est émis quand un agent reçoit le message ACCEPT_COMPOSE.
- Le message REFUSE_COMPOSE permet à un agent de refuser l'invitation de composition. Le message est émis quand un agent reçoit le message REQUEST_COMPOSE et il ne veut pas se composer.
- Le message REQUEST_REDUCTION permet à un agent de demander à agent père de détruire à cause des changements dans son groupe de *Composition*.
- Le message RESIGNATION permet à un agent d'avertir son agent père de sa disparition.
- Le message ROUTE_REQUEST permet à un agent élémentaire de demander un chemin. Le message contient une cible (un agent récursif) et une liste d'agents où le message est parvenu.
- Le message ROUTE_REPLY permet à un agent de répondre le message ROUTE_REQUEST.

Les figures 5.16 et 5.17 illustrent la circulation des messages du protocole, qui sont présentés dans le tableau 5.1. La Figure 5.16 présente le processus de *Composition* et de *Réduction* dans laquelle la circulation des messages REQUEST_COMPOSE, ACCEPT_COMPOSE, REFUSE_COMPOSE, REQUEST_REDUCTION et RESIGNATION sont impliqués. L'utilisation des messages APPLImsg_REQUEST, ApplicativeMessage, RECURSIVE_STATE_REQUEST, RECURSIVE_STATE_REPLY et RECURSIVE_STATE_UPDATE est présentée dans la Figure 5.17.

Le problème de la résolution d'adresse

Chaque agent composé dispose de sa propre adresse. Les agents élémentaires disposent d'une adresse physique qui est celle de l'agent applicatif. Chaque agent ne peut communiquer qu'avec les agents qui lui sont directement connectés.

Un problème interne à l'envoi de message est la résolution d'une adresse de niveau n en adresses de niveau $n-1$.

Considérons la figure 5.18. L'agent a_1^2 souhaite envoyer la donnée *bonjour* à son agent voisin (i.e. connecté a_6^2). Au niveau 1, les agents a_1^1 et a_3^1 doivent traiter cet envoi. On peut alors imaginer deux solutions :

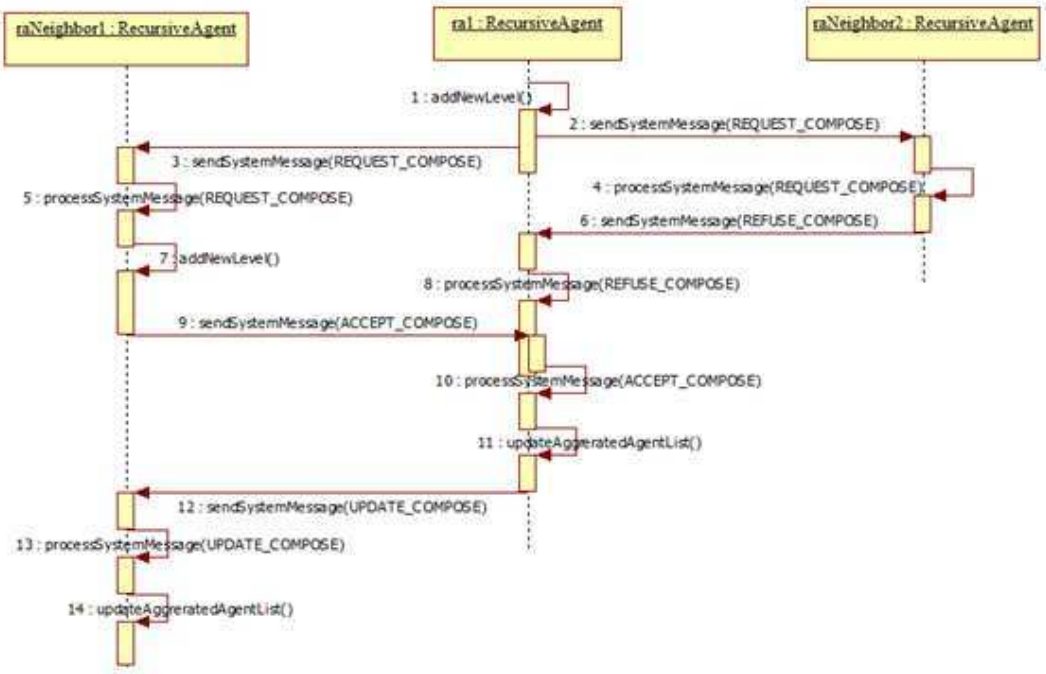


Diagramme de séquence du processus *Composition*

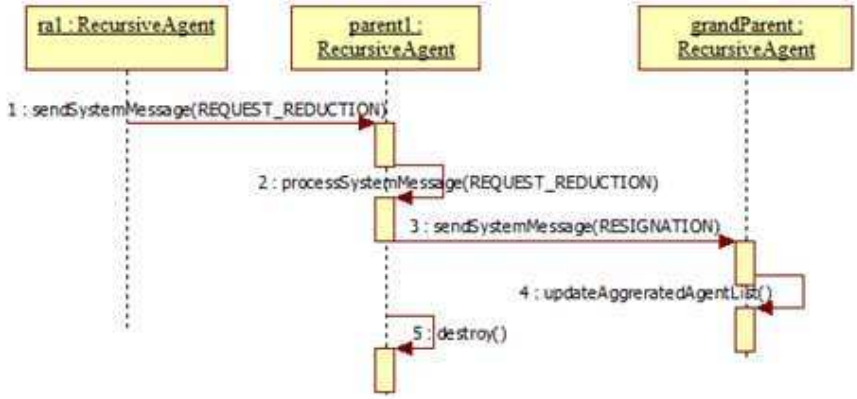


Diagramme de séquence du processus *Réduction*

FIGURE 5.16 – Diagramme des interactions pour les processus de *Composition* et de *Réduction*

– la première solution consiste à demander à tous les agents agrégés dans l'émetteur a_1^2 (ici les agents a_1^1 et a_3^1) de transmettre le message à leurs propres voisins. On remarque que si a_1^2 est voisin de a_6^2 nécessairement il existe dans le niveau inférieur un agent agrégé dans l'émetteur qui est voisin d'un agent lui même agrégé dans le destinataire (ici ce sont les agents a_3^1 et a_6^1). Le résultat de l'utilisation de cette solution est illustré en figure 5.18

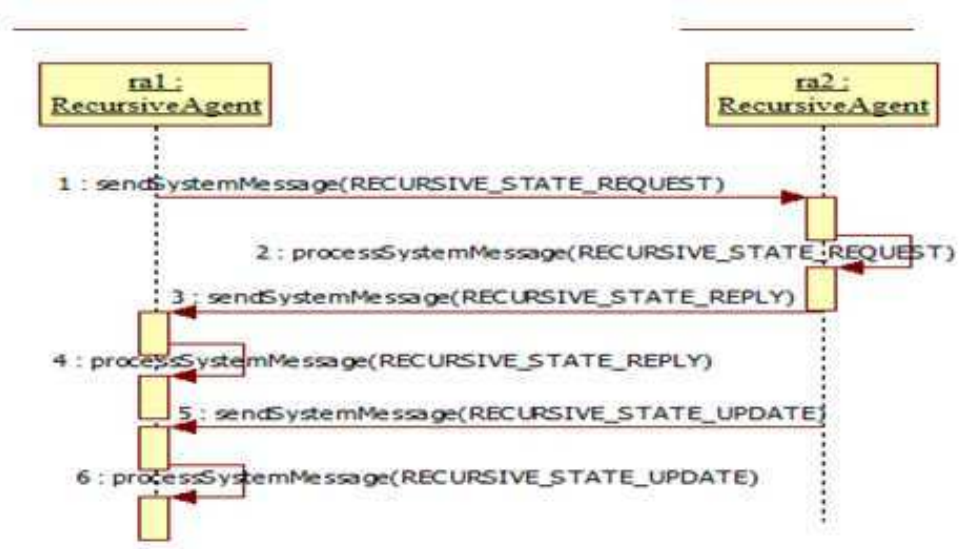


Diagramme de séquence du processus de demande des états récursifs

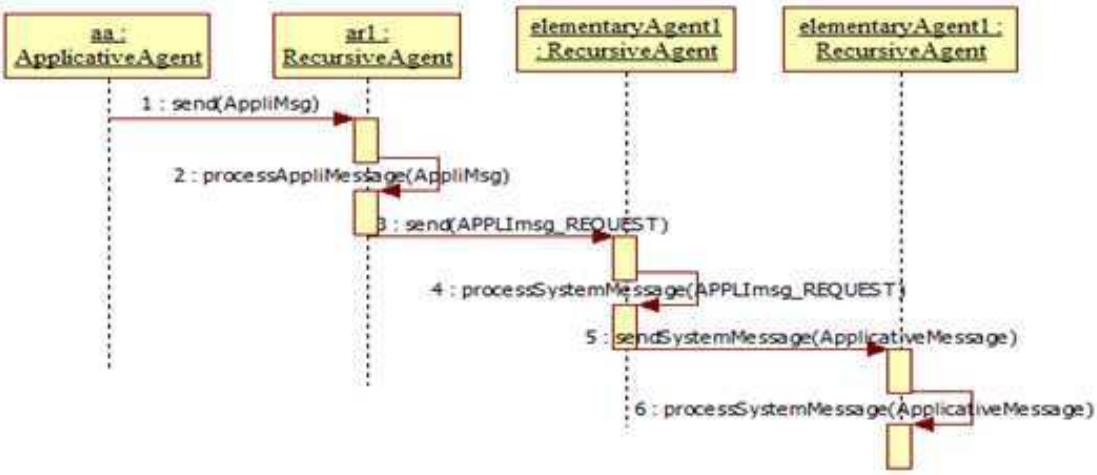


Diagramme de séquence du processus d'envoie un message applicatif

FIGURE 5.17 – Diagramme des interactions pour les messages applicatifs et les demandes d'états

– la seconde solution consiste à choisir un agent agrégé dans l'émetteur a_1^2 et à lui demander de transmettre le message. Il faut que cet agent utilise alors un protocole utilitaire qui permet de sélectionner un agent de niveau 1 agrégé dans le destinataire a_6^2 (ici a_1^1 ou à a_3^1). Il faut répéter l'opération aux niveaux inférieurs. Cette technique s'apparente à une technique de routage et son résultat est illustré en figure 5.19.

Dans l'esprit du modèle il est plus cohérent d'utiliser la première méthode car elle est purement récursive. Elle est de plus très simple à implanter par rapport à la seconde. Cette solution peut par

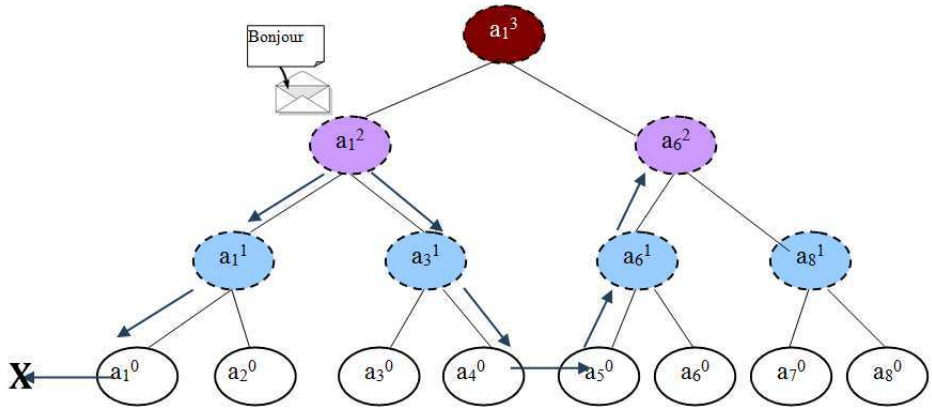


FIGURE 5.18 – Circulation des messages (solution 1)

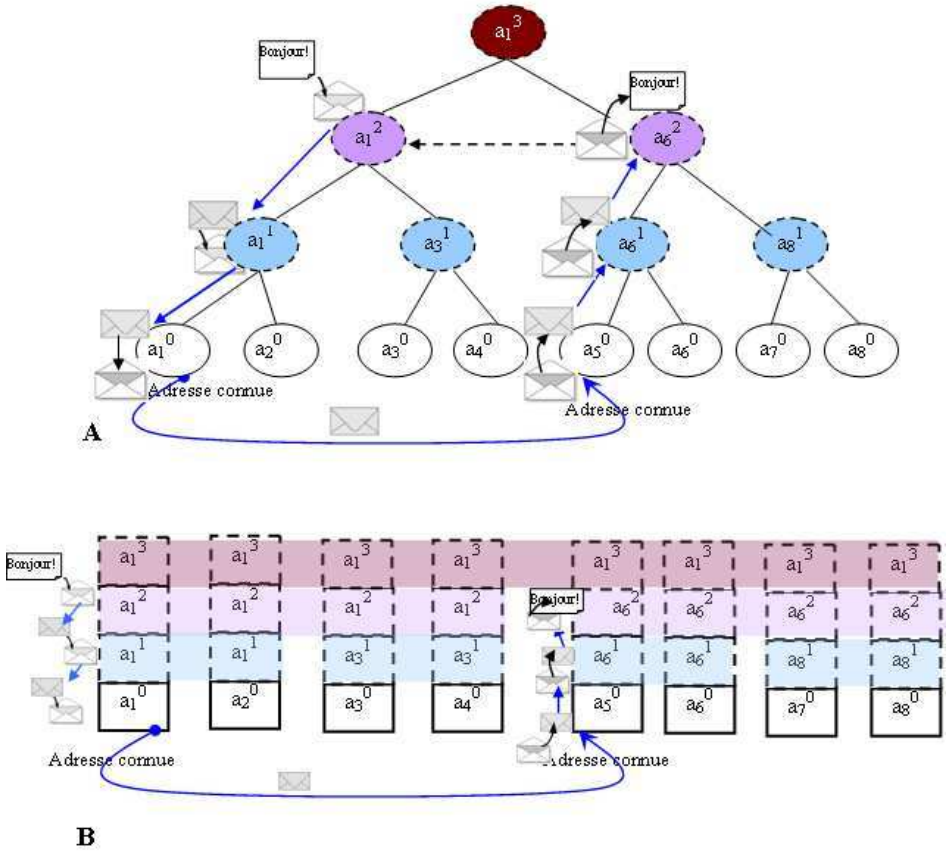


FIGURE 5.19 – Circulation des messages (solution 2)

contre paraitre très couteuse car elle repose sur un mécanisme d'inondation. Elle peut cependant être assez simplement améliorée. En effet, quand un agent agrégé n'a que des voisins qui sont eux aussi agrégés dans la même abstraction de niveau supérieur, il ignore la demande d'acheminement de message. Ainsi seuls les agents élémentaires (niveau 0) qui ne sont pas dans la même abstraction de niveau 2 envoient concrètement le message. Les agents élémentaires qui recevront ces messages, s'ils ne sont pas impliqués dans l'abstraction destinataire ignoreront les messages. On remarque que cette technique peut entrainer la duplicité de messages en réception mais ce problème, bien connu dans les réseaux, se traite simplement en utilisant dans les entêtes une numérotation de messages.

La seconde solution, avec la multiplicité des niveaux, est elle aussi couteuse car, à moins de maintenir des caches sur chaque niveau qui feront perdre en réactivité lors de modifications de voisinage, elle implique de nombreuses interactions entre des agents de niveaux différents (des mécanismes d'inondation).

5.3 Conclusion

Notre proposition vise à fournir un framework générique récursif multi-niveau, intégrable à un SMA existant. L'objectif de nombreuses approches consiste à construire un environnement (**?**) pour la conception de systèmes ou pour la simulation (**?**).

Contrairement à eux, notre objectif est de développer un framework enfichable sur un système à grande échelle décentralisé. Une façon de traiter la récursivité pourrait consister à utiliser la réflexion. La réflexion vise à décrire un processus qui est capable de se représenter et de raisonner sur lui-même. Dans notre modèle, chaque couche assure cette tâche de réflexion.

Le rôle du module d'observation est de détecter l'état d'un agent à composer ou à réduire. Dans ce sens, certaines de nos notions peuvent être comparés à l'utilisation de représentant dans le mécanisme de réflexion AGR (**?**) basé sur l'organisation. GEAMAS (**?**) utilise de la même façon des mécanismes d'observation, décomposition, recomposition basés sur des propriétés récurrentes. Cependant, le nombre de niveaux de la récursion est fixé à trois par les concepteurs de l'architecture. GEAMAS n'exploite donc pas en totalité le potentiel récursif qu'il introduit qui devrait conduire l'architecture à adapter dynamiquement son nombre de niveaux en fonction de la nature du problème.

Réalisant une auto-organisation holonique de SMA, le mécanisme basé sur une boucle de rétroaction de contrôle proposé par Tianfield (**?**) poursuit les mêmes objectifs que notre travail, mais d'une façon centralisée. Un des défis est la synthèse à chaque niveau de la décision d'un ensemble organisé d'agents et non pas d'un seul agent ou d'un observateur extérieur. Notre modèle n'a pas besoin de stocker des hiérarchies complètes comme dans (**?**) où chaque système d'exécution maintient une hiérarchie d'agents qui est mise en œuvre sous forme d'arborescence où chaque noeud contient un nom, un programme d'agent, et les références des de ses agents fils. A chaque niveau, nous stockons uniquement des informations sur l'agent parent et les agents fils. En outre, les holons peuvent participer à plusieurs hiérarchies en même temps. Notre choix est de proposer un middleware totalement décentralisé. La plupart des middleware multi-agents génériques proposent des capacités de communication entre les agents. Certaines d'entre elles intègrent les fonctionnalités organisationnelles, très peu adressent des capacités de récursion.

Relativement à ces contributions, notre travail peut être considéré comme un middleware décentralisé. Le framework empile sur chaque agent pour chaque niveau d'abstraction une couche en possession d'informations organisationnelles locales sur le niveau et de décision primitives récursives. Le framework encapsule les agents applicatifs et peut donc être totalement indépendant de l'application.

Dans le chapitre suivant, nous allons appliquer notre modèle SMA-R et son architecture à un système de supervision de réseau de capteurs sans fils.

Troisième partie

Application et validation

Chapitre 6

Application à la supervision d'un système réseau de capteurs sans fil

Ce chapitre est consacré à une application du framework SMA-R à la supervision d'un réseau de capteurs sans fil. Dans une première section nous présentons l'application réalisée dans le cadre du projet EnvSys **Env**ironment **Sys**tem) [1]. Nous décrivons le modèle MWAC (Multi-Wireless-Agent Communications) (**?**) qui propose une décomposition AEIO du réseau de capteurs destinée à maintenir l'intégrité fonctionnelle du système et à gérer la communication entre les agents-capteurs en économisant l'énergie. Dans la seconde partie du chapitre, nous détaillons l'application de SMA-R à ce système multi-agent.

6.1 Le réseau de capteurs sans fil

6.1.1 Le projet EnvSys

Ce projet a pour but l'instrumentation sans fil de réseaux hydrographiques souterrains. Ce système permet de collecter des données afin d'assurer une gestion des risques. Les paramètres importants à mesurer dans un réseau hydrographique souterrain sont nombreux ((**?**)) : température de l'air et de l'eau, pression de l'air et éventuellement de l'eau pour les réseaux noyés, degré de pollution par certains polluants classiques, débit d'eau, vitesse du courant d'air, etc.

L'objectif du système est de recueillir toutes ces informations et de les collecter à la sortie immédiate du réseau via une station de travail type PC. Elle pourront être ensuite traitées pour déclencher des alarmes, étudier la progression d'une pollution en fonction des divers paramètres mesurés.

La mesure des différents paramètres d'un réseau hydrographique souterrain fait face aux diffi-

1. Projet soutenu par le Fonds d'Incitation au Transfert de Technologie (FITT) de la région Rhône-Alpes

cultés suivantes :

- L'accès aux réseaux souterrains est difficile car leur structure est souvent fort chaotique. La pose des éléments de réseaux de communications est obligatoirement manuelle et difficilement modifiable,
- La propagation des ondes sous terre est rendue difficile par les propriétés des roches. Même les techniques récentes imposent d'avoir à gérer des perturbations.

Il est donc nécessaire d'étudier la faisabilité d'un réseau maillé de capteurs à partir de la couche de communication existante assurant un maintien adaptatif de l'intégrité du système et une gestion intelligente de l'énergie.

L'environnement est le réseau hydrographique souterrain. Les constituants du système sont les suivants :

- une station de collecte (type PC) : reçoit les messages des capteurs qui contiennent des données. Elle permet à l'utilisateur de gérer des données ou de prendre connaissance de la topologie du réseau.
- des capteurs : interagissent avec l'environnement pour en extraire des données. Ils doivent ensuite les faire parvenir à la station de collecte. Chaque capteur est situé mais ne peut percevoir sa position géographique, la localisation absolue n'est pas envisageable dans ce contexte. Le capteur peut cependant percevoir les localisations relatives de sess voisins.

Une illustration du réseau sans fil est présentée dans la figure 6.1 (page 96).

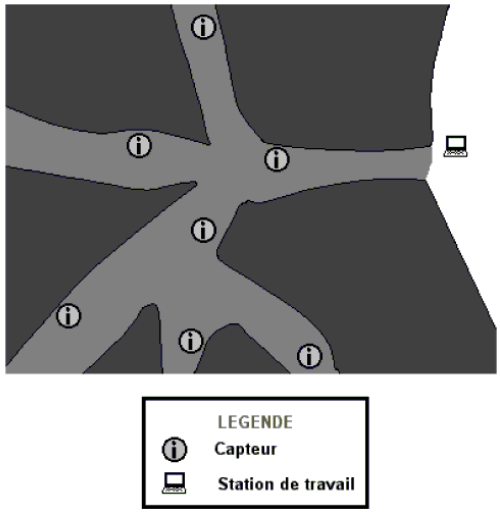


FIGURE 6.1 – Le sous-terrain instrumenté (**?**)

Le problème majeur d'un capteur réside dans le fait qu'il ne dispose que d'une quantité d'énergie limitée. Cette quantité d'énergie doit lui permettre d'effectuer des mesures et d'acheminer les messages au destinataire. Pour ce faire, il sollicite l'aide d'autres capteurs et doit lui-même aider les autres capteurs à faire parvenir leurs données à la station de collecte. Il doit choisir s'il est plus

important pour satisfaire l'objectif global, qu'il aide les autres ou se concentre sur ses mesures.

Les capteurs devront donc assurer deux fonctions principales :

– une fonction mesure : c'est le rôle premier d'un capteur. Les capteurs interagissent avec l'environnement pour en extraire des données ;
– une fonction relais : chaque capteur a une portée de transmission limitée. Les capteurs doivent donc aider les autres capteurs à transférer les données vers la station de collecte.

Généralement les éléments du réseau sont indépendants du point de vue de leur énergie et ne peuvent être rechargés de façon simple comme dans les réseaux cellulaires. Cette contrainte forte doit être prise en compte à tous les niveaux de la conception et de la réalisation. Chaque capteur prend en charge sa mission essentielle de mesure. La deuxième fonction (fonction relais) est utilisée pour résoudre le problème de gestion d'énergie. (**?**) détermine deux problèmes :

– Comment router au mieux les messages ? Chacun des capteurs ne peut physiquement pas communiquer avec la station de collecte des informations. Afin de maîtriser la dépense énergétique, lesquels devront prendre la décision de relayer l'information ?
– Quelle intelligence donner au réseau pour gérer au mieux cet environnement complexe ?

Depuis plusieurs années, il existe de plus en plus de mise en commun des deux domaines de recherche que sont les systèmes multiagents et les réseaux de capteurs sans fil. La nature distribuée et ouverte des réseaux de capteurs rend l'approche multi-agent particulièrement adaptée (**?**). (**?**) a aussi identifié plusieurs domaines d'applications des systèmes multi-agents dans le contexte des réseaux sans fil : le routage de l'information, la fusion de données, la recherche et la description de services. J.P. Jamont et M.Occello dans l'article (**?**) ont montré l'intéret de l'application des SMA pour résoudre les problèmes des réseaux de capteurs sans fil. Ils cherchent dans ce travail à répondre aux questions de routage des messages et d'application intelligentes aux réseaux de capteurs sans fil. La décomposition AEIO qu'ils proposent pour un SMA réseau de capteurs sans fil est détaillée dans la section qui suit.

6.1.2 La décomposition AEIO du réseau de capteurs sans fil

6.1.2.1 Le E (Environnement)

L'axe environnement regroupe les informations permettant de qualifier l'espace commun aux agents du système. Dans le contexte du projet *EnvSys*, l'environnement est fait de toutes les informations mesurables par les agents ou des objets qui empêchent les agents de communiquer.

6.1.2.2 Le O (Organisation)

Dans le projet *EnvSys*, l'organisation des agents est une structure hiérarchique, semblable à celle que l'on trouve dans les clusters utilisés aujourd'hui dans le réseau de capteurs (**?**). L'objectif est

de localiser au mieux l'inondation et ainsi obtenir un gain conséquent en énergie. La localisation d'un agent consiste à déterminer le groupe auquel appartient l'agent. Dans ce type d'application, l'organisation a priori n'est pas contrôlée : la relation entre les agents émerge de leurs interactions et de l'évolution de leurs états.

Dans le cadre du projet *EnvSys*, les capteurs sont autonomes en énergie. La durée de la vie des capteurs sans fil dépend de leur énergie. Leur énergie est consommée chaque fois qu'ils réalisent une tâche de mesure ou de communication. La consommation est importante et parfois inutile lorsqu'ils relayent inefficacement un message. Une des questions importantes du projet *EnvSys* est de diminuer le relais inutile de messages pour épargner au mieux l'énergie. Pour résoudre un tel problème, le modèle *MWAC* (Multi-Wireless-Agent Communications) pour la gestion des communications est proposé par Jamont (**?**).

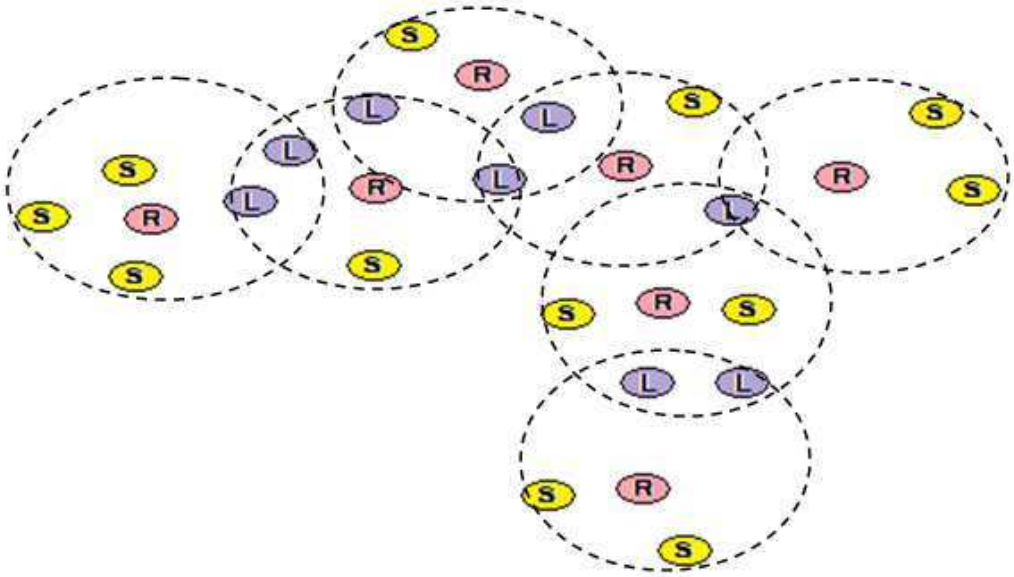


FIGURE 6.2 – L'organisation MWAC

Dans MWAC, l'auteur définit trois rôles d'agent différents Représentant (R), Liaison (L) et Simple Membre (S) :

- Les agents représentants (*R*) se trouvent au centre du groupe jouent un rôle d'administration. Ils gèrent la communication dans leur groupe et renvoient les données des membres du groupe si nécessaire,
- Les agents liaisons (*L*) se situent aux frontières de deux groupes (ou plusieurs groupes). Ils appartiennent donc à plusieurs groupes. Ils aident les autres agents de leur groupe à transférer les messages à la demande des agents (*R*),
- Les agents simple membre (*S*) ne jouent qu'un rôle de mesure. Ils peuvent également envoyer, recevoir et traiter les messages qui leurs sont destinés. Ils ne participent pas au relais des

messages.

Chaque groupe de l'organisation MWAC est composé de :

- un agent *R* ;
- un ou plusieurs agents *L* ;
- aucun ou plusieurs agents *S* (voir la Figure 6.2, page 98).

Les agents fonctionnent tous sur le même schéma mais leur rôle influe sur leurs comportements spécifiques face aux messages qui viennent des autres agents.

Le rôle des capteurs est modifié selon leur état (énergie, position, capacité de communication) suite à l'action de l'environnement et du temps. L'organisation évolue dynamiquement. Le chemin de routage local est modifié en même temps que l'organisation.

6.1.2.3 Le A (Agents)

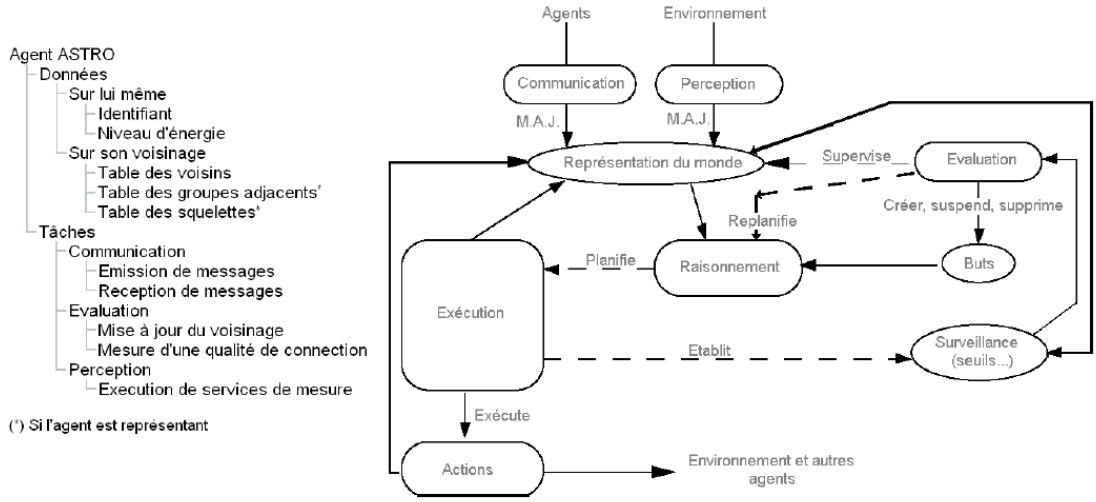


FIGURE 6.3 – Arbre des données/tâches et architecture d'un agent basé sur ASTRO (**?**)

Dans EnvSys, les capteurs sont représentés par des agents. Ils sont conçus à l'aide d'une extension de l'architecture hybride ASTRO ((**?**)) baptisé e-ASTRO (embedded ASTRO) (voir figure 6.3).

Les agents sont munis de deux fonctions : la gestion de la communication et la réalisation des tâches applicatives.

- La fonction de communication est celle de transmettre, si nécessaire, l'information à d'autres agents ou tout simplement de relayer des messages pour le compte d'autres agents. Tous les agents ont des capacités de communication mais leur comportement face à un message dépend de leur rôle (soit *R*, *L* ou *S*) dans l'organisation ;
- Les tâches spécifiques consistent en la mesure de différents paramètres de l'environnement, enregistrement de données etc. En dehors des données sur soi comme son identifiant, son

statut dans le groupe (son rôle) et sa table de voisins, les agents enregistrent les données sur les autres. Les données enregistrées sur les autres sont liées à leur rôle dans le groupe :

1. Un agent représentant (*R*) possède en plus une liste des groupes adjacents qui lui permet de connaître l'identifiant des groupes voisins. Il possède aussi une table d'optimisation qui lui permet de mémoriser un certain nombre (3) de chemins utilisés fréquemment. Cette route est en fait la liste des identifiants des représentants des groupes traversés pour joindre le destinataire ;
2. Un agent simple membre (*S*) connait le rôle et le groupe d'appartenance de ses voisins car il doit probalement remettre son rôle en cause l'organisation établie ;
3. Un agent liaison (*L*) connait les groupes voisins grâce à l'identifiant des représentants qu'il voit.

6.1.2.4 Le I (Interactions)

Dans EnvSys, les agents interagissent seulement avec les agents en accointance avec eux. Un agent est en accointance avec un autre si et seulement s'il est conscient de son existence c'est-à dire présent dans son aire d'émission/réception (**?**). Ils interagissent via l'envoi de messages. Il existe 13 types de messages différents (**?**) :

1. Le message QuiSontMesVoisins est un message qui permet à un agent de demander à ses voisins de se faire connaître. Ce message est émis quand un agent est créé (le premier but d'un nouvel agent est de connaître ses voisins). Ensuite, ce message ne sera utilisé que lorsque l'agent aura l'impression (qu'elle soit justifiée ou non) que sa table des voisins ne coïncide plus avec la réalité (par exemple s'il émet un message et qu'il ne reçoit pas d'ACK, depuis un moment qu'il juge trop long, il n'entend pas de communication autour de lui...).
2. Le message JeSuisUnDeTesVoisins permet à un agent de répondre à la requête précédente. Avec ce message, il fournit son identifiant, son rôle et son groupe d'appartenance.
3. Le message JeChangeRôle est utilisé par tout agent qui a changé le rôle pour en avertir ses voisins. Ce message contient comme information l'identifiant de l'agent, son rôle et son groupe d'appartenance.
4. Le message DemandeGrpAgentLiaison est utilisé par les représentants qui veulent remettre à jour leur connaissance sur les groupes voisins. Elle oblige les agents de liaison à répondre.
5. Le message ReponseGrpAgentLiaison est utilisé par un agent de liaison pour signaler à un représentant la liste des groupes que cet agent peut contacter.
6. Le message VerifieCoherenceGroupeVoisin est envoyé par un agent qui croit avoir détecté une incohérence avec un groupe voisin. Il y a l'incohérence entre deux groupes quand deux

simples membres de groupes différents se voient et que leur représentant ne peuvent pas communiquer.

7. Le message ResolutionConflitRepresentant est un message utilisé par un représentant en conflit avec un ou plusieurs pour communiquer son score.
8. Le message PasseRepresentant est un ordre donné par un représentant à un des agents de son groupe. Il peut donner cet ordre s'il s'aperçoit par exemple qu'un groupe voisin est isolé et que la seule manière de briser cet isolement est qu'un agent particulier devienne représentant.
9. Le message ChercheRoute est utilisé par un représentant qui veut connaître la route à prendre (succession de couples (agent de liaison, représentant)) pour joindre un individu.
10. Le message ResultatChercherRoute est la réponse du représentant qui à le destinataire à sa charge.
11. Le message ACKMessage est un message de configuration qui confirme à l'émetteur que son message est parvenu à destination. Il intervient donc dans la gestion de la cohérence de l'organisation.
12. Le message DonneesEncapsulees est un message qui encapsule des données.
13. Le message RouteErronee est un message envoyé par un représentant qui s'est aperçu d'un problème. Ce message porte la route erronée.

6.2 SMA récursif pour le réseau de capteurs sans fil

La problématique du réseau est que chaque capteur peut être en panne, ne plus avoir l'énergie ou se déplacer à cause du roulement de l'eau... Ce contexte complique des tâches simples comme l'envoi d'un message à certains capteurs pour obtenir des informations ou la recherche d'un chemin vers un endroit précis. Un nombre de capteurs important renforce d'autant les difficultés. Pour les simplifier, nous proposons d'utiliser le modèle SMA récursif pour l'observation et la supervision du système.

6.2.1 SMA-R appliqué à EnvSys

Les cinq parties du modèle SMA-Récursif vont nous permettre d'exprimer les propriétés *PA*, *PE*, *PI* et *PO*. A travers des parties K, R, C, nous allons exprimer la propriété récurrente *PA*. La partie *IR* nous permet d'écrire la propriété récurrente *PI*.
Pour l'application au réseau de capteurs sans fils, nous avons décidé de prendre l'organisation MWAC comme base pour la règle de génération d'une abstraction de niveau supérieur. Une fois l'organisation MWAC stabilisée, les groupes MWAC formés permettent de générer un agent de niveau supérieur représentant le groupe.

Les cinq parties de l'agent récursif de la Table 4.1 appliquées au modèle MWAC sont détaillées comme suit :

– **(*K*)** : contient les attributs et fonctions de l'agent MWAC, les fonctions dynamiques, les états de récursion, les conditions de *Composition* et *Réduction* :

1. les fonctions dynamiques sont explicitées dans les algorithmes 4, 5, 6 et 7.

Algorithme 4 *$F_{Complexité}$: Fonction de complexité*

1: $F_{Complexité}$.init(0)
2: *Si son rôle est Représentant (R) et*
3: *Si dans son groupe, il y a un agent Liason (L), alors $F_{Complexité}$=1* ;
4: *Si $F_{Complexité}$==1 mais il n'y a qu'un seul agent, alors $F_{Complexité}$=0.*

Algorithme 5 F_{SA} : Fonction de *stabilité*

1: F_{SA}.init(0) ;
2: *Dans un période de sûreté, s'il y a des changements de rôle, de la liste des voisins, des changement de son groupe ;*
3: *Alors, l'agent est instable F_{SA}= 0 ;*
4: *S'il n'y a aucun changement, l'agent est stable, alors F_{SA}=1.*

Algorithme 6 $F_{Satisfaction}$: Fonction de *satisfaction* par rapport à son objectif

1: $F_{Satisfaction}$.init(1)
2: *Dans un temps d'attente, le désir n'est pas satisfait, alors $F_{Satisfaction}$=0, sinon $F_{Satisfaction}$=1.*

Algorithme 7 F_{SS} : La fonction de *stabilité* de la société pour les agents composés

1: F_{SS} = Max(F_{SA1} de tous les agent dans la liste d'agents agrégés).

2. La condition de *Composition* est explicitée par l'algorithme 8.

Algorithme 8 La condition de *Composition*

1: *($F_{Stabilité}$ ==1) et ($F_{Complexité}$ ==1)*
2: *et ($F_{Satisfaction}$==0)*
3: *et (getNext()==null)*

3. La condition de *Réduction* est explicitée spar l'algorithme 9).

Algorithme 9 La condition de *Réduction*

1: *($F_{Stabilité}$ ==1) et ($F_{Complexité}$ ==0)*
2: *et (getPrevious() !=null)*

– **(O)** : Si les conditions *Composition* (resp. *Réduction*) sont satisfaites, le mécanisme *Composition* (resp. *Réduction*) est lancé.

– **(C)** : Crée une nouvelle instance de la classe *RecursiveAgent* et l'assigne à *next* pour construire une nouvelle couche. La propriété récurrente *PA* définit la liste *aggregatedAgentList*.
– **(R)** : Envoye un message pour demander à son agent père de se réduire.
– **(IR)** : Gère deux types d'interactions :
  – Interaction système : Les messages (REQUEST_COMPOSE, ACCEPT_COMPOSE, REFUSE_COMPOSE, UPDATE_COMPOSE,... voir la Table 5.1, page 87) ;
  – Interaction applicative : les 13 messages de MWAC.

  La partie *IR* nous permet d'écrire la propriété récurrente *PI* :
  – Pour les messages MWAC dont le destinataire est "tout le monde" : le message va être distribué à tous les agents élémentaires qui se situent sur les feuilles dont la racine est l'agent émetteur du message. Les agents élémentaires vont envoyer le message de l'émetteur. L'algorithme 10 détaille ce traitement.

Algorithme 10 Méthode permettant d'envoyer une trame

```
public void sendFrame(Frame frm)
{
  if(frm.getLevel()==0)
      return this.getCommInterface(CommInterface.DEFAULT_INTERFACE)
    .send(frm.toByteArray()) ;
  else
  {
     LinkedList<RecursiveAgent> aggregatedAgent=this.getAgreggratedAgents() ;
     for(int i=0 ;i<this.getAgreggratedAgents() ;i++)
      aggregatedAgent.sendFrame(
            new Frame(
                  (int[])aggregatedAgent.toArray() /* senders */ ,
               ( (frm.getReceiver()==Frame.BROADCAST) ? Frame.BROADCAST :
 ( int[]) this.adresseResolution(frm.getReceiver()).toArray()) /* receivers */,
                  frm /* content */)) ;
  }
}
```

  – Pour les messages exprimant une requête, il est nécessaire d'expliciter la méthode de fusion des informations. Par exemple, pour la requête "demander une mesure de température à l'environnement", la température mesurée d'un agent composé est définie par la moyenne arithmétique des températures captées des agents dans la liste *aggregatedAgentList*. L'algorithme 11 détaille ce traitement.

L'architecture SMA-R appliquée à MWAC

L'application de notre modèle SMA-R est illustrée dans la Figure 6.4, page 104. Chaque agent MWAC est encapsulé par un agent élémentaire. Les capacités sur les niveaux d'abstraction néces-

Algorithme 11 Méthode permettant d'acquérir une mesure

```
public double getTemperature()
{
  if(this.level==0)
      return this.getSensor(Sensor.TEMPERATURE).getMesure() ;
  else
  {
      LinkedList<RecursiveAgent> aggregatedAgent=this.getAgreggratedAgents() ;
      for(int i=0 ;i<this.getAgreggratedAgents() ;i++)
          moyenne+=aggregatedAgent.get(i)/this.getAgreggratedAgents.size() ;
  return moyenne ;
  }
}
```

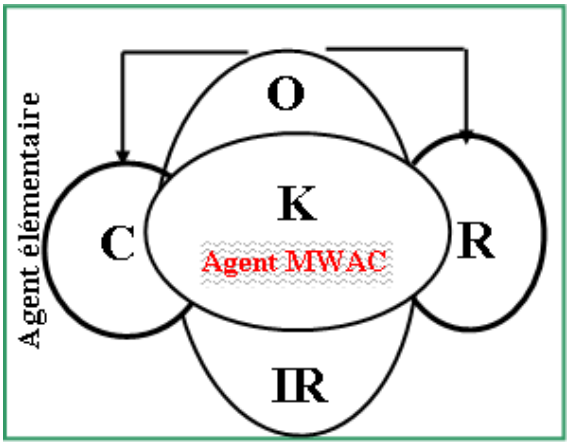


FIGURE 6.4 – L'agent MWAC est encapsulé par un agent élémentaire

saires MWAC sont localisées évidement dans la partie *K*.

Le diagramme de classe MWAC Récursive se présente dans la Figure 6.6, page 105. La classe *ApplicativeAgent* est remplacée par la classe *MWACAgent*. La classe *MWACMessage* est utilisée et remplacée la classe générique *ApplicativeMessage*.

Physiquement, les agents élémentaires communiquent par le canal de communication des agents MWAC (figure 6.5).

6.3 Extension du simulateur MASH pour SMA-R

6.3.1 L'outil MASH

MASH (MultiAgent Hardware Software Simulator) (**?**) est un outil de simulation et de mise au point de systèmes multi-agents logiciels/matériels élaboré dans l'équipe. Les agents et les objets du SMA peuvent interagir avec un environnement (SimNetwork) via une couche d'abstraction (Si-

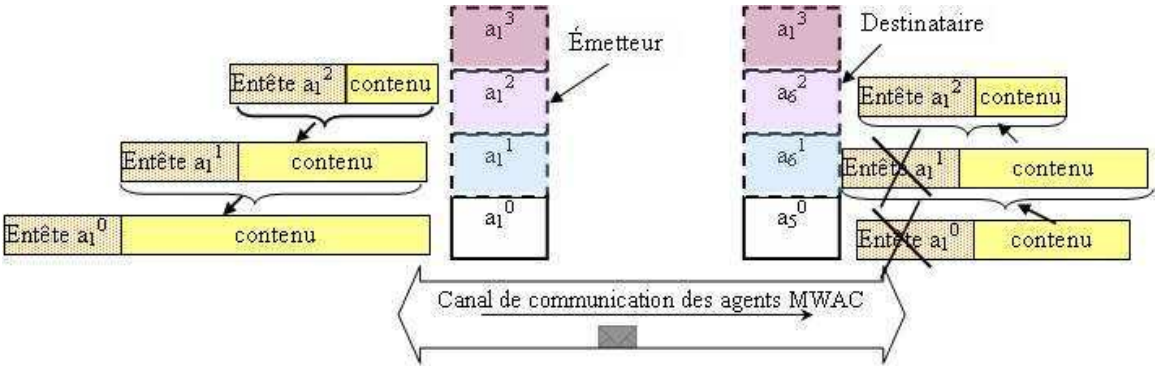


FIGURE 6.5 – La communication entre les couches récursives

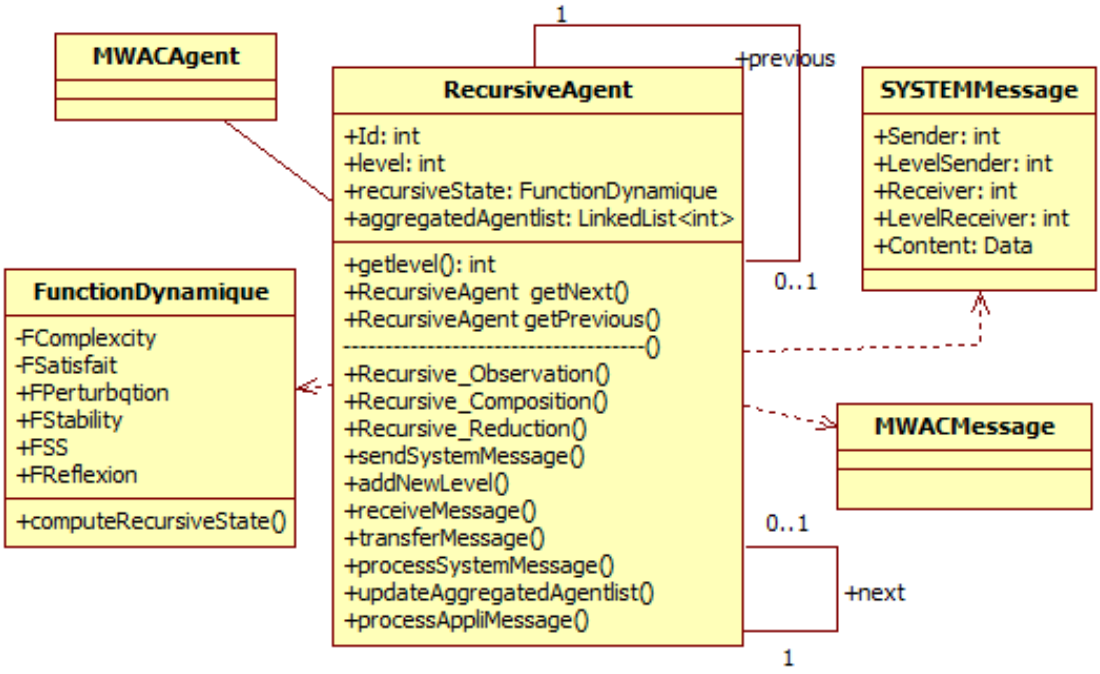


FIGURE 6.6 – Le diagramme de classe d'agent MWAC Récursif

mAgent). L'architecture de base de MASH est décrite sur la Figure 6.7. MASH permet de construire des SMA de trois façons différentes : par simulation logicielle, par simulation matérielle et par simulation hybride ou mixte.
MASH contient deux couches principales : la couche *SimNetwork* et la couche *SimAgent*. La couche SimNetwork fournit un modèle d'environnement réel et d'interaction d'agent. Un des principaux objectifs de cette couche est de trouver les voisins d'un agent qui peuvent recevoir physiquement le message transmis par l'expéditeur.

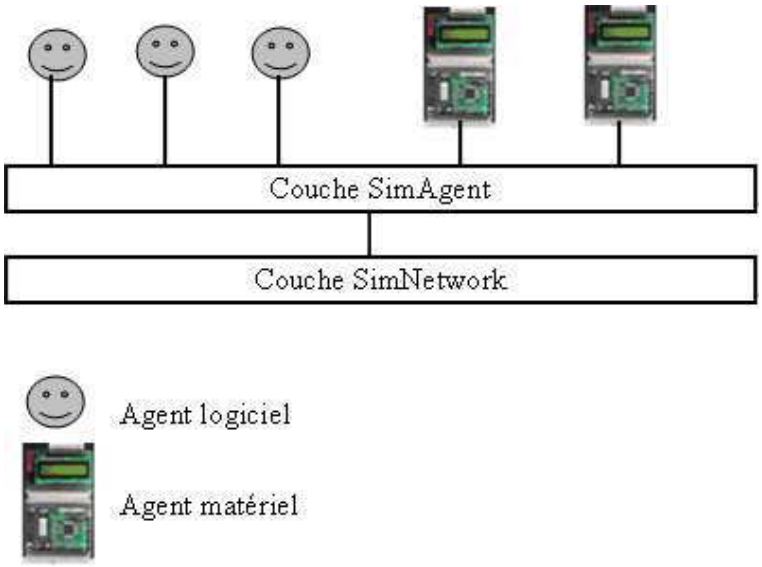


FIGURE 6.7 – L'architecture de base du simulateur MASH

La couche *SimNetwork* peut fournir des valeurs lisibles des agents et permettre de modifier l'état de l'environnement. MASH fournit aussi un outil de statistique sur les messages qui circulent dans le système ou sur les messages envoyés/reçus par un agent. La couche *SimAgent* permet la simulation des agents logiciels/matériels. Chaque agent possède son propre modèle et sa propre architecture. Un agent peut être mis en œuvre par un agent logiciel (une classe Java) ou d'un agent matériel qui est branché sur le simulateur avec un port série. Nous n'utilisons pas ici cette fonctionalité, mais elle nous permet de dire que notre framework est directement applicable à un système réel grâce à MASH

L'utilisateur doit implémenter la *partie Applicative* de l'agent en dérivant une nouvelle classe de la classe *Application* et de la classe *messagesApplication*. Pour l'application capteurs MWAC, les agents sont implémentés par une classe Java *MWACAgent*. Dans le simulateur MASH, les agents MWAC peuvent communiquer grâce à la couche *SimNetWork*. Ils utilisent un protocole (voir les messages MWAC en section 6.1.2.4). Ils décident de leur rôle, réalisent leur tâche (mesurer) et en-

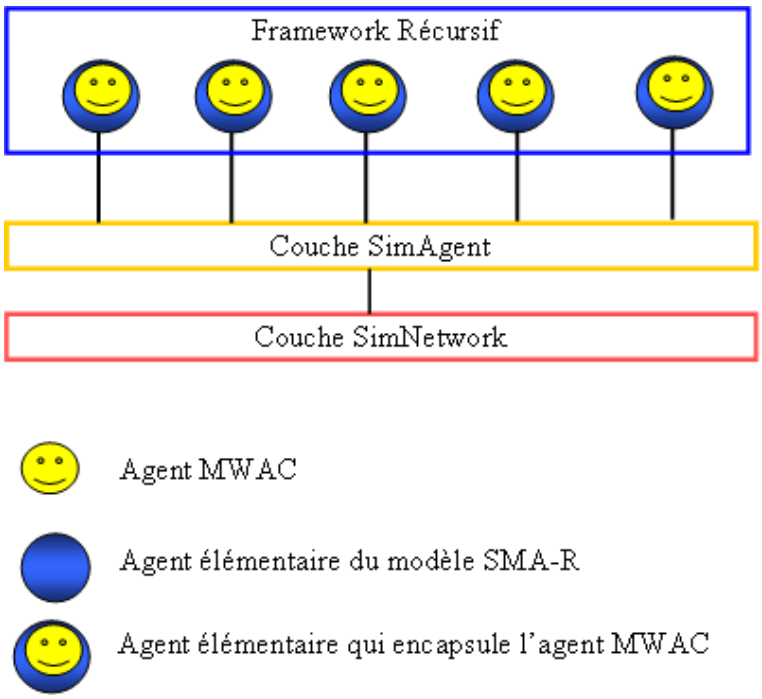


FIGURE 6.8 – Connexion à MASH des agents récursifs SMA-R et des agents applicatifs MWAC

voient les informations à la *station de collecte*.

6.3.2 L'intégration du framework SMA-R et sa visualisation

Pour expérimenter notre modèle SMA-R, nous avons appliqué ce framework au système de *capteurs sans fil MWAC*. Le framework qui implémente le modèle SMA-R et son architecture décentralisée est générique. Il permet aux agents réels de communiquer par son intermédiaire pour construire plusieurs niveaux d'observation. Les couches d'abstraction sont implémentées sur les agents du SMA réel. La structure du framework contient :

- une classe *RecursiveAgent* qui exprime le modèle SMA-R,
- une classe *RecursiveMessage* qui exprime les messages récursifs décrivant dans la partie *Interaction Récursive - IR* du modèle.

La classe *RecursiveAgent* est directement reliée à la classe *MWACAgent* pour l'encapsulation (voir la Figure 6.8). Les agents de la classe *RecursiveAgent* écoutent les messages et détectent les changements de leur agent MWAC pour décider la construction des niveaux abstraits.

Nous avons développé un outil pour visualiser l'organisation multi-niveau par l'adaptation de MASH. L'outil permet d'observer le système à un niveau choisi à l'aide d'un curseur de niveau. L'intégration du framework au MASH et au système applicatif est illustrée par le diagramme de package dans la Figure 6.9. Ce diagramme présente trois packages. Le package *MASH* représente

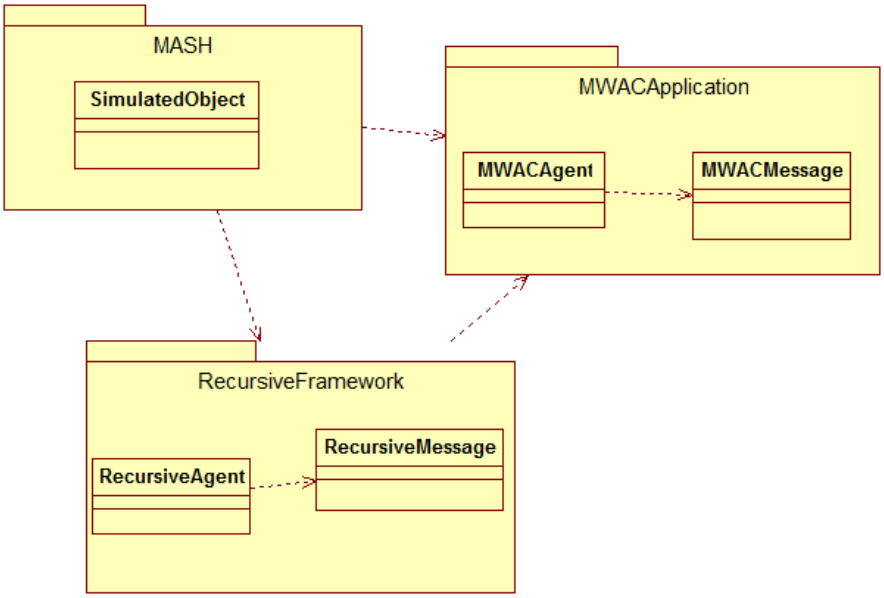


FIGURE 6.9 – L'extension du framework pour la visualisation du système de capteurs sans fil

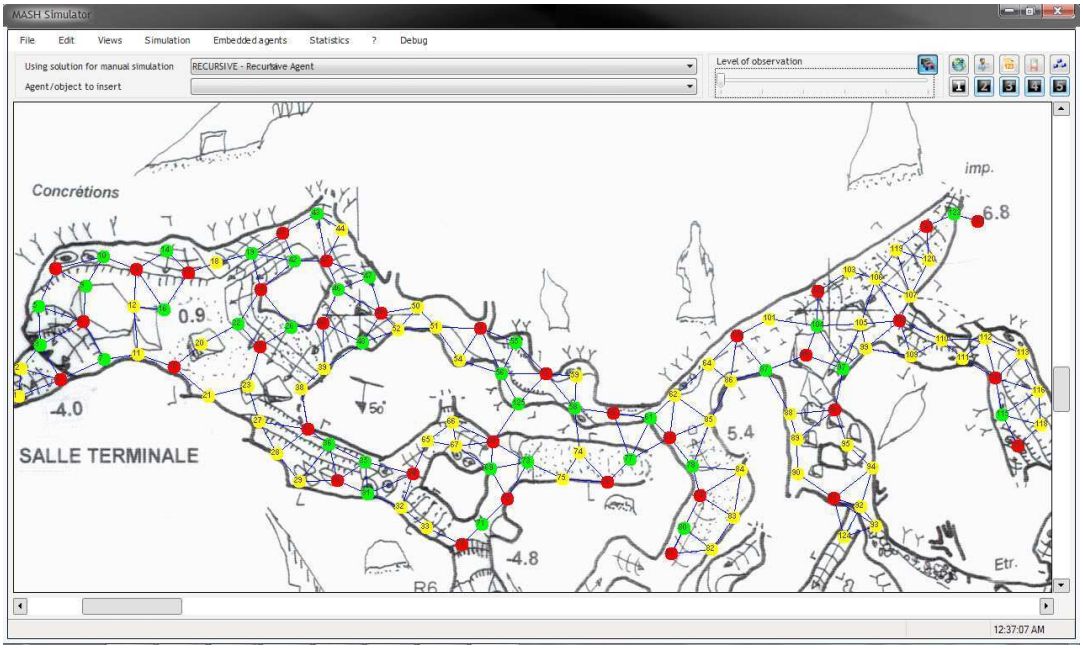


FIGURE 6.10 – La simulation de l'application réseau de capteurs MWAC récursifs dans MASH

le simulateur MASH. Le package *MWACApplication* contient des classes de l'application *MWACAgent*. Le package *FrameWorkRecursive* est un package générique permettant d'appliquer notre modèle SMA-R aux systèmes complexes. On peut voir dans la figure 6.10 un exemple de l'application au réseau de capteurs sans fil. On remarquera sur la figure 6.11 que cette interface intègre un curseur permettant de choisir son niveau de répresentation. Le premier écran présente le système MWAC au niveau 0 et le deuxième écran indique le niveau 1.

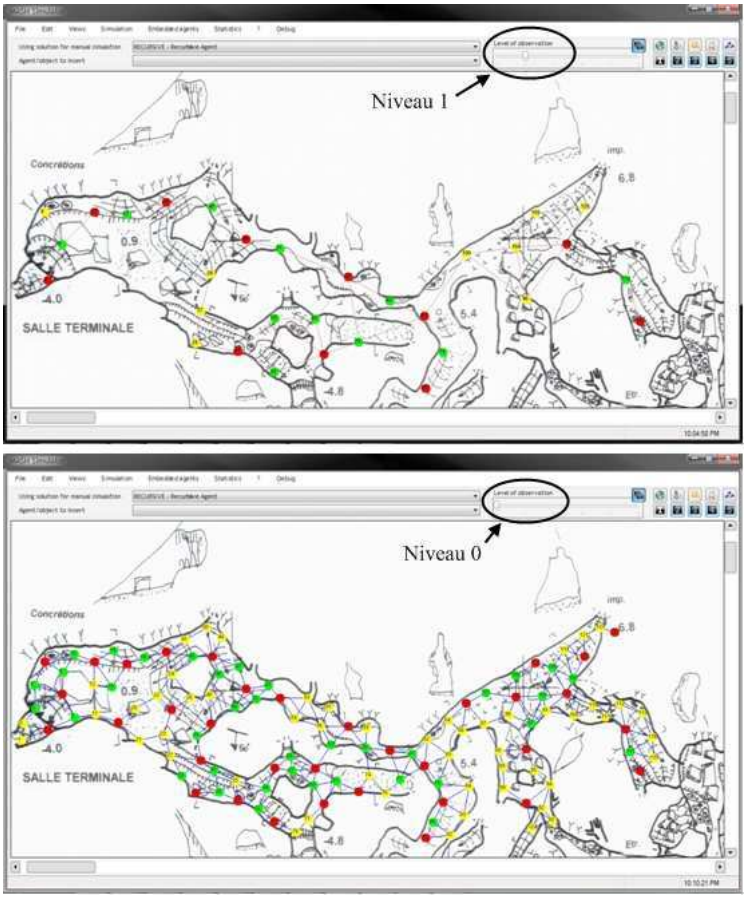


FIGURE 6.11 – MWAC Récursive sur le simulateur MASH

6.4 Conclusion

Dans ce chapitre, nous avons appliqué de notre modèle SMA-R à un système de réseau de capteurs sans fil. Nous avons présenté le système réel qui consiste en des capteurs non filaires à la fois physiques et logiciels. Les capteurs prennent en charge la mesure des paramètres de l'environnement où ils se situent et les envoyent à une station de collecte. L'agentification de ce problème a été décrite. Le SMA peut être de grande taille et il doit d'être supervisé de loin ou de près selon les

besoins du superviseur. L'observation des niveaux abstraits est importante pour focaliser et interagir globalement ou sur des zones spécifiques.

Nous avons montré comment notre approche peut être utilisée sur un cas pratique, en essayant de distinguer les parties du modèle qui sont génériques, et celles liées à l'application. Nous avons détaillé chacune de ces parties pour ce problème. Puis les architectures interne et externe de SMA-R appliqué ont été présentées. Nous avons également abordé l'implémentation distribuée du réseau envisagé comme un vrai système d'entités physiquement décentralisées et autonomes. Enfin nous avons décrit la réalisation attachée à l'outil MASH qui en permet à la fois sa visualisation multi-niveau mais aussi la possibilité d'être instancié sur des systèmes physiques.

Ce chapitre est resté volontairement descriptif. Dans le chapitre suivant, nous nous livrons à des explications détaillées du comportement du système sur des scénarios précis et nous essayons d'évaluer le coût de ce framework.

Chapitre 7

Évaluation et Validation

Ce chapitre est dédié à la présentation de l'évaluation et de la validation du modèle proposé. Nous abordons les trois axes suivants :

- une évaluation fonctionnelle (section 7.1) du modèle destinée à illustrer le comportement du framework sur des situations d'intérêt précises,
- une évaluation des performances du modèle (section 7.2) destinée à estimer le coût de mise en oeuvre du framework,
- une évaluation méthodologique du modèle (section 7.3) destinée à estimer la réutilisabilité du framework.

7.1 Évaluation fonctionnelle du modèle

Cette section est consacrée à la description précise du comportement du modèle dans certaines situations particulières. Nous proposons de développer quelques scénarios issus de l'application de réseau de capteurs sans fils étendue au multi-niveau avec SMA-R. Le détail du fonctionnement du prototype est établi à partir des traces d'exécution. Un exemple de trace sur un cas réduit (10 agents) est donné en annexe A. Les scénarios choisis illustrent notamment la construction des niveaux, l'adaptation des niveaux et l'interaction avec l'utilisateur.

7.1.1 La construction des niveaux

Un point clé de la supervision à l'aide de SMA-R est d'offrir une observation accessible par la construction de niveaux à partir d'un SMA de base. Appliquer le modèle SMA-R implique, pour le concepteur, de définir des *fonctions dynamiques*, des règles de *Composition* et de *Réduction* à partir desquelles le modèle construit les niveaux d'abstraction qui fournissent une visualisation abstraite.

Considérons le système *réseau de capteurs sans fils* présenté dans le chapitre 6 permettant d'instrumenter un réseau hydrographique souterrain. L'expérimentation met en jeu 125 agents capteurs organisés selon le modèle MWAC, elle est visualisée sur la figure 7.1 par une copie d'écran de l'outil MASH utilisé dans notre contexte.

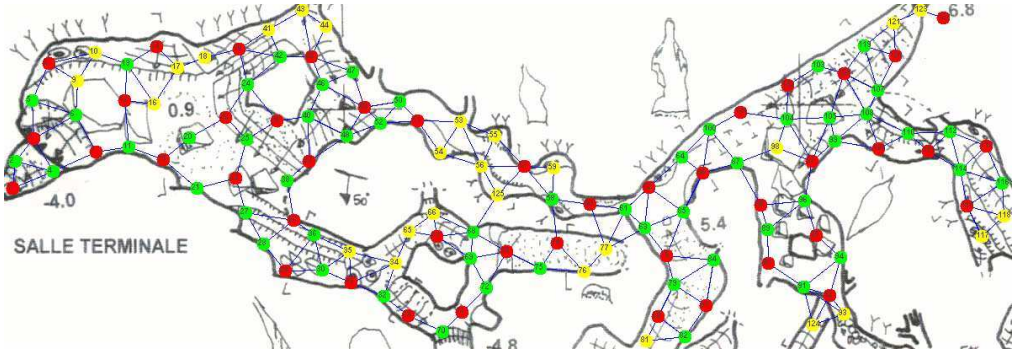


FIGURE 7.1 – La grotte instrumentée à l'aide des 125 agents capteurs

Nous ne représenterons plus visuellement l'environnement du SMA afin de simplifier la lecture des illustrations. La figure 7.2 donne une image du système multi-agent sans abstraction visible (niveau 0).

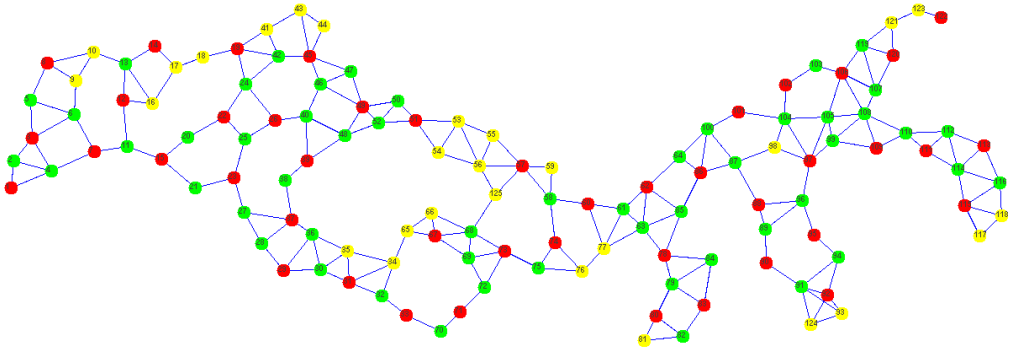


FIGURE 7.2 – Réseau hydrographique instrumenté - Visualisation du SMA au niveau 0

Nous souhaitons maintenant réduire la quantité d'information visualisée par l'utilisateur afin de simplifier la compréhension de l'état du système global.

Au niveau 0, la totalité des informations est accessible via les capteurs représentés graphiquement. Sur un niveau abstrait construit en utilisant le modèle SMA-R, la réduction de la complexité consiste en la création d'agents abstraits fusionnant les informations portées par les agents du niveau directement inférieur, c'est à dire représentant certaines zones.

Les agents, implantant le modèle SMA-R, vont donc interagir et construire des regroupements. Chacun des regroupements est vu comme un agent abstrait. Ces abstractions formeront le niveau d'observation supérieur à celui dans lequel étaient les agents qui se sont regroupés.

Le premier niveau abstrait est visible en figure 7.3. Dans ce niveau, le nombre d'agents est moindre (44 au lieu de 125). Les données que portent les agents abstraits consistent en une fusion des données portées par les agents agrégés du niveau inférieur.

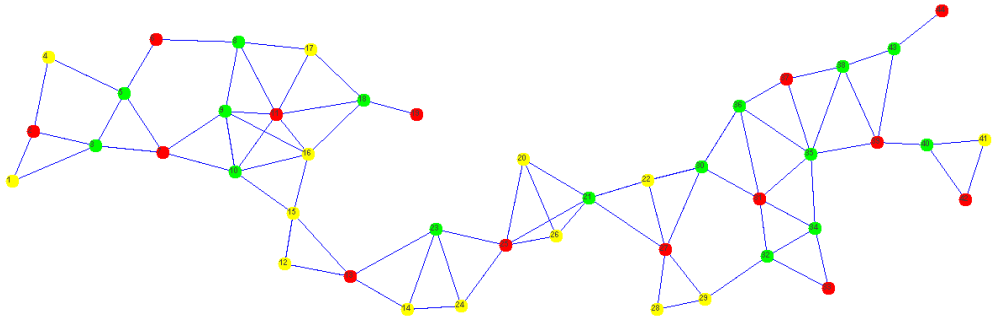


FIGURE 7.3 – Réseau hydrographique instrumenté - Visualisation du SMA au niveau 1

L'ensemble des agents de chacun des niveaux abstraits applique le comportement hérité du modèle MWAC. Quand un agent abstrait considère que l'organisation ainsi créée est stable dans son voisinage, et si sa *condition de composition* renvoie vrai, alors un nouveau niveau d'abstraction est créé (ce comportement est codé par l'algorithme 12).

Algorithme 12 Règle de Composition 1

1: Si le temps écoulé depuis la création du groupe est supérieur à la constante *DELAI_POUR_FIN_CREATION*
2: et si le temps écoulé depuis la dernière modification des voisins est supérieur à la constante *DELAI_POUR_VERIFICATION_INCOHERENCE_ORGANISATION*
3: et si le rôle est REPRESENTANT
4: et si un de ses voisins a le rôle LIAISON
5: et si l'agent n'est pas encore agrégé avec d'autres,
6: alors la *condition de composition* = 1,
7: sinon la *condition de composition* = 0.

Au niveau 5, la *fonction de satisfaction* renvoie 1 car l'agent n'a plus aucun voisin. La *condition de composition* renvoie donc faux, la composition s'arrête.

La figure 7.4 montre les différents niveaux créés. Notons que la section 7.2 présentera une analyse quantitative du modèle (coût, performances...).

7.1.2 L'adaptation du système

Nous nous intéressons à l'adaptation des niveaux abstraits à des disparations et ajouts d'agents sur le niveau 0. Nous considérons toujours ici l'application aux réseaux de capteurs sans fils MWAC. Nous étudions la situation du système de 23 agents introduits dans la section 7.1.2, reprise figure 7.5.

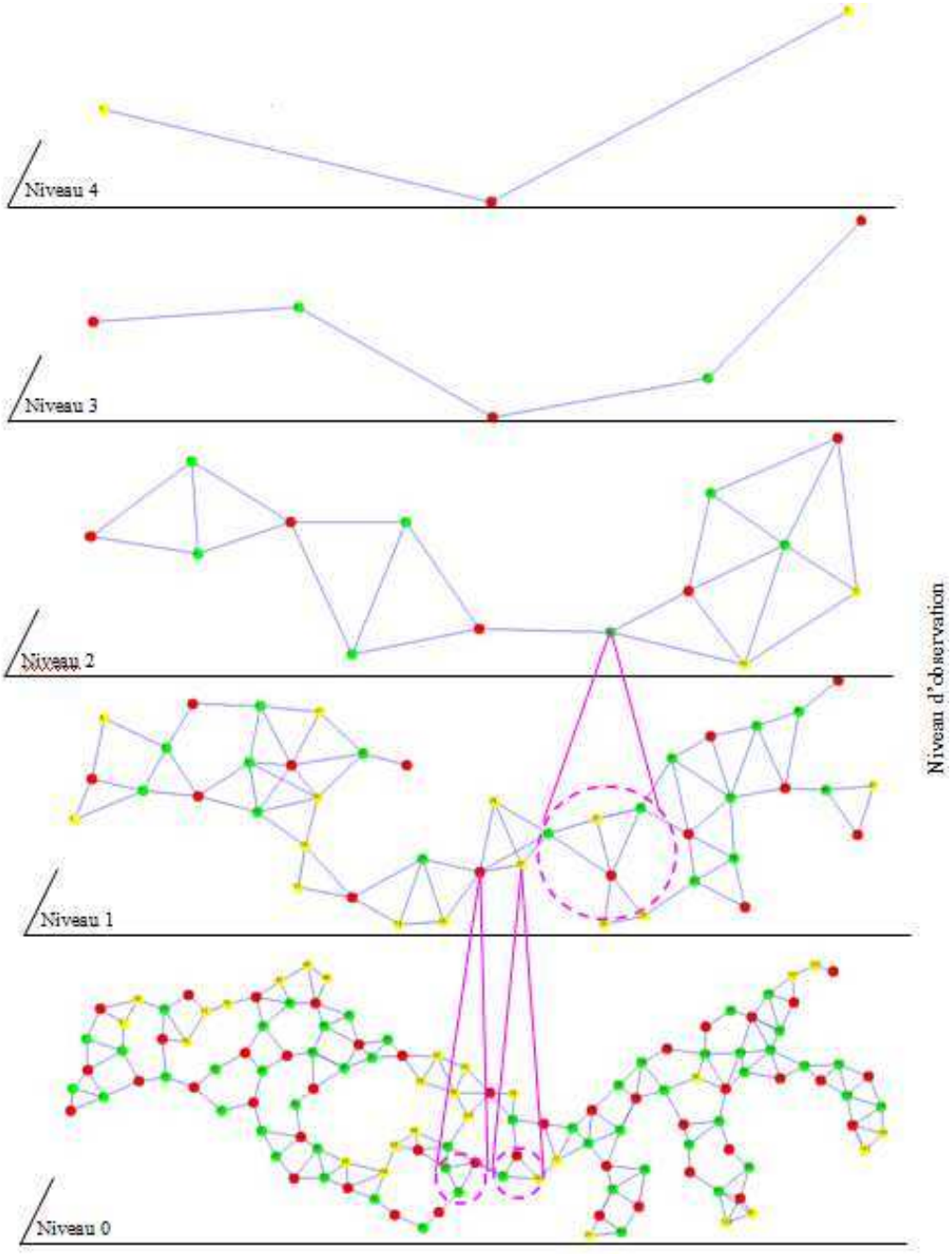


FIGURE 7.4 – Les visualisations de 5 niveaux d'abstraction du système de capteurs

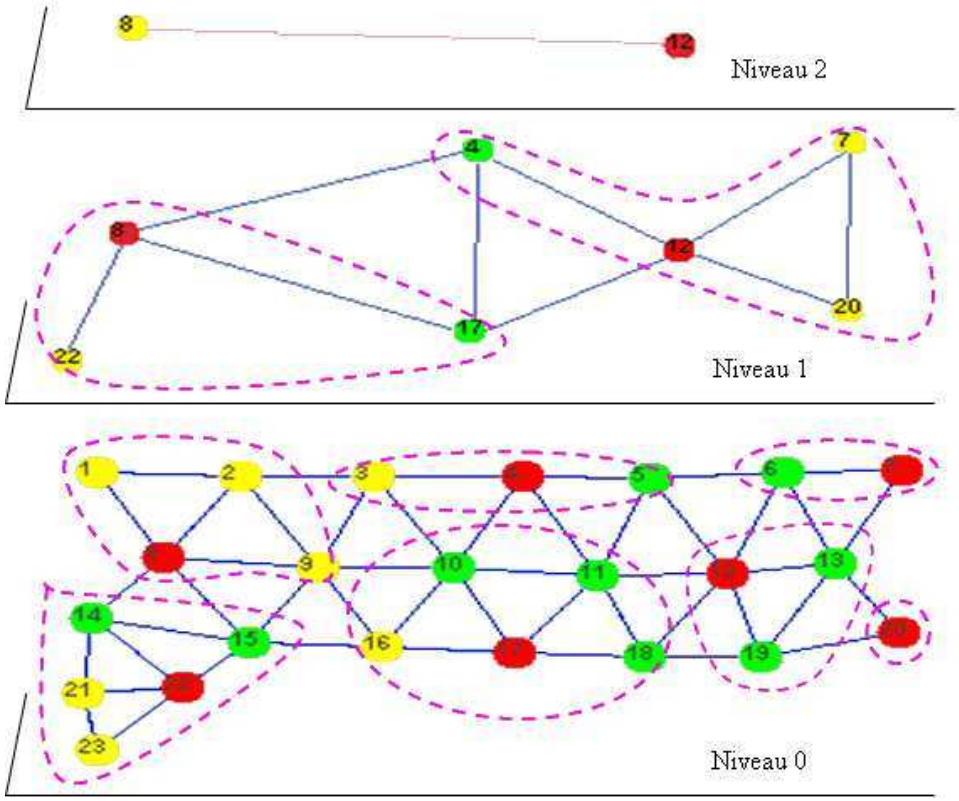


FIGURE 7.5 – Les 23 agents MWAC et les trois niveaux abstraits

Type de Groupe	Nombre d'agents	Contenu	Commentaire
1	1	R	un seul agent
2	2	R, L	
3	2	R, S	
4	3	R, L, S	
5	3	R, S, S	extension de 3
6	3	R, L, L	extension de 2

Tableau 7.1 – Type des groupes MWAC pour la règle de *Composition 1*

Les règles de construction nous conduisent à 3 niveaux d'abstraction avec six configurations de groupes différentes (Table 7.1).

Pour simplifier, nous retenons seulement les quatre groupes indiqués sur la figure 7.6. Les quatre groupes sont typiques de par leur nombre d'agents et la variété de leur rôles.

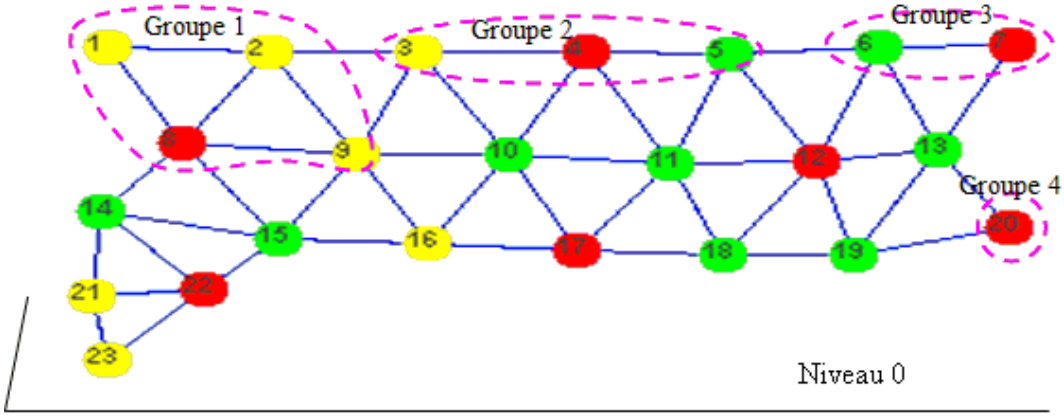


FIGURE 7.6 – Les quatre groupes considérés typiques de 23 agents MWAC

Considérons la disparition d'un agent sur chacun de ces cas :

1. Les groupes 1, 2, 3 :
   Si un seul des agents du groupe disparait, il n'y a pas de changement dans la population des agents abstraits au niveau supérieur. Les rôles sont éventuellement recalculés.
2. Le groupe 4 :
   Si l'agent 20 sur le niveau 0 disparait, alors le groupe 4 n'existe plus, le représentant (agent abstrait 20) sur le niveau 1 satisfait la règle de *Réduction 1*, il doit alors disparaitre. Les agents du groupe auquel il appartenait sur le niveau 1 doivent donc recalculer leurs rôles. La structure des niveaux d'abstraction est donc modifiée comme indiqué sur la figure 7.7.

Considérons maintenant l'ajout d'un agent :

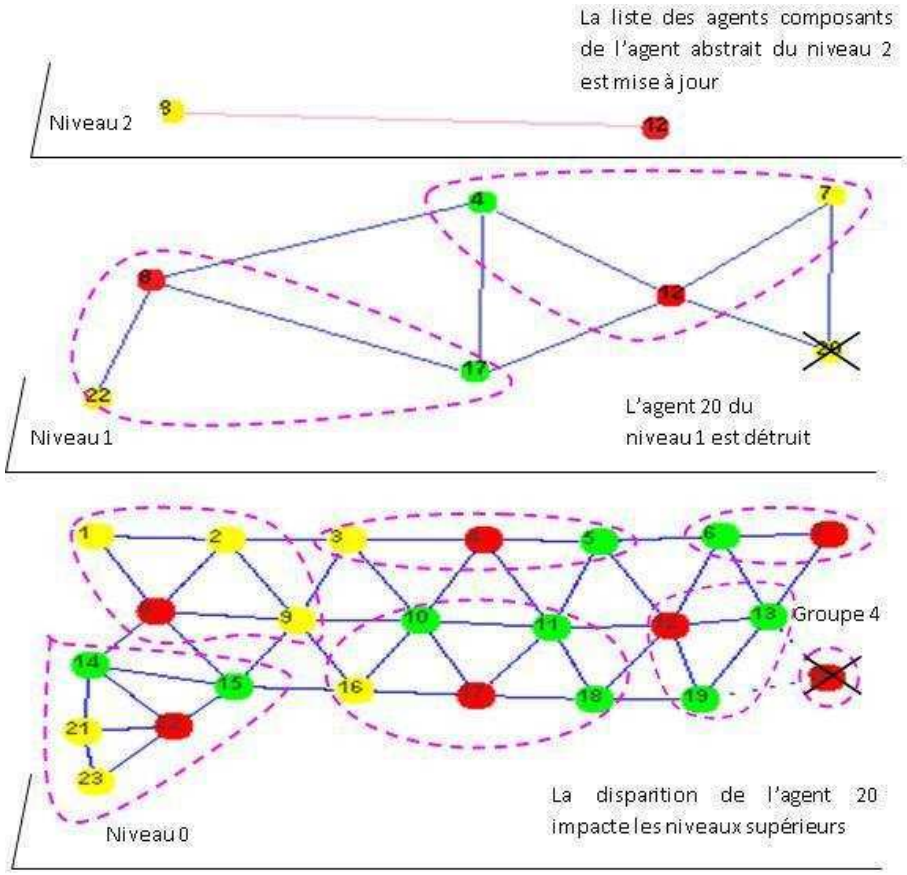


FIGURE 7.7 – Suppression de l'agent du groupe 4

– Si l'agent ajouté prend le rôle *S* ou *L*, il participe au groupe MWAC auquel il appartient. Aucun agent ne satisfait la règle *Composition 1*. Aucun agent abstrait nouveau n'est donc créé au niveau 1. Aucun changement n'est provoqué sur le niveau supérieur.
La figure 7.8 donne un exemple de ces situations. Dans la configuration (A), les deux agents 24 et 25 viennent d'être ajoutés. L'agent 24 prend le rôle *S*. L'agent 25 prend le rôle *L*. Dans la configuration (B), l'agent 24 intègre le groupe MWAC de l'agent 22. Il devient un agent fils de l'agent 22 du niveau 1. L'agent 25 intègre un des deux groupes auxquels il peut se rallier, par exemple le groupe de l'agent 7. Il devient un agent fils de l'agent 7 du niveau 1.

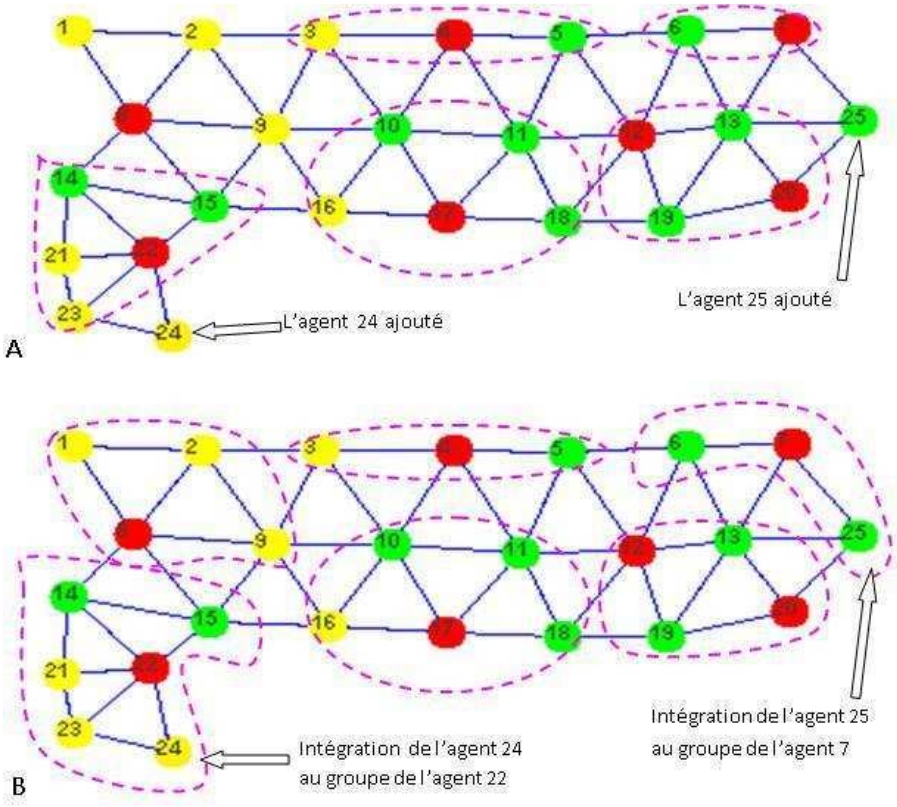


FIGURE 7.8 – Exemple d'ajout d'un agent de rôle *S* ou *L*

– Si l'agent ajouté prend le rôle *R*, un nouveau groupe est formé. On distingue deux cas :

1. Le nouvel agent est seul dans son groupe, alors il déclenche une composition. Dans ce cas, l'agent abstrait créé satisfait la règle de *Réduction 1*, il décide alors de sa disparition. L'agent représentant rejoint le groupe voisin. La figure 7.9 illustre l'exemple de l'ajout de l'agent 26 alors que son seul voisin l'agent 24 a intégré le groupe de l'agent 22. L'agent 26 intègre le groupe de l'agent 22. Il n'y a aucun changement dans les niveaux supérieurs.

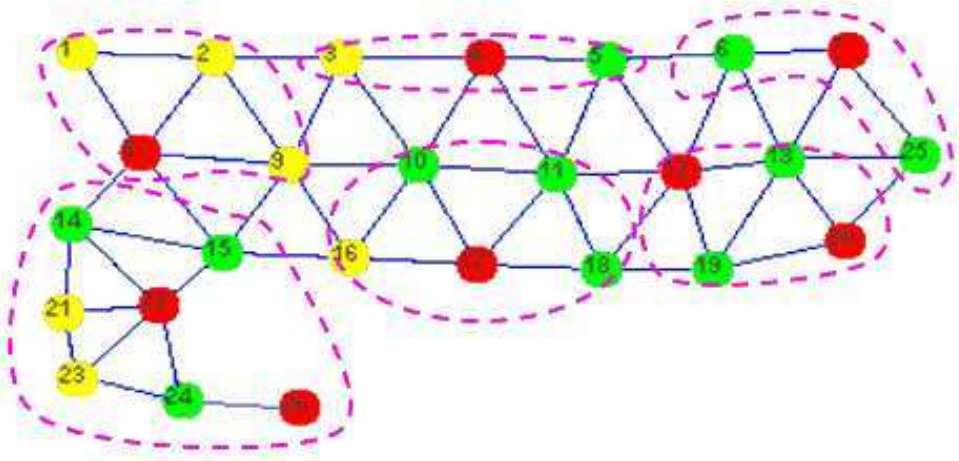


FIGURE 7.9 – Exemple d'ajout d'un agent de rôle *R* sans création d'agent abstrait

2. Si le groupe du représentant est non vide, et contient des agents non encore en cours de composition, l'agent représentant invite les autres membres à rejoindre son groupe. Si le nouveau groupe satisfait la règle de *Composition 1*, l'agent représentant initie la création d'un nouvel agent récursif du niveau 1. La structure des niveaux abstraits est modifiée.

   La figure 7.10 illustre l'ajout de l'agent 26 alors que l'agent 24 n'appartient encore à aucun groupe MWAC. L'agent 26 prend le rôle *R* et invite l'agent 24 à intégrer son groupe. Un nouveau groupe est donc formé. La règle composition validée, provoque la modification des niveaux supérieurs.

Impact du choix des règles de construction de niveau

Le choix des *règles de composition* ou de *réduction* impacte la profondeur du système construit.

Nous reprenons la première *règle de composition* décrite dans l'algorithme 12, page 113.

Nous définissons comme suit une seconde *règle de composition* (algorithme 13, 120) pour le même problème.

La règle de *Réduction* est décrite de la même façon dans l'algorithme 14, page 123 :

Le nombre de niveaux d'observation créés pour la population de 125 agents MWAC par l'application de la règle de *Composition 1* est de 4 alors qu'il est de 5 avec la règle de *Composition 2*. L'organisation des agents abstraits diffère de même comme on peut le voir sur les figures 7.11 et 7.12.

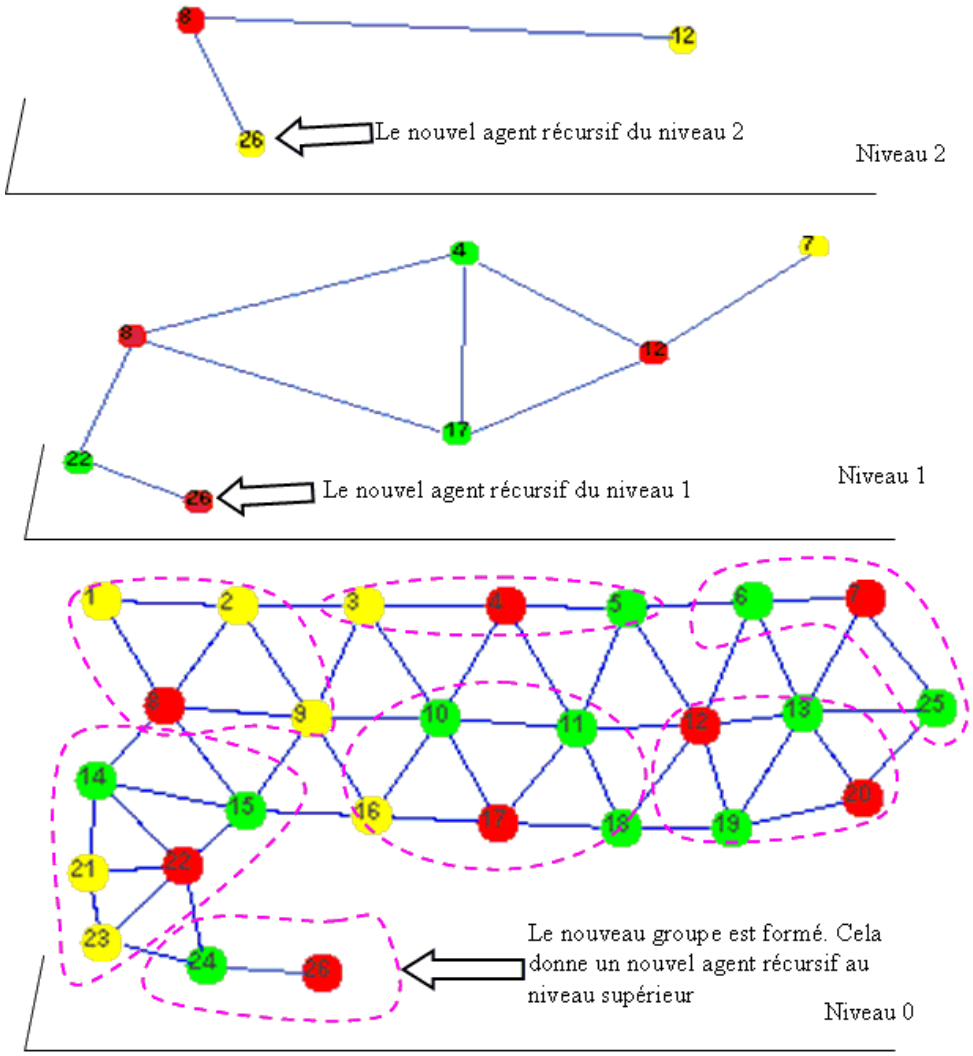


FIGURE 7.10 – Exemple d'ajout d'un agent de rôle *R* avec création d'un agent abstrait

Algorithme 13 La règle de Composition 2

1: Si le temps écoulé depuis la création du groupe est supérieur à la constante DELAI_POUR_FIN_CREATION
2: et si le temps écoulé depuis la modification des voisins est supérieur à la constante DELAI_POUR_VERIFICATION_INCOHERENCE_ORGANISATION
3: et si son rôle est REPRESENTANT ou LIASON
4: et s'il a au moins un voisin
5: et si l'agent n'est pas encore agrégé avec d'autres,
6: alors la *condition de composition* = 1,
7: sinon la *condition de composition* = 0.

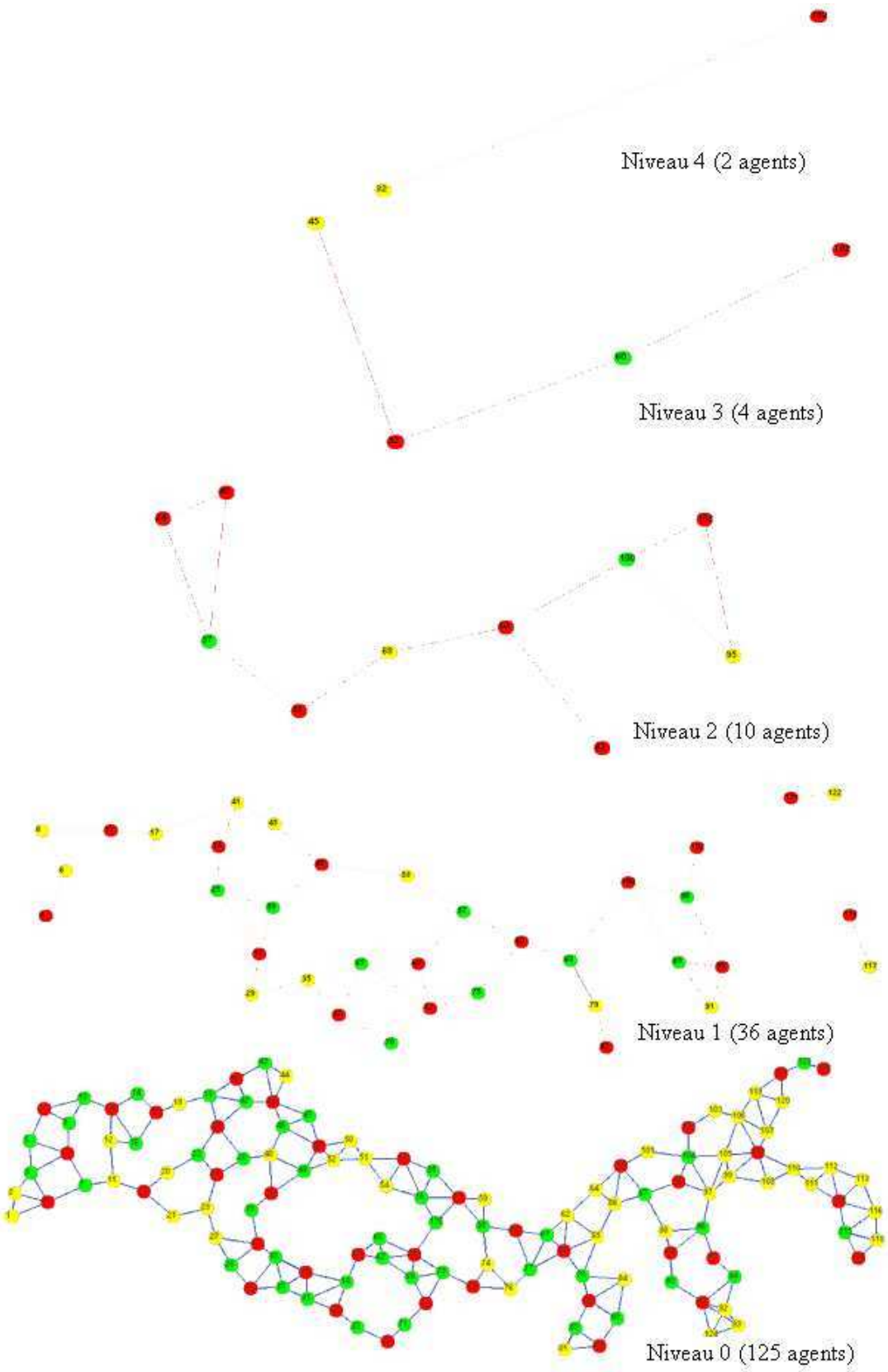


FIGURE 7.11 – Visualisation des 125 capteurs MWAC en utilisant les règles de Composition 1

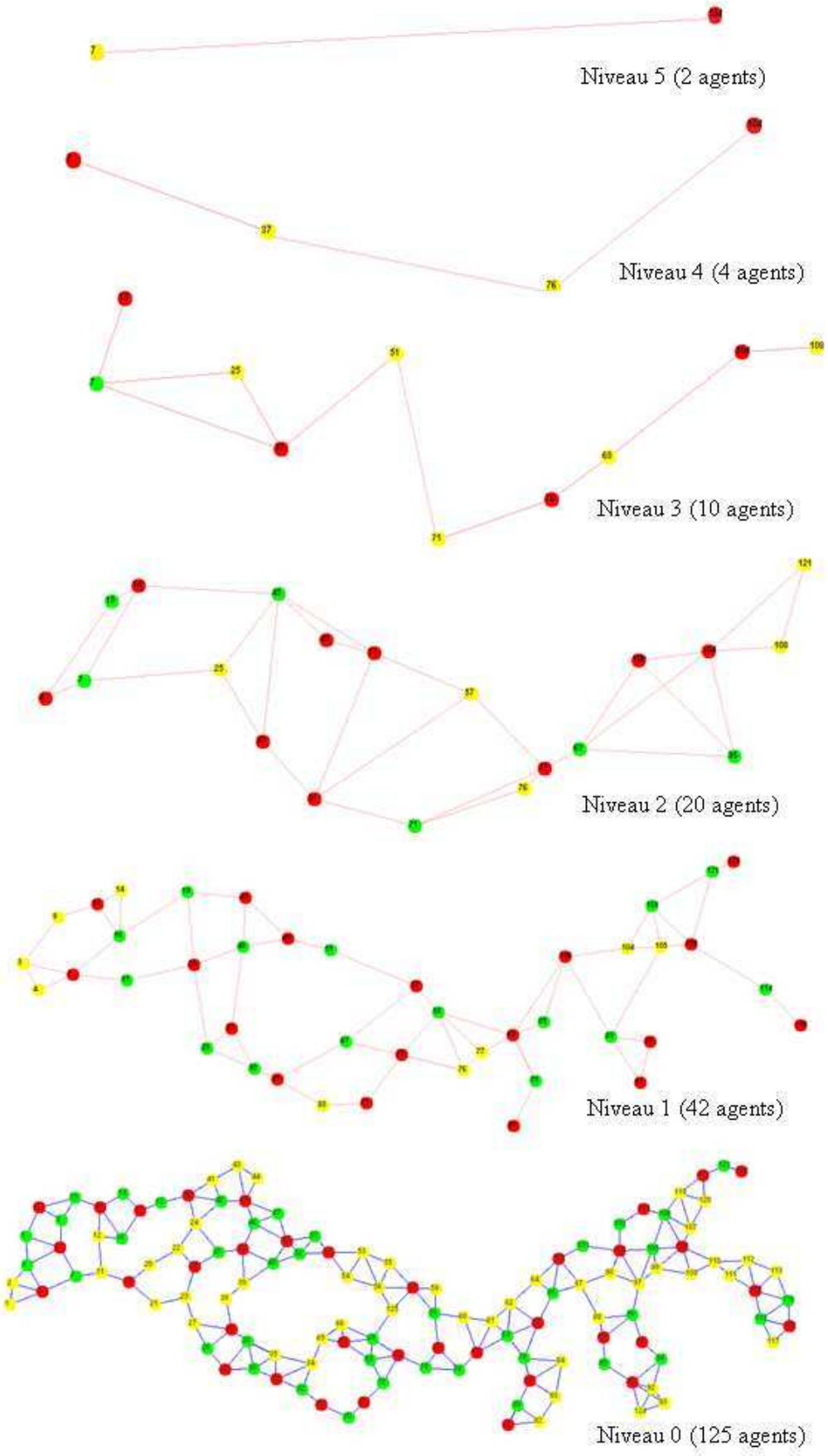


FIGURE 7.12 – Visualisation des 125 capteurs MWAC en utilisant les règles de Composition 2

Algorithme 14 La règle de *Réduction 1*

(Si le temps écoulé depuis la création du groupe est supérieur à la constante *DELAI_POUR_FIN_CREATION*
et si le temps écoulé depuis la modification des voisins est supérieur à la constante *DELAI_POUR_VERIFICATION_INCOHERENCE_ORGANISATION*
et si la construction du niveau supérieur est finie
et s'il n'y a qu'un seul agent dans son groupe)
OU
(Si le temps écoulé depuis la création du groupe est supérieur à la constante *DELAI_POUR_FIN_CREATION*
et si le temps écoulé depuis la modification des voisins est supérieur à la constante *DELAI_POUR_VERIFICATION_INCOHERENCE_ORGANISATION*
et si la construction du niveau supérieur est finie
et s'il y a plus d'un agent *R* et des agents L reliant les agent R dans son groupe)
alors la *condition de réduction* = 1,
sinon la *condition de réduction* = 0.

7.1.3 L'interaction avec l'utilisateur

Cette section veut illustrer comment est traitée l'interaction entre l'utilisateur et un niveau d'abstraction donné. L'utilisateur peut interagir avec les agents applicatifs, il est donc capable d'interagir avec un agent élémentaire. Supposons que les agents applicatifs mesurent la température de leur environnement. Si l'utilisateur veut connaitre la température moyenne d'une zone, il doit envoyer la même demande à tous les agents applicatifs de cette zone et collecter les mesures retournées par ces agents applicatifs. L'utilisateur doit répéter la même demande et intégrer les résultats pour obtenir une valeur moyenne pour la zone.

L'emploi du modèle SMA-R permet aux utilisateurs d'observer le système à plusieurs niveaux d'abstraction. Les agents récursifs jouent un rôle de représentant des groupes d'agents applicatifs dans une zone précise. Ils sont comme des images abstraites des agents applicatifs. Les agents récursifs ne suppriment pas la capacité d'interagir avec les agents applicatifs. Les agents récursifs doivent fournir de la même façon que les agents applicatifs une interface permettant à l'utilisateur d'envoyer une demande à une zone dont ces agents récursifs sont représentants. Après avoir circonscrit la zone qui doit être observée, l'utilisateur choisit un niveau abstrait du système et trouve facilement sur quel agent composé il doit intervenir pour obtenir des informations considérées. L'utilisateur envoie une demande seulement à cet agent récursif. Cette demande est propagée de façon "descendante" en focalisant sur les agents applicatifs de la zone dont l'agent récursif est représentant. L'agent récursif retourne une valeur de la zone en calculant les résultats retournés par les agents fils. Le calcul est défini par le concepteur dans la propriété récurrente *PI* comme une moyenne arithmétique.

Reprenons l'exemple des 125 capteurs (Figure 7.1). Les capteurs MWAC sont capables de mesurer la température de l'environnement où ils se situent. Grâce à la visualisation abstraite du système,

l'utilisateur détermine l'agent récursif du niveau 2 qui représente des capteurs de la zone considérée (figure 7.13, page 125). Supposons que l'utilisateur souhaite connaitre la température de la zone. La température de la zone est définie égale à la moyenne arithmétique des températures captées de la zone (voir l'algorithme 15)

Algorithme 15 Formulaire de recalcul de la température défini dans *PI*

1: $temperature = (temperature + newTemperature)/(nombreReponsesRecues + 1)$
2: $nombreReponsesRecues = nombreReponsesRecues + 1$

La demande de l'utilisateur est propagée à tous les agents élémentaires de la zone (figure 7.14). Le message portant la demande de l'utilisateur est "descendu" jusqu'au niveau 0. Il est ensuite passé aux agents applicatifs. Les agents applicatifs interprètent le message et mesurent la température de l'environnement réel.

Le message portant la température est "remonté" au niveau supérieur jusqu'au destinataire - l'agent représentant de la zone. Les agents composés qui reçoivent le message doivent le transformer en recalculant la température et "remontent" un nouveau message, qui porte la température calculée jusqu'à l'agent représentant de la zone. Le résultat de la mesure est retourné à l'agent représentant de la zone (figure 7.15).

7.2 Évaluation de la performance du modèle

Nous proposons dans cette section une évaluation du processus de construction des niveaux d'abstraction. Nous décrivons le protocole expérimental dans une première partie puis nous évaluons pour chacune des configurations proposées le fonctionnement des mécanismes mis en œuvre pour la récursion.

7.2.1 Protocole expérimental

Nous allons nous intéresser à 5 configurations initiales permettant d'évaluer l'approche pour une population de 100, 200, 400, 600 et 800 agents.

Phase 1. Dans un premier temps, nous allons nous intéresser au fonctionnement et au coût de la construction de l'organisation multi-niveau. Pour l'ensemble des configurations, à l'instant $t_0 = 0s$, 90% des agents de la population sont initialisés et démarrent leur exécution. Ils vont construire les différents niveaux d'abstraction. Pour cela ils déterminent leurs voisinages, ils choisissent un rôle et forment/rejoignent un groupe. Quelle que soit la configuration, le dernier niveau d'abstraction qui sera créé est celui qui contient exactement un agent.

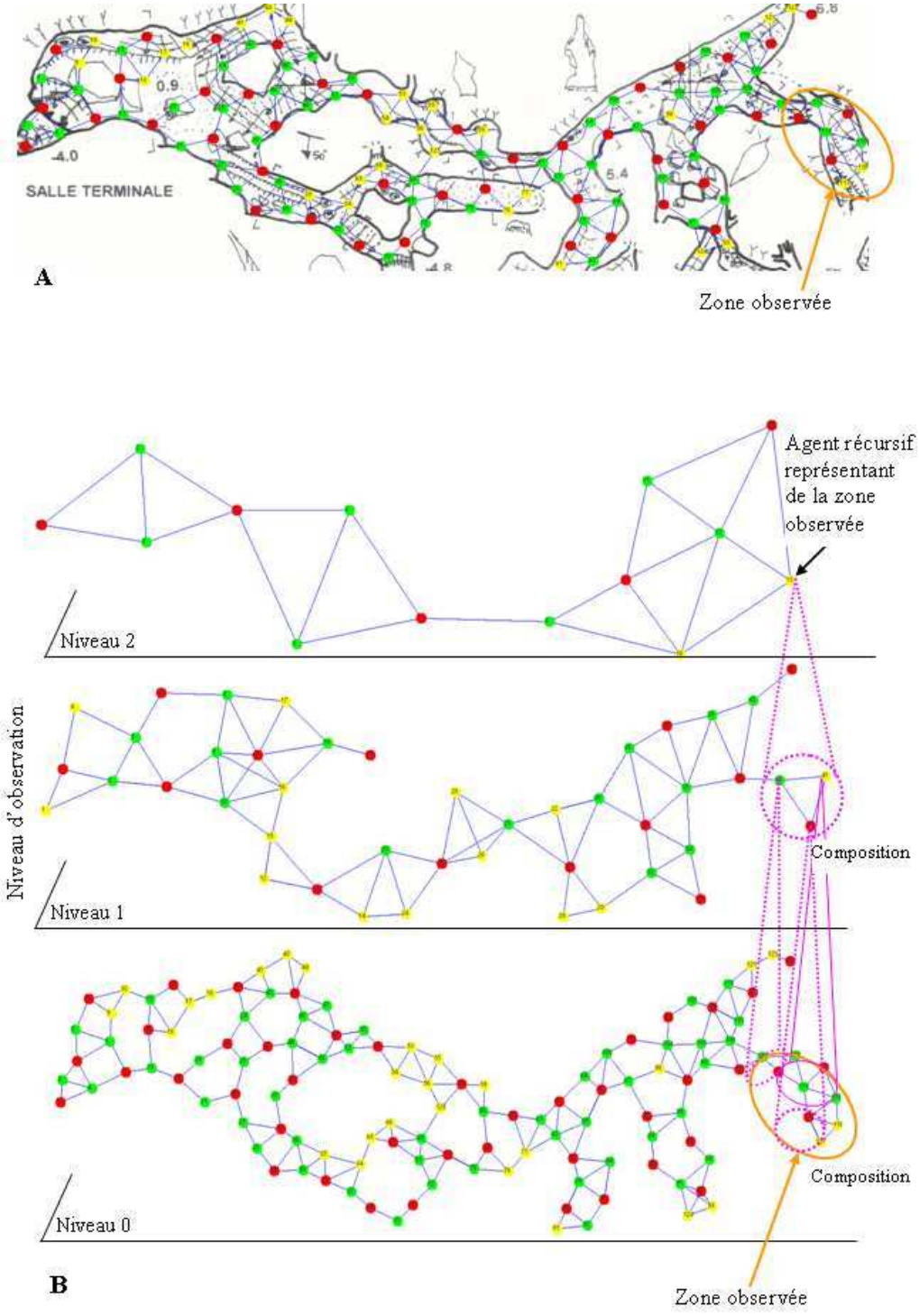


FIGURE 7.13 – (A) Détermination de la zone observée et de ses abstractions (B)

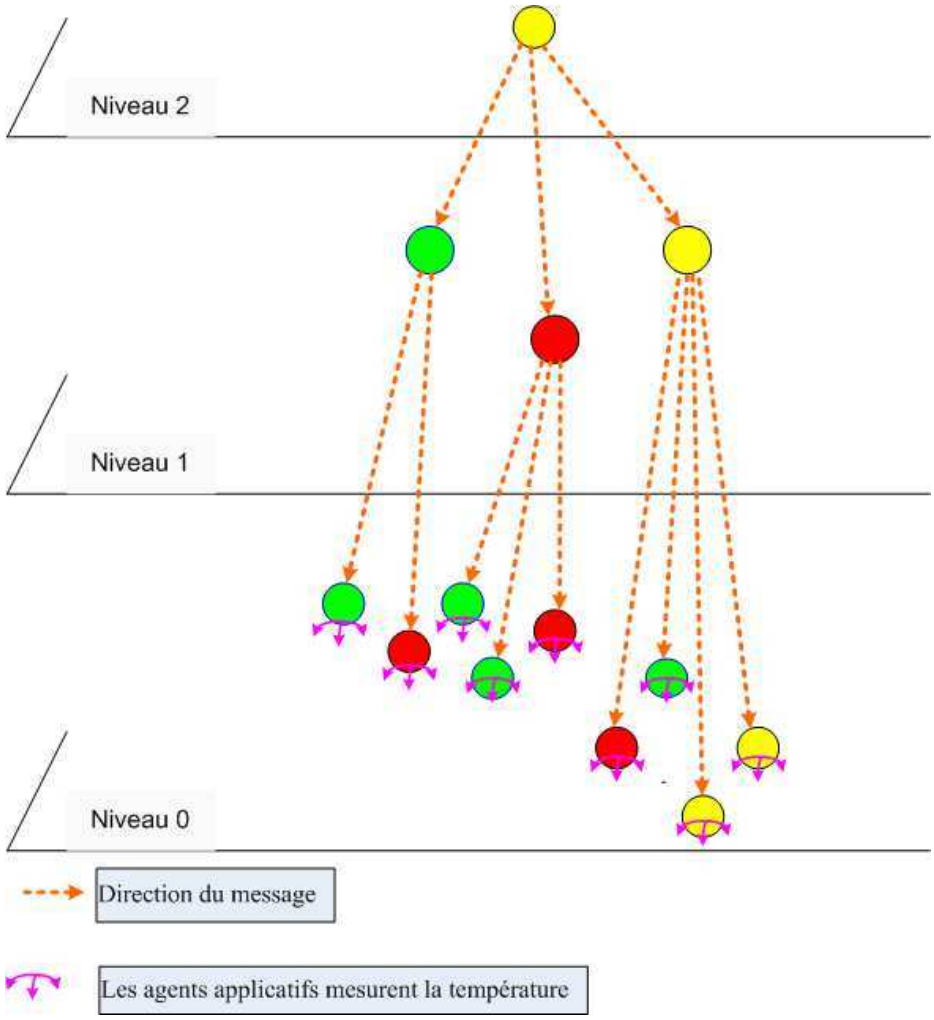


FIGURE 7.14 – Propagation de la requête vers les agents applicatifs

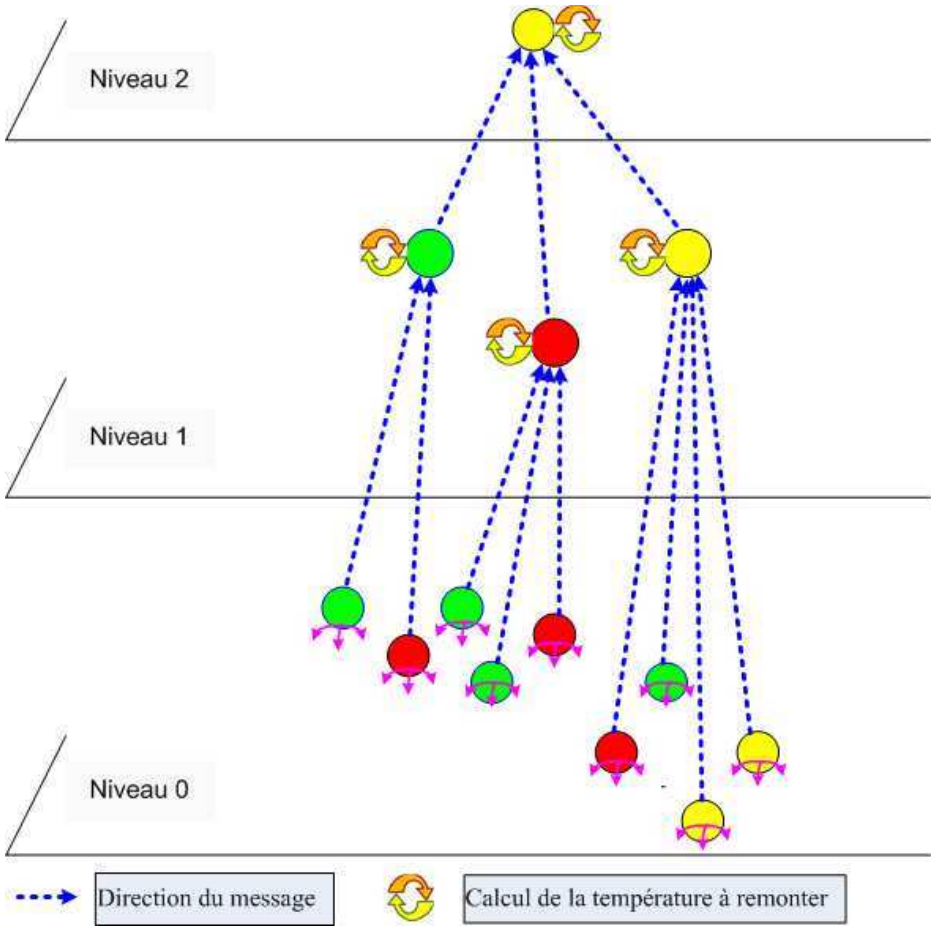


FIGURE 7.15 – Remontée et transformation de la mesure vers le destinataire.

Tableau 7.2 – Les configurations de simulation au début de chacune des phases

Configuration	1	2	3	4	5
Topologie 1 :					
- Nombre d'agents	90	180	360	540	720
- Densité (moyenne)	8.24	7.25	8.23	8.02	8.33
- Densité (écart-type)	2.91	2.95	2.59	2.45	2.35
Topologie 2 :					
- Nombre d'agents	100	200	400	600	800
- Densité (moyenne)	8.2	7.7	8.34	8.35	8.53
- Densité (écart-type)	2.97	2.75	2.44	2.37	2.41

Phase 2. Dans un second temps, nous allons porter notre attention à l'adaptation d'une organisation existante à l'ajout de nouveaux agents. Ces agents ne sont dans aucune organisation. A la date $t = 240s$, les 10% d'agents restants sont ajoutés au système, ils sont initialisés et démarrent leur exécution.

Phase 3. Dans un troisième et dernier temps, nous allons nous intéresser à l'adaptation d'une organisation existante à la suppression d'agents. A la date $t = 300s$, les 10% d'agents qui avaient été ajoutés à la date $t = 240s$ quittent "proprement" le système.

La tableau 7.2 résume les différentes configurations et propose un indicateur de la répartition géographique des agents que l'on appelle densité, le nombre moyen d'agents constituant le voisinage d'un agent. La phase 1 consiste en passer d'un système vide à un système ayant la topologie 1. La phase 2 consiste en l'ajout d'agents faisant évoluer le système de la topologie 1 à la topologie 2. La phase 3 consiste à revenir de la topologie 2 à la topologie 1.

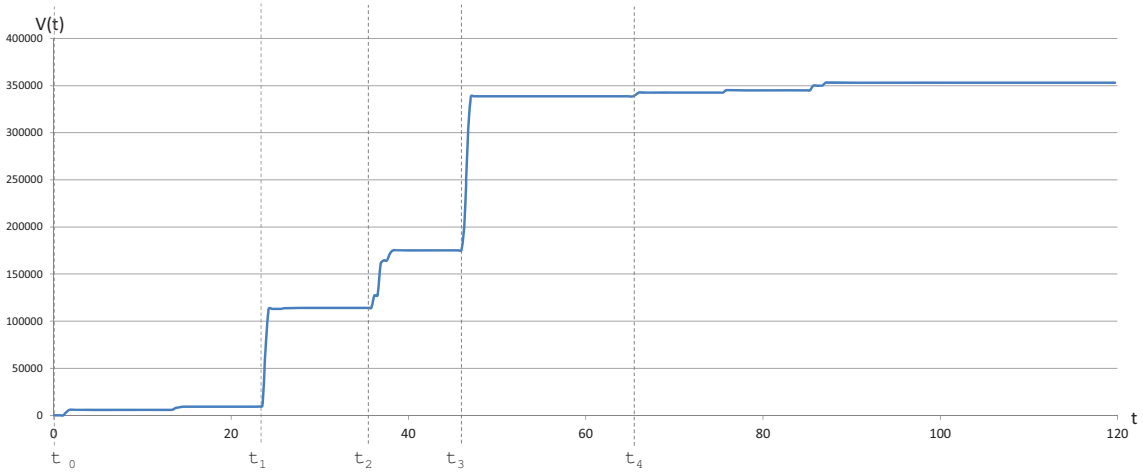
Les résultats de simulation sont présentés comme suit. Dans une première partie nous insistons sur la construction de l'organisation puis, dans un second temps, sur l'adaptation à l'ajout d'agents et pour finir à l'adaptation au retrait d'agents. A chaque fois, nous commencerons par porter notre attention sur la configuration 1 seulement. Cette configuration, contenant par rapport aux autres peu d'agents, simplifie l'identification et donc l'interprétation de la mise en œuvre des mécanismes impliqués dans la création des abstractions.

7.2.2 Construction de l'organisation

Dans cette section, nous analysons les résultats de simulation liés à la phase 1.

7.2.2.1 Description détaillée de la configuration 1

La figure 7.16 montre le volume cumulé de données échangées par l'ensemble des agents du SMA en fonction du temps. Ce graphique permet donc de mesurer en partie [1] le coût de transmission de la mise en œuvre de notre architecture.



t : temps (en secondes) V(t) : volume total de données échangées par les agents (en octets)

FIGURE 7.16 – Construction des abstractions (configuration 1).

A la date t_0, 90 agents démarrent et construisent le niveau 0 du système : les groupes sont construits et l'organisation se stabilise.

A partir du temps t_1, chaque agent qui perçoit localement la stabilité de son voisinage et construit son abstraction de niveau supérieur. La période de sureté permettant de s'assurer de la stabilité correspond au dernier tiers de l'intervalle $[t_0;t_1]$ (aucun message n'est échangé).

On remarque, en observant le comportement externe des agents (concrètement les messages transmis) dans l'intervalle $[t_0;t_1]$ qu'il est impossible de déterminer si les agents implantent le modèle MWAC seul ou l'extension multi-niveau car seuls des messages applicatifs circulent dans le système.

De la même manière, aux dates t_2 et t_3 les agents stables commencent à construire respectivement les niveaux d'abstraction 2 et 3.

A la date t_4, l'agent applicatif de niveau 3 tente de découvrir son voisinage afin d'être sûr qu'aucune abstraction supplémentaire n'est nécessaire.

1. Pour avoir une mesure complète de ce coût il serait nécessaire de mesurer non seulement le coût d'émission mais aussi le coût de réception des messages

7.2.2.2 Résultats obtenus pour les 5 configurations

Nous présentons en figure 7.20 les courbes V(t) associées aux autres configurations. L'analyse de telles courbes n'est pas aussi évidente que pour le cas de la configuration à 100 capteurs car il peut y avoir des recouvrements entre les périodes de construction des différents niveaux d'abstraction. Plusieurs raisons sont à l'origine de ce problème notamment :

– l'étendue du réseau qui entraine des différences importantes dans les dates d'identification par les agents de stabilités organisationnelles locales,
– l'augmentation des temps de transit (temps perdu par un message dans le traitement par un agent relais dans l'acheminement du message) à cause d'un plus grand remplissage des files de réception/émission de messages.

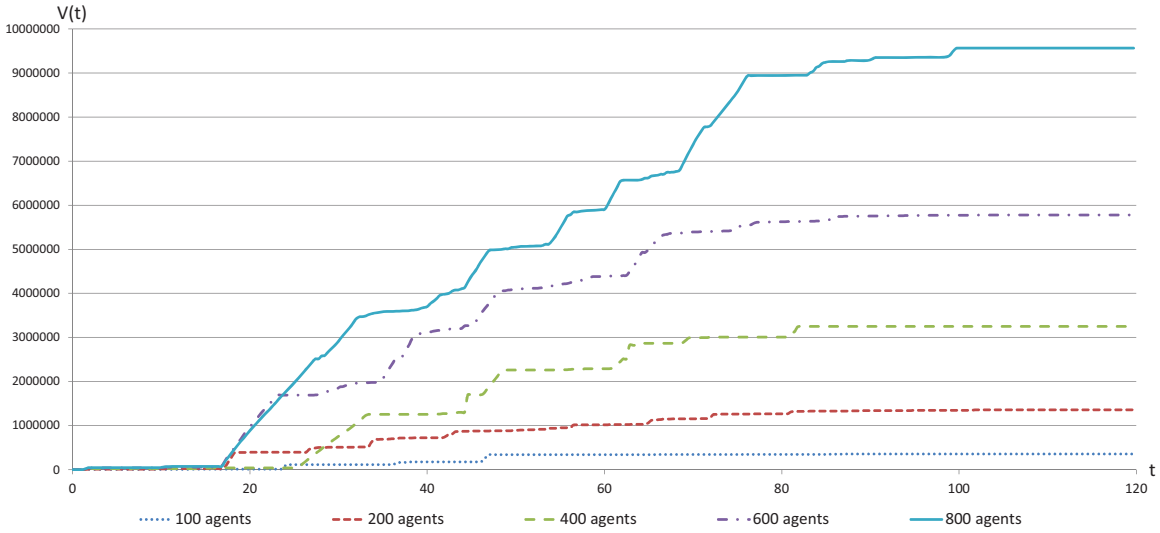


FIGURE 7.17 – Construction des abstractions (configuration 1,2,3,4 et 5).

7.2.3 Adaptation de l'organisation à l'ajout d'agents

Dans cette section, nous analysons les résultats de simulation liés à la phase 2.

7.2.3.1 Description détaillée de la configuration 1

A la date t_5 (figure 7.18), 10 nouveaux agents sont introduits dans l'organisation : durant ce processus les agents doivent déterminer leur voisinage, choisir un rôle et créer/rejoindre un groupe. Le volume de communication associé est donc faible et donc non visible sur le graphique. Les agents vont ensuite, en fonction de leurs positions géographiques et donc de leurs accointances avec des agents déjà impliqués dans la construction de l'abstraction, soit rejoindre des agents abstraits existants, soit en créer de nouveaux. A la date t_5 les agents ont déjà construits leurs niveaux 0. Ces agents

vont créer l'abstraction de niveau 1. A t_6 et t_7 aucun agent abstrait supplémentaire de niveau 2 et 3 n'est créé : les agents abstraits de niveaux inférieurs se sont agrégés aux agents abstraits existants sur les niveaux supérieurs. Le volume de message que l'on peut identifier correspond à une mise à jour des listes d'agents agrégés. Les données véhiculées dans $[t_8;t_9]$ sont les messages envoyés par l'agent de niveau 3 pour vérifier qu'aucun voisin n'existe et qu'il n'est donc pas nécessaire de créer une nouvelle abstraction.

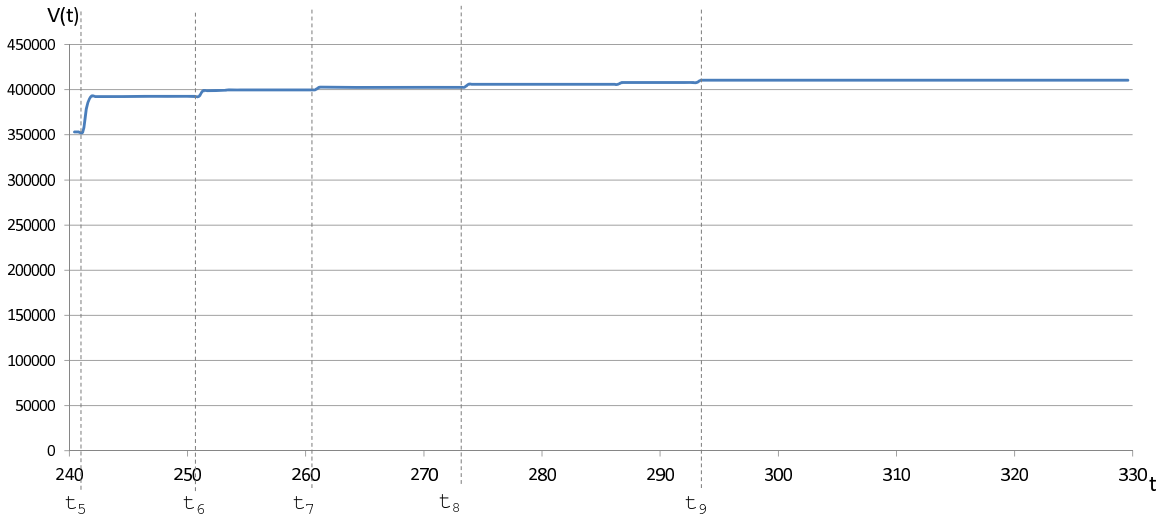


FIGURE 7.18 – Adaptation à l'ajout d'agents (configuration 1).

L'ensemble des niveaux construits est visible en figure 7.19.

On constate sur ce diagramme que le système multi-niveau converge bien pour l'ensemble des configurations.

7.2.3.2 Résultats obtenus pour les 5 configurations

Nous présentons en figure 7.20 les courbes V(t) associées aux autres configurations.

Si l'on considère le résultat des phases 1 et 2, on remarque (tableau 7.3) que le surcoût (en volume échangé par agent) pour la création d'un niveau est à peu près constant, de l'ordre de 18 fois le volume échangé par agent pour le système de base sans visualisation multi-niveau. Le volume généré n'explose donc pas avec l'accroissement du nombre de niveaux.

On remarquera aussi que le coût moyen par niveau généré reste inférieur à 2000 octets échangés par agent ce qui reste tout à fait intéressant en regard du service rendu. On pourrait mener une étude plus poussée autour de l'impact de la densité des agents ou de celui du choix des règles de construction des abstractions. Cependant il est important de noter que les mesures étant très dépendantes de l'application, ces études ne permettraient pas d'apporter d'éléments plus précis sur le mécanisme lui-même, le mêlant inévitablement au comportement de l'application.

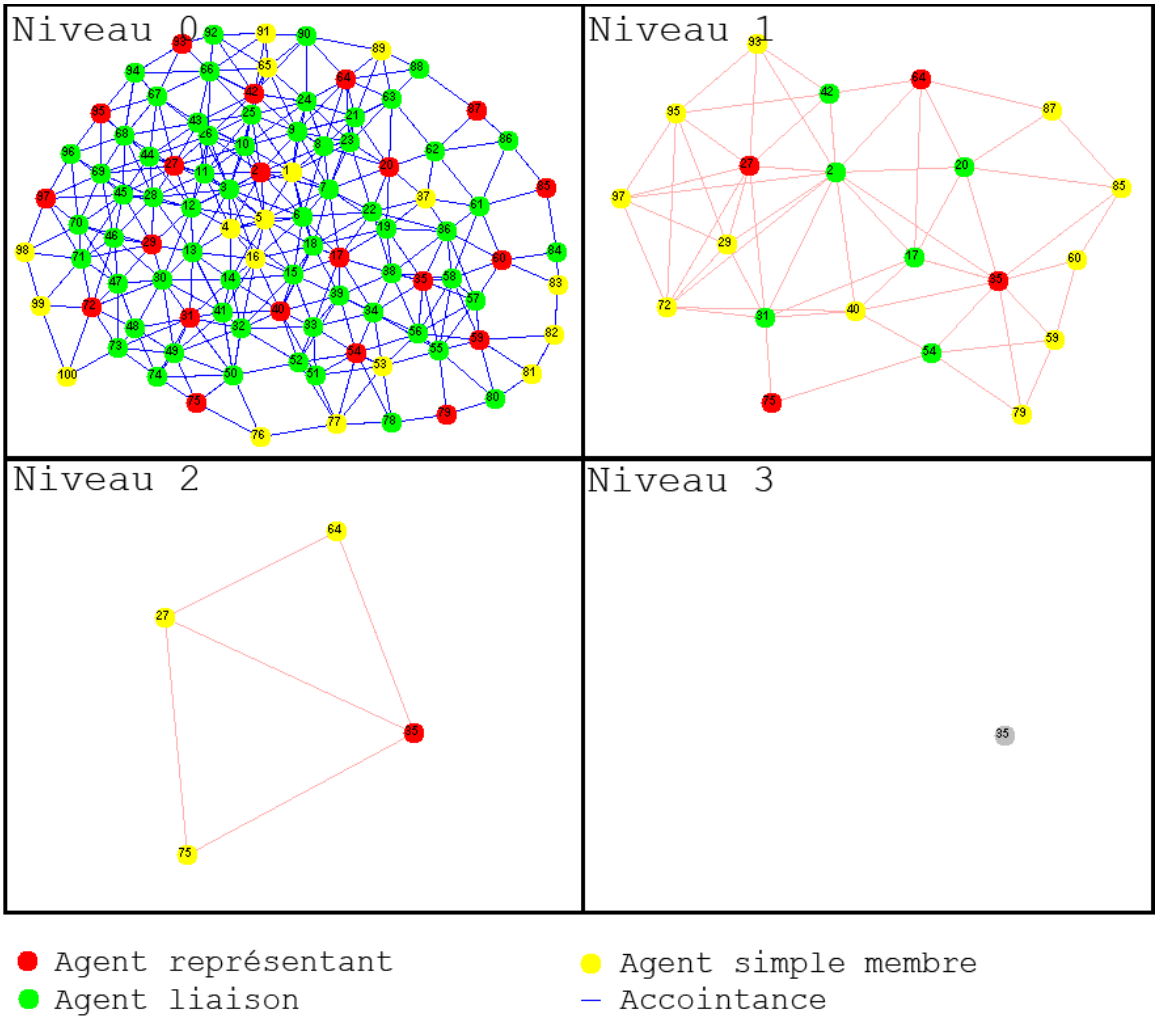


FIGURE 7.19 – Les 4 niveaux construits (configuration 1)

Le tableau 7.4 présente des informations détaillées sur les différents niveaux observées dans les simulations des différents scénarios présentés (phases 1 et 2 terminées).

7.2.4 Adaptation de l'organisation au retrait d'agents

Dans cette section, nous analysons les résultats de simulation liés à la phase 3.

D'une manière générale, le coût du retrait des agents est assez faible (figure 7.21). A la date t_{10}, les agents qui avaient été ajouté à la phase 2, soit 10% de la population, demandent à se réduire à des entités de niveau 0 uniquement. Les agents retirés signalent qu'ils quittent l'organisation. Pour cela, ils envoient un message de configuration aux autres agents agrégés pour signaler qu'ils quittent l'abstraction. Ensuite, ils détruisent leurs propres représentations des abstractions. S'il peut encore joindre ses voisins de niveau *n*, le dernier membre d'une abstraction de niveau *n* qui se décompose

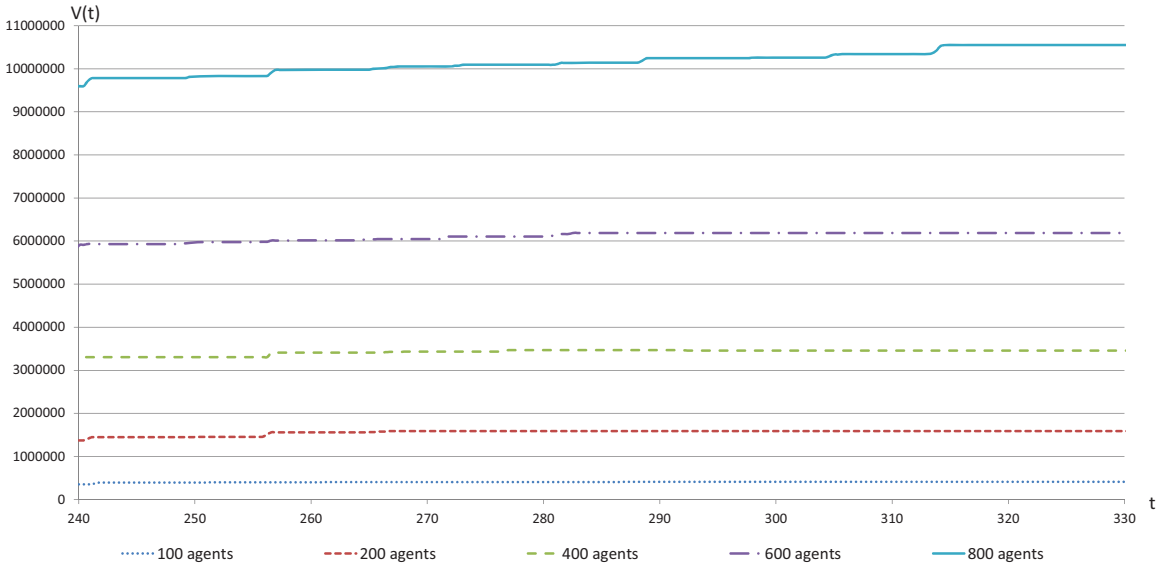
Tableau 7.3 – Coût de construction des niveaux (en volume échangé par agent)

Configuration	1	2	3	4	5
Nombre de niveaux	4	5	5	6	7
MWAC sans multi-niveau	92	97	92	95	88
Tous les niveaux	4010	7944	8643	10314	13187
Coût de création d'un niveau récursif (volume MWAC $\times$)	$\times$ 10.90	$\times$16.38	$\times$ 18.79	$\times$18.09	$\times$21.41

Tableau 7.4 – Informations sur les niveaux mesurées en simulation (toutes configurations)

Configuration	1	2	3	4	5
Nombre de niveaux	4	5	5	6	7
Niveau 0					
Nombre d'agents	100	200	400	600	800
Densité (moyenne)	8.2	7.7	8.34	8.35	8.53
Densité (écart-type)	2.97	2.75	2.44	2.37	2.41
Nombre d'agents agrégés (moyenne)	0	0	0	0	0
Nombre d'agents agrégés (écart-type	0	0	0	0	0
Niveau 1					
Nombre d'agents	21	42	77	119	155
Densité (moyenne)	5.523	5.09	5.53	5.64	5.30
Densité (écart-type)	2.293	2.06	1.51	1.73	1.96
Nombre d'agents agrégés (moyenne)	4.76	4.76	5.19	5.04	5.16
Nombre d'agents agrégés (écart-type)	1.54	1.73	1.79	1.81	1.88
Niveau 2					
Nombre d'agents	4	10	18	28	37
Densité (moyenne)	2.5	1.9	4.55	4.14	2.7
Densité (écart-type)	0.577	0.99	1.54	1.19	1.83
Nombre d'agents agrégés (moyenne)	5.25	4.2	4.27	4.25	4.18
Nombre d'agents agrégés (écart-type)	1.5	1.56	1.61	1.65	1.71
Niveau 3					
Nombre d'agents	1	3	4	6	14
Densité (moyenne)	0	1.9	2.5	1.33	3.28
Densité (écart-type)	0	0.9	0.57	0.51	1.58
Nombre d'agents agrégés (moyenne)	4	3.33	4.5	4.66	2.67
Nombre d'agents agrégés (écart-type)	0	1.42	1.56	1.58	1.21
Niveau 4					
Nombre d'agents		1	1	2	5
Densité (moyenne)		0	0	1	2.83
Densité (écart-type)		0	0	0	1.32
Nombre d'agents agrégés (moyenne)		3	4	3	2.8
Nombre d'agents agrégés (écart-type)		0	0	1.41	1.30
Niveau 5					
Nombre d'agents				1	2
Densité (moyenne)				0	1
Densité (écart-type)				0	0
Nombre d'agents agrégés (moyenne)				2	2.5
Nombre d'agents agrégés (écart-type)				0	0.57
Niveau 6					
Nombre d'agents					1
Densité (moyenne)					0
Densité (écart-type)					0
Nombre d'agents agrégés (moyenne)					2
Nombre d'agents agrégés (écart-type)					0

Nombre d'agents agrégés : Nombre d'agents de niveau $n-1$ agrégés pour créer l'abstraction de niveau n.



t : temps (en secondes) V(t) : volume total de données échangées par les agents (en octets)

FIGURE 7.20 – Adaptation à l'ajout d'agents (configuration 1,2,3,4 et 5).

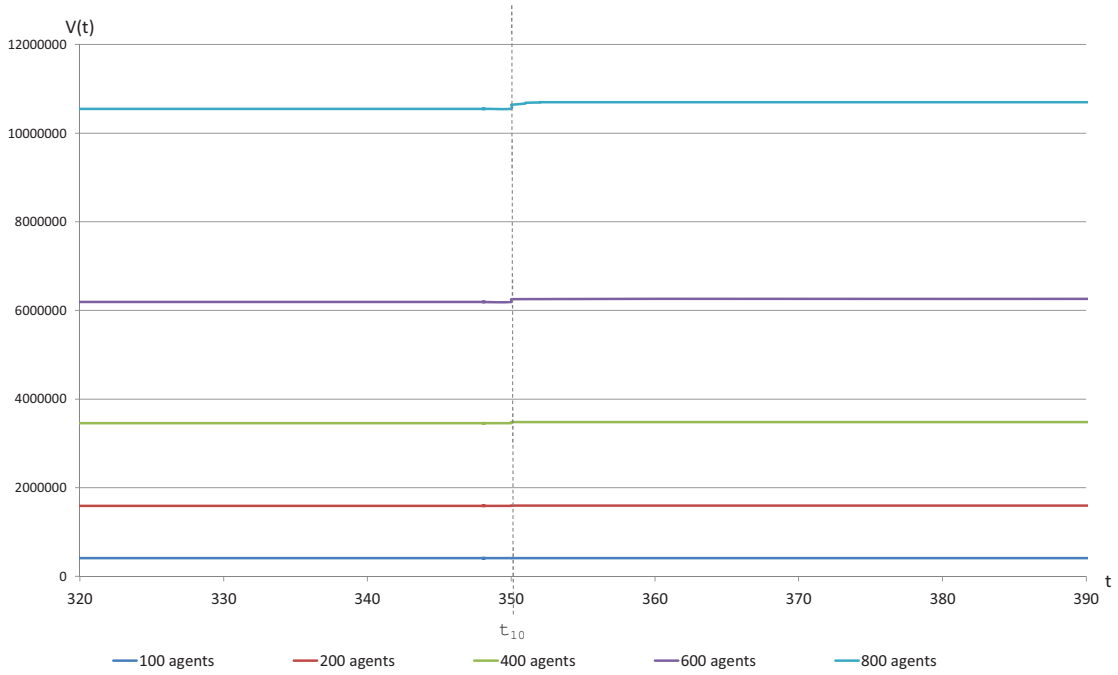
signale à ces voisins que l'entité n'existe plus.

Le coût pour détruire les abstractions est de 200 octets en moyenne par agent pour la configuration 1 (population de 100 agents) à 1800 octets pour la configuration 6 (population de 800 octets). Cela vient du fait que les agents qui sortent de l'organisation récursive n'étaient impliqués que dans 2 niveaux d'abstraction dans la première configuration contre 4 pour la dernière.

7.2.5 Conclusion

Nous avons présenté le coût de la construction d'une structure multi-niveau d'observation d'un réseau de capteurs organisé selon le modèle MWAC ainsi que son adaptation à l'ajout et au retrait d'agents. Même si ce coût peut paraitre globalement important, rapporté au nombre d'agent il apparait acceptable.

Dans notre simulation, nous n'avons pas introduit de mobilité au niveau des agents. Dans une première approche, on pourrait penser que ce problème se résume au retrait d'un agent à un groupe et à son ajout à un autre. Cependant, un problème subsisterait. Il serait nécessaire de définir un protocole de renommage des abstractions. En effet, à chaque abstraction est affecté un nom (une adresse) construit à partir de l'ensemble des noms des agents agrégés. Dans notre implantation, le nom choisi est celui de l'agent agrégé dont le rôle est "représentant". Si cet agent est déplacé, il peut donner naissance en un autre lieu du réseau à une abstraction de même niveau portant le même nom. Ce problème n'a pas été traité dans cette implantation du modèle.



t : temps (en secondes) V(t) : volume total de données échangées par les agents (en octets)

FIGURE 7.21 – Adaptation au retrait d'agents (configuration 1,2,3,4 et 5).

7.3 Évaluation méthodologique du modèle

Dans cette section, nous tentons de montrer la généricité et la réutilisabilité du modèle SMA-R et de son architecture récursive. Dans les sections précédentes, nous avons étudié l'application de notre modèle SMA-R à un système de réseau de capteurs sans fils - mettant en jeu des agents embarqués dans un contexte physique. Dans cette section, nous montrons l'application à une autre nature d'application mettant en jeu des agents logiciels dans le contexte des architectures logicielles. Notre objectif est de montrer la généricité du modèle SMA-R sur des applications de natures différentes afin de mettre en évidence les parties génériques réutilisables de notre framework mais aussi son mode de mise en œuvre. Nous allons appliquer le modèle SMA-R à la *composition de services web* qui est un sujet en cours d'étude dans l'équipe. Nous allons présenter dans un premier temps la notion de *composition de services web*. Ensuite, nous détaillerons l'analyse d'une application du modèle SMA-R et de son architecture à une ébauche de système de composition de services web.

7.3.1 Une application à la composition de services web

7.3.1.1 Introduction au contexte de la composition de service Web

Un service Web (selon W3C [2]) est un système logiciel conçu pour supporter les interactions entre les applications à travers le réseau. Les services Web fournissent un moyen standard pour l'interopérabilité entre différentes applications logicielles, en cours d'exécution sur une variété de plates-formes. Ils disposent d'une interface décrite dans un format machine transformable (spécifiquement WSDL-Web Services Description Language).

L'objectif des services web est de mettre des ressources à disposition sur l'Internet, via un protocole d'échange standardisé. Ces ressources sont accessibles à distance par des programmes écrits en des langages diverts qui s'exécutent sur une plate-forme quelconque. qu'une description interprétable/compréhensible automatiquement par la machine grâce au standard XML. Les autres systèmes d'interagir avec le service web de la manière prescrite par sa description en utilisant des messages SOAP (Simple Object Access Protocol), typiquement véhiculées via le protocole HTTP.

Les requêtes lancées sur l'Internet sont souvent complexes. Très souvent, la réalisation d'une requête complexe n'est pas effectuée par un seul service web. Il faut donc une composition de plusieurs services Web pour répondre à ces requêtes complexes. Une composition des services web est donc une combinaison des fonctionnalités de plusieurs services au sein d'un processus métier (Business Process) pour répondre à une demande complexe (**?**). Le problème de la composition de services Web est un défi important à cause de caractéristiques comme la distribution, l'hétérogénéité, l'évaluation, la dynamique... Un service Web peut être avantageusement agentifié (**?**), depuis des années, les SMAs sont utilisée comme un outil pour résoudre des problèmes de compositions de services web (**?**), (**?**), (**?**), (**?**).

La composition de service web est un problème complexe par sa nature dynamique et distribuée aux composants en nombre variable. Les services sont distribués et détenus par des fournisseurs différents. Ils sont hétérogènes et dynamiques. Les fournisseurs de services peuvent modifier ou supprimer leurs services, qui ne sont pas forcément disponibles au moment de la composition. dont la composition des services web est un sujet en cours d'étude. Nous essayons de défendre de l'application de notre framework aux problèmes de composition de service web (**?**). Nous reprenons les travaux d'Abrougui (**?**) dont nous ne retiendrons que l'aspect fonctionnel sans prendre en compte la gestion des contraintes de l'utilisateur hors de notre cadre d'étude. Ce travail propose un modèle en trois couches : la *Couche Service* contient des *agents services* encapsulant des services web ; la *Couche Composition* contient des agents métiers qui peuvent décider de composer ; la *Couche Application* contient un agent utilisateur.

2. World Wide Web Consortium, http ://www.w3c.org/

Nous nous intéressons à la *couche Service* du modèle où un SMA est utilisé pour modéliser les services web. Nous appliquons le modèle SMA-R sur cette couche. Les agents se composent grâce aux connaissances métiers fournies par l'utilisateur pour répondre à une requête complexe. L'objectif est de fournir une visualisation multi-niveau de requêtes faisant appel à une hiérarchie complexe de services. Nous allons présenter la couche Service et l'application de notre modèle SMA-R à cette approche de composition de services web.

7.3.1.2 Le SMA de la couche service

L'utilisateur lance une requête. Si la requête est complexe, un service web seul ne peut pas la résoudre.

La couche service contient des agents services qui peuvent participer à des compositions dont ils ont une représentation partielle. Des connaissances appelées "métiers" sont fournies par le concepteur. Il s'agit de "mots-clés" ou *sollicitations* correspondant à des thèmes que le service peut résoudre et des arbres partiels de tâches précisant les sous-tâches auxquelles le service peut participer. Ces connaissances "métiers" réalisent en fait une distribution dans les agents des tâches complexes (des plans partiels).

Nous analysons le SMA de la couche service selon une décomposition AEIO :

- Les agents (A) : Il y a autant d'*agents services* que de *services web*. Un *agent service as* est un agent qui :
  - encapsule un *service Web* réalisant une opération *OPSW* alors il connait : les paramètres *input* nécessaires à l'exécution du *service Web* ; les paramètres *output* résultant de l'exécution du *service Web*.
  - réalise une *sollicitation élémentaire* en invoquant par un objet opération OP_{SW}
  - possède des descriptions syntaxiques et sémantiques de son *services Web* fournies par le fournisseur du *service Web*.
  - possède des connaissances métier fournies par le concepteur.
  - possède des relations fonctionnelles avec les autres agents services.
- L'environnement (E) : Il contient les objets de sollicitations et opération qui réalisent les sollicitations :
  - un *Objet Sollicitation OS* est un objet créé par l'agent utilisateur pour propager la sollicitation de l'utilisateur ;
  - un *Objet Opération* OPW_S est un objet créé par un agent service pour regrouper les agents services qui ont le même objet opération. Chaque objet opération *OP* réalise une sollicitation ;
- Les Interactions (I) : les agents communiquent en échangeant des messages ;
- L'Organisation (O) : Les agents sont organisés en groupes. Tous les agents qui peuvent réali-

ser une même opération se regroupent. Un groupe contient donc un objet *opération*, un objet *sollicitation* et les *agents services* qui sont intéressés par l'objet opération. Il y a autant de groupes que de *sollicitations*.

La description du comportement de l'agent service

Une requête de l'utilisateur est traduite en un objet sollicitation *os* placé dans l'environnement. Au lancement les agents services sont dans l'état *désactivé*. Ils cherchent l'objet *os* pour changer son état et cherchent l'objet *op* pour changer son rôle. Leurs comportements sont détaillés ci-dessous :

- **Changement de l'état de** *os* : Les agents observent et détectent la présence du nouvel objet *os*. Dès qu'ils le trouvent, ils changent leur état à *actif*. Ils consultent ensuite l'objet sollicitation *os* pour récupérer la sollicitation *s* et déterminent s'ils peuvent participer à la réalisation de *s*. Dans ce cas, leur état devient *interessé* sinon ils sont *désactivés*.
- **Changement du rôle de** *op* : Les agents recherchent l'objet opération *op* correspondant à leur compétence et s'y connectent. S'ils ne trouvent pas d'objet *op* alors ils créent un nouvel objet opération *op* dans l'environnement (les autres agents qui ont la même compétence peuvent s'y connecter). Les rôles de l'agent service sont donc : *Créateur* s'il a créé *op* ; *Responsable* s'il est responsable de *op* et peut changer ou supprimer *op* ; *Membre* s'il est connecté au groupe grâce à *op*.

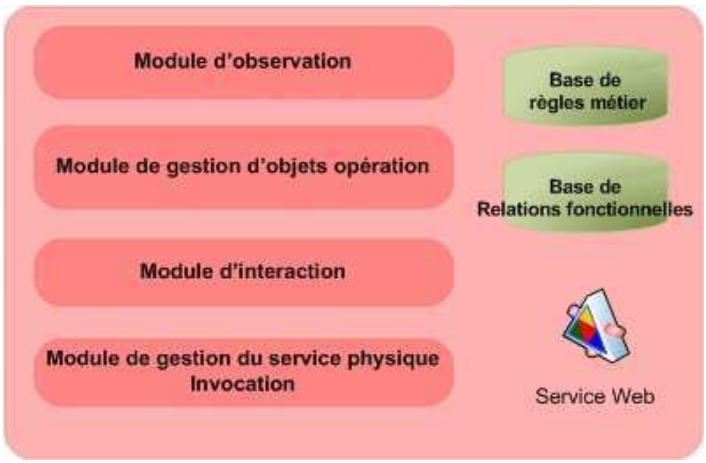


FIGURE 7.22 – L'architecture fonctionnelle de l'agent service (d'après A. Abrougui)

7.3.2 Une observation multi-niveau pour le système de composition de services

Nous proposons d'utiliser le modèle SMA-R pour la construction et l'observation de la composition. Notre modèle envisage de regrouper les services existants qui satisfont les tâches et de les composer afin de réaliser le service. Un agent récursif représente une composition des services

web (via les agents récursifs du niveau inférieur) qui peut résoudre une partie du service demandé. Un niveau abstrait donne aussi une vision abstraite de la composition des services web qui peuvent participer à la réalisation du service demandé par l'utilisateur.

7.3.2.1 La modélisation avec SMA-R

Pour ce problème, nous avons décidé de prendre l'organisation d'un groupe d'objet opération *op* comme base pour les règles déclenchant la création des agents abstraits. Un groupe associé à l'objet *op* contient cet objet et les agents connectés à l'objet. La création de l'objet *op* est basée sur les connaissances de l'agent service. La règle de *composition* est dépendante des *relations fonctionnelles* de l'agent service.

L'idée principale est que les groupes sont formés :

- quand un agent a une organisation stabilisée,
- que son objectif n'est pas satisfait, (l'agent ne peut pas réaliser la requête globale de l'utilisateur),
- que l'agent a trouvé dans son groupe tous les agents nécessaires pour réaliser une sous-sollicitation de la requête.

Chaque groupe construit un nouvel agent abstrait. La propriété récurrente *PA* construit une liste des agents et un objet opération *op* auquel les agents se connectent. Les objets *os* ne sont pas transformés. Ils sont présentés identiquement à tous les niveaux.

Les fonctions dynamiques, les conditions de *composition* et *réduction* de l'agent récursif appliquées au système service web sont détaillées comme suit :

1. les fonctions dynamiques : $F_{Complexité}$ (algorithme 16), F_{SA} (algorithme 17), $F_{Satisfaction}$ (algorithme 18),

Algorithme 16 FComplexité

1: $F_{Complexit}$ *.init(0)*
2: Si dans le groupe, il y a un *op*, si son rôle est *Créateur* ou *Responsable*
et son état est *intéressé* et son groupe peut réaliser l'*op* sans avoir besoin aucun d'autre agent,
3: alors $F_{Complexit} = 1$

Algorithme 17 F_{SA}

1: F_{SA}.init(0)
2: Si durant la période de *sûreté*, il y a des changements de rôles et de la liste des agents services intéressés par l'opération *op*
3: Alors l'agent est instable $F_{SA} = 0$
4: S'il n'y a aucun changement, l'agent est stable, alors $F_{SA} = 1$

La fonction F_{SS} utilisée est celle par défaut.

Algorithme 18 $F_{Satisfaction}$

1: $F_{Satisfaction}$.intit(0)
2: Si durant le *temps d'attente*, il peut résoudre la sollicitation *s* sans besoin d'autre agent, alors $F_{Satisfaction} = 1$

2. La condition de *composition* (algorithme 19).

Algorithme 19 La condition de *composition*

1: *(getNext()==null)* et
2: *($F_{Stabilité}$==1) et ($F_{Complexité}$==1)*
3: *et ($F_{Satisfaction}$==0)*

3. la condition de *réduction* (algorithme 20).

Algorithme 20 La condition de *réduction*

1: *(getPrevious() !=null)* et
2: *($F_{Stabilité}$==1) et ($F_{Complexité}$==0)*

Les primitives de *composition* et de *réduction* ainsi que d'*interaction récursive* sont génériques.

7.3.2.2 Un cas d'illustration

Nous prenons l'exemple de l'*isolation des combles* d'une maison qui est en cours de développement dans les travaux d'Abrougui. Nous supposons que chaque tâche donne lieu a des services web d'information, de conseil, de réservations, d'achat... Les services suivants peuvent participer à la réalisation du service complexe, certains services sont obligatoires et d'autres sont optionnels :
1. la recherche de conseil pour acheter les matériaux,
2. l'achat des matériaux,
3. Le transport des matériaux,
4. la recherche de guides pour la pose de l'isolant,
5. la pose de l'isolant,
6. le paiement du service.

La Figure 7.23 présente la structure des connaissances pour l'isolation des combles.

Dans cette structure, afin de mettre en œuvre une isolation, il faut l'acheter et la poser. Pour la pose, on peut soit demander à un expert de pose, soit le faire soi-même en consultant un guide de pose. Pour la pose par un expert, il faut chercher l'artisant, le réserver et le payer... Pour la livraison de la marchandise, on peut faire appel à un transporteur ou bien louer un véhicule et transporter soi-même la marchandise. Le service "Isolation" est décomposée en des sollicitations. Les sollicitations sont présentées dans la Table 7.6

Ordre	Sollicitation
1	Isolation
2	Pose
3	Achat Matériau
4	Pose Expert
5	Recherche Guide
6	Livraison
7	Location Véhicule
8	Transport
9	Paiement Achat Matériau
10	Commande Achat Matériau
11	Catalogue Achat Matériau
12	Conseil Expert
13	Catalogue Transporteur
14	Paiement Transporteur
15	Réservation Transporteur
16	Catalogue Véhicule
17	Réservation Véhicule
18	Paiement Location Véhicule
19	Recherche Expert
20	Réservation Expert
21	Paiement Pose Expert

Tableau 7.5 – Les sollicitations pour l'isolation des combles

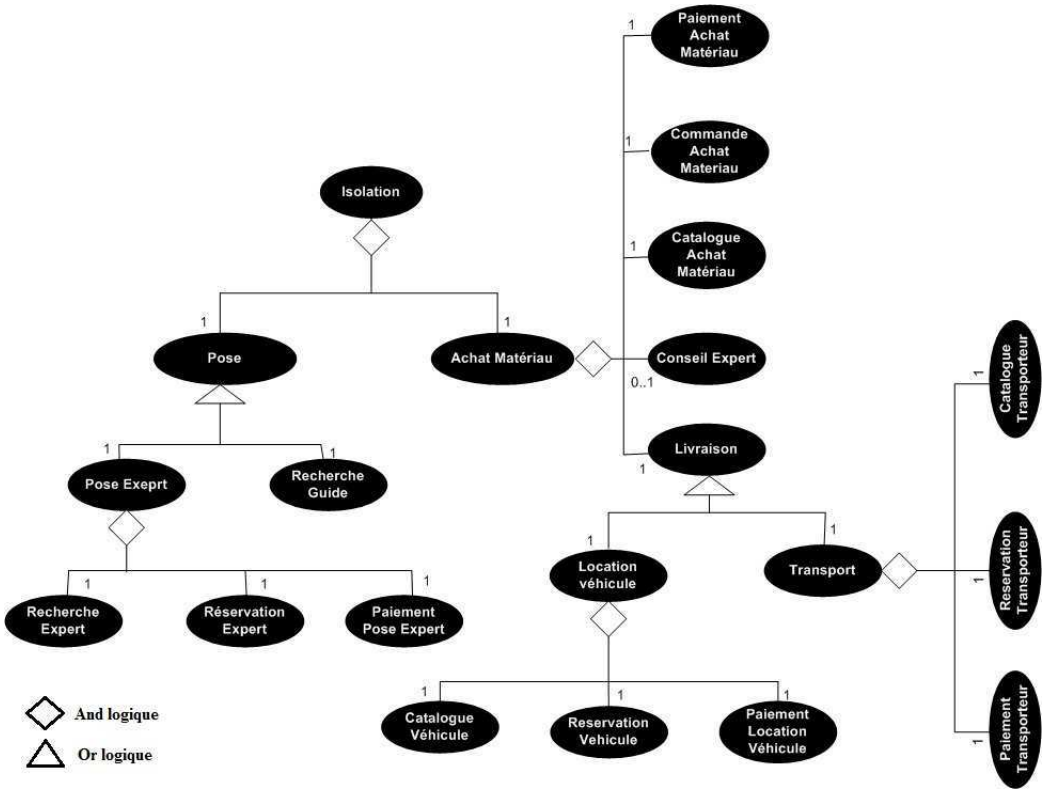


FIGURE 7.23 – Structure des connaissances métiers fournies par le concepteur

Parmi les sollicitations dans la Table 7.6, il y a les sollicitations élémentaires qui sont résolues par un seul service web. Ces services web sont encapsulés par des *agents services*. Ils peuvent réaliser l'opération équivalente. Les agents services web consultent et récupèrent les connaissances métiers liées à leur opération. La Table 7.6 présente les sollicitations élémentaires, les services web disponibles (nous supposons qu'il y a trois services web disponibles pour résoudre une sollicitation élémentaire), les agents services, les objets opération créés par les agents services web et les connaissances des agents.

Pour atteindre le but global *Isolation*, pas à pas, les agents services récursifs créent les *objets opération* en regardant dans leurs connaissances métiers. Lorsqu'un objet opération est créé par un agent récursif quelconque, les autres agents services abstraits détectent la présence du nouvel objet et s'y connectent s'ils peuvent participer à réalisation de l'opération. Nous pouvons voir dans la Table 7.7, à partir des connaissances métiers récupérées dans l'objet sollicitation *os*, les agents services abstraits créés. Les agents services abstraits se composent suivant la règle de *composition* implémentée dans l'*agent service récursif* :

– Le niveau 1 contenant quatorze agents récursifs est construit grâce au regroupement des agents services élémentaires. Ces agents créent quatre objets *op* (figure 7.24 et les connaissances de la Table 7.7) :

Ordre	Sollicitation	Service Web	Agent service	Objet d'operation	Connaissances
1	Catalogue Achat Matériau	$CA_{1,2,3}$	$as_{CA1,2,3}$	Op_{CA}	Catalogue Achat Matériau → Achat Matériau → Isolation
2	Recherche Expert	$CE_{1,2,3}$	$as_{CE1,2,3}$	Op_{CE}	Recherche Expert → Pose Expert → Pose → Isolation
3	Recherche Guide	$CG_{1,2,3}$	$as_{CG1,2,3}$	Op_{CG}	Guide Recherche Guide → Pose → Isolation
4	Conseil Expert	$CO_{1,2,3}$	$as_{CO1,2,3}$	Op_{CO}	[Conseil Expert] → Achat Matériau → Isolation
5	Catalogue Transporteur	$CT_{1,2,3}$	$as_{CT1,2,3}$	Op_{CT}	Catalogue Transporteur → Transport → Livraison → Achat Matériau → Isolation
6	Catalogue Véhicule	$CV_{1,2,3}$	$as_{CV1,2,3}$	Op_{CV}	Catalogue Véhicule → Location Véhicule → Livraison → Achat Matériau → Isolation
7	Paiement Achat Matériau	$PA_{1,2,3}$	$as_{PA1,2,3}$	Op_{PA}	Paiement Achat Matériau → Achat Matériau → Isolation
8	Paiement Pose Expert	$PE_{1,2,3}$	$as_{PE1,2,3}$	Op_{PE}	Paiement Pose Expert → Pose Expert→ Pose → Isolation
9	Paiement Location Véhicule	$PL_{1,2,3}$	$as_{PL1,2,3}$	Op_{PL}	Paiement Location Véhicule → Location Véhicule → Livraison → Achat Materiau → Isolation
10	Paiement Transporteur	$PT_{1,2,3}$	$as_{PT1,2,3}$	Op_{PT}	Paiement Transporteur → Transport → Livraison → Achat Materiau → Isolation
11	Réservation Expert	$RE_{1,2,3}$	$as_{RE1,2,3}$	Op_{RE}	Réservation Expert → Pose Expert → Pose → Isolation
12	Commande Achat Matériau	$CM_{1,2,3}$	$as_{CM1,2,3}$	Op_{CM}	Commande Achat Matériau → Achat Matériau → Isolation
13	Réservation Véhicule	$RV_{1,2,3}$	$as_{RV1,2,3}$	Op_{RV}	Réservation Véhicule → Location Véhicule → Livraison → Achat Matériau → Isolation
14	Réservation Transporteur	$RT_{1,2,3}$	$as_{RT1,2,3}$	Op_{RT}	Réservation Transporteur → Transport → Livraison → Achat Materiau → Isolation

Tableau 7.6 – Sollicitations élémentaires, Services web et Agents services correspondants

Ordre	Sollicitation	Composition des services web	Agent service récursif	Objet d'opération	Connaissances
1	Pose Expert	*PO*	as_{PO}	Op_{PO}	Pose Expert → Pose → Isolation
2	Location Véhicule	*LV*	as_{LV}	Op_{LV}	Location Véhicule → Livraison → Achat Matériau → Isolation
3	Transport	*TP*	as_{TP}	Op_{TP}	Transport → Livraison → Achat Matériau → Isolation
4	Pose	*PS*	as_{PS}	Op_{PS}	Pose → Isolation
5	Livraison	*LIV*	as_{LIV}	Op_{LIV}	Livraison → Achat Matériau → Isolation
6	Isolation	*IS*	as_{IS}	Op_{IS}	Isolation

Tableau 7.7 – Sollicitations et connaissances des agents services pour l'isolation des combles

1. l'objet op_{PO} réalise la sollicitation *Pose Expert*, il intéresse les agents *PE*, *CE* et *RE* qui se composent et créent l'agent abstrait *PO* du niveau 2 ;
2. l'agent *LV* réalise la sollicitation *Location Véhicule*. L'objet op_{LV} intéresse les agents *CV, RV* et *PL* qui se composent et créent l'agent abstrait *LV* du niveau 2 ;
3. l'agent *PT* réalise la sollicitation *Transport*. L'objet op_{TP} intéresse les agents *PT, RT* et *CT* qui se composent et créent l'agent abstrait *TP* du niveau 2 ;
4. l'agent *AM* réalise la sollicitation *Achat Matériau*. L'objet op_{AM} intéresse les agents *CA, PA, CO* et *CM* qui ne se composent pas car ils ont besoin d'un autre service.

– Le niveau 2 contient trois agents récursifs qui créent deux objets *op* (figure 7.25) :

1. l'agent *PS* réalise la sollicitation *Pose*. L'objet op_{PO} intéresse l'agent *PO* et l'agent *CG* du niveau 1 qui se composent et créent l'agent abstrait *PS* du niveau 3 ;
2. l'agent *LIV* réalise la sollicitation *Livraison*. L'objet op_{LIV} est intéressé par les agents *LV* et *TP* qui se composent et créent un agent récursif *LIV* du niveau 3 ;

– Le niveau 3 contient 2 agents récursifs. Quatre agents *CA, PA CO* et *CM* du niveau 1 et l'agent *LIV* du niveau 3 sont connectés à l'objet *op* OP_{AM} et se composent en l'agent *AM* du niveau 4.
– Le niveau 4 contient un agent récursif. Il crée op_{IS} qui réalise la sollicitation *Isolation*, cet objet op_{PO} intéresse l'agent *PS* du niveau 3 et l'agent *AM* du niveau 4 qui se composent et créent l'agent abstrait *IS* du niveau 5 ;
– Le niveau 5 contient un seul agent abstrait *IS* qui peut réaliser la requête de l'utilisateur.

Grâce aux connaissances des agents services, les agents récursifs construisent les niveaux de composition abstraits pour réaliser la requête "*Isolation*". La Table 7.7 montre comment les agents

services récursifs composés sont créés selon les connaissances métiers récupérées dans l'objet sollicitation *os*. Les agents services récursifs se composent suivant la *règle de composition* implémentée dans l'*agent service récursif*. Les résultats sont illustrés dans les figures 7.24 et 7.25.

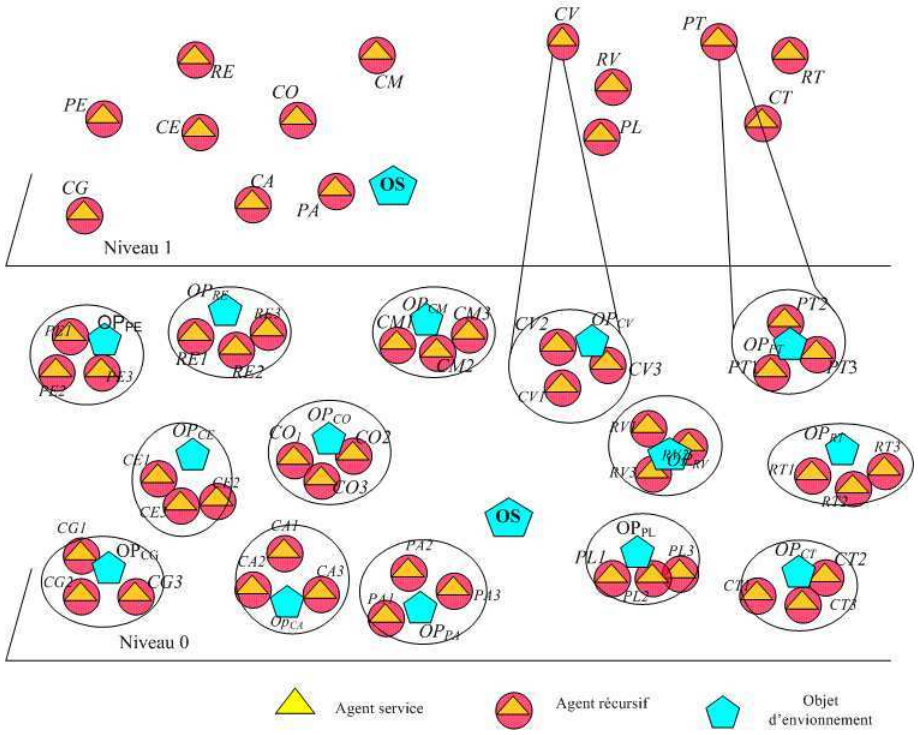


FIGURE 7.24 – Les agents services récursifs du niveau 0 et la construction du niveau 1

7.3.3 La réutilisabilité de SMA-R

Nous avons appliqué notre modèle SMA-R au système de composition des services web - un système complexe artificiel. Nous avons pris l'exemple de l'*Isolation des combles* pour montrer comment appliquer notre modèle et notre framework. Notre système SMA service web récursif trouve des compositions adéquates pour la demande de l'utilisateur. Le SMA récursif donne une visualisation de la composition pour arriver à réaliser le service global. Cette étude montre que notre modèle peut être appliqué à plusieurs sortes de systèmes complexes munis des structures multi-échelles dans des contextes différents.

– **Une partie du modèle dépendante du contexte des applications :**
  Après avoir étudié les deux applications présentées, nous avons montré précédement que les *fonctions dynamiques*, la détermination du *groupe de composition*, les *interactions des messages récursifs* qui portent les messages applicatifs fonctionnels (comme les messages de demande de mesure de la température et de retour du résultat dans l'application de réseau de capteur sans fil) sont dépendants des applications. Ils trouvent place dans les parties *K* et *IR*

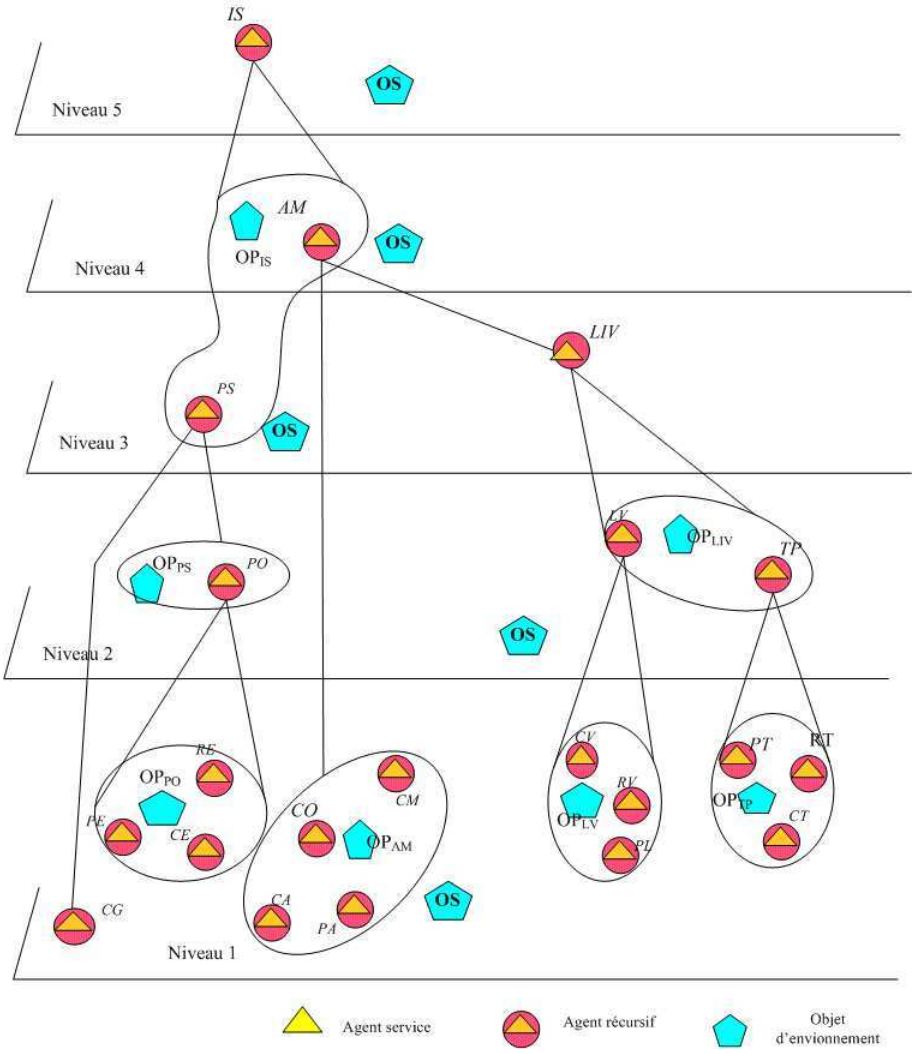


FIGURE 7.25 – La construction des niveaux 2 à 5

du modèle.

– **Une partie du modèle générique :** Le modèle propose des *parties génériques* comme : *la fonction d'observation, le mécanisme de Composition et le mécanisme de Réduction*. La fonction d'*observation* utilise le calcul des *fonctions dynamiques*. Grâce aux valeurs des fonctions dynamiques, cette partie déclenche les mécanismes *Composition* ou *Réduction*. Ces mécanismes trouvent place dans les parties *O*, *C* et *R* du modèle.

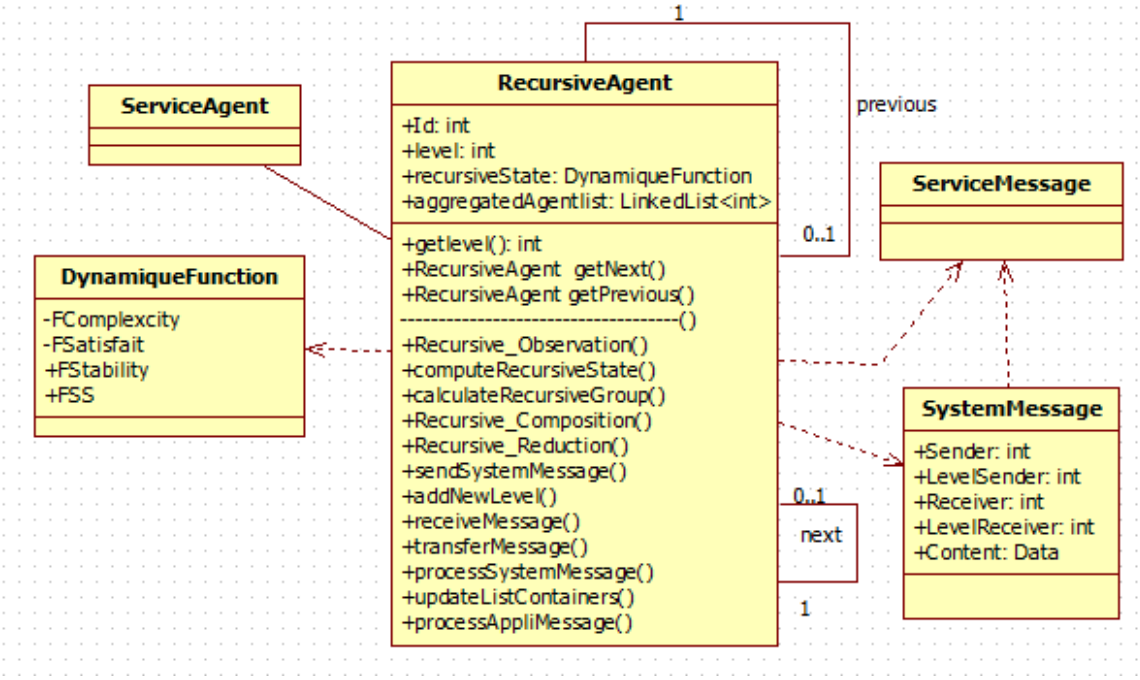


FIGURE 7.26 – Diagramme de classes d'un *Agent service web récursif*

Utilisation du framework pour l'implémentation du modèle SMA-R

Le diagramme de classes dans la figure 7.26 implémente le modèle SMA-R sur le système de composition des services web. Ce diagramme contient six classes dont deux classes *ServiceAgent* et *ServiceMessage* sont des classes de l'application. Les quatre classes, qui restent, viennent du modèle SMA-R. La classe *DynamiqueFunction* est indépendante de l'application. Dans la classe *RecursiveAgent*, les méthodes *computeRecursiveState()*, *calculateRecursiveGroup()*, *processSystemMessage()* et *processApplliMessage()* sont dépendantes des applications. Ici, les concepteurs doivent participer au développement de ces fonctions selon le contexte de l'application et les spécifications du concepteur. Nous décrivons les méthodes dela classe *RecursiveAgent* qui ont besoin de l'inter-

vention du concepteur pour une personnalisation selon les applications :

- La méthode *computeRecursiveState()* calcule les *fonctions dynamiques* qui sont basées sur les états des agents applicatifs et leur organisation ;
- La méthode *calculateRecursiveGroup()* cherche les agents dans l'organisation applicative pour les inviter à se composer ;
- La méthode *processAppliMessage()* renvoie les messages applicatifs à l'agent applicatif, elle doit donc être interfacée avec l'agent applicatif ;
- La méthode *processSystemMessage()* traite les messages récursifs portant les messages applicatifs fonctionnels, elle doit récupérer et transformer les valeurs retournées par les messages applicatifs.

Les méthodes génériques dans la classe *RecursiveAgent* sont : *Recursive_Observation(), Recursive_Composition(), Recursive_Reduction(), sendSystemMessage(), addNewLevel(), receiveMessage(), transferMessage(), updateListContainers()*. Elles sont indépendantes du contexte des applications. Elles sont réutilisables pour toutes les applications.

7.4 Conclusion

Ce chapitre a été consacré à l'évaluation et la validation de notre modèle SMA-R. Nous nous somme intéressés à trois aspects d'évaluation du modèle qui sont : l'évaluation fonctionnelle du modèle, l'évaluation des performances du modèle et l'évaluation méthodologique du modèle.

Concernant l'évaluation fonctionnelle du modèle, nous l'avons dédié à la visualisation et à l'interaction avec l'utilisateur. Grâce aux agents récursifs, le système construit des niveaux abstraits dont chacun nous donne une visualisation abstraite du système. A un niveau plus abstrait, le système est moins complexe. Quant à l'interaction avec l'utilisateur, on sait que les agents récursifs jouent le rôle de représentant des groupes d'agents applicatifs dans une zone précise. Nous avons présenté comment une demande de l'utilisateur donnée à un agent récursif peut se propager jusqu'aux agents applicatifs de la zone correspondante. De plus, nous avons montré comment le résultat de la demande de l'utilisateur est retourné à l'agent récursif par lequel l'utilisateur lance la demande.

L'évaluation de performance du modèle concerne les réactions du système aux modifications du niveau 0, l'impact des règles de construction et de réduction à la construction des niveaux d'abstraction et le rendement des messages récursifs par rapport aux volumes des messages du système non appliqué le modèle SMA-R. Pour s'adapter à la caractéristique d'ouverture des systèmes complexes, le modèle SMA-R doit réagir dynamiquement aux changements du système au niveau applicatif, c'est-à-dire au niveau 0. Nous avons étudié l'application de réseau de capteurs sans fils MWAC

pour l'évaluation de la performance de notre modèle. Nous avons appliqué notre modèle à plusieurs configurations et lancé des scénarios de disparition et d'ajout d'agents en cours d'exécution. Suivant les rôles des agents MWAC, les règles de composition et de réduction, l'ajout et la suppression des agents provoquent le changement des niveaux abstraits. La construction des niveaux est également dépendante des règles de construction et de réduction données par le concepteur. Pour l'instant, nous n'avons pas tenté d'améliorer l'efficacité du système. Poutant, malgré l'accroissement du nombre des messages récursifs qui circulent dans le système pour construire des niveaux abstraits, le rendement des messages est supportable.

Pour l'évaluation de la méthodologie du modèle, nous avons appliqué le modèle à une autre nature d'application mettant en jeu des agents logiciels - l'application de *composition de services web*. Nous n'avons mené à bien que l'analyse de ce problème, mais par l'application du modèle à deux systèmes existants, nous avons voulu montrer la généricité du modèle SMA-R et mis en évidence les parties génériques réutilisables de notre framework mais aussi son mode de mise en œuvre. Les parties génériques et les parties dépendantes des applications sont clairement identifiées. Le framework fournit le cadre et les parties génériques réutilisables, le concepteur sera invité à participer au développement des parties dépendantes de l'application.

Conclusions et Perspectives

Contexte

L'origine de cette thèse est un intérêt pour des problèmes de modélisation, de supervision et d'observation à plusieurs niveaux de systèmes informatiques décentralisés auto-organisés que nous qualifions de systèmes complexes artificiels. Parmi ces systèmes, se trouvent les travaux sur des systèmes physiques comme les réseaux de capteurs sans fils et des systèmes de services logiciels.

La supervision de ces systèmes (leur observation et leur contrôle) devient difficile dès lors qu'ils atteignent des tailles importantes. Une des solutions que nous avons envisagé est de les voir et d'agir sur eux à travers plusieurs niveaux d'abstraction. Plus particulièrement, en fait, il s'agit de les considérer comme des systèmes complexes multi-échelles.

Nous avons adopté très naturellement une approche multi-agent pour la modélisation de ces systèmes.

Nous avons défendu dans ce travail que l'intégration de l'aspect récursif dans le modèle est une façon de construire des niveaux d'observation dont les dépendances et les déductions entre les niveaux sont basées sur les propriétés récurrentes. Ces formulations récursives nous emmènent à construire un mécanisme générique récursif permettant d'établir une architecture générique de SMA récursif pour répondre à des exigences des systèmes multi-échelles. Nous avons développé, pour ce modèle, un framework décentralisé générique permettant aux agents récursifs de communiquer avec les agents réels pour construire plusieurs niveaux d'observation.

Contributions

Ce travail a débouché sur plusieurs contributions :

Des préconisations pour un modèle multi-agent multi-niveau générique : Notre première tâche a été de caractériser les besoins à satisfaire pour construire un système complexe multi-échelle. Nous avons examinés de nombreuses approches multi-échelles dans les domaines d'application les plus variés. Il est apparu indispensable de fournir des capacités pour :

– La construction dynamique des niveaux,

– L'observation de chaque niveau,
– L'interaction entre les niveaux,
– L'insertion ou la suppression d'un élément sur n'importe quel niveau,
– L'action sur n'importe quel niveau.

Convaincus de la pertinence de l'approche multi-agent pour les systèmes complexes artificiels multi-échelles, nous avons tenté d'établir des préconisations pour un modèle multi-agent multi-niveau. La généricité de notre approche étant un pré-requis incontournable, nous avons, pour cela, analysé et comparé des approches multi-agents génériques utilisées ou utilisables dans un contexte multi-niveau. Nous avons retenu les critères qui sont devenu la base du modèle :

– une architecture récursive,
– une création et une adaptation dynamique et ouverte des niveaux en fonction du problème
– une architecture décentralisée
– une interaction inter-niveau par message
– une interaction avec l'environnement par l'intermédiaire du niveau le plus bas

Cette étude et les préconisations qui en résultent constituent la première contribution de cette thèse.

Une tentative de définition formelle d'un SMA récursif : Nous avons proposé une vision de l'abstraction d'un SMA, proposé un modèle de SMA récursif capable de l'exprimer et nous avons tenté d'en donner une expression plus formelle qui constitue une deuxième contribution de ce travail. Nous avons introduit des propriétés récurrentes sur les quatre axes A, E, I et O de décomposition d'un SMA comme un moyen d'induire les différents niveaux d'observation du SMA. Le passage entre les niveaux se fait grâce aux propriétés récurrentes *PA* et *PE*. Les interactions sont transformées grâce à la propriété *PI*. La propriété *PO* transforme l'organisation entre les niveaux.

Le modèle SMA-R : La troisième contribution de cette thèse est le modèle de SMA récursif générique **SMA-R**. Basé sur un modèle d'agents récursifs SMA-R permet de construire des niveaux abstraits et fournit des mécanismes génériques pour leur adaptation. Une gestion récursive de l'interaction entre les niveaux permet à l'utilisateur d'agir sur n'importe quel niveau abstrait.

L'architecture décentralisée pour SMA-R : Notre but est d'appliquer ce modèle à des SMA physiquement décentralisés. Nous avons adopté les concepts du modèle OSI qui met en avant les capacités que nous cherchons : multi niveau, encapsulation, échange de messages virtuels et physiques... Les niveaux abstraits du système sont organisés en couches enfichées sur chaque agents applicatifs sous la forme d'un middleware réparti. Les messages transitent au travers des couches récursives adjacentes jusqu'à leur couche élémentaire chargée de l'acheminement vers l'agent applicatif. Les mécanismes de cette architecture forment la quatrième contribution de ce travail.

Résultats Pratiques

Ces modèles fondamentaux ont conduits à des réalisations pratiques :

Un framework générique : Nous avons construit un framework générique qui implémente l'architecture décentralisée du modèle SMA-R. Ce framework est réutilisable et peut s'appliquer à différentes applications. Ce framework présente une partie générique indépendante de l'application et une partie confiée au concepteur dépendante de l'application. Il est fourni sous la forme d'un package JAVA ;

Une extension de l'outil MASH : L'outil MASH existant dans l'équipe pour la simulation et la mise au point de SMA a été étendu à la visualisation d'une organisation multi-niveau. L'outil permet d'observer le système à un niveau choisi à l'aide d'un curseur de niveau. Cette extension a été utilisée avec l'intégration du framework pour visualiser des niveaux abstraits de l'application "réseau de capteurs sans fils" et réaliser nos expérimentations ;

Deux applications : Le modèle SMA-R a été appliqué à un premier problème, le SMA de supervision de réseau de capteurs sans fil MWAC, dont le développement complet a été réalisé sur l'outil MASH. Il constitue une application à un système physique. L'analyse d'une seconde application complète les réalisations : le système de composition des services web qui adresse un système logiciel. Seule l'analyse de ce cas a été menée à bien, pour montrer comment le framework peut se réutiliser.

Limitations et améliorations

Cependant, malgré les résultats obtenus dans cette thèse, nous reconnaissons certaines limitations qui ne sont pas encore résolues. Nous présentons par la suite les limitations considérées et les améliorations qui peuvent être apportées tant au niveau théorique que méthodologique ou technique. D'un point de *théorique* nous pensons que la vision de l'abstraction d'un SMA telle que nous l'avons introduite est originale. L'utilisation des transformations appliquées récursivement pour générer des niveaux d'abstraction en est le point central. Les propriétés de récurrence sont définies pour chacunes des dimensions A, E, I, O. Malheureusement, nos exemples portent sur des cas où seuls les dimensions A ou E et quelquefois I sont récursifs. Trouver un exemple contenant quatre axes A, E, I, O récursifs simultanément serait un plus pour valider plus avant notre modèle.

L'approche est conçue pour des systèmes complexes artificiels éventuellement en relation avec le monde physique modélisés sous la forme d'un SMA. Ce choix impose certaines caractéristiques sur la nature du SMA comme une interaction par messages et une granularité importante. Cela limite donc le caractère universel du modèle qui ne peut d'emblée pas s'appliquer à des SMA réactifs et donc ne couvre pas les champs de la simulation SMA de système complexes naturels où l'approche réactive est de mise.

Du coté *méthodologique*, il est, actuellement obligatoire, d'appliquer le traitement par abstraction sur l'ensemble du SMA de base, même pour en exploiter qu'une partie. Cela peut poser des problèmes si la taille de ce SMA est très importante, non pas en termes de performances, puisque le framework est complètement décentralisé, mais en termes pratiques, puisque une observation partielle ne peut se faire sans lancer le processus sur l'ensemble du système.

Au niveau *technique*, nous avons réalisé une évaluation sur l'application du réseau de capteurs sans fils. Nous avons obtenu des résultats satisfaisants en terme de performances confirmant la validité de l'idée défendue dans cette thèse. Dans nos évaluation nous considérons une faible dynamicité de la population d'agent, ce qui est le plus souvent le cas dans les domaines visés. Une très forte dynamicité dans le SMA de base pourrait remettre en question certains choix et ces performances :

- l'organisation deviendrait très coûteuse à maintenir , en particulier si la topologie était soumise à une forte mobilité des noeuds, car il faudrait adapter sans cesse l'organisation de base donc les niveaux. Cela serait le cas pour les capteurs dans un environnement ad-hoc ou dans une moindre mesure pour des services à interchanger dans un éco-système de services ambiants.
- la convention de nommage des groupes devrait être revue, car actuellement on identifie les agents abstraits par référence à l'agent du niveau inférieur ayant initié la composition. Une forte mobilité des agents impliquerait des conflits de noms.

Dans tous les cas, un travail sur l'amélioration de l'efficacité du système peut être menée. Une contribution sur ce point pourrait être de contourner le traitement récursif des messages dans le cas où les transformation sur l'interaction sont définies par l'identité. Cela en éviterait l'encapsulation multiple par l'établissement de tunnels et diminuerait le nombre de ré-émissions.

Une réalisation complète a été réalisée sur l'application du réseau de capteurs sans fils. Le framework générique a été "pluggé" sur le SMA existant MWAC qui contient des agents embarqués. Dans cet exemple, seul les agents sont récursifs, l'environnement n'est pas récursif, c'est à dire que la transformation entre les niveaux est en fait l'*Identité*. Même si les mécanismes sont prévus et implémentés, les interférences éventuelles entre les effets croisés sur les A E I et O mériterait d'être plus complètement testées sur des cas plus complets.

Perspectives

Nos perspectives à plus long terme peuvent aussi être déclinée selon les axes théoriques, méthodologiques et techniques.

Au niveau *théorique* tout d'abord, il serait intéressant d'étudier les modalités de cohabitation des abstractions parallèles. Cela reviendrait en pratique à autoriser l'observation du système selon plu-

sieurs vues simultanées. Conceptuellement, on peut affirmer cela possible, mais la viabilité mérite d'être étudiée.

L'étude des aspects temporels est aussi un sujet d'intérêt, le temps pouvant être important pour les application visées.

Plus loin, il serait intéressant de revenir sur les aspects formels de la récursivité sur les composante A, E, I et O. Nous avions ouvert quelques pistes sur la base des travaux de notre master (**?**), qu'il serait intéressant de creuser dans l'avenir.

D'un côté *méthodologique*, il pourrait être intéressant de rendre le framework plus facilement utilisable en complétant l'outil MASH par une fonctionalité d'aide à la définition des propriétés récursives de façon à rendre plus systématique la possibilité d'utiliser le modèle SMA-R. Il faudrait pour cela contraindre beaucoup plus la spécification de certaines méthodes laissées à la libre appréciation du concepteur dans l'état actuel du framework. Cela pourrait être envisagé en restreignant l'utilisation à une catégorie spécifique d'application, par exemple celles des réseaux de capteurs que nous circonscrivons le mieux.

Dans un souci de supervision plus automatisée, on peut aussi se pencher sur le couplage de l'abstraction par des systèmes automatisés de surveilance ou des SIG. Un travail pourra être mené sur la nature des informations échangées avec les systèmes et sur les modes d'interaction avec les systèmes de décisions.

Mais la principale perspective est d'éprouver SMA-R dans des *développements futurs* sur différents types d'application afin d'en roder la mise en oeuvre générique. Cela peut démarrer avec ce qui concerne la composition des services web pour laquelle il faut trouver plusieurs scénarios et implémenter l'application en utilisant le framework. Nous espérons aussi très rapidement le diffuser, et dans un premier temps, sur les différentes applications en cours de l'équipe dans ces domaines en particulier dans la supervision de réseaux de distribution d'eau [3] ou d'environnements instrumentés [4].

3. Projet PHC-Tassili 11MDU838.
4. Projet ECOS-Nord M10-M02

Bibliographie

C. Athale and T. S. Deisboeck. The effects of egf-receptor density on multiscale tumor growth patterns. *Journal of Theoretical Biology*, 238(4) :771–779, 2006.

C. Athale, Y. Mansury, and T. S. Deisboeck. Simulating the impact of a molecular 'decision-process' on cellular phenotype and multicellular patterns in brain tumors. *Journal of Theoretical Biology*, 233(4) :469–481, 2005.

Emmanuel Adam, Rene Mandiau, and Christophe Kolski. Application of a holonic multi-agent system for cooperative work to administrative processes 1. *Journal of Applied Systems Studies*, 2(1998) :1–15, 2001.

Anissa Abrougui, Annabelle Mercier, Michel Occello, and Marc-Philippe Huget. Recursive multi-agent system for dynamic and adaptative web services composition. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems (MEDES'09)*, pages 295–299, 2009.

Harold Abelson and Gerald J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, USA, 2nd edition, 1996.

C. Baeijs. *Fonctionnalité Emergente dans une Société d'Agents Autonomes*. Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, 1998.

B. Brugger, R. Barrera, A. Frank, Beard K., and M Ehlers. Research topic on multiple representations. In *Workshop on Multiple Representations Initiative 3*, pages 53–67, 1989.

Boualem Benatallah, Fabio Casati, Farouk Toumani, and Rachid Hamadi. Conceptual modeling of web service conversations. In *Proceedings of the 15th international conference on Advanced information systems engineering*, pages 449–467. Springer, 2003.

C. Bertelle, M. Flouret, V. Jay, D. Olivier, and J.-L. Ponty. Automata with multiplicities as behaviour model in multi-agent simulations. In *The 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'2001)*, Orlando, Florida, USA, July 2001.

V. Botti and A. Giret. *ANEMONA : A Multi-agent Methodology for Holonic Manufacturing Systems*. Springer Series in Advanced Manufacturing. Springer-Verlag, 2008.

Ajay Bansal, Srividya Kona, M Brian Blake, and Gopal Gupta. An agent-based approach for composition of semantic web services. In *IEEE 17th Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises*, pages 12–17. IEEE, 2008.

C. Bertelle, S. Lerebourg, D. Olivier, and P. Tranouez. Déploiement des systèmes multi-agents : vers un passage à l'échelle. In *Journées Francophones sur les Systèmes Multi-Agents 2003 (JFSMA 03)*, November 2003.

Nouredine Bensaid and Philippe Mathieu. Magique : A hybrid and hierarchical multi-agent architecture model. In *Proceedings of the second International Conference and Exhibition on*

the Practical Application of Intelligent Agents and Multi Agent Technology (PAAM'97), pages 145–155, 1997.

Olivier Barreteau, Garin Patrice, Dumontier Alexandre, Abrami Geraldine, and Cernesson Flavie. Agent-based facilitation of water allocation : Case study in the drome river valley. *Group Decision and Negotiation*, 12(5) :441–461, Septembre 2003.

Yaneer Bar-Yam. *Dynamics of Complex Systems*. Longman, 1997.

Y. Bar-Yam. *Dynamics of Complex Systems*. Studies in Nonlinearity. Westview Press, 2003.

A. Chaturvedi, J. Chi, M. Shailendra, and D. Dolk. Samas : Scalable architecture for multi-resolution agent-based simulation. In *International Conference of Computational Science*, volume LNCS 3038, pages 779–788, Berlin, 2004. Springer-Verlag.

M. R. Jean (collective name). Emergence et SMA. In *JFIADSMA - Journées Francophones IAD et SMA*, Nice, 1997. Hermes.

C C Chiang, H K Wu, W Liu, and M Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings of IEEE Singapore international conference on networks (SICON)*, pages 197–211, 1997.

Y. Demazeau and A.C. Rocha Costa. Populations and organisations in open multi-agent systems. In *1st Symposium on Parallel and Distributed AI*, Hyderabad, India, July 1996.

Alexis Drogoul and Christophe Dubreuil. *ECO-Problem-Solving Model : Results of the N-puzzle*, pages 283–295. Elsevier Science Publishers B.V., 1992.

Y. Demazeau. From interactions to collective behavior in agent-based systems. In *European Conference on Cognitive Science*, Saint-Malo France, 1995.

T. Devogele. *Processus d'intégration et d'appariement de bases de données Géographiques. Application à une base de données routières multi-échelles*. Thèse de doctorat, Université de Versailles, 1997.

Q J Duan. A concept of multi-robots multi-scale coordination model. In *Fourth International Conference on Natural Computation*, pages 602–604, 2008.

J. Ferber. *Les Systèmes Multi-Agents : vers une intelligence collective*. InterEditions, 1995.

K. Fernandes. *Systèmes Multi-Agents Hybrides : Une approche pour la Conception des Systèmes Complexes*. Thèse de doctorat, Université de Joseph Fourier - Grenoble 1, Grenoble, France, 2001.

S. Franklin and A. Grasser. Is it an agent, or just a program ? : A taxonomy of autonomous agents. In *Proceedings of Third International Workshop on Agent Theories, Architectures and Languages*, Springer-Verlag, 1996.

J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *ICMAS '98 : Proceedings of the 3rd International Conference on Multi Agent Systems*, pages 128–135, Washington, DC, USA, 1998. IEEE Computer Society.

J. Ferber and J. P. Muller. Influences and réaction : a model of situated multiagent systems. In *Proceedings of International Conference on Multi- agent Systems*, pages 72–79, 1996.

Jacques Ferber, Fabien Michel, and José-Antonio Báez-Barranco. Agre : Integrating environments with organizations. In Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, editors, *Environments for Multi-Agent Systems*, volume 3374 of *Lecture Notes in Computer Science*, pages 48–56. Springer Berlin / Heidelberg, 2005.

Nicolas A. Gaud. *Systèmes multi-agents holoniques : de l'analyse à l'implantation. Méta-modèle, méthodologie, et simulation multi-niveaux.* Thèse de doctorat, Université de Franche-Comté et Université de Technologie de Belfort-Montbéliard, 2007.

Adriana Giret and Vicente Botti. Towards a recursive agent oriented methodology for large-scale mas. *Agent Oriented Software Engineering IV*, pages 135–161, 2003.

Mario Giampietro, Kozo Mayumi, and Jesus Ramos-Martin. Multi-scale integrated analysis of societal and ecosystem metabolism (musiasem) : Theoretical concepts and basic rationale. *Energy*, 34(3) :313–322, 2009.

Sylvain Guerraz, Frank Perbet, David Raulo, François Faure, and Marie-Paule Cani. A procedural approach to animate interactive natural sceneries. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*, 2003.

N Goldenfeld and D Rind. Complex system. *History*, pages 1–5, 2008.

J. Glimm and D. H Sharp. Multi-scale science : A challenge for the twenty-first century. *SIAM News*, 30(6) :1–7, 1997.

Christian Gerber, Jorg H. Siekmann, and Gero Vierke. Holonic multi-agent systems. Technical report, Deutsches Forschungszentrum fur Kunztliche Inteligenz - GmbH, Postfach 20 80, 67608 Kaiserslautern, FRG, May 1999.

Julien Guitton. Planification multi-agent pour la composition dynamique de services web. *Intelligence*, 2006.

Christophe Guéret, Shenghui Wang, Paul Groth, and Stefan Schlobach. Multi-scale analysis of the web of data : a challenge to the complex system's community. *Advances in Complex Systems*, 14(04) :587, 2011.

M Henry, Jean François Cosson, and Jean Marc Pons. Modelling multi-scale spatial variation in species richness from abundance data in a complex neotropical bat assemblage. *Ecological Modelling*, 221(17) :2018–2027, 2010.

T-T-H. Hoang and M. Occello. Towards the specification of recursive multi-agent systems using type theory. In *WI-IAT'09 : Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 297–300, Washington, DC, USA, 2009. IEEE Computer Society.

Thi-Thanh-Ha Hoang, Michel Occello, and Jean-Paul Jamont. A generic decentralized recursive multiagent model for multi-scale organization of large scale complex systems. In *ICAART (2)*, pages 418–421, 2011.

Thi-Thanh-Ha Hoang, Michel Occello, and Jean-Paul Jamont. A generic recursive multiagent model to simplify large scale multi-level systems observation. In *IAT*, pages 155–158, 2011.

Thi-Thanh-Ha Hoang, Michel Occello, and Jean-Paul Jamont. Un modèle multi-agent générique récursif pour simplifier la supervision de systèmes décentralisés multi-niveaux. In *JFSMA*, pages 41–50, 2011.

Ha Hoang Thi Thanh and Michel Occello. Applying theory of types to formal specification of recursive multiagent systems. In *IEEE-RIVF International Conference on Computing and Telecommunication Technologies*, pages –, Danang, Viet Nam, 2009.

C. Iglesias, M. Garrijo, and J. Gonzales. A survey of agent oriented methodologies. In *Proceedings of ATAL 98 - Workshop on Agent Theories, Architectures, and Languages*, volume LNAI 1555, pages 163–176, Paris, France, July 1998. Springer-Verlag.

ISO. International standard ISO/IEC 7498-1, 2nd edition, 1989.

J.P. Jamont. Applications des systèmes multi-agents à un réseau maillé de capteurs. DEA informatique, Université de Savoie, France, 2002.

Jean-Paul Jamont. *DIAMOND : Une approche pour la conception de systèmes multi-agents embarqués*. PhD thesis, Institut National Polytechnique de Grenoble (INP Grenoble), September 2005.

Jean-Paul Jamont and Michel Occello. Une approche multi-agents pour la gestion de la communication dans les réseaux de capteurs sans fil. *Revue Technique et Science Informatiques*, 25(5) :661–690, 2006.

Jean-Paul Jamont and Michel Occello. A multiagent tool to simulate hybrid real/virtual embedded agent societies. In *WI-IAT '09 : Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 501–504, Washington, DC, USA, 2009. IEEE Computer Society.

J.-P. Jamont, M. Occello, and A. Lagrèze. A multiagent approach to manage communication in wireless instrumentation systems. *Measurement*, 43(4) :489 – 503, 2010.

Yoann Kubera, Philippe Mathieu, and Sébastien Picault. Ioda : an interaction-oriented approach for multi-agent based simulations. *Autonomous Agents and Multi-Agent Systems*, 23 :303–343, 2011.

Arthur Koestler. *The Ghost in the Machine*. Hutchinson, 1967.

J. A. Krumhansl. Multiscale science : Materials in the 21st century. *Materials science forum*, 327 - 328 :1–7, 2000.

Allyson L Lister, Ruchira S Datta, Oliver Hofmann, Roland Krause, Michael Kuhn, Bettina Roth, and Reinhard Schneider. Live coverage of intelligent systems for molecular biology/european conference on computational biology (ismb/eccb) 2009. *PLoS Computational Biology*, 6(1) :4, 2010.

Jinghai Li. Exploring complex systems in chemical engineering the multi-scale methodology. *Chemical Engineering Science*, 58(3-6) :521–535, 2003.

S. Lamy, A. Ruas, Y. Demazeau, M. Jackson, W. Mackaness, and R. Weibel. The application of agents in automated map generalisation. In *Proc. 19th Int. Cartographic Conference*, pages 47–53, Ottawa , CANADA, 1999.

T.W. Malone. Modeling coordination in organizations and markets. *Management science*, 33 :1317–1332, 1987.

D. Mark. Multiple views of multiple representations. In *Workshop on Multiple Representations*, number 89-3, pages 68–71, 1989.

P. MARCENAC. Modelisation de systemes complexes par agents. *Revue de l'AFCET Technique et Science Informatique,*, 43(4), 1997.

Nelson Minar, Rogert Burkhart, Chris Langton, and Manor Askenazi. The swarm simulation system : A toolkit for building multi-agent simulations. Technical Report 96-06-042, Santa Fe Institute, Jun 1996.

Evaldinolia Moreira, Sergio Costa, Ana Aguiar, Gilberto Cemara, and Tiago Carneiro. Dynamical coupling of multiscale land change models. *Landscape Ecology*, 24 :1183–1194, 2009.

Pierre Marcenac and Sylvain Giroux. Geamas : A generic architecture for agent-oriented simulations of complex processes. *Applied Intelligence*, 8(3) :247–267, 1998.

M. Mezura-Godoy. Recursivité dans les systèmes multi-agents. DEA informatique, IMAG, INPG, Grenoble, France, 1998.

Z Maamar, S K Mostefaoui, and H Yahyaoui. Toward an agent-based and context-oriented approach for web services composition. *IEEE Transactions on Knowledge and Data Engineering*, 17(5) :686–697, 2005.

M. Occello. Towards a generic recursive agent model. In *Proceedings of International Conference on Artificial Intelligence*, Las Vegas, USA, August 2000.

M. Occello and Y. Demazeau. Vers une approche de conception et de description récursive en univers multi-agents. In *5èmes journées Francophone sur l'Intelligence Artificielle Distribuée et les système Multi-Agents*, La Colle sur Loup, France, 1997.

M. Occello and Y. Demazeau. Modelling decision making systems using agents for cooperation in a real time constraints. In *3rd IFAC Symposium on Intelligent Autonomous Vehicles*, volume 1, pages 51–56, Madrid, Spain, March 1998.

Damien Pellier and Humbert Fiorino. Un modèle de composition automatique et distribuée de services web par planification. *Intelligence*, pages 13–46, 2009.

M. Purvis, N. Mariusz, and S. Cranefield. A multi-level approach and infrastructure for agent-oriented software development. In *AAMAS'02 : Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 88–89, New York, NY, USA, 2002. ACM.

S. Picault, P. Mathieu, and Y. Kubera. Padawan, un modèle multi-échelles pour la simulation orientée interactions. In M. Occello and L. Rejeb, editors, *Systèmes Mutli-Agent et Défis Sociétaux*, pages 193–202, 2010.

Sebastian A. Rodriguez. *From analysis to design of holonic multi-agent systems : A framework, methodological guidelines and applications*. PhD thesis, Université de Technologie de Belfort-Montbéliard, Université de Franche-Compté, December 2005.

Clare Sansom. Meeting review : The european conference on computational biology 2002. *Comparative and Functional Genomics*, 4(1) :20–24, 2003.

Ichiro Satoh. A formalism for hierarchical mobile agents. *Software Engineering for Parallel and Distributed Systems, International Symposium on*, 0 :165, 2000.

M Schillo and K Fischer. Holonic multiagent systems. *Artificial Intelligence*, 8(13) :1–2, 2002.

P. Souquiere. La carte électronique : définitions et principes. *Bulletin du Comité français de cartographie*, (139) :47–60, 1994. fre.

C. G. Taylor. Fleshing out. *Artificial Life II*, pages 26–38, 1992.

V. Teles. *Construction de réservoirs aquifères alluviaux par modèle génétique de mise en place des sédiments*. Thèse de doctorat, Université de Paris 6, France, 1999.

H. Tianfield. A new framework of holonic self-organization for multi-agent systems. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 753–758, Montréal, Canada, 2007.

S. Valette, J. Dardenne, R. Prost, and F. Peyrin. Graph-based multi-scale analysis of plates and rods in human trabecular bone. In *Proceedings of 2010 IEEE 17th International Conference on Image Processing*, pages 2289–2292, 2010.

M.F. Wood and S.A. DeLoach. An overview of the multiagent systems engineering methodology. In *Proceedings of The First International Workshop on Agent-Oriented Software Engineering*, 2000.

M.J. Wooldridge, N.R. Jennings, and D. Kinny. The GAIA methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3 :285–312, 2000.

M.J. Wooldridge. Intelligent agents. In Gerhard Weiss, editor, *Multiagent systems ? : A modern approach to Distributed Artificial Intelligence*. MIT Press, 1999.

Charif Yasmine. *Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs*. Thèse de doctorat, Université Pierre et Marie Curie Paris VI, 2007.

Le Zhang, L. Leon Chen, and Thomas S. Deisboeck. Multi-scale, multi-resolution brain cancer modeling. *Mathematics and Computers in Simulation*, 79 :2021–2035, March 2009.

Siliang Zhang, Bang-Ce Ye, Ju Chu, Yingping Zhuang, and Meijin Guo. From multi-scale methodology to systems biology : to integrate strain improvement and fermentation optimization. *Journal of Chemical Technology and Biotechnology*, 81(5) :734–745, 2006.

Annexe A

Trace d'exécution

Cette annexe présente des traces d'exécution de la construction des niveaux d'abstraction d'une population de 10 agents. Cette population est volontairement réduite afin de simplifier la compréhension des traces. 3 niveaux d'abstraction sont construits (voir figure A.1). L'étape initiale de création d'une abstraction est mise en évidence dans les traces à l'aide de lignes ne comportant que des étoiles.

Certains messages ne correspondent pas tout à fait à la définition faites en table 5.1. Ainsi un message COMPOSE_JOIN est un cas particulier de COMPOSE_ACCEPT émis par un agent qui souhaite de composer avec un groupe mais qui n'a pas recu d'invitation COMPOSE_REQUEST.

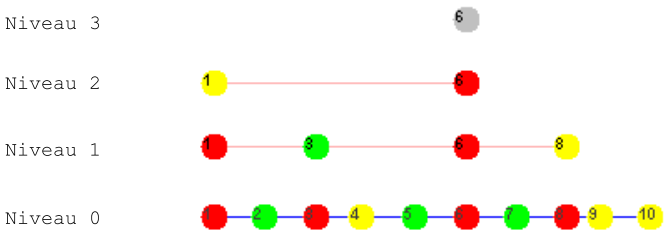


FIGURE A.1 – Les niveaux construits.

00 :00 :00.0000 : Démarrage de l'agent recurssif 1 de niveau 0
00 :00 :00.0000 : Démarrage de l'agent recurssif 2 de niveau 0
00 :00 :00.0000 : Démarrage de l'agent recurssif 3 de niveau 0
00 :00 :00.000 : Démarrage de l'agent recurssif 4 de niveau 0
00 :00 :00.000 : Démarrage de l'agent recurssif 5 de niveau 0
00 :00 :00.000 : Démarrage de l'agent recurssif 6 de niveau 0
00 :00 :00.000 : Démarrage de l'agent recurssif 7 de niveau 0
00 :00 :00.000 : Démarrage de l'agent recurssif 8 de niveau 0
00 :00 :00.000 : Démarrage de l'agent recurssif 9 de niveau 0
00 :00 :00.000 : Démarrage de l'agent recurssif 10 de niveau 0
00 :00 :01.0001 : Démarrage de l'agent applicatif
00 :00 :01.0001 : Démarrage de l'agent applicatif
00 :00 :01.0001 : Démarrage de l'agent applicatif
00 :00 :01.001 : Démarrage de l'agent applicatif
00 :00 :01.001 : Démarrage de l'agent applicatif
00 :00 :01.001 : Démarrage de l'agent applicatif
00 :00 :01.001 : Démarrage de l'agent applicatif
00 :00 :01.001 : Démarrage de l'agent applicatif
00 :00 :01.001 : Démarrage de l'agent applicatif
00 :00 :01.001 : Démarrage de l'agent applicatif
00 :00 :01.656 : (1) traite le message d'introduction Introduction message instancied by 2 to all surrounding neighboors

00 :00 :01.656 : (2) traite le message d'introduction Introduction message instancied by 1 to all surrounding neighboors
00 :00 :01.657 : (2) traite le message d'introduction Introduction message instancied by 3 to all surrounding neighboors
00 :00 :01.659 : (3) traite le message d'introduction Introduction message instancied by 2 to all surrounding neighboors
00 :00 :01.659 : (2) traite le message d'introduction Introduction message instancied by 4 to all surrounding neighboors
00 :00 :01.659 : (3) traite le message d'introduction Introduction message instancied by 4 to all surrounding neighboors
00 :00 :01.660 : (3) traite le message d'introduction Introduction message instancied by 5 to all surrounding neighboors
00 :00 :01.663 : (4) traite le message d'introduction Introduction message instancied by 2 to all surrounding neighboors
00 :00 :01.663 : (4) traite le message d'introduction Introduction message instancied by 3 to all surrounding neighboors
00 :00 :01.664 : (4) traite le message d'introduction Introduction message instancied by 5 to all surrounding neighboors
00 :00 :01.666 : (4) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE_REPRESENTATIVE,group=3]
00 :00 :01.666 : (4) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 2 : [role=ROLE_REPRESENTATIVE,group=2]
00 :00 :01.673 : (5) traite le message d'introduction Introduction message instancied by 3 to all surrounding neighboors
00 :00 :01.674 : (5) traite le message d'introduction Introduction message instancied by 4 to all surrounding neighboors
00 :00 :01.674 : (5) traite le message d'introduction Introduction message instancied by 6 to all surrounding neighboors
00 :00 :01.675 : (5) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE-_REPRESENTATIVE,group=3]
00 :00 :01.675 : (5) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 4 : [role=ROLE-_LINK,group=2,3]
00 :00 :01.676 : (6) traite le message d'introduction Introduction message instancied by 5 to all surrounding neighboors
00 :00 :01.676 : (6) traite le message d'introduction Introduction message instancied by 7 to all surrounding neighboors
00 :00 :01.677 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 5 : [role=ROLE-_SIMPLEMEMBER,group=3]
00 :00 :01.680 : (7) traite le message d'introduction Introduction message instancied by 6 to all surrounding neighboors
00 :00 :01.681 : (7) traite le message d'introduction Introduction message instancied by 8 to all surrounding neighboors
00 :00 :01.681 : (7) traite le message d'introduction Introduction message instancied by 9 to all surrounding neighboors
00 :00 :01.681 : (7) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 6 : [role=ROLE-_REPRESENTATIVE,group=6]
00 :00 :01.684 : (8) traite le message d'introduction Introduction message instancied by 7 to all surrounding neighboors
00 :00 :01.684 : (8) traite le message d'introduction Introduction message instancied by 9 to all surrounding neighboors
00 :00 :01.685 : (8) traite le message d'introduction Introduction message instancied by 10 to all surrounding neighboors
00 :00 :01.685 : (8) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 7 : [role=ROLE-_SIMPLEMEMBER,group=6]
00 :00 :01.688 : (9) traite le message d'introduction Introduction message instancied by 7 to all surrounding neighboors
00 :00 :01.689 : (9) traite le message d'introduction Introduction message instancied by 8 to all surrounding neighboors
00 :00 :01.689 : (9) traite le message d'introduction Introduction message instancied by 10 to all surrounding neighboors
00 :00 :01.690 : (9) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 7 : [role=ROLE-_SIMPLEMEMBER,group=6]
00 :00 :01.690 : (9) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 8 : [role=ROLE_-REPRESENTATIVE,group=8]
00 :00 :01.691 : (10) traite le message d'introduction Introduction message instancied by 8 to all surrounding neighboors
00 :00 :01.691 : (10) traite le message d'introduction Introduction message instancied by 9 to all surrounding neighboors
00 :00 :01.692 : (10) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 8 : [role=ROLE_-REPRESENTATIVE,group=8]
00 :00 :01.692 : (10) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 9 : [role=ROLE_-SIMPLEMEMBER,group=8]
00 :00 :01.763 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 2 : [role=ROLE_-REPRESENTATIVE,group=2]
00 :00 :01.764 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 4 : [role=ROLE_-LINK,group=2,3]
00 :00 :01.764 : (1) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 2 : [role=ROLE_-REPRESENTATIVE,group=2]
00 :00 :01.764 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 5 : [role=ROLE_-SIMPLEMEMBER,group=3]
00 :00 :01.765 : (2) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE_-REPRESENTATIVE,group=3]
00 :00 :01.766 : (2) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 1 : [role=ROLE_-REPRESENTATIVE,group=1]
00 :00 :01.766 : (2) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 4 : [role=ROLE_-LINK,group=2,3]
00 :00 :01.767 : (4) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 5 : [role=ROLE_-SIMPLEMEMBER,group=3]
00 :00 :01.768 : (2) GAGNE le conflict resolution Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)
00 :00 :01.768 : (2) PERDS le conflict resolution Conflict resolution message instancied by 3 to all Representative neighboors (score=30000)
00 :00 :01.776 : (5) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]
00 :00 :01.777 : (5) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 4 : [role=ROLE_-LINK,group=2,3]

00 :00 :01.778 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 7 : [role=ROLE_-SIMPLEMEMBER,group=6]
00 :00 :01.778 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 5 : [role=ROLE_-LINK,group=3,6]
00 :00 :01.782 : (7) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 8 : [role=ROLE_-REPRESENTATIVE,group=8]
00 :00 :01.783 : (7) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 9 : [role=ROLE_-SIMPLEMEMBER,group=8]
00 :00 :01.786 : (8) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 9 : [role=ROLE_-SIMPLEMEMBER,group=8]
00 :00 :01.786 : (8) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 10 : [role=ROLE_-SIMPLEMEMBER,group=8]
00 :00 :01.786 : (8) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 7 : [role=ROLE_-LINK,group=6,8]
00 :00 :01.791 : (9) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 10 : [role=ROLE_-SIMPLEMEMBER,group=8]
00 :00 :01.791 : (9) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 7 : [role=ROLE_-LINK,group=6,8]
00 :00 :01.865 : (1) PERDS le conflict resolution Conflict resolution message instancied by 2 to all Representative neighboors (score=30000)
00 :00 :01.865 : (3) GAGNE le conflict resolution Conflict resolution message instancied by 2 to all Representative neighboors (score=30000)
00 :00 :01.866 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 4 : [role=ROLE_-LINK,group=2,3]
00 :00 :01.867 : (3) traite le message d'introduction Introduction message instancied by 2 to all surrounding neighboors
00 :00 :01.867 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 5 : [role=ROLE_-LINK,group=3,6]
00 :00 :01.868 : (4) traite le message d'introduction Introduction message instancied by 2 to all surrounding neighboors
00 :00 :01.869 : (4) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 5 : [role=ROLE_-LINK,group=3,6]
00 :00 :01.869 : (4) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE_-REPRESENTATIVE,group=3]
00 :00 :01.877 : (5) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE_-REPRESENTATIVE,group=3]
00 :00 :01.878 : (5) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 4 : [role=ROLE_-SIMPLEMEMBER,group=3]
00 :00 :01.879 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 7 : [role=ROLE_-LINK,group=6,8]
00 :00 :01.880 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 5 : [role=ROLE_-LINK,group=3,6]
00 :00 :01.968 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 4 : [role=ROLE_-SIMPLEMEMBER,group=3]
00 :00 :01.969 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 5 : [role=ROLE_-LINK,group=3,6]
00 :00 :01.970 : (4) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 5 : [role=ROLE_-LINK,group=3,6]
00 :00 :02.119 : (2) traite le message d'introduction Introduction message instancied by 1 to all surrounding neighboors
00 :00 :02.120 : (2) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE_-REPRESENTATIVE,group=3]
00 :00 :02.120 : (2) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 4 : [role=ROLE_-SIMPLEMEMBER,group=3]
00 :00 :02.169 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 2 : [role=ROLE_-SIMPLEMEMBER,group=3]
00 :00 :02.170 : (4) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 2 : [role=ROLE_-SIMPLEMEMBER,group=3]
00 :00 :02.216 : (1) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 2 : [role=ROLE_-SIMPLEMEMBER,group=3]
00 :00 :02.221 : (2) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 1 : [role=ROLE_-REPRESENTATIVE,group=1]
00 :00 :02.269 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 2 : [role=ROLE_-LINK,group=3,1]
00 :00 :02.270 : (4) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 2 : [role=ROLE_-LINK,group=3,1]
00 :00 :02.317 : (1) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 2 : [role=ROLE_-LINK,group=3,1]
00 :00 :13.750 : COMPOSE_JOIN provenant de ((*lvl=0,id=8,role=R**) recu par représentant (id=8,level=0) previous JOIN state=-1 ecart=1337787003390
****

* Création de la première abstraction de niveau 1 (Agent 8)

```
**********************************************************************************************
00 :00 :13.800 : ((*lvl=0,id=8,role=R**) est prêt à lancer une composition
00 :00 :13.807 : ((lvl=0,id=8,role=R) Creation de l'abstraction suite a une operation COMPOSE 8 de niveau 1
00 :00 :13.850 : COMPOSE_REQUEST recu par ((*lvl=0,id=7,role=R**)
00 :00 :13.850 : COMPOSE_JOIN provenant de ((*lvl=0,id=6,role=R**) recu par représentant (id=6,level=0) previous JOIN state=-1
ecart=1337787003490
00 :00 :13.851 : ((*lvl=0,id=7,role=R**) acception la composition avec 8
00 :00 :13.851 : COMPOSE_REQUEST recu par ((*lvl=0,id=9,role=R**)
00 :00 :13.851 : COMPOSE_REQUEST recu par ((*lvl=0,id=10,role=R**)
00 :00 :13.851 : ((*lvl=0,id=9,role=R**) acception la composition avec 8
00 :00 :13.851 : ((*lvl=0,id=10,role=R**) acception la composition avec 8
00 :00 :13.852 : ((lvl=0,id=7,role=R) Creation de l'abstraction suite a une operation COMPOSE 8 de niveau 1
00 :00 :13.852 : ((lvl=0,id=9,role=R) Creation de l'abstraction suite a une operation COMPOSE 8 de niveau 1
00 :00 :13.852 : ((lvl=0,id=10,role=R) Creation de l'abstraction suite a une operation COMPOSE 8 de niveau 1
00 :00 :13.854 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N) créer l'agent ABSTRAIT Agent #8 : energy=100,00 range=60
00 :00 :13.854 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N) créer l'agent ABSTRAIT Agent #8 : energy=100,00 range=60
00 :00 :13.854 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N) créer l'agent ABSTRAIT Agent #8 : energy=100,00 range=60
00 :00 :13.858 : COMPOSE_JOIN provenant de ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=N) recu par représentant (id=8,level=0) pre-
vious JOIN state=1337787003389 ecart=108
00 :00 :13.858 : COMPOSE_JOIN provenant de ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=N) recu par représentant (id=8,level=0) pre-
vious JOIN state=1337787003497 ecart=0
00 :00 :13.859 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=7,level=0)
to (id=8level=1)
00 :00 :13.859 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=9,level=0)
to (id=8level=1)
00 :00 :13.859 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=10,level=0)
to (id=8level=1)
**********************************************************************************************
* Création de la 2éme abstraction de niveau 1 (Agent 6)
**********************************************************************************************
00 :00 :13.901 : ((*lvl=0,id=6,role=R**) est prêt à lancer une composition
00 :00 :13.901 : ((lvl=0,id=6,role=R) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 1
00 :00 :13.904 : COMPOSE_REQUEST recu par ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)
00 :00 :13.952 : COMPOSE_JOIN provenant de ((*lvl=0,id=3,role=R**) recu par représentant (id=3,level=0) previous JOIN state=-1
ecart=1337787003591
00 :00 :13.952 : COMPOSE_REQUEST recu par ((*lvl=0,id=5,role=R**)
00 :00 :13.952 : COMPOSE_JOIN provenant de ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=N) recu par représentant (id=6,level=0) pre-
vious JOIN state=1337787003490 ecart=101
00 :00 :13.952 : ((*lvl=0,id=5,role=R**) acception la composition avec 6
00 :00 :13.953 : ((lvl=0,id=5,role=R) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 1
00 :00 :13.953 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N) créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :14.0014 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=5,level=0)
to (id=6level=1)
00 :00 :14.252 : COMPOSE_JOIN provenant de ((*lvl=0,id=3,role=R**) recu par représentant (id=3,level=0)
previous JOIN state=1337787003591 ecart=300
00 :00 :14.301 : COMPOSE_JOIN provenant de ((*lvl=0,id=1,role=R**) recu par représentant (id=1,level=0)
previous JOIN state=-1 ecart=1337787003941
00 :00 :14.302 : COMPOSE_JOIN provenant de ((*lvl=0,id=3,role=R**) recu par représentant (id=3,level=0) previous JOIN state=1337787003891
ecart=50
**********************************************************************************************
* Création de la 4éme abstraction de niveau 1 (agent 3)
**********************************************************************************************
00 :00 :14.302 : ((*lvl=0,id=3,role=R**) est prêt à lancer une composition
00 :00 :14.303 : COMPOSE_REQUEST recu par ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)
00 :00 :14.304 : ((lvl=0,id=3,role=R) Creation de l'abstraction suite a une operation COMPOSE 3 de niveau 1
00 :00 :14.351 : COMPOSE_REQUEST recu par ((*lvl=0,id=4,role=R**)
00 :00 :14.351 : COMPOSE_REQUEST recu par ((*lvl=0,id=2,role=R**)
**********************************************************************************************
* Création de la 3éme abstraction de niveau 1 (agent 1)
**********************************************************************************************
00 :00 :14.351 : ((*lvl=0,id=1,role=R**) est prêt à lancer une composition
00 :00 :14.351 : ((*lvl=0,id=4,role=R**) acception la composition avec 3
00 :00 :14.351 : ((*lvl=0,id=2,role=R**) acception la composition avec 3
00 :00 :14.352 : ((lvl=0,id=1,role=R) Creation de l'abstraction suite a une operation COMPOSE 1 de niveau 1
00 :00 :14.352 : ((lvl=0,id=4,role=R) Creation de l'abstraction suite a une operation COMPOSE 3 de niveau 1
00 :00 :14.353 : ((lvl=0,id=2,role=R) Creation de l'abstraction suite a une operation COMPOSE 3 de niveau 1
00 :00 :14.353 : ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N) créer l'agent ABSTRAIT Agent #3 : energy=100,00 range=60
```

00 :00 :14.353 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N) créer l'agent ABSTRAIT Agent #3 : energy=100,00 range=60
00 :00 :14.353 : COMPOSE_REQUEST recu par ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)
00 :00 :14.354 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=4,level=0) to (id=3level=1)
00 :00 :14.354 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=2,level=0) to (id=3level=1)
00 :00 :14.411 : [COMPOSE UPDATE from (id=8,level=0) : role is ROLE_REPRESENTATIVE members are 7,8,9,10
00 :00 :14.504 : [COMPOSE UPDATE from (id=6,level=0) : role is ROLE_REPRESENTATIVE members are 5,6
00 :00 :14.864 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=N) Stability : false => je veux créer une abstraction
00 :00 :14.904 : COMPOSE_REQUEST recu par ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)
00 :00 :14.904 : COMPOSE_REQUEST recu par ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)
00 :00 :14.904 : COMPOSE_REQUEST recu par ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)
00 :00 :14.904 : [COMPOSE UPDATE from (id=3,level=0) : role is ROLE_REPRESENTATIVE members are 2,3,4
00 :00 :14.954 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=N) Stability : false => je veux créer une abstraction
00 :00 :15.0015 : COMPOSE_REQUEST recu par ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)
00 :00 :15.0015 : COMPOSE_REQUEST recu par ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)
00 :00 :15.302 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=N) Stability : true => je veux créer une abstraction
00 :00 :15.305 : COMPOSE_REQUEST recu par ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)
00 :00 :15.305 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=N) Stability : false => je veux créer une abstraction
00 :00 :15.355 : COMPOSE_REQUEST recu par ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)
00 :00 :15.355 : COMPOSE_REQUEST recu par ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N)
00 :00 :15.355 : COMPOSE_REQUEST recu par ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)
00 :00 :23.815 : ((lvl=0,id=8,role=R)(*lvl=1,id=8,role=N**) envoie une trame applicative Introduction message instancied by 8 to all surrounding neighboors
00 :00 :23.817 : In ((lvl=0,id=8,role=R)(*lvl=1,id=8,role=N**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=8,layer=1) to (id=-1,layer=1) data=Introduction message instancied by 8 to all surrounding neighboors] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=8,layer=1) labelized #0 to (id=-1,layer=1) data=[RecMSG applicative frame transport message from (id=8,layer=1) to (id=-1,layer=1) data=Introduction message instancied by 8 to all surrounding neighboors]
00 :00 :23.905 : ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=N**) TRAITEMENT du message APPLICATIF recu Introduction message instancied by 8 to all surrounding neighboors
00 :00 :23.906 : (6) traite le message d'introduction Introduction message instancied by 8 to all surrounding neighboors
00 :00 :23.907 : ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**) envoie une trame applicative Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :23.907 : In ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=6,layer=1) labelized #0 to (id=-1,layer=1) data=[RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]]
00 :00 :23.955 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)) fait remonter une trame a son abstraction (6,1) pour traitement [RecMSG applicative frame transport message from (id=8,layer=1) to (id=-1,layer=1) data=Introduction message instancied by 8 to all surrounding neighboors...
00 :00 :23.955 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N) ... et je repete le message de transport interne (6,1) pour traitement [RecMSG applicative frame transport message from (id=8,layer=1) to (id=-1,layer=1) data=Introduction message instancied by 8 to all surrounding neighboors
00 :00 :23.956 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]...
00 :00 :23.956 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N) ... et je repete le message de transport interne (8,1) pour traitement [RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]
00 :00 :23.972 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]...
00 :00 :23.972 : ((lvl=0,id=8,role=R)(*lvl=1,id=8,role=N**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :23.973 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=N) ... et je repete le message de transport interne (8,1) pour traitement [RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]
00 :00 :23.973 : (8) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]
00 :00 :23.974 : ((lvl=0,id=8,role=R)(*lvl=1,id=8,role=R**) envoie une trame applicative Presentation message instancied by 8 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :23.974 : In ((lvl=0,id=8,role=R)(*lvl=1,id=8,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=8,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 8 : [role=ROLE_SIMPLEMEMBER,group=6]] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=8,layer=1) labelized #1 to (id=-1,layer=1) data=[RecMSG applicative frame transport message from (id=8,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 8 : [role=ROLE_SIMPLEMEMBER,group=6]]

00 :00 :24.0024 : ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=N**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :24.0024 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]...
00 :00 :24.0024 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]
00 :00 :24.0024 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N) ... et je repete le message de transport interne (8,1) pour traitement [RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :24.0024 : ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**) envoie une trame applicative Presentation message instancied by 3 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :24.0024 : In ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=3,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 3 : [role=ROLE_SIMPLEMEMBER,group=6]] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=3,layer=1) labelized #0 to (id=-1,layer=1) data=[RecMSG applicative frame transport message from (id=3,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 3 : [role=ROLE_SIMPLE-MEMBER,group=6]]
00 :00 :24.0024 : ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 8 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :24.0024 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 8 : [role=ROLE_-SIMPLEMEMBER,group=6]
00 :00 :24.024 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]...
00 :00 :24.024 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N) ... et je repete le message de transport interne (3,1) pour traitement [RecMSG applicative frame transport message from (id=6,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]
00 :00 :24.024 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)) fait remonter une trame a son abstraction (6,1) pour traitement [RecMSG applicative frame transport message from (id=8,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 8 : [role=ROLE_-SIMPLEMEMBER,group=6]...
00 :00 :24.024 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N) ... et je repete le message de transport interne (6,1) pour traitement [RecMSG applicative frame transport message from (id=8,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 8 : [role=ROLE_-SIMPLEMEMBER,group=6]
00 :00 :24.024 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)) fait remonter une trame a son abstraction (6,1) pour traitement [RecMSG applicative frame transport message from (id=3,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 3 : [role=ROLE_-SIMPLEMEMBER,group=6]...
00 :00 :24.024 : ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 3 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :24.024 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R) ... et je repete le message de transport interne (6,1) pour traitement [RecMSG applicative frame transport message from (id=3,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 3 : [role=ROLE_-SIMPLEMEMBER,group=6]
00 :00 :24.024 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE_-SIMPLEMEMBER,group=6]
00 :00 :24.102 : ((lvl=0,id=1,role=R)(*lvl=1,id=1,role=N**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 3 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :24.103 : (1) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE_-SIMPLEMEMBER,group=6]
00 :00 :24.103 : ((lvl=0,id=1,role=R)(*lvl=1,id=1,role=R**) envoie une trame applicative Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]
00 :00 :24.103 : In ((lvl=0,id=1,role=R)(*lvl=1,id=1,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=1,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=1,layer=1) labelized #0 to (id=-1,layer=1) data=[RecMSG applicative frame transport message from (id=1,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 1 : [role=ROLE_-REPRESENTATIVE,group=1]]
00 :00 :24.108 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=R)) fait remonter une trame a son abstraction (3,1) pour traitement [RecMSG applicative frame transport message from (id=1,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 1 : [role=ROLE_-REPRESENTATIVE,group=1]...
00 :00 :24.108 : ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]
00 :00 :24.108 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=R) ... et je repete le message de transport interne (3,1) pour traitement [RecMSG applicative frame transport message from (id=1,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 1 : [role=ROLE_-REPRESENTATIVE,group=1]
00 :00 :24.109 : (3) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 1 : [role=ROLE_-REPRESENTATIVE,group=1]
00 :00 :24.110 : ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**) envoie une trame applicative Presentation message instancied by 3 : [role=ROLE_LINK,group=6,1]
00 :00 :24.110 : In ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame trans-

port message from (id=3,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 3 : [role=ROLE_LINK,group=6,1]] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=3,layer=1) labelized #1 to (id=-1,layer=1) data=[RecMSG applicative frame transport message from (id=3,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 3 : [role=ROLE_LINK,group=6,1]]
00 :00 :24.157 : ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [RecMSG applicative frame transport message from (id=1,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 1 : [role=ROLE_-REPRESENTATIVE,group=1]...
00 :00 :24.157 : ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N) ... et je repete le message de transport interne (3,1) pour traitement [RecMSG applicative frame transport message from (id=1,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 1 : [role=ROLE_-REPRESENTATIVE,group=1]
00 :00 :24.161 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)) fait remonter une trame a son abstraction (6,1) pour traitement [RecMSG applicative frame transport message from (id=3,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 3 : [role=ROLE_-LINK,group=6,1]...
00 :00 :24.161 : ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 3 : [role=ROLE_LINK,group=6,1]
00 :00 :24.162 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R) ... et je repete le message de transport interne (6,1) pour traitement [RecMSG applicative frame transport message from (id=3,layer=1) to (id=-1,layer=1) data=Presentation message instancied by 3 : [role=ROLE_-LINK,group=6,1]
00 :00 :24.162 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE_-LINK,group=6,1]
00 :00 :24.204 : ((lvl=0,id=1,role=R)(*lvl=1,id=1,role=R**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 3 : [role=ROLE_LINK,group=6,1]
00 :00 :24.205 : (1) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 3 : [role=ROLE_-LINK,group=6,1]
00 :00 :36.036 : COMPOSE_JOIN provenant de ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**) recu par représentant (id=6,level=1) previous JOIN state=-1 ecart=1337787025654
00 :00 :36.036 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)) fait remonter une trame a son abstraction (6,1) pour traitement [COMPOSE JOIN from (id=8,level=1)...
00 :00 :36.036 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N) ... et je repete le message de transport interne (6,1) pour traitement [COMPOSE JOIN from (id=8,level=1)
00 :00 :36.165 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)) fait remonter une trame a son abstraction (6,1) pour traitement [COMPOSE JOIN from (id=3,level=1)...
00 :00 :36.165 : COMPOSE_JOIN provenant de ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**) recu par représentant (id=6,level=1) previous JOIN state=1337787025653 ecart=151
00 :00 :36.165 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R) ... et je repete le message de transport interne (6,1) pour traitement [COMPOSE JOIN from (id=3,level=1)
****
* Création de la 3éme abstraction de niveau 2 (agent 6)
****
00 :00 :36.166 : ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**) est prêt à lancer une composition
00 :00 :36.167 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 2
00 :00 :36.206 : COMPOSE_JOIN provenant de ((lvl=0,id=1,role=R)(*lvl=1,id=1,role=R**) recu par représentant (id=1,level=1) previous JOIN state=-1 ecart=1337787025846
****
* Création de la 3éme abstraction de niveau 2 (agent 1)
****
00 :00 :36.207 : ((lvl=0,id=1,role=R)(*lvl=1,id=1,role=R**) est prêt à lancer une composition
00 :00 :36.207 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R) Creation de l'abstraction suite a une operation COMPOSE 1 de niveau 2
00 :00 :36.208 : COMPOSE_REQUEST recu par ((lvl=0,id=7,role=R)(*lvl=1,id=8,role=N**)
00 :00 :36.208 : COMPOSE_REQUEST recu par ((lvl=0,id=2,role=R)(*lvl=1,id=3,role=N**)
00 :00 :36.208 : ((lvl=0,id=7,role=R)(*lvl=1,id=8,role=N**) acception la composition avec 6
00 :00 :36.208 : ((lvl=0,id=2,role=R)(*lvl=1,id=3,role=N**) acception la composition avec 1
00 :00 :36.209 : ((lvl=0,id=7,role=R)(lvl=1,id=8,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 2
00 :00 :36.209 : ((lvl=0,id=2,role=R)(lvl=1,id=3,role=N) Creation de l'abstraction suite a une operation COMPOSE 1 de niveau 2
00 :00 :36.209 : ((lvl=0,id=7,role=R)(*lvl=1,id=8,role=N**)(lvl=2,id=6,role=N) créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :36.210 : ((lvl=0,id=2,role=R)(*lvl=1,id=3,role=N**)(lvl=2,id=1,role=N) créer l'agent ABSTRAIT Agent #1 : energy=100,00 range=60
00 :00 :36.211 : ((lvl=0,id=5,role=R)(lvl=1,id=6,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 2
00 :00 :36.211 : ((lvl=0,id=5,role=R)(*lvl=1,id=6,role=N**)(lvl=2,id=6,role=N)a entendu un COMPOSE REQUEST qui l'implique et va donc créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :36.213 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=R)) fait remonter une trame a son abstraction (3,1) pour traitement [COMPOSE REQUEST from (id=1,level=1)...
00 :00 :36.213 : COMPOSE_REQUEST recu par ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**)
00 :00 :36.213 : ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**) acception la composition avec 1
00 :00 :36.214 : ((lvl=0,id=3,role=R)(lvl=1,id=3,role=R) Creation de l'abstraction suite a une operation COMPOSE 1 de niveau 2
00 :00 :36.215 : ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**)(lvl=2,id=1,role=N) créer l'agent ABSTRAIT Agent #1 : energy=100,00 range=60

00 :00 :36.215 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=R)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (3,1) pour traitement [COMPOSE REQUEST from (id=1,level=1)
00 :00 :36.216 : COMPOSE_REQUEST recu par ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**)(lvl=2,id=1,role=N)
00 :00 :36.217 : ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**)(lvl=2,id=6,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=8,level=1) to (id=6level=2)
00 :00 :36.228 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)) fait remonter une trame a son abstraction (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)...
00 :00 :36.228 : COMPOSE_REQUEST recu par ((lvl=0,id=8,role=R)(*lvl=1,id=8,role=R**)
00 :00 :36.229 : ((lvl=0,id=8,role=R)(*lvl=1,id=8,role=R**) acception la composition avec 6
00 :00 :36.229 : ((lvl=0,id=8,role=R)(lvl=1,id=8,role=R) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 2
00 :00 :36.230 : ((lvl=0,id=8,role=R)(*lvl=1,id=8,role=R**)(lvl=2,id=6,role=N) créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :36.230 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)
00 :00 :36.257 : ((lvl=0,id=1,role=R)(*lvl=1,id=1,role=R**)(lvl=2,id=1,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=3,level=1) to (id=1level=2)
00 :00 :36.259 : ((lvl=0,id=9,role=R)(lvl=1,id=8,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 2
00 :00 :36.259 : ((lvl=0,id=4,role=R)(lvl=1,id=3,role=N) Creation de l'abstraction suite a une operation COMPOSE 1 de niveau 2
00 :00 :36.259 : ((lvl=0,id=9,role=R)(*lvl=1,id=8,role=N**)(lvl=2,id=6,role=N)a entendu un COMPOSE ACCEPT qui l'implique et va donc créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :36.259 : ((lvl=0,id=4,role=R)(*lvl=1,id=3,role=N**)(lvl=2,id=1,role=N)a entendu un COMPOSE ACCEPT qui l'implique et va donc créer l'agent ABSTRAIT Agent #1 : energy=100,00 range=60
00 :00 :36.260 : ((lvl=0,id=10,role=R)(lvl=1,id=8,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 2
00 :00 :36.260 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)...
00 :00 :36.260 : ((lvl=0,id=10,role=R)(*lvl=1,id=8,role=N**)(lvl=2,id=6,role=N)a entendu un COMPOSE ACCEPT qui l'implique et va donc créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :36.260 : COMPOSE_REQUEST recu par ((lvl=0,id=9,role=R)(*lvl=1,id=8,role=N**)(lvl=2,id=6,role=N)
00 :00 :36.261 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)...
00 :00 :36.261 : COMPOSE_REQUEST recu par ((lvl=0,id=2,role=R)(*lvl=1,id=3,role=N**)(lvl=2,id=1,role=N)
00 :00 :36.261 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)
00 :00 :36.261 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (3,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)
00 :00 :36.261 : ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [COMPOSE REQUEST from (id=1,level=1)...
00 :00 :36.262 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)...
00 :00 :36.262 : COMPOSE_REQUEST recu par ((lvl=0,id=4,role=R)(*lvl=1,id=3,role=N**)(lvl=2,id=1,role=N)
00 :00 :36.262 : COMPOSE_REQUEST recu par ((lvl=0,id=10,role=R)(*lvl=1,id=8,role=N**)(lvl=2,id=6,role=N)
00 :00 :36.262 : ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (3,1) pour traitement [COMPOSE REQUEST from (id=1,level=1)
00 :00 :36.262 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)
00 :00 :36.262 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,1) pour traitement [COMPOSE ACCEPT from (id=8,level=1) to (id=6level=2)...
00 :00 :36.263 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,1) pour traitement [COMPOSE ACCEPT from (id=8,level=1) to (id=6level=2)
00 :00 :36.264 : COMPOSE_REQUEST recu par ((lvl=0,id=4,role=R)(*lvl=1,id=3,role=N**)(lvl=2,id=1,role=N)
00 :00 :36.267 : ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**)(lvl=2,id=6,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=8,level=1) to (id=6level=2)
00 :00 :36.267 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,1) pour traitement [COMPOSE ACCEPT from (id=3,level=1) to (id=1level=2)...
00 :00 :36.268 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,1) pour traitement [COMPOSE ACCEPT from (id=3,level=1) to (id=1level=2)
00 :00 :36.268 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,1) pour traitement [COMPOSE ACCEPT from (id=3,level=1) to (id=1level=2)...
00 :00 :36.269 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,1) pour traitement [COMPOSE ACCEPT from (id=3,level=1) to (id=1level=2)
00 :00 :36.307 : ((lvl=0,id=1,role=R)(*lvl=1,id=1,role=R**)(lvl=2,id=1,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=3,level=1) to (id=1level=2)
00 :00 :36.316 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,1) pour traitement [COMPOSE ACCEPT from (id=8,level=1) to (id=6level=2)...
00 :00 :36.316 : ((*lvl=0,id=5,role=R**)(lvl=1,id=6,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,1) pour traitement [COMPOSE ACCEPT from (id=8,level=1) to (id=6level=2)
00 :00 :37.169 : ((lvl=0,id=6,role=R)(*lvl=1,id=6,role=R**)(lvl=2,id=6,role=N) Stability : false => je veux créer une abstraction

00 :00 :37.207 : ((lvl=0,id=1,role=R)(*lvl=1,id=1,role=R**)(lvl=2,id=1,role=N) Stability : false => je veux créer une abstraction
00 :00 :37.211 : COMPOSE_REQUEST recu par ((lvl=0,id=7,role=R)(*lvl=1,id=8,role=N**)(lvl=2,id=6,role=N)
00 :00 :37.213 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)...
00 :00 :37.213 : COMPOSE_REQUEST recu par ((lvl=0,id=2,role=R)(*lvl=1,id=3,role=N**)(lvl=2,id=1,role=N)
00 :00 :37.213 : COMPOSE_REQUEST recu par ((lvl=0,id=9,role=R)(*lvl=1,id=8,role=N**)(lvl=2,id=6,role=N)
00 :00 :37.213 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)
00 :00 :37.215 : ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [COMPOSE REQUEST from (id=1,level=1)...
00 :00 :37.215 : COMPOSE_REQUEST recu par ((lvl=0,id=4,role=R)(*lvl=1,id=3,role=N**)(lvl=2,id=1,role=N)
00 :00 :37.215 : ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (3,1) pour traitement [COMPOSE REQUEST from (id=1,level=1)
00 :00 :37.216 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=R)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [COMPOSE REQUEST from (id=1,level=1)...
00 :00 :37.216 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)...
00 :00 :37.216 : COMPOSE_REQUEST recu par ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**)(lvl=2,id=1,role=N)
00 :00 :37.216 : COMPOSE_REQUEST recu par ((lvl=0,id=10,role=R)(*lvl=1,id=8,role=N**)(lvl=2,id=6,role=N)
00 :00 :37.216 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=R)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (3,1) pour traitement [COMPOSE REQUEST from (id=1,level=1)
00 :00 :37.217 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)
00 :00 :37.219 : [COMPOSE UPDATE from (id=6,level=1) : role is ROLE_REPRESENTATIVE members are 6,8
00 :00 :37.230 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)...
00 :00 :37.230 : COMPOSE_REQUEST recu par ((lvl=0,id=8,role=R)(*lvl=1,id=8,role=R**)(lvl=2,id=6,role=N)
00 :00 :37.230 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (8,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)
00 :00 :37.264 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [COMPOSE UPDATE from (id=6,level=1) : role is ROLE_REPRESENTATIVE members are 6,8...
00 :00 :37.264 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (8,1) pour traitement [COMPOSE UPDATE from (id=6,level=1) : role is ROLE_REPRESENTATIVE members are 6,8
00 :00 :37.266 : COMPOSE_REQUEST recu par ((lvl=0,id=4,role=R)(*lvl=1,id=3,role=N**)(lvl=2,id=1,role=N)
00 :00 :37.267 : COMPOSE_REQUEST recu par ((lvl=0,id=3,role=R)(*lvl=1,id=3,role=R**)(lvl=2,id=1,role=N)
00 :00 :37.267 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [COMPOSE UPDATE from (id=6,level=1) : role is ROLE_REPRESENTATIVE members are 6,8...
00 :00 :37.267 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (8,1) pour traitement [COMPOSE UPDATE from (id=6,level=1) : role is ROLE_REPRESENTATIVE members are 6,8
00 :00 :37.281 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (8,1) pour traitement [COMPOSE UPDATE from (id=6,level=1) : role is ROLE_REPRESENTATIVE members are 6,8...
00 :00 :37.281 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (8,1) pour traitement [COMPOSE UPDATE from (id=6,level=1) : role is ROLE_REPRESENTATIVE members are 6,8
00 :00 :37.307 : [COMPOSE UPDATE from (id=1,level=1) : role is ROLE_REPRESENTATIVE members are 1,3
00 :00 :37.313 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)...
00 :00 :37.313 : COMPOSE_REQUEST recu par ((lvl=0,id=2,role=R)(*lvl=1,id=3,role=N**)(lvl=2,id=1,role=N)
00 :00 :37.313 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (3,1) pour traitement [COMPOSE REQUEST from (id=6,level=1)
00 :00 :37.314 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [COMPOSE UPDATE from (id=6,level=1) : role is ROLE_REPRESENTATIVE members are 6,8...
00 :00 :37.314 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (3,1) pour traitement [COMPOSE UPDATE from (id=6,level=1) : role is ROLE_REPRESENTATIVE members are 6,8
00 :00 :37.316 : ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [COMPOSE UPDATE from (id=1,level=1) : role is ROLE_REPRESENTATIVE members are 1,3...
00 :00 :37.316 : ((*lvl=0,id=4,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (3,1) pour traitement [COMPOSE UPDATE from (id=1,level=1) : role is ROLE_REPRESENTATIVE members are 1,3
00 :00 :37.318 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=R)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (3,1) pour traitement [COMPOSE UPDATE from (id=1,level=1) : role is ROLE_REPRESENTATIVE members are 1,3...
00 :00 :37.318 : ((*lvl=0,id=3,role=R**)(lvl=1,id=3,role=R)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (3,1) pour traitement [COMPOSE UPDATE from (id=1,level=1) : role is ROLE_REPRESENTATIVE members are 1,3
00 :00 :46.170 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=N**) envoie une trame applicative Introduction message instancied by 6 to all surrounding neighboors
00 :00 :46.170 : In ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=N**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 6 to all surrounding neighboors] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=6,layer=2) labelized #4 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 6 to all surrounding

neighboors]
00 :00 :46.208 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=N**) envoie une trame applicative Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.208 : In ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=N**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=1,layer=2) labelized #4 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors]
00 :00 :46.221 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.221 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=N**) TRAITEMENT du message APPLICATIF recu Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.221 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.222 : (6) traite le message d'introduction Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.222 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) envoie une trame applicative Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :46.222 : In ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=6,layer=2) labelized #5 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]]
00 :00 :46.261 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.261 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.265 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.265 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 6 to all surrounding neighboors...
00 :00 :46.265 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.265 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 6 to all surrounding neighboors
00 :00 :46.268 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.268 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.283 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.283 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.308 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 6 to all surrounding neighboors...
00 :00 :46.308 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=N**) TRAITEMENT du message APPLICATIF recu Introduction message instancied by 6 to all surrounding neighboors
00 :00 :46.308 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 6 to all surrounding neighboors
00 :00 :46.308 : (1) traite le message d'introduction Introduction message instancied by 6 to all surrounding neighboors
00 :00 :46.309 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**) envoie une trame applicative Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]
00 :00 :46.309 : In ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_-

REPRESENTATIVE,group=1]] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=1,layer=2) labelized #5 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]]
00 :00 :46.316 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]...
00 :00 :46.316 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :46.323 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]...
00 :00 :46.323 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]
00 :00 :46.323 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]
00 :00 :46.323 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 1 : [role=ROLE_-REPRESENTATIVE,group=1]
00 :00 :46.324 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) envoie une trame applicative Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)
00 :00 :46.324 : In ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=6,layer=2) labelized #6 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)]
00 :00 :46.359 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]...
00 :00 :46.359 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :46.359 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :46.360 : (1) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 6 : [role=ROLE_-REPRESENTATIVE,group=6]
00 :00 :46.360 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**) envoie une trame applicative Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)
00 :00 :46.360 : In ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=1,layer=2) labelized #6 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)]
00 :00 :46.362 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]...
00 :00 :46.362 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]
00 :00 :46.366 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]...
00 :00 :46.366 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]
00 :00 :46.369 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]...
00 :00 :46.369 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]
00 :00 :46.384 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]...
00 :00 :46.384 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_REPRESENTATIVE,group=1]

00 :00 :46.417 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)...
00 :00 :46.417 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)
00 :00 :46.424 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)...
00 :00 :46.424 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) TRAITEMENT du message APPLICATIF recu Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)
00 :00 :46.424 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)
00 :00 :46.424 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) envoie une trame applicative Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)
00 :00 :46.424 : In ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=6,layer=2) labelized #7 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)]
00 :00 :46.425 : (6) GAGNE le conflict resolution Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)
00 :00 :46.460 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)...
00 :00 :46.460 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**) TRAITEMENT du message APPLICATIF recu Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)
00 :00 :46.460 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)
00 :00 :46.461 : (1) PERDS le conflict resolution Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)
00 :00 :46.461 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=N**) envoie une trame applicative Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.462 : In ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=N**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=1,layer=2) labelized #7 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors]
00 :00 :46.462 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)...
00 :00 :46.462 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)
00 :00 :46.466 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)...
00 :00 :46.466 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)
00 :00 :46.469 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)...
00 :00 :46.469 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)
00 :00 :46.484 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)...
00 :00 :46.484 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 1 to all Representative neighboors (score=10000)
00 :00 :46.517 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)...
00 :00 :46.517 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message

instancied by 6 to all Representative neighboors (score=10000)
00 :00 :46.525 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.525 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) TRAITEMENT du message APPLICATIF recu Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.525 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.526 : (6) traite le message d'introduction Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.526 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) envoie une trame applicative Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :46.526 : In ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=6,layer=2) labelized #8 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]]
00 :00 :46.563 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.563 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.567 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.567 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.570 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.570 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.585 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors...
00 :00 :46.585 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Introduction message instancied by 1 to all surrounding neighboors
00 :00 :46.617 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]...
00 :00 :46.617 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :46.762 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)...
00 :00 :46.762 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=N**) TRAITEMENT du message APPLICATIF recu Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)
00 :00 :46.762 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Conflict resolution message instancied by 6 to all Representative neighboors (score=10000)
00 :00 :46.763 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]...
00 :00 :46.763 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=N**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :46.763 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=N) ... et je repete le message de transport interne (1,2) pour traitement [RecMSG applicative frame transport message from (id=6,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :46.763 : (1) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 6 : [role=ROLE_REPRESENTATIVE,group=6]
00 :00 :46.764 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**) envoie une trame applicative Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]

00 :00 :46.764 : In ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=1,layer=2) labelized #8 to (id=-1,layer=2) data=[RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]]
00 :00 :46.827 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]...
00 :00 :46.827 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) TRAITEMENT du message APPLICATIF recu Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :46.827 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :46.828 : (6) traite le message de presentation/whoAreMyNeighboor Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :46.865 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]...
00 :00 :46.865 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :46.868 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]...
00 :00 :46.868 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :46.871 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]...
00 :00 :46.871 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :46.886 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]...
00 :00 :46.886 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [RecMSG applicative frame transport message from (id=1,layer=2) to (id=-1,layer=2) data=Presentation message instancied by 1 : [role=ROLE_SIMPLEMEMBER,group=6]
00 :00 :58.828 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)...
00 :00 :58.828 : COMPOSE_JOIN provenant de ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) recu par représentant (id=6,level=2) previous JOIN state=-1 ecart=1337787048468
00 :00 :58.828 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)
****
* Création de la 3éme abstraction de niveau 3 (agent 1)
****
00 :00 :58.829 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**) est prêt à lancer une composition
00 :00 :58.829 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.865 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)...
00 :00 :58.865 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)
00 :00 :58.866 : ((lvl=0,id=7,role=R)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.866 : ((lvl=0,id=7,role=R)(lvl=1,id=8,role=N)(*lvl=2,id=6,role=N**)(lvl=3,id=6,role=N)a entendu un COMPOSE REQUEST qui l'implique et va donc créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :58.869 : ((lvl=0,id=5,role=R)(lvl=1,id=6,role=N)(lvl=2,id=6,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.869 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)...
00 :00 :58.869 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)
00 :00 :58.869 : ((lvl=0,id=5,role=R)(lvl=1,id=6,role=N)(*lvl=2,id=6,role=N**)(lvl=3,id=6,role=N)a entendu un COMPOSE REQUEST qui l'implique et va donc créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60

00 :00 :58.870 : ((lvl=0,id=9,role=R)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.870 : ((lvl=0,id=9,role=R)(lvl=1,id=8,role=N)(*lvl=2,id=6,role=N**)(lvl=3,id=6,role=N)a entendu un COMPOSE REQUEST qui l'implique et va donc créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :58.871 : COMPOSE_REQUEST recu par ((lvl=0,id=3,role=R)(lvl=1,id=3,role=R)(*lvl=2,id=1,role=N**)
00 :00 :58.871 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)...
00 :00 :58.871 : ((lvl=0,id=3,role=R)(lvl=1,id=3,role=R)(*lvl=2,id=1,role=N**) acception la composition avec 6
00 :00 :58.871 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)
00 :00 :58.872 : ((lvl=0,id=3,role=R)(lvl=1,id=3,role=R)(lvl=2,id=1,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.872 : ((lvl=0,id=10,role=R)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.872 : ((lvl=0,id=3,role=R)(lvl=1,id=3,role=R)(*lvl=2,id=1,role=N**)(lvl=3,id=6,role=N) créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :58.872 : ((lvl=0,id=10,role=R)(lvl=1,id=8,role=N)(*lvl=2,id=6,role=N**)(lvl=3,id=6,role=N)a entendu un COMPOSE REQUEST qui l'implique et va donc créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :58.888 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)...
00 :00 :58.888 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE JOIN from (id=1,level=2)
00 :00 :58.889 : ((lvl=0,id=8,role=R)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.889 : ((lvl=0,id=8,role=R)(lvl=1,id=8,role=R)(*lvl=2,id=6,role=N**)(lvl=3,id=6,role=N)a entendu un COMPOSE REQUEST qui l'implique et va donc créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :58.918 : COMPOSE_REQUEST recu par ((lvl=0,id=4,role=R)(lvl=1,id=3,role=N)(*lvl=2,id=1,role=N**)
00 :00 :58.918 : ((lvl=0,id=2,role=R)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.918 : ((lvl=0,id=4,role=R)(lvl=1,id=3,role=N)(*lvl=2,id=1,role=N**) acception la composition avec 6
00 :00 :58.918 : ((lvl=0,id=2,role=R)(lvl=1,id=3,role=N)(*lvl=2,id=1,role=N**)(lvl=3,id=6,role=N)a entendu un COMPOSE ACCEPT qui l'implique et va donc créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :58.919 : ((lvl=0,id=4,role=R)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.919 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [COMPOSE REQUEST from (id=6,level=2)...
00 :00 :58.919 : ((lvl=0,id=4,role=R)(lvl=1,id=3,role=N)(*lvl=2,id=1,role=N**)(lvl=3,id=6,role=N) créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :58.919 : COMPOSE_REQUEST recu par ((lvl=0,id=2,role=R)(lvl=1,id=3,role=N)(*lvl=2,id=1,role=N**)(lvl=3,id=6,role=N)
00 :00 :58.919 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (1,2) pour traitement [COMPOSE REQUEST from (id=6,level=2)
00 :00 :58.930 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.930 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**)(lvl=3,id=6,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.930 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.931 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.931 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**)(lvl=3,id=6,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.931 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.965 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R)) fait remonter une trame a son abstraction (1,2) pour traitement [COMPOSE REQUEST from (id=6,level=2)...
00 :00 :58.965 : COMPOSE_REQUEST recu par ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**)
00 :00 :58.965 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**) acception la composition avec 6
00 :00 :58.966 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R) Creation de l'abstraction suite a une operation COMPOSE 6 de niveau 3
00 :00 :58.966 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.966 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.966 : ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**)(lvl=3,id=6,role=N) créer l'agent ABSTRAIT Agent #6 : energy=100,00 range=60
00 :00 :58.967 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (1,2) pour traitement [COMPOSE REQUEST from (id=6,level=2)

00 :00 :58.967 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.967 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.970 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.970 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.971 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.971 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.972 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.972 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.973 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.973 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.989 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.989 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :58.990 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :58.990 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :59.059 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :59.059 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**)(lvl=3,id=6,role=N) traite un COMPOSE_ACCEPT [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :59.059 : ((*lvl=0,id=6,role=R**)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :59.059 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :59.059 : ((*lvl=0,id=7,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :59.059 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :59.059 : ((*lvl=0,id=9,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :59.059 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :59.059 : ((*lvl=0,id=10,role=R**)(lvl=1,id=8,role=N)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :59.059 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)...
00 :00 :59.059 : ((*lvl=0,id=8,role=R**)(lvl=1,id=8,role=R)(lvl=2,id=6,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (6,2) pour traitement [COMPOSE ACCEPT from (id=1,level=2) to (id=6level=3)
00 :00 :59.833 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(*lvl=2,id=6,role=R**)(lvl=3,id=6,role=N) Stability : false => je veux créer une abstraction
00 :00 :59.872 : COMPOSE_REQUEST recu par ((lvl=0,id=3,role=R)(lvl=1,id=3,role=R)(*lvl=2,id=1,role=N**)(lvl=3,id=6,role=N)
00 :00 :59.920 : COMPOSE_REQUEST recu par ((lvl=0,id=4,role=R)(lvl=1,id=3,role=N)(*lvl=2,id=1,role=N**)(lvl=3,id=6,role=N)
00 :00 :59.920 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [COMPOSE REQUEST from (id=6,level=2)...
00 :00 :59.920 : COMPOSE_REQUEST recu par ((lvl=0,id=2,role=R)(lvl=1,id=3,role=N)(*lvl=2,id=1,role=N**)(lvl=3,id=6,role=N)
00 :00 :59.920 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (1,2) pour traitement [COMPOSE REQUEST from (id=6,level=2)
00 :00 :59.967 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [COMPOSE REQUEST from (id=6,level=2)...
00 :00 :59.967 : COMPOSE_REQUEST recu par ((lvl=0,id=1,role=R)(lvl=1,id=1,role=R)(*lvl=2,id=1,role=R**)(lvl=3,id=6,role=N)
00 :00 :59.967 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (1,2) pour traitement [COMPOSE REQUEST from (id=6,level=2)
00 :01 :00.433 : [COMPOSE UPDATE from (id=6,level=2) : role is ROLE_REPRESENTATIVE members are 1,6
00 :01 :00.521 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [COMPOSE UPDATE from (id=6,level=2) : role is ROLE_REPRESENTATIVE members are 1,6...

00 :01 :00.521 : ((*lvl=0,id=2,role=R**)(lvl=1,id=3,role=N)(lvl=2,id=1,role=N)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (1,2) pour traitement [COMPOSE UPDATE from (id=6,level=2) : role is ROLE_REPRESENTATIVE members are 1,6
00 :01 :00.568 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R)(lvl=3,id=6,role=N)) fait remonter une trame a son abstraction (1,2) pour traitement [COMPOSE UPDATE from (id=6,level=2) : role is ROLE_REPRESENTATIVE members are 1,6...
00 :01 :00.568 : ((*lvl=0,id=1,role=R**)(lvl=1,id=1,role=R)(lvl=2,id=1,role=R)(lvl=3,id=6,role=N) ... et je repete le message de transport interne (1,2) pour traitement [COMPOSE UPDATE from (id=6,level=2) : role is ROLE_REPRESENTATIVE members are 1,6
00 :01 :08.834 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(*lvl=3,id=6,role=N**) envoie une trame applicative Introduction message instancied by 6 to all surrounding neighboors
00 :01 :08.834 : In ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(*lvl=3,id=6,role=N**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=6,layer=3) to (id=-1,layer=3) data=Introduction message instancied by 6 to all surrounding neighboors] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=6,layer=3) labelized #12 to (id=-1,layer=3) data=[RecMSG applicative frame transport message from (id=6,layer=3) to (id=-1,layer=3) data=Introduction message instancied by 6 to all surrounding neighboors]
00 :01 :18.835 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(*lvl=3,id=6,role=N**) envoie une trame applicative Introduction message instancied by 6 to all surrounding neighboors
00 :01 :18.835 : In ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(*lvl=3,id=6,role=N**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=6,layer=3) to (id=-1,layer=3) data=Introduction message instancied by 6 to all surrounding neighboors] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=6,layer=3) labelized #13 to (id=-1,layer=3) data=[RecMSG applicative frame transport message from (id=6,layer=3) to (id=-1,layer=3) data=Introduction message instancied by 6 to all surrounding neighboors]
00 :01 :28.837 : ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(*lvl=3,id=6,role=N**) envoie une trame applicative Introduction message instancied by 6 to all surrounding neighboors
00 :01 :28.837 : In ((lvl=0,id=6,role=R)(lvl=1,id=6,role=R)(lvl=2,id=6,role=R)(*lvl=3,id=6,role=N**) :Transport d'un message applicatif [[RecMSG applicative frame transport message from (id=6,layer=3) to (id=-1,layer=3) data=Introduction message instancied by 6 to all surrounding neighboors] => [[RecMSG INTERNAL MESSAGE TRANSPORT from (id=6,layer=3) labelized #14 to (id=-1,layer=3) data=[RecMSG applicative frame transport message from (id=6,layer=3) to (id=-1,layer=3) data=Introduction message instancied by 6 to all surrounding neighboors]