Analyse des performances d'un système multi-agents par visualisation

Hussein Joumaa

Université de Grenoble

N° attribué par la bibliothèque

Thèse

pour obtenir le grade de

Docteur de l'Université de Grenoble

Spécialités : « Informatique »

préparée au Laboratoire d'Informatique de Grenoble

dans le cadre de l'école doctorale

« Mathématiques, Sciences et Technologies de l'Information, Informatique (MSTII) »

présentée et soutenue publiquement

par

Hussein Joumaa

le 13 Octobre 2010

Analyse des performances d'un système multi-agents par visualisation

---

Directeurs de thèse :

M. Yves Demazeau (Directeur)

M. Jean-Marc Vincent (Co-directeur)

Jury

M.	Yves Demazeau	Examinateur	DR - CNRS
M.	Bernard Espinasse	Rapporteur	Professeur - Université Paul Cézanne
M.	Jean-Claude Fernandez	Président	Professeur - Université Joseph Fourier
Mme.	Marie-Pierre Gleizes	Examinateur	Professeur - Université Paul Sabatier
M.	René Mandiau	Rapporteur	Professeur - Université de Valenciennes
M.	Jean-Marc Vincent	Examinateur	MCF - Université Joseph Fourier

A Dieu

Remerciements

Je tiens d'abord à remercier mes directeurs de thèse Yves Demazeau et Jean-Marc Vincent pour m'avoir permis de mener à bien cette étude.

Je remercie les membres de mon jury : Monsieur Bernard Espinasse, Professeur à l'Université Paul Cézanne, et Monsieur René Mandiau, Professeur à l'Université de Valenciennes, pour avoir accepté de rapporter cette thèse ; Madame Marie-Pierre Gleizes, Professeur à l'Université Paul Sabatier d'avoir accepté de faire partie de mon jury.

Je tiens à remercier chaleureusement tous mes collègues de travail, les membres (passé et présent) de l'équipe MAGMA LIG : Sylvie Pesty, Julie Dugdale, Humbert Fiorino, Alexandra Berger-Masson, Xavier Clerc, Joris Deguet, Guillaume Piolle, Shadi Abras, Ludivine Crepin, Cyrille Martin, Jérémy Rivière.

Je souhaite remercier mes amis qui m'ont aidé à supporter cette thèse par leur encouragement lors de mon séjour à Grenoble : Hassan Bazzi, Ibrahim Safieddine, Ali Hajj Hassan, Abeer Naserddine, Dr Saleh Amro, Dr Solayman Dahchan, Dr Nabil Abu Shamalla, Jamal Barafi, Wassim Deeb, Hussein Ftouni, Mohammad Ghassani, Nabil Feghali, Fadel Bassal, Hussein Khansa et sa famille.

Je tiens à remercier ma grand-mère, partie à la vie éternelle pendant les années de travail sur cette thèse. Je ne t'oublierais jamais. Je te remercie pour avoir été toujours à mes côtés, m'avoir aidé, m'aimé et éclairé mes jours d'enfance.

Mes remerciements s'adressent à ma famille et tout particulièrement à mon père Jaafar et ma mère Fadia, mes exemples dans cette vie. Ils m'ont soutenu dans tous les moments de la thèse. Grâce à leur confiance, j'ai pu suivre ce chemin difficile. Je ne sais pas si je peux les récompenser de leur longue attente par ces quelques mots de remerciement. Je remercie ma soeur Zeinab pour tous les encouragements. Je remercie mon grand-père Abdul Rahman ; mes tantes et mes oncles surtout Adel, Ahmad, Ali, Mohammad, Nadia, Alia, Siham, Fatmeh, Sabah, Mona et Rima. Je n'oublierais jamais de remercier ma chère Hanaa. Cette thèse est un peu la leur. Merci à toute la famille d'être toujours à côté de moi.

Enfin, je remercie tous les gens que j'aurais involontairement oubliés de remercier.

Table des matières

Synopsis

Chapitre 2 : *Évaluation de performances des SMA: état de l'art*

La question de l'évaluation de performances des SMA n'a été que très peu posée, et n'a motivé que peu de travaux. Dans ce chapitre, nous présentons les principaux travaux réalisés dans le cadre de l'évaluation des performances des SMA. Ensuite, nous situons notre travail par rapport à ceux existants.

Nous distinguons, dans ce chapitre, différents types des travaux d'évaluation existants dans le cadre des SMA:

- Évaluation des méthodes de conception des SMA
- Évaluation des plates-formes d'exécution des SMA
- Évaluation des applications SMA

Chapitre 3 : *La visualisation de l'exécution dans les SMA*

Habituellement, les systèmes multi-agents sont considérés, dans les travaux d'évaluation, comme une boîte noire où les entités interagissent pour achever un but global. Cependant, cette approche ne fournit pas les informations nécessaires à l'analyse complète de l'architecture et du comportement d'un SMA. En effet, l'étude du comportement interne des entités qui constituent un SMA est primordiale pour une compréhension du système en terme de:

- Analyse du comportement des agents dans un SMA;
- Présentation des indices de performances au niveau des agents;
- Présentation des informations de l'utilisation des ressources durant l'exécution;
- Détection des sources de défaillance dans la conception du système observé.

Dans ce chapitre, nous démontrons que la compréhension du comportement interne des entités qui composent un SMA peut être facilitée par la technique de

visualisation, largement utilisée en évaluation des performances des systèmes parallèles.

Section 3.1 : *La visualisation de l'exécution des applications parallèles*

La visualisation de l'exécution des applications parallèles/distribuées se fonde sur le modèle événement/état. Ce modèle ressort la distinction entre deux types des objets traités, pour les entités construisant l'application, durant le processus de visualisation: les états et les événements. La visualisation de l'exécution, dirigée par l'objectif de visualiser les SMA, comporte les phases suivantes :

Définition du modèle d'observation et d'instrumentation :

Le modèle d'observation définit un ensemble des règles décrivant les événements qui doivent être observés et collectés durant l'exécution. Plusieurs méthodes d'observation peuvent être utilisées: le chronométrage, l'échantillonnage, le comptage, le traçage. Le choix d'une de ces méthodes dépend à la fois du niveau de détail des informations demandées et du niveau de perturbation introduite au moment de l'observation.

L'observation des données peut se faire par des techniques appelées techniques d'instrumentation. L'instrumentation, l'insertion du code qui détecte et enregistre les événements, peut se faire aux différents stades du processus de construction du SMA. Nous distinguons: l'instrumentation directe du code source du programme, l'instrumentation pendant la compilation, l'instrumentation de la bibliothèque de communication, l'instrumentation directe de l'image exécutable.

Collecte des données :

La collecte de données est l'enregistrement des événements d'une exécution dans un fichier. La qualité du fichier résultant et sa taille dépendent des choix effectués au niveau de la méthode d'observation et de la technique d'instrumentation.

Analyse les données pour synthétiser et pour estimer les indices de performances :

L'information collectée pendant l'exécution d'un système doit être présentée aux programmeurs sous forme compréhensible. Dans la plupart des cas, l'information telle qu'elle est collectée n'est pas prête à être visualisée et doit subir un certain traitement. Ce traitement vise d'une part à reconstituer la séquence des états du programme observé et à en calculer des indices de performances, d'autre part à réduire les données.

Visualisation des traces et présentation des indices de performance aux développeurs :

La visualisation des traces comporte plusieurs étapes :

- La lecture des traces d'exécution qui extrait les informations du fichier des traces;
- La simulation reproduisant les états du programme à partir des événements des traces;
- Le filtrage qui sélectionne les informations à visualiser;
- La visualisation de l'exécution qui présente les informations à l'utilisateur.

Cette phase est réalisée par un outil de visualisation. Cet outil impose un format du fichier de traces.

Section 3.2 : *Visualisation des SMA*

Dans cette partie, les systèmes multi-agents, fondés sur l'architecture VOYELLES, sont modélisés par des événements et des états. Cette modélisation est fondée sur la représentation des entités (agents et environnement) par des automates.

Ensuite, nous définissons le modèle d'observation pour les SMA. A ce niveau, la visualisation fondée sur les traces est choisie. En effet, cette technique permet d'obtenir le plus d'informations sur le comportement d'un SMA.

Pour les méthodes d'instrumentation, nous adoptons une technique hybride composée de l'instrumentation directe du code source du programme et de l'instrumentation de la bibliothèque de communication. Ce choix est un compromis entre la facilité de l'implantation et la diminution de l'influence de collecte des traces sur l'exécution des SMA.

Puis, nous traitons de la collecte des données et l'analyse de ces données. Ces deux étapes sont plutôt dans la phase technique et seront discutées en détails dans le chapitre 5.

Ensuite, nous traitons du problème de présentation des données de l'exécution d'un SMA. Nous nous sommes orientés vers l'utilisation des outils de visualisation développés et utilisés dans les systèmes distribués/parallèles, vue la similarité avec notre domaine et la difficulté de création d'un outil de visualisation dédié aux SMA dans le cadre de notre travail. Pour démontrer cette similarité nous présentons le modèle de calcul des programmes parallèles (Le modèle de programmation offerte par ATHAPASCAN-0) en le comparant avec le modèle de calcul des SMA fondé sur VOYELLES.

Par la suite, nous listons les critères du choix d'un outil qui s'adapte au SMA. Le choix de cet outil est lié à ses caractéristiques d'extensibilité et de modularité. Alors, un outil adapté aux SMA doit avoir:

- La possibilité de représenter l'exécution des systèmes durant une longue période;

- La possibilité de traiter la visualisation de l'exécution pour un grand nombre des entités constituant le système;
- La possibilité d'ajouter des nouvelles fonctionnalités et d'étendre les fonctionnalités existantes sans la nécessité de changer en totalité l'outil;
- La possibilité de visualiser des entités qui se créent et se détruisent durant l'exécution du système.

Section 3.3 : ***Visualisation de l'exécution dans un SMA: résultats et discussion***

Pour conclure, l'adaptation de la visualisation aux SMA, la modélisation des SMA selon le modèle événement/état et les résultats obtenus suite à l'étape de la présentation des données montrent l'apport de cette technique pour :

- L'analyse du comportement pour un SMA;
- La détection des erreurs du comportement et le débogage;
- La présentation des indices de performance;
- L'utilisation des ressources par les différentes entités;
- La distribution des tâches sur les entités;
- La comparaison entre différents systèmes résolvant le même problème.

Ces résultats sont validés une première fois sur l'application Proies/Prédateurs.

Chapitre 4 : ***Étude de la communication dans l'exécution d'un SMA***

Dans ce chapitre, nous montrons que l'étude et la modélisation des caractéristiques, spécifiques aux SMA, sont intéressantes pour élaborer des indices de performances au niveau de l'ensemble des entités et les liens entre elles. Cette étude est une tâche complémentaire à la technique de visualisation pour avoir une vision globale et complète sur le comportement des SMA.

L'étude et l'évaluation de la plupart des caractéristiques, représentant l'ensemble des entités, se fondent sur l'interaction (chapitre 2). Dans ce chapitre, nous considérons, comme exemple des caractéristiques à modéliser, l'interaction simple fondée sur la communication, et les conversations entre agents. Nous nous concentrons donc sur la dimension interaction de l'architecture VOYELLES [Demazeau (1995)]. Cette dimension représente la suite de la dimension agent traitée dans le chapitre précédent.

Ce chapitre est composé de deux parties. Dans une première partie, une communication est traitée comme une unité isolée de tout ce qui se passe avant son émission et après son traitement.

Dans la deuxième partie, nous considérons les conversations, l'ensemble des communications conformes à un protocole, pour étudier les séquences des communications plutôt que des communications discrètes.

Section 4.1 : *La communication comme une problématique d'évaluation*

Section 4.1.1 : *Modélisation de la communication*

La plupart des travaux d'évaluation dans le domaine des SMA traitent de la communication comme critère de comparaison entre les différents systèmes [Mylopoulos *et al.* (2002a); Cernuzzi et Rossi (2002); Davidsson et Johansson (2002); Jurasovic *et al.* (2006a); Bincheng *et al.* (2004), etc.]. La liste des travaux montre l'importance de la communication comme un critère d'évaluation. Toutefois, la communication est considérée comme un critère de base. Des caractéristiques plus complexes (coopération, coordination, ...) sont évaluées sur la base de sa mesure (une étude au niveau de coopération entre les agents nécessite une étude de base sur les communications).

Trois problèmes peuvent être élevés dans les évaluations des SMA fondées sur la communication:

- Le problème de la subjectivité, en associant une valeur comme résultat de la fonction de quantification de la communication par l'évaluateur;
- Le problème de mesure de la quantité des communications plutôt que la mesure du poids de l'information portée par les unités de communication;
- La considération de la communication comme un critère lié au domaine d'application et pas comme un composant fondamental des SMA.

Comme conséquence de ces problèmes, l'évaluation des SMA au niveau de leurs communications souffre des nombreuses lacunes :

- La mesure de la quantité de communication plutôt que leurs poids ne permet pas de comparer les différents systèmes entre eux. La quantité des communications varie d'un système à un autre selon l'approche de conception utilisée;
- Les approches d'évaluation de la communication sont dépendantes du domaine d'application ce qui rend leurs utilisations ailleurs difficile;
- La subjectivité empêche les évaluations de refléter la réalité de l'application.

Donc nous distinguons certaines situations liées à l'évaluation de la communication, qui sont à la base des problèmes d'évaluations cités ci-dessus. Ces situations sont distinguées dans le but de proposer une approche indépendante du domaine d'application en essayant de minimiser l'effet de subjectivité dans l'évaluation. Les situations de la communication distinguées sur plusieurs SMA sont :

- L'effet d'une unité de communication dans un système est équivalent à "n" unités de communication dans un autre système. (problème 1)
- Les unités de communication reçues et s'avérant non pertinentes pour l'agent (problème 2)

Section 4.2 : *Approches d'évaluation*

La première approche traite de la quantité des communications. Cette approche est utilisée dans les évaluations traitant de la communication réalisées sur les SMA. Le but dans cette approche est de comparer l'évolution de communications durant l'exécution, par la mesure du nombre des messages échangés entre les agents. Cette approche est intéressante pour identifier les liens entre les différents agents et la comparaison entre l'utilisation des canaux de communications par les agents.

Une première idée, pour proposer une approche traitant de la quantité des informations portées par les messages, est de diviser l'ensemble des messages reçus par un agent en des sous-ensembles ayant le même type, ayant alors le même effet sur l'agent. Ensuite, le poids est associé à un message selon son type. La distinction entre les types des messages dépend de l'application des SMA à évaluer.

Le problème principal de cette approche est que la quantité des informations portées par une unité de communication est considérée comme statique et fixe selon le type. Dans la majorité des applications, le même message peut avoir deux effets différents à deux instants différents.

Ensuite, nous proposons les solutions pour les problèmes présentés. Pour traiter du problème 2, nous introduisons la notion de *communication pertinente* où un message pertinent est un message qui change l'état mental d'un agent ou déclenche une action. La solution proposée pour le problème 2 est de mesurer la quantité d'information pertinente plutôt que de mesurer la quantité d'information totale. En ce qui concerne le problème 1, une fonction Q(m) est définie pour affecter un poids aux messages pertinents (le poids 0 est affecté aux autres messages). Une étude sur Q(m) est faite pour élaborer la relation entre les messages reçus et les effets sur les agents. Cette relation est établie en tenant compte du changement de l'état mental de l'agent et des actions

Ce chapitre est divisé en 3 parties. Dans la première partie, nous présentons un comparatif entre différents outils de visualisation candidats pour la visualisation des traces des SMA. Ensuite, après la discussion des motifs de choix, nous présentons l'outil de visualisation Paje. Cet outil est utilisé dans la suite pour visualiser les SMA. Le rôle, l'architecture et le format de données acceptés par cet outil seront ensuite présentés.

Dans la deuxième partie, nous proposons d'automatiser la tâche de visualisation en construisant un système de visualisation des SMA (MAS-Paje). Une architecture de MAS-Paje est proposée et les moyens techniques qui la supportent sont choisis et argumentés. Finalement, nous présentons le rôle de MAS-Paje dans la visualisation d'un SMA sur l'application Proies/Prédateurs.

Chapitre 6 : *La mise en oeuvre de la solution proposée pour évaluer la communication*

Dans ce chapitre, nous présentons les étapes de la réalisation de la plate-forme d'évaluation de la communication dans un SMA. Cette plate-forme est un raffinement de MAS-Paje. Nous nous appuyons sur les informations collectées par MAS-Paje et obtenues sous forme d'un fichier (sous le format SDDF accepté par Paje) et nous étendons les modules de MAS-Paje pour obtenir et pour traiter des informations supplémentaires (les informations concernant l'effet de la communication sur l'agent et les conversations dans les SMA). Puis, nous traitons le cas de l'évaluation des conversations.

Section 6.1 : *Plate-forme d'évaluation de la communication*

Dans cette partie, nous traitons le problème de mise en oeuvre de la plate-forme d'évaluation de la communication, en étendant les travaux faits au niveau de MAS-Paje dans le chapitre précédent. Nous précisons les tâches à réaliser par la plate-forme, assurant une évaluation de communication selon les approches: approche classique, approche type, approche poids des messages et conversations en se fondant sur l'étude théorique du chapitre 4. Des traitements supplémentaires au niveau de la collecte des données, du traitement et de l'analyse de ces données sont nécessaires.

Plate-forme d'évaluation des conversations

Dans cette partie, nous discutons l'architecture générale des tâches à réaliser par la plate-forme effectuant l'extraction de l'ensemble des conversations à partir d'un fichier de traces d'exécution d'un SMA. Les tâches sont présentées dans la figure 1. Comme pour les communications, nous considérons que cette plate-forme est une extension de MAS-Paje. Nous discutons les modifi-

cations au niveau de la collecte des données et du traitement et l'analyse de ces données. Puis, nous présentons les détails techniques.

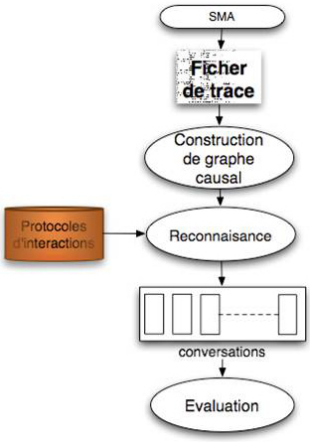

FIG. 1 – *Tâches à réaliser pour étudier les conversations*

Conclusion

Dans ce chapitre, nous avons montré l'applicabilité des solutions présentées pour modéliser et analyser la communication et les conversations dans un SMA. Nous avons proposé des plates-formes qui sont des raffinements de MAS-Paje pour réaliser l'analyse de ces caractéristiques. Ce travail a complété la tâche réalisée par MAS-Paje dans le chapitre précédent. En plus de l'architecture interne des agents, nous avons proposé l'analyse de certaines des caractéristiques représentant les liens (communications simples et conversations) entre les agents. Ce travail nécessite d'être enrichi par l'étude et l'analyse des autres caractéristiques SMA [Demazeau (2004b)], utiles pour l'évaluation des SMA.

Chapitre 7 : *Expérimentation de MAS-Paje sur l'application de collecte de minerai*

Dans le chapitre 7, nous présentons dans un premier temps l'utilité de la visualisation pour obtenir des informations sur les comportements des agents et l'utilisation des ressources par les différentes entités du SMA. Ces informations servent à détecter des problèmes dans l'architecture interne des SMA et d'extraire des schémas significatifs de l'exécution. Dans un deuxième temps, nous utilisons les informations de visualisation pour comparer différents applications SMA résolvant le même problème.

Ce chapitre est constitué de quatre parties, la première partie présente les critères pour choisir le SMA à expérimenter, le problème de collecte de minerai

choisi en fonction de ces critères, les arguments de choix de ce problème et la présentation d'un système multi-agents résolvant le problème. Puis, le protocole expérimental et le modèle de visualisation de ce système sont détaillés dans une deuxième partie. Ensuite, dans la troisième partie nous présentons les résultats de visualisation de l'exécution de ce système en analysant ces résultats. Finalement, nous présentons en bref des visualisations sur deux autres systèmes résolvant le même problème et nous exposons un comparatif des trois systèmes présentés.

Chapitre 8 : ***Expérimentation et Évaluation de l'application de collecte de minerai au niveau de la communication***

Le but de ce chapitre est de valider l'idée principale défendue que l'étude des communications complète le travail réalisé sur la visualisation des SMA et facilite l'étude et la compréhension de comportements des entités dans un SMA. Dans ce chapitre, Nous démontrons l'intérêt de l'évaluation de la communication et des conversations pour:

- La détection des problèmes au niveau des communications (communications non pertinentes, poids des communications);
- L'importance de la visualisation des communications dans un SMA et le couplage entre l'évaluation de la communication et la visualisation de l'exécution;

Puis, nous traitons la modélisation des protocoles de communication et l'étude des communications selon leurs rôles durant les conversations.

Ce chapitre est constitué de 2 parties. Dans la première partie, nous présentons la dimension communication pour l'application de collecte de minerai. La deuxième partie traite de la visualisation et l'étude de la communication et des conversation pour cette application.

Chapitre 9 : ***Évaluation du travail***

Dans ce chapitre, nous présentons une évaluation du travail réalisé. Nous présentons les choix adoptés durant le travail et nous discutons les extensions possibles au niveau de théorie et au niveau de technique.

Ce chapitre est divisé en 4 parties. Dans la première partie, nous présentons une évaluation du modèle de visualisation présenté dans le chapitre 3. Dans la deuxième partie, nous présentons une évaluation du modèle de l'évaluation de la communication présenté dans le chapitre 4. Dans la troisième partie, nous présentons une évaluation de l'architecture de MAS-Paje présenté dans les chapitres 5 et 6. Finalement, nous concluons notre chapitre.

Chapitre 1

Introduction générale

L'Intelligence Artificielle (IA) est un domaine de l'informatique dont le but est de concevoir des systèmes capables de reproduire le comportement intelligent de l'humain dans ses activités de raisonnement. L'approche classique de l'Intelligence Artificielle aborde la résolution d'un problème d'une manière centralisée. Un seul processus se charge d'accomplir les tâches, de gérer les connaissances et de manipuler les ressources. Cette approche centralisée a été remise en cause vu les inconvénients qu'elle présente.

Ces études ont donné naissance à l'Intelligence Artificielle Distribué (IAD) où les processus de raisonnement sont distribués sur plusieurs entités. Les Systèmes Multi-Agents (SMA), issus de l'IAD, cherchent à étudier la manière de répartir un problème sur un certain nombre d'entités autonomes et coopérantes appelées agents. Ces systèmes étudient également la manière de coordonner les comportements intelligents de ces agents selon des lois sociales [Demazeau (1995)].

Les caractéristiques que possèdent les SMA (distribution de l'intelligence, autonomie, coopération et coordination entre les différentes entités, etc. [Demazeau (2004b)]) n'ont fait qu'élargir leurs domaines d'applications. Par contre, cette multiplicité a rendu les SMA complexes, difficiles à analyser dans leur comportement et délicats à évaluer dans leurs performances.

La performance est un critère très important qu'il faut prendre en considération lors de la conception et du développement de tout système informatique. En effet, les concepteurs et les développeurs visent toujours à construire le système qui permette d'atteindre les meilleurs résultats à moindre coût. Reed [Reed (1994)] remarque que les programmeurs admettent des performances "acceptables" et ne cherchent pas à optimiser leurs programmes pour atteindre des meilleures performances. Le domaine des SMA souffre du manque de travaux en matière d'évaluation des performances. Pour toutes ces raisons,

il est nécessaire de développer ou d'adapter les terminologies et les techniques d'évaluation afin de les appliquer dans les SMA.

La visualisation de l'exécution est une des techniques largement utilisées en évaluation des performances pour des systèmes parallèles [ChassinDeKergommeaux *et al.* (2001)]. Cette technique fait partie de la phase de mise au point pour les performances. Elle constitue une aide importante pour la compréhension des comportements et par conséquent pour l'étude, la comparaison et l'amélioration des performances des systèmes informatiques.

Dans le domaine des SMA, la visualisation de l'exécution n'a jamais été traitée véritablement comme un problème de recherche. C'est dans le cadre d'une tentative de construction d'un processus de visualisation de l'exécution d'un SMA que s'inscrit ce travail.

Plan du manuscrit de thèse

Le manuscrit de thèse est organisé en dix chapitres dont le chapitre 1 est cette introduction.

Le deuxième chapitre présente le contexte de travail et l'état de l'art relatif à l'évaluation de performance des Systèmes Multi-Agents selon trois axes :

- L'évaluation des méthodes de conception des SMA: présente des travaux traitant de l'évaluation de cycle de vie de la conception des SMA, de a complexité et du coût de développement des applications SMA.
- l'évaluation des plates-formes d'exécution des SMA: présente des travaux qui traite de l'évaluation des plates-formes d'exécution dédiées aux SMA.
- l'évaluation des applications multi-agents: présente des évaluations des applications SMA.

Le troisième chapitre traite du problème de la visualisation des SMA, afin d'analyser et de comprendre le comportement interne au niveau des entités construisant un SMA. Ce chapitre présente un modèle des SMA pour la visualisation de l'exécution et traite des étapes de la visualisation de l'exécution de ces systèmes.

Le quatrième chapitre traite de la compréhension de l'ensemble des entités et également des liens entre elles. Ce chapitre modélise l'interaction simple fondée sur la communication, et les conversations entre agents. Cette modélisation est effectuée dans le but de présenter les liens entre les agents dans une évaluation fondée sur la visualisation.

Dans le cinquième chapitre, nous présentons la dimension technique du travail. Une dimension technique du travail est nécessaire pour:

- Réaliser la visualisation des SMA;

- Garantir, dans le processus de visualisation, le fait de minimiser l'effet de l'intrusion due aux techniques de collecte utilisées;
- Garantir, dans le processus de visualisation, le fait d'obtenir le maximum d'informations sur le comportement des entités dans un SMA.

Ce chapitre présente MAS-Paje, architecture de plate-forme de visualisation des SMA que nous avons développée.

Dans le sixième chapitre, nous présentons la mise en oeuvre des solutions proposées au niveau des liens entre les agents présentées dans le quatrième chapitre. Cette dimension est une extension de l'architecture de MAS-Paje présentée dans le cinquième chapitre.

Dans le septième chapitre, nous présentons des expérimentations de MAS-Paje sur l'application de collecte de minerai. Ces expérimentations ont pour but de montrer l'utilité de la visualisation pour obtenir des informations sur les comportements des agents et l'utilisation des ressources par les différentes entités du SMA. Ces informations servent à détecter des problèmes dans l'architecture interne des SMA et d'extraire des schémas significatifs de l'exécution.

Le huitième chapitre présente l'expérimentation et l'évaluation de l'application de collecte de minerai au niveau de la communication.

Le neuvième chapitre présente une évaluation du travail et finalement nous concluons la thèse dans le chapitre 10 dans lequel nous exposons nos perspectives du travail.

Chapitre 2

Évaluation des performances des SMA: état de l'art

La question de l'évaluation des performances des SMA n'a été que très peu posée, et n'a motivée que peu de travaux. Dans ce chapitre, nous présentons les principaux travaux réalisés dans le cadre de l'évaluation des performances des SMA. Ensuite, nous situons notre travail par rapport à ceux existants.

La première remarque, lorsque l'on parle d'évaluation de SMA, est qu'on peut avoir deux types d'évaluation : la première évaluant les SMA en tant que paradigme, face à d'autres paradigmes, comme par exemple évaluer l'approche multi-agent face à l'approche objet. La deuxième évaluant les SMA entre eux, principalement dans le sous-domaine de la simulation multi-agents. A ce niveau, nous distinguons différents types des travaux existants :

- Évaluation des méthodes de conception des SMA;
- Évaluation des plates-formes d'exécution des SMA;
- Évaluation des SMA implémentant des applications.

Dans ce chapitre, nous présentons dans premier temps quelques définitions servant pour notre distinction entre les différents types d'évaluation des SMA. Ensuite, nous présentons les travaux d'évaluation dans les SMA. Nous commençons par une présentation des travaux d'évaluation des méthodes de conception des SMA. Ensuite, nous présentons les travaux d'évaluation des Plates-formes d'exécution des SMA. Puis, les travaux d'évaluation des SMA implémentant des applications. Finalement, nous situons notre travail et concluons le chapitre.

2.1 Classification des évaluations des SMA

La plupart des évaluations existantes des systèmes multi-agents se trouvent dans le sous-domaine des simulations multi-agents. Nous présentons quelques définitions concernant les modèles et les théories et les techniques qui se rattachent à cette discipline afin de classifier les évaluations des SMA et de les présenter. Fishwick [Fishwick (1997)], définit la simulation informatique comme un processus composé de trois tâches fondamentales fortement interdépendantes :

- Elaboration du modèle.
- Exécution du modèle sur ordinateur.
- Analyse de l'exécution du modèle et des résultats obtenus.

Cette définition est illustrée et augmentée par une méthode distinguant les différentes étapes dans le processus de simulation dans [Shannon (1998)].

D'après ces méthodes nous soulignons déjà la forte distinction entre conception du modèle et implémentation et exécution du modèle sur l'ordinateur.

D'après ce qui est dit dans ces travaux nous distinguons entre deux types d'évaluation: l'évaluation des méthodes de conception des systèmes concernant la partie modélisation et l'évaluation de l'exécution et la comparaison des systèmes implémentant les applications tels que résultant des méthodes de conception.

Les simulations multi-agents suivent les mêmes étapes distinguées dans les travaux présentés. La figure 2.1 présente les parties impliquées dans l'élaboration d'un SMA. La simulation d'un système multi-agents se fonde donc sur un problème à résoudre. Ce problème est modélisé par un SMA en suivant une méthode de conception des SMA. La méthode de conception réfère à la partie modélisation de [Fishwick (1997)] et de [Shannon (1998)]. Cette méthode élabore des architectures des systèmes multi agents qui sont ensuite implémentés et exécutés en se fondant sur une plate-forme d'exécution d'un SMA.

Comme nous nous intéressons dans cette thèse à l'analyse des performances des SMA, nous considérons la partie propre aux SMA de la figure 2.1. Nous abandonnons les parties concernant l'évaluation du support exécutif.

La question qui se pose alors est pourquoi présenter des travaux d'évaluation des méthodes de conception et des plates-formes d'exécution des SMA dans un travail d'analyse de performances des SMA?

Les SMA sont représentatifs des méthodes utilisées pour les concevoir. La comparaison et l'évaluation des méthodes de conception amènent donc à comparer et évaluer indirectement les SMA. De plus, l'évaluation des méthodes met en avant et clarifie ce qu'offrent réellement les SMA. En ce qui concerne les plates-formes, ils ont une influence négligeable sur la performance des systèmes étant l'intermédiaire entre le SMA et le support exécutif. Ces plates-

formes constituent un medium d'échange de communications et d'interaction entre les agents.

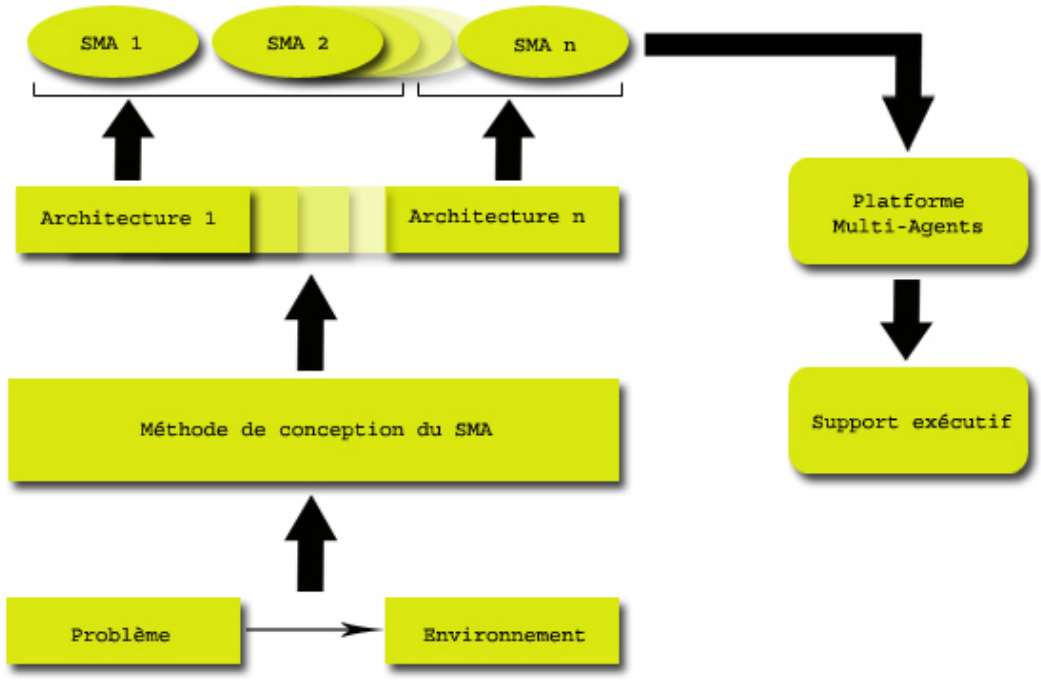

FIG. 2.1 – *Les parties impliquées dans l'élaboration d'un SMA*

2.2 Évaluation des méthodes de conception des SMA

L'importance des méthodes de conception des SMA est discutée dans de nombreux travaux [Jennings *et al.* (1998); Jennings et Wooldridge (2000); Iglesias *et al.* (1999a); Bresciani *et al.* (2003); Bayer et Svantesson (2001)]. Par exemple, selon [Iglesias *et al.* (1999a)], l'ingénierie logicielle orientée agent (AOSE) est une condition nécessaire pour introduire les SMA dans l'industrie comme une approche effective de conception. L'évaluation des méthodes de conception des SMA contribue au niveau dans la communauté de recherche SMA à trouver une méthode de conception standard.

Une comparaison entre les méthodes des conceptions des SMA est disponible dans [Bayer et Svantesson (2001)]. Ce travail examine la similarité entre les méthodes GAIA [Wooldridge *et al.* (2000)] et MAS-CommonKADS [Iglesias *et al.* (1998)]. Ce travail n'évalue pas ces méthodes et ne propose pas des

techniques pour une telle évaluation. Une comparaison similaire est présenté dans [Cox et DeLoach (2000)]: les auteurs comparent GAIA et MaSE [DeLoach *et al.* (2001)] et concluent que MaSE est plus détaillée que GAIA. Mais dans ce travail il ne proposent pas d'étapes pour comparer des méthodes de conception. [Bayer et Svantesson (2001)] et [Cox et DeLoach (2000)] se focalisent sur l'expressivité de la méthode examiné.

Le travail de [Sabas *et al.* (2002)] propose une approche d'évaluation des méthodes de conception des SMA. Cette approche vise à régler l'incomplétude des méthodes de conception des SMA en essayant de les comparer pour ensuite obtenir une méthode standard.

L'approche est composée de six dimensions:

- Dimension Méthodologie (étapes de processus, modèles de développement, approches de développement, degré d'implication de l'utilisateur, moment d'implication de l'utilisateur, réutilisabilité, disponibilité de supports logiciels et méthodologiques)
- Dimension Représentation (découpage du système, formalisme, séquencement des modèles, qualité des modèles)
- Dimension Agent (nature des agents, types d'agents, attributs des agents, attributions des agents)
- Dimension Organisation (image d'organisation, nature de l'environnement, caractéristiques des données)
- Dimension Coopération (type de communication, modèle de communication, langage de communication, modèle de coopération, type de contrôle, interaction)
- Dimension Technologique (mode de traitement, type d'interface homme-machine, programmation, type d'application visée)

Suite à ce travail les auteurs ont comparé 9 méthodes de conception. Ils ont obtenu des résultats démontrant qu'aucun méthode de conception n'est tout à fait complète. Par la suite, ils ont proposé une piste de standardisation des méthodes de conception des SMA. Un exemple des comparaisons entre les méthodes de conception selon la dimension agent est présenté dans la figure 2.2.

[nhu Numi Tran *et al.* (2003)] proposent une approche fondée sur les dimensions : processus de conception, technique de conception, modèle proposée, et support de la méthode.

D'autres travaux existent proposant des comparaisons entre les différents méthodes de conception comme dans [Sudeikat *et al.* (2004); Iglesias *et al.* (1999b); Sturm et Shehory (2003)].

		METHODOLOGIES								
Criteria	**Values of criteria**	**GAIA**	**MaSE**	**MMTS**	**HLIM**	**CoMoMAS**	**MASB**	**MAS-Common KADS**	**AOME M**	**Cassiopeia**
Nature of the agents	homogeneous	N	N	N	N	N	N	N	N	N
	heterogeneous	Y	P	Y	Y	Y	Y	Y	Y	Y
Type of agents	intelligent agents	P	P	Y	Y	Y	Y	Y	Y	Y
	interface agents	P	P	Y	Y	Y	Y	Y	Y	Y
	mobile agents	P	P	Y	P	Y	P	Y		Y
	information agents	P	Y	Y	Y	P	Y	Y	Y	P
	autonomous agents	P	P	Y	P	Y	Y	Y	Y	Y
Agents attributes	adaptability	P	P	P	P	Y	P	P	P	P
	autonomy	P	P	Y	Y	Y	Y	Y	Y	Y
	cooperative behaviour	P	P	Y	Y	Y	Y	Y	Y	Y
	inferential capability	P	P	Y	Y	Y	Y	Y	Y	Y
	communication ability	P	P	Y	Y	Y	Y	Y	Y	P
	mobility									
	personality	P	P	Y	Y	Y	Y	Y	Y	Y
	reactivity	P	P	Y	Y	Y	Y	Y	Y	Y
	temporal continuity	P	P	N			N			Y
	deliberative behaviour	P	P				N	Y	P	P
Agents attributions	physical stance	N	N		N	N	Y	P		P
	design stance	Y	Y	Y	Y	Y	Y	Y	Y	Y
	intentional stance	Y		Y	Y	Y	Y	Y	Y	Y

FIG. 2.2 – *Comparaisons entre les méthodes de conception, selon la dimension agent [Sabas* et al. *(2002)]*

2.3 Évaluation des plates-formes d'exécution des SMA

Dans cette partie nous présentons des évaluations des plates-formes d'exécutions des SMA.

L'importance de plate-forme d'exécution d'un SMA est discuté dans [Leszczyna (2004); Mulet *et al.* (2006)].

[Jurasovic *et al.* (2006b)] a pour objectif d'analyser les performances et de comparer deux plates-formes de développement des systèmes multi-agents : Grasshopper et JADE. Pour chaque plate-forme, le travail est consacré d'une part à la mesure de la moyenne du temps d'aller-retour requis pour l'échange circulaire de messages entre deux agents, d'autre part, sur l'estimation du trafic engendré durant cette procédure.

Les tests ont été réalisés en utilisant des messages de différentes tailles et en utilisant des techniques de codage différentes. Le but de ce travail était de pouvoir déterminer la quantité maximale de données pouvant être transférée via le réseau durant une période de temps fixée. Les résultats obtenus ont pu montrer que la plate-forme JADE est plus performante quand il s'agit de l'envoi de messages de taille importante alors que Grasshopper l'est lorsqu'il s'agit d'envoyer des messages de petite taille.

[Trillo *et al.* (2007)] proposent une comparaison entre les plates-formes Aglets, Voyager, Grasshoper, Tryllian, JADE, Tracy, Springs. Cette comparaison a pour but de trouver la plate-forme la plus appropriée pour l'utilisation des agents mobiles.

Les agents mobiles constituent une technologie pour développer des systèmes composés des entités autonomes et mobiles. La mobilité implique le déplacement physique de l'agent dans le réseau ou dans l'architecture du système.

D'autres travaux d'évaluation des plates-formes d'exécution des SMA en suivant les mêmes principes existent encore dans [Mulet *et al.* (2006); Nguyen et Dang (2002)].

2.4 Évaluation des SMA implémentant des applications

Dans cette partie nous nous intéressons à l'évaluation et à l'analyse des SMA. Nous présentons des outils et des méthodes proposés pour évaluer ces SMA.

Évaluation des styles d'organisation pour le projet TROPOS

Un style d'architecture permet de décrire les différents modules qui composent un système et les liens qui peuvent exister entre eux. Dans un contexte multi-agents, ces modules sont vus comme des organisations d'agents interagissant les unes avec les autres. Lors de l'élaboration de la méthode TROPOS [Bresciani *et al.* (2004)], il a été important de savoir quel style d'architecture organisationnelle proposer à l'utilisateur. C'est ce que [Mylopoulos *et al.* (2002b)] ont fait en proposant une évaluation de plusieurs styles d'organisation dans le but de les comparer.

En tout, dix styles ont été alors présentés et neuf critères ont été proposés pour évaluer les différentes organisations. Ces critères sont :

- Prédictibilité : les comportements des agents et du système sont-ils prévisibles ?
- Sécurité : le système sait-il reconnaître des données erronées et protéger ses propres données ?
- Adaptabilité : les agents sont-ils capables de s'adapter aux changements de leur environnement ?
- Coopérativité : les agents sont-ils capables de coopérer pour arriver à un but commun ?
- Compétitivité : les agents se mettent-ils en compétition pour obtenir le meilleur résultat ?
- Disponibilité : les services offerts par le système sont-ils disponibles sans interruption ?
- Intégrité : le système se montre-il robuste en présence de problèmes ?
- Modularité : est-ce que le système est modulable ?
- Agrégabilité : existe-t-il des agrégations entre les agents, augmentant l'efficacité mais diminuant la flexibilité ?

Ensuite, pour chacun des dix styles d'organisations proposés, une valeur est associée à chaque critère. Les valeurs pouvant être attribuées sont : partiel positif, suffisant positif, partiel négatif et suffisant négatif, respectivement notées +, ++, - ,- -. L'objectif initial est alors atteint : l'utilisateur est capable de choisir entre différents styles d'architecture proposés par la méthode TROPOS, selon ses besoins.

Outils d'analyse (BIOMAS/GEAMAS)

Le système multi-agents Biomas a été développé pour simuler des flux de matières organiques (MO) [Courdier et al., 2002].

L'objectif est d'aider les agriculteurs à expérimenter des options de gestion par simulation. Ce travail propose un outil d'analyse de simulation. L'outil contient un traceur de messages et un grapheur de courbes d'évolution d'attributs dynamiques. De plus, il propose une représentation de l'ensemble des agents et des objets positionnés sur le territoire.

La plate-forme GEAMAS est équipée de fonctionnalités de paramétrage de l'outil d'observation qui peut être utilisé en cours de simulation ou après simulation.

Quantification des caractéristiques des SMA et expérience

Le travail de [Bonnet et Tessier (2008)] propose une formalisation des critères d'évaluation relatifs aux notions de *performance*, de *stabilité* et de *capacité d'extension*. La théorie est utilisée pour évaluer un algorithme de coordination dans le domaine des satellites d'observation.

- La performance comporte des indicateurs statistiques comme la consommation des ressources, le temps de réponse, le temps de calcul et la charge de communication.
- La stabilité est liée au concept d'équilibre. Lorsqu'un évènement vient affecter le système, ce dernier est réputé stable s'il parvient à revenir à un état d'équilibre.
- La capacité d'extension représente la capacité du système à accroitre ses performances alors que sa taille augmente elle aussi.

Il existe d'autres travaux pour appréhender la notion d'évaluation et de tests de performances dans le cadre des systèmes multi-agents en quantifiant les caractéristiques usuelles de ces derniers. Un des tous premiers travaux est celui de [Berreur (2005)] qui consiste à mesurer l'*ouverture* et l'*intelligibilité* du système multi-agents.

L'ouverture est une caractéristique qui possède trois aspects : l'ouverture sur l'environnement, l'ouverture sur l'utilisateur et l'ouverture sur les autres agents. Chacun de ces aspects est quantifié selon le nombre d'échanges entre l'agent et l'une des trois facettes : environnement, utilisateur, agents.

L'intelligibilité représente l'aptitude du système à présenter à l'utilisateur une vision compréhensible de son état. Elle est vue comme étant le rapport du nombre de représentations graphiques différentes d'un agent sur le nombre de ses différents états internes.

Un autre travail [Joumaa (2006)] consiste à trouver une fonction de mesure pour évaluer l'*adaptation* du système. L'adaptation étant la capacité du

système à réagir aux changements de son environnement pour l'adapter à ce dernier. Elle est vue comme étant la variation des valeurs de performance dans des intervalles de temps significatifs qui correspondent à des changements dans l'environnement.

Un dernier travail comportant un analyse des critères d'évaluation de systèmes multi-agents adaptatifs est présenté dans [E. Kaddoum *et al.* (2009)].

Outils d'observation et de visualisation des SMA

Les travaux de visualisation dans les SMA sont principalement des supports graphiques pour le développement et l'implémentation associés à des plates-formes de développement des SMA. Chaque plate-forme propose des interfaces graphiques facilitant et accélérant le développement et l'implémentation. Elles peuvent servir à la création du modèle, à la création des agents, à l'élaboration de conversations, au déploiement des agents sur les différentes plates-formes, etc. Par la suite, nous présentons des exemples des interfaces graphiques d'observation et de visualisation associés à ces plates-formes.

Madkit [O. Gutknecht (2000)] est doté d'un modèle graphique fondé sur des composants graphiques indépendants. Chaque agent est responsable de ses propres interfaces graphiques. Un noyau graphique se charge de lancer les interfaces et les gérer dans une interface globale. L'interface graphique de Madkit propose de visualiser les traces des mouvements des agents dans l'environnement et leurs actions sur l'environnement.

JADE [Bellifemine *et al.* (2001)] propose deux outils graphiques pour déboguer les systèmes multi-agents réalisés : l'agent Dummy et l'agent Sniffer. Dummy est un outil pour inspecter les échanges de messages entre agents, et offre une interface graphique pour l'édition, l'écriture et l'envoi des messages ACL vers les agents, permet de recevoir et lire les messages des autres agents, et éventuellement de sauvegarder ces messages. Sniffer trace l'échange de messages et donne une interface graphique pour afficher les échanges de messages entre les différents groupes d'agents on utilisant une notation proche d'UML.

Zeus [ZEUS (2000)] propose une interface graphique pour visualiser la société des agents et les agents individuels. Cette interface propose un certain nombre de graphes représentant les interactions par le nombre et le type des messages échangés.

Ces plates-formes proposent des visualisations qui sont :

- Limitées : les informations présentées dans les visualisations et leurs formes de présentation sont en nombre et en qualité limitées;

- Statiques : montrent une vision unique, inchangeable de la performance d'un SMA implémentant une application. Cette dimension est particulièrement claire dans les visualisations proposées par Zeus et Jade.
- Non interactives : les visualisations ne permettent pas de présenter différents niveaux d'abstraction en changeant l'échelle de temps. Cette dimension est claire dans les visualisations proposées par Zeus et Jade.

Finalement, les visualisations présentées ne servent que rarement à sortir des indices de performances (comme dans Madkit).

2.5 Conclusion

Tout le long de ce chapitre nous avons présenté un tour d'horizon sur les différents types des travaux d'évaluation dans le domaine des SMA.

Dans une première partie, nous avons présenté l'évaluation des méthodes de conception des SMA. Ensuite, les travaux d'évaluation des plates-formes d'exécution des SMA et finalement les évaluations des applications SMA.

Suite aux travaux présentés nous remarquons que:

- Le domaine des SMA souffre de lacunes en termes de travaux d'évaluation;
- Les travaux d'évaluation des SMA indépendant d'un domaine d'application sont rares. Les travaux qui existent proposent un modèle d'évaluation dépendant du domaine d'application;
- Les travaux d'évaluation des méthodes de conception et des SMA implémentants des applications sont fortement liés. Les systèmes sont représentatifs des méthodes utilisées pour les concevoir. Une comparaison des systèmes se ramène souvent à comparer indirectement des méthodes de conception entre elles;
- Les travaux d'évaluation reste au niveau de proposer des critères qualitatifs. L'absence de mesure directe sur l'application et l'intervention de l'évaluateur dans ce type des critères augmente la subjectivité de l'évaluation;
- Les techniques proposées à l'observation et l'analyse des SMA sont rares et associés à des domaines d'application spécifiques;

Notre travail se situe au niveau de l'évaluation des SMA implémentant des applications. Nous entamons à présent une nouvelle partie de notre rapport qui consiste à proposer une solution à la problématique posée, en répondant aux remarques tirées de chapitre.

Chapitre 3

La visualisation de l'exécution dans les SMA

Les travaux d'évaluation des systèmes multi-agents (chapitre 2), particulièrement dans les sous-domaines comme la simulation, traitent ces systèmes comme une boîte noire où les entités interagissent pour réaliser un but global. Les indices et les critères marquant leurs comportements sont étudiés du point de vue externe. Cette observation résulte du fait que c'est le comportement global du système qui est recherché et donc observé.

Cependant, la vision "boîte noire" paraît insuffisante pour le domaine des SMA. Il ne suffit pas d'appréhender un SMA par ses seules entrées et sorties, ou encore par les services requis et les services fournis. Cette vision ne fournit pas les informations nécessaires à une analyse complète de structure et du comportement des entités dans un SMA. Une étude de point de vue interne (approche boîte blanche) des entités qui constituent un SMA est primordiale pour une compréhension du système en termes de :

- Analyse du comportement interne des agents dans un SMA;
- Présentation des indices de performances au niveau de l'architecture interne des agents;
- Présentation des informations concernant l'utilisation des ressources par les agents durant l'exécution;
- Détection des sources de défaillance dans la conception du système observé.

Dans ce chapitre, nous montrons que la compréhension du comportement interne des entités qui composent un SMA est facilitée par la technique de visualisation de traces d'exécution, largement utilisée en évaluation de performances des systèmes parallèles.

Ce chapitre est composé de trois parties. Dans la première partie nous présentons la visualisation de l'exécution des systèmes parallèles/distribués, dirigée par l'objectif de visualiser les SMA.

Dans la deuxième partie, nous présentons une abstraction des SMA pour la visualisation de l'exécution et nous traitons des étapes de la visualisation de l'exécution de ces systèmes.

Finalement, nous concluons sur l'adaptation de cette technique aux SMA en montrant son apport pour l'évaluation des SMA par rapport au quatre points présentés ci-dessus.

3.1 La visualisation de l'exécution des applications parallèles

La visualisation est un moyen pour présenter à l'utilisateur les informations sur le comportement de l'application au cours de son exécution dans le but de les analyser [ChassinDeKergommeaux *et al.* (2001)]. La qualité des visualisations proposées par l'environnement d'évaluation de performances influence beaucoup la facilité et la précision avec laquelle cette analyse peut être menée.

En effet, nous n'avons pas la possibilité de représenter les informations telles qu'elles sont dans la réalité de l'exécution d'une application parallèle sur une machine parallèle [Post et Hin (1991)]. Ce qui entraîne une diversité et une richesse des types des représentations proposées. La façon de présenter les informations afin qu'elles soient facilement compréhensibles constitue le principal problème de la visualisation pour l'évaluation de performances [Miller (1993)]. Chaque environnement d'évaluation de performances possède sa propre représentation du comportement de l'application, principalement en le caractérisant par des indices de performances et en se fondant sur des **abstractions** représentant l'exécution. Ces abstractions utilisées sont réalisées pour être les plus proches possibles de la manière dont l'utilisateur représente son application et ses caractéristiques afin de faciliter la compréhension (figure 3.1).

Une visualisation de l'exécution des systèmes parallèles se fonde donc sur une abstraction des données du **modèle d'exécution** de ces systèmes. Un modèle d'exécution définit les propriétés essentielles d'un système parallèle. Par essentiel, nous désignons les communications entre les processus (communication par message, par appel procédural, par mémoires reparties, par diffusion), les types de comportement des processus (synchrone ou asynchrone), etc. Parmi les modèles d'exécution existant pour les systèmes parallèles, nous citons les modèles de programmation à mémoire partagée [Levrouw *et al.* (1994); Mellor-Crummey (1989)] et par passage de messages [Netzer et Miller (1992); Hurfin *et al.* (1992); Leu et Shiper (1992)].

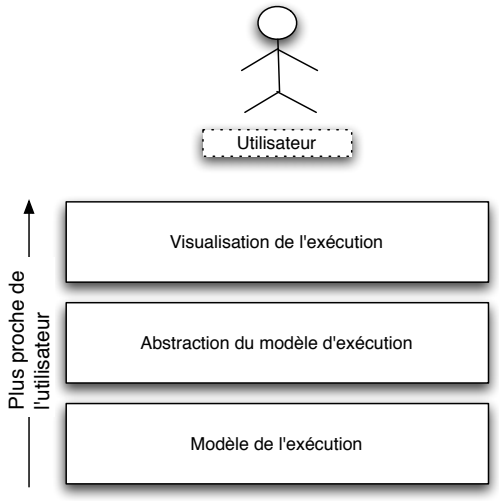

FIG. 3.1 – *Modèle de l'execution, abstraction et visualisation selon l'utilisateur*

À partir d'une abstraction de l'exécution, il est possible de définir un type de visualisation représentant l'exécution d'un système parallèle. Différentes abstractions sont proposées pour les programmes parallèles.

Une abstraction considérée comme la plus complète dans [ChassinDeKergommeaux *et al.* (2001)] est celle présentée par É. Leu [Leu (1992)]. Dans cette abstraction une exécution de programme parallèle X est modélisée par :

- Une séquence d'états (l'état initial est S_0^X);
- L'ensemble E^X des événements élémentaires de X;
- Une relation R^X d'ordre partiel entre les événements de X appelée relation de précédence directe.

Une exécution pour chaque processus p_i de X est représentée comme une séquence des événements, nommée "local history" du processus :

$$h_i^X = e_{i,0}^X, e_{i,1}^X, e_{i,2}^X, ...$$

Avec $e_{i,k}^X$ représente le *kième* événement du processus p_i. L'ensemble des événements est noté :

$$E^X = \{e_{i,k}^X / 1 \leq i \leq N, 1 \leq k \leq |h_i^X|\}$$

$|h_i^X|$ est le nombre des événements possibles dans un "*local history*" h_i^X.

Chaque événement $e_{i,k}^X \in E^X$ représente l'exécution d'un ensemble d'instructions. Il est considéré comme une fonction ϕ qui change l'état local d'un

processus p_i de $s^X_{i,j-1}$ en un état local $s^X_{i,j}$ (l'état initial du processus p_i est défini comme $s^X_{i,0}$).

Un **événement élémentaire de calcul** est défini comme un ensemble d'instructions ne comprenant aucun accès à la mémoire partagée, aucun envoi et aucune réception de messages.

Un **événement élémentaire de communication** est défini comme l'une des trois actions suivantes : envoi de message, réception de message ou accès à une unité de mémoire partagée.

Une **relation de précédence directe** R existe entre deux événements $e_{i,j}$ et $e_{k,l}$ si et seulement si l'une au moins des trois conditions suivantes est satisfaite :

- $e_{i,j}$ et $e_{k,l}$ sont des événements consécutifs d'un même processus : $e_{k,l} = e_{i,j+1}$;
- $e_{i,j}$ est l'émission d'un message et $e_{k,l}$ sa réception;
- $e_{i,j}$ et $e_{k,l}$ sont deux accès consécutifs à une même unité de mémoire partagée m

Cette abstraction est connue sous le nom de "modèle événement/état". Ce modèle est à la base de la plupart des visualisations d'exécution dans les systèmes parallèles.

Pour conclure, le comportement d'une application, selon cette abstraction, est représenté par l'évolution de son état au cours de son exécution. Un changement d'état intervient lorsqu'on a l'occurrence d'un événement, par exemple lorsque deux tâches échangent des données ou passent d'une phase de calcul à une autre.

Suite à cette abstraction, et dans le but de visualiser le comportement d'une application parallèle, un certain nombre de questions se posent. Nous répondons par des "réponses rapides", qui seront interprétées dans la section suivante :

- Cette abstraction permet la construction d'une représentation de l'exécution d'un programme parallèle fondée sur les états et les événements. Quelles sont les données à observer qui sont nécessaires à la production de cette représentation?
  - *Définir un modèle d'observation.*
- Comment ces données sont-elles obtenues?
  - *Définir une technique d'instrumentation.*

- Par quels moyens les transforme-t-on en indicateurs visuels?
  - *Analyser les données collectées.*
- Avec quels outils et sous quelle forme effectue-t-on la visualisation de ces informations et leurs analyses?
  - *Présenter les informations sur l'application.*

Le processus de visualisation de l'exécution des systèmes parallèles constitue donc l'ensemble des étapes qui répondent à ces questions, et sert à obtenir une représentation des données caractérisant l'exécution. Différentes alternatives et solutions sont proposées pour répondre aux questions posées ci-dessus. La facilité et la pertinence de l'application de ces alternatives dépendent beaucoup du domaine d'application.

Dans ce qui suit, nous présentons les phases de la visualisation de l'exécution des programmes parallèles dans l'objectif de visualiser les SMA. Nous abordons donc les alternatives utiles pour les SMA sans présenter toutes les alternatives existantes pour visualiser les systèmes parallèles de manière exhaustive.

3.1.1 Modèles, méthodes d'observation et techniques d'instrumentation

Modèles d'observation

Un événement est défini comme un ensemble d'instructions exécutées par un processus. Le modèle d'observation définit l'ensemble des instructions décrivant les événements (de l'abstraction) qui doivent être observés et collectés durant l'exécution.

Plusieurs méthodes d'observation peuvent être utilisées pour collecter les informations concernant les événements durant l'exécution : chronométrage, échantillonnage, comptage ou traçage des événements. Le choix d'une méthode parmi ces quatre dépend à la fois :

- Du niveau de détail des informations demandées;
- Du niveau de perturbation introduite au moment de l'observation.

A ce stade, nous ne traitons pas du problème de datation des événements. Ce problème est le problème principal au niveau de la collecte des données. La méthode d'observation considère à ce niveau qu'il existe une horloge pour l'évaluation du temps.

Le chronométrage :

Le chronométrage consiste à mesurer le temps passé entre deux événements et à enregistrer la somme du temps passé dans chaque état "intéressant" pour les entités constituant le système. Par exemple, pour obtenir le temps cumulé passé dans chaque état de l'exécution d'une entité du système, la date de l'événement "fin de l'état", soustrait à la date de l'événement "début de l'état" correspondant, est accumulée dans une variable associée à chaque état. À la fin de l'exécution du programme, ou périodiquement pendant l'exécution, l'ensemble de ces variables est enregistré.

L'échantillonnage :

L'échantillonnage consiste à observer périodiquement l'état de l'exécution des entités d'un système et à incrémenter un compteur associé à chaque état observé. Le temps passé dans un état est supposé proportionnel à la valeur du compteur associé.

Le comptage :

Le comptage consiste à enregistrer le nombre d'événements de chaque type qui ont eu lieu pendant l'exécution d'un système. Nous comptons, par exemple, le nombre de fois où une entité passe dans un état donné.

Le traçage :

Le traçage est la technique d'observation de l'exécution de programmes parallèles la plus générale. Elle consiste à engendrer une séquence d'événements. Chaque événement correspond à un ensemble d'instructions qui modifie l'état du système (par exemple, des communications entre les processus). Cette séquence d'événements est appelée trace et peut être enregistrée et exploitée pour permettre de reconstruire les états du système et donc de calculer les métriques de performances puisqu'on obtient la totalité des informations que l'on peut obtenir sur une exécution. Chaque enregistrement d'événement contient les attributs suivants [Guilloud (2004); Maillet (1990); Reed (1993a)] :

- L'identification de l'action et de l'état dans laquelle elle est effectuée;
- La date d'occurrence de l'événement;
- Les informations supplémentaires caractéristiques de l'événement (par exemple, l'identifiant du ou des entités de destination dans le cas d'une action d'envoi de message).

La trace permet donc de savoir quelle action a lieu, où et quand. Elle permet d'obtenir le graphe des changements d'états des entités et fournit également la suite des interactions entre les entités et toutes les autres informations qui peuvent être considérées comme utiles dans un système.

Techniques d'instrumentation

L'instrumentation, l'insertion du code qui détecte et enregistre les événements, peut se faire aux différents stades du processus de construction des systèmes. Nous distinguons : l'instrumentation directe du code source du programme, l'instrumentation pendant la compilation, l'instrumentation de la bibliothèque de communication, l'instrumentation directe de l'image exécutable.

L'instrumentation directe du code source du programme :

Elle consiste à modifier le programme de l'utilisateur en y insérant, avant la compilation, le code qui engendre les événements. Par conséquent, cette approche requiert un composant, qui effectue la traduction du code source vers un code équivalent instrumenté. Typiquement, un tel composant insère du code d'instrumentation à chaque appel de l'API de communication ainsi qu'aux entrées et sorties des états (à l'occurrence des événements provoquant l'entrée ou la sortie des états).

L'instrumentation pendant la compilation :

Elle consiste à modifier le compilateur pour insérer le code qui engendre les événements durant la compilation. Cette approche facilite l'intégration d'informations sémantiques (changement d'états, variables etc.) dans les traces [Larus (1993)].

L'instrumentation de la bibliothèque de communication :

Elle consiste à insérer le code qui engendre les événements dans la bibliothèque de communication. Cette technique ne requiert aucune modification, ni même de recompilation du programme. Il suffit, en général, de relancer l'application en activant le code de traçage compilé dans la bibliothèque.

L'instrumentation directe de l'image exécutable :

Elle consiste à effectuer l'instrumentation pendant l'exécution du programme indépendamment du langage de programmation et sans la nécessité de recompilation du programme.

3.1.2 Collecte des données

La collecte des données est l'enregistrement des événements d'une exécution dans un fichier. La collecte des données traite des problèmes présentés selon 3 axes :

– La qualité des informations enregistrées

- La quantité des informations enregistrées
- Le format de l'enregistrement de ces informations

Qualité des informations enregistrées

Dans cette partie, nous présentons brièvement les facteurs qui affectent la qualité de l'information collectée. Ces facteurs seront discutés dans la section suivante dans le cas d'un SMA.

La qualité du fichier résultant et sa taille dépendent initialement des choix faits au niveau de la méthode d'observation et de la technique d'instrumentation. En plus de ces facteurs, il existe plusieurs niveaux de collecte des données [Hofmann (1996)] : matériel, logiciel et hybride.

La collecte au niveau logiciel est le niveau de collecte le plus répandu car le plus facile à mettre en oeuvre. Cependant, il rend difficile l'obtention des informations de haute qualité en raison de :

- L'absence d'horloge globale dans la plupart des systèmes répartis
- L'intrusion provoquée par l'enregistrement et le transport des informations concernant l'exécution des systèmes parallèles.

Dans le cas idéal, les événements enregistrés sont datés avec une horloge globale de précision infinie et il n'existe aucune intrusion due à la collecte des données. Cependant ce n'est pas le cas en général et particulièrement pas dans le cas du traçage logiciel.

Absence d'horloge globale :

Le défaut d'horloge globale dans un système parallèle à mémoire répartie peut avoir comme conséquence des incohérences entre les événements enregistrés, s'ils sont datés en utilisant les horloges locales des processeurs non synchronisées. Par exemple, la date de réception d'un message entre processeurs différents peut être inférieure à sa date d'émission. De telles incohérences rendent difficile ou impossible l'analyse des traces d'exécution par des outils d'évaluation de performances. Sur des traceurs matériels ou hybrides, ce problème est résolu en utilisant le matériel dédié [Hofmann (1996)]. Sur des traceurs logiciels, ce problème peut être résolu par la mise en place d'un algorithme logiciel de correction d'horloge [Jacobson *et al.* (1986); Maillet (1996)].

Intrusion due à la collecte :

La collecte de données perturbe l'exécution des programmes parallèles observés. Il est difficile d'estimer l'intrusion du traceur puisqu'elle dépend du

programme tracé et du nombre d'événements tracés. En cas de collecte hybride ou matérielle, on suppose que l'intrusion reste limitée à une fraction négligeable du temps d'exécution et qu'elle ne change pas assez le comportement de l'exécution du programme observé pour empêcher l'utilisation des traces d'exécution par les outils d'évaluation de performances [Hofmann (1996)].

L'intrusion du traceur ne peut pas être négligée dans le cas du traçage logiciel. Plusieurs propositions ont été faites pour modéliser et corriger l'effet de l'intrusion des traceurs logiciels [Yan (1994); Malony et A. Reed (1992); Maillet (1996, 1995)]. L'objectif de telles propositions est de produire, à partir de la trace d'exécution correspondant au comportement perturbé de l'application, une trace d'exécution qui soit le plus proche possible de la trace d'exécution idéale qui serait obtenue par une collecte non-intrusive.

Quantité des informations enregistrées

La quantité de données tracées dépend du nombre d'événements tracés : elle peut être limitée quand le traçage est restreint aux événements de communication. Une quantité énorme de données peut être produite si des mesures plus détaillées sont nécessaires. Pour faire face à ce problème, quelques systèmes d'observation, tels que Pablo [Reed (1993b)], ajustent dynamiquement la fréquence d'enregistrement des traces et peuvent remplacer le traçage d'événements par un comptage quand la fréquence d'occurrence des événements tracés devient trop haute. Dans tous les cas, une quantité potentiellement grande de données de trace doit être enregistrée et extraite du système parallèle [Maillet (1996)].

Format du fichier de traces

Le format du fichier de traces décrit les données qui sont enregistrées dans le fichier, dans quel ordre et sous quelle forme (binaire, textuelle). Pour pouvoir lire le fichier de traces et en extraire les informations, il faut en connaître le format. De nombreux formats de trace ont été développés à ce jour. Deux catégories des formats peuvent être distinguées :

- Les formats spécifiques aux environnements de programmation : la raison pour le développement d'un format de traces spécifique est d'enregistrer le minimum d'informations nécessaires pour exprimer les actions du programme dans son environnement. Dans un format spécifique, les informations transportées par ce format possèdent une sémantique spécifique à l'environnement de programmation.

– Les formats de trace développés avec pour objectif la généralité : ils ont la capacité d'exprimer les différents paradigmes des environnements de programmation. Le format des traces ne contient aucune sémantique des informations qu'il transporte. Le format définit uniquement la façon de représenter les données, pas sa signification. Ces formats sont généralement auto-descriptifs : la structure des événements est définie dans les entêtes des fichiers de trace. Le format auto-descriptif qui a eu le plus de succès est SDDF [Aydt (1992)], développé pour l'outil Pablo [Reed (1993b)].

3.1.3 Synthétiser et estimer les indices de performance

L'information collectée pendant l'exécution d'un système doit être présentée au programmeur sous une forme compréhensible. Dans la plupart des cas, l'information telle qu'elle est collectée n'est pas prête à être visualisée et doit subir un certain traitement. Ce traitement vise d'une part à réduire les données et d'autre part à reconstituer la séquence des états du programme observé, à en calculer des indices de performances.

Les traitements des données peuvent donc être décomposés en plusieurs étapes : lecture des traces d'exécution, qui extrait l'information du fichier de traces, simulation qui reproduit les états du programme à partir des événements de la trace, analyse de données, qui produit des indices de performances et fait la réduction des données, filtrage, qui sélectionne les informations à visualiser et visualisation, qui présente ces informations à l'utilisateur.

Lecture des traces

Avant de pouvoir faire quoi que ce soit avec les données collectées lors de l'exécution du programme, il faut accéder à ces données. Dans la plupart des cas, ces données sont mises à disposition dans un fichier (ou plusieurs), dans un ordre qui correspond en général à l'ordre chronologique de génération des données.

Le principal problème dans la lecture de traces est la connaissance du format du fichier de traces. La plupart des outils de visualisation sont capables de lire un ou plusieurs formats de traces. Il faut donc que :

– L'environnement d'exécution utilisé pour le développement du programme parallèle fournisse une trace dans le format accepté par l'outil ; ou
– La trace fournie sera convertie à un format accepté, avant de pouvoir être analysée.

Simulation de l'exécution

La simulation a pour but de reconstituer le comportement des divers objets du programme parallèle dont les actions ont été enregistrées. Les objets considérés dépendent de l'environnement de programmation utilisé par le programme parallèle, et représentent en général les processeurs, les processus et les liens de communication, mais peuvent être aussi des fils d'exécution, des objets de synchronisation ou des objets significatifs pour l'application, définis par le programmeur.

La simulation n'est pas toujours présente dans le processus de visualisation. Elle est présente lorsqu'il est nécessaire de restituer la séquence des états du programme observés à partir de l'enregistrement des événements tracés. Par contre, si les données collectées ne sont pas des événements mais des compteurs ou des valeurs d'indices de performances, ou bien si on ne veut pas visualiser les changements d'états, la simulation n'a alors pas de raison d'être.

Analyse de données

L'analyse de données a pour but de produire des indices de performances qui synthétisent les données brutes produites par la simulation ou enregistrées pendant l'exécution du programme. De nombreux indices de performances peuvent être obtenus en analysant ces données brutes, tels que le volume de données échangées durant l'exécution du programme, le degré d'utilisation des processeurs, l'évolution des files de messages, divers comptages, des calculs statistiques, etc. Cette analyse des données est en général simple, la difficulté est dans le choix des indices à produire [Maillet (1996)], pour qu'ils soient représentatifs de l'exécution observée et susceptibles d'aider le programmeur à mieux comprendre son programme et à en améliorer les performances.

Filtrage des données

Le filtrage sert à sélectionner les données à visualiser. Au contraire de l'analyse, qui produit des nouvelles données à partir des données existantes, le filtrage ne fait que masquer certaines données, empêchant leur visualisation. L'idée est d'éliminer les informations qui ne sont pas intéressantes à un instant donné et de laisser celles qui le sont. Ce choix est en général laissé au programmeur au moyen de filtres permettant d'effectuer ce choix selon différents critères.

Comme la production des données, le filtrage peut se faire à plusieurs moments : pendant la collecte des données, pendant la lecture des traces, pendant la simulation, pendant l'analyse ou bien juste avant l'affichage. Plus on filtre

tôt, moins on aura de données à traiter et à visualiser, et plus rapide sera leur traitement. Par contre, le choix de ce qu'il faut filtrer est plus difficile, vu qu'on ne sait pas toujours d'avance ce que l'utilisateur va vouloir visualiser. Les conséquences d'un filtrage au début du processus de visualisation peuvent être lourdes : si on découvre que le filtrage réalisé pendant la collecte des traces a été trop " filtrant " et qu'on a besoin de données éliminées, il faut réexécuter le programme avec une configuration des filtres plus "perméable".

Le filtrage tardif, au contraire, permet un choix plus sélectif. Le grand avantage du filtrage tardif (couplé à la production de données abstraites par l'analyse de données) est de permettre la construction d'un outil qui, à partir d'une vision simple, facilement compréhensible, des données abstraites de l'exécution, permet à son utilisateur d'explorer des données de plus en plus détaillées, là où il le juge nécessaire. De cette façon, l'utilisateur n'est pas forcé d'examiner des données qui ne sont pas nécessaires ou voulues, ce qui est une grande source d'insatisfaction avec les outils de visualisation [Reed (1994)]. Un désavantage majeur d'un filtrage tardif réside au niveau du temps de traitement et l'espace de stockage des données à filtrer.

Il faut donc trouver un bon équilibre entre ne pas avoir trop de données à traiter et en avoir suffisamment pour une visualisation de qualité.

3.1.4 Visualisation des traces et présentation des indices de performance aux développeurs

La présentation des indices de performances constitue l'étape finale du processus de visualisation. La qualité des visualisations offertes par un environnement et la facilité d'interaction avec ces visualisations sont fondamentales pour aider le programmeur à comprendre, à trouver et à corriger les problèmes de son programme.

Reed propose une classification des présentations de performances fondée sur trois axes [Reed et Ribler (1998)] : *dynamique*, *cardinalité* et *ordinalité*.

Dynamique : une visualisation statique montre une vision unique, inchangeable de la performance d'une application. Une visualisation dynamique, par contre, met sa représentation à jour pour refléter un changement dans les données. Un exemple de visualisation dynamique est un graphe de l'utilisation moyenne des processeurs pendant un intervalle donné de l'exécution, qui est réactualisé quand l'utilisateur change l'intervalle de temps sélectionné.

Cardinalité : les présentations se distinguent selon le nombre de variables présentées simultanément dans la visualisation (présentations des valeurs simples ou des relations entre plusieurs variables).

Ordinalité : 2 types de visualisations possibles:

- Des visualisations de données numériques, ordonnées, tels que le nombre de messages échangés ou le taux d'utilisation d'un processeur;
- Des visualisations de données non-numériques et sans ordre total, qui représentent des catégories, comme les états d'un processus (actif, bloqué, etc.) ou l'identification des procédures exécutées [Ribler *et al.* (1995); Couch (1993)].

Un cas très commun de visualisation à plusieurs variables se présente quand une de ces variables est le temps. Ces visualisations montrent l'évolution des valeurs d'autres variables pendant l'exécution du programme observé. Ces visualisations sont appelées temporelles ou historiques. Un exemple de visualisation temporelle est le diagramme espace-temps, présent dans beaucoup d'outils de visualisation. Dans un tel diagramme, en général en deux dimensions, un axe représente le temps et l'autre représente l'"espace", l'évolution des valeurs des variables visualisées dans le temps.

Un grand nombre d'outils a déjà été développé pour visualiser l'exécution des programmes parallèles. Cette prolifération est due en partie à la complexité de la tâche et à la variété des domaines et des perspectives à considérer, mais aussi au manque de possibilités d'extension des outils. Pablo et Svpablo sont des exemples de ces outils. Des présentations plus exhaustives peuvent être trouvées dans la bibliographie [NHSE (1999); Browne *et al.* (2001); Kraemer et Stasko (1993); Hayes *et al.* (1996)].

3.2 Visualisation des SMA

La différence majeure entre les systèmes parallèles et les SMA, en termes de visualisation de traces, est liée aux informations à visualiser. Concernant les systèmes parallèles, les tâches réalisées par les processus durant l'exécution d'un programme parallèle sont précises et prévues. Ces tâches sont décrites pour chaque processus d'une façon séquentielle par un ensemble d'instructions. La visualisation, pour ces systèmes, se fait au niveau des processus communiquants. Le rôle de la visualisation pour ces systèmes est de présenter l'état d'un processus (processus bloqués, processus actifs, etc.), les informations sur la communication et l'accès à la mémoire partagée. Dans le cas d'un SMA, une visualisation avec ce type d'information n'est pas très significative. La connaissance du blocage et de l'activation de l'agent dans un système multi-agent n'a pas assez de sens. Ce qui est le plus important à visualiser dans ces systèmes c'est le *comportement non prévu* des agents. Par le comportement non prévu, nous désignons les tâches réalisées par les agents durant la résolution du pro-

blème. Donc, la nature des objets à visualiser et les informations associées à ces objets à étudier sont différentes.

Dans ce qui suit, nous suivons le même processus de visualisation, présenté pour les systèmes parallèles, pour définir la visualisation des SMA. Nous commençons par définir une abstraction des systèmes multi-agents. Ensuite, nous traitons des autres parties du processus de visualisation :

- Méthode et modèle d'observation
- Technique d'instrumentation
- Collecte des données
- Visualisation de l'exécution

3.2.1 Abstraction des SMA

Pour spécifier les objets à visualisés, il est nécessaire de proposer une abstraction des systèmes multi-agents. Dans cette partie nous présentons cette abstraction en structurant selon l'architecture "VOYELLES" des SMA [Demazeau (1995)].

Cette architecture modélise un SMA selon quatre dimensions fondamentales: *Agent*(A), *Environnement*(E), *Interaction*(I) et *Organisation*(O). Dans ce qui suit, nous discutons l'abstraction d'un SMA selon ces 4 dimensions.

Agents(A):

Les définitions classiques de l'*agent* [Demazeau (1995); Demazeau et Costa (1996); Ferber (1995); Wooldridge et Jennings (1995); Wooldridge et Ciancarini (2000)] le considèrent comme une entité logicielle ou physique à laquelle est *attribuée une certaine mission* qu'elle est capable d'accomplir de manière autonome et en coopération avec d'autres agents. *Le comportement de l'agent* est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

À partir de cette définition, on peut distinguer différents types des agents, constituant un SMA, selon la mission attribuée (selon le rôle de l'agent). De plus, le comportement d'un agent est traité comme dépendant de ses observations, de ses connaissances et de ses interactions avec les autres entités. Les différents types de comportements possibles, que peut subir un agent durant l'exécution du système, désignent les états comportementaux de l'agent. Un état comportemental d'un agent correspond à un ensemble des actions que peut réaliser l'agent et qui ont un sens particulier pour réaliser le but local de l'agent.

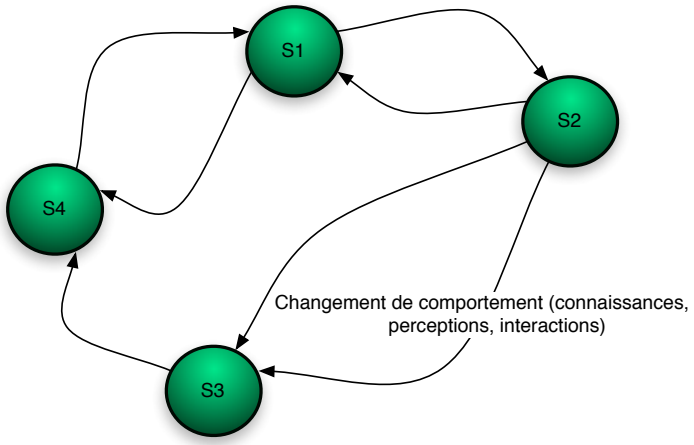

FIG. 3.2 – *Les états comportementaux de l'agent*

Nous modélisons le comportement d'un agent par un graphe d'états où les états représentent les états comportementaux de l'agent (figure 3.3). Les transitions représentent les changements de comportements d'un agent.

Les seuls moyens d'échanger les informations entre un agent et son environnement logique (les autres agents et l'environnement physique) sont la perception et l'interaction. Le changement des connaissances est dépendant soit d'un échange d'informations (perception ou interaction avec les autres entités) soit à des événements de calcul interne sur ses propres connaissances. Le changement de comportement est représenté par:

$$ChangementDeComportement = f(connaissances,perceptions,interactions)$$

Notons $Type_{agents}$ l'ensemble des types des agents alors:

$$Type_{agents} = \{type_1,type_2,...,type_n\}$$

Nous considérons que l'ensemble des types des agents est un ensemble fini. Il n'existe pas un agent sans type défini. De plus, tous les types possibles d'agents sont des éléments dans l'ensemble $Type_{agents}$.

Soit S^{type_i} l'ensemble des états comportementaux de l'agent de $type_i$. Alors l'exécution d'un agent A de $type_i$ est représentée par:

$$h_A = s_1,s_2,...,s_i,...,s_n$$

Avec:

$s_i \in S^{type_i}$

En plus des états, les événements visualisables, au niveau de l'agent, sont les événements de changement d'état. Alors que ces événements sont des événements de calcul, des perceptions ou encore des interactions. Notons pour la suite E^{type_i} l'ensemble des événements d'observation pour le $type_i$.

Environnement(E):

L'*environnement* est l'espace où sont situés les agents. L'environnement est également l'espace contenant les entités passives manipulées par les agents. Le modèle de l'environnement utilisé dans ce travail est illustré par la figure 3.3. Deux états sont visualisables pour l'environnement: l'état d'attente actif des requêtes des agents, et l'état de réponse aux requêtes. Comme au niveau des agents, le comportement au niveau de l'environnement est représenté par une séquence d'états. L'ensemble des états de l'environnement est:

$$S^{Environnement} = \{s_{reponse}, s_{attente}\}$$

Avec:

- $s_{reponse}$: l'état dans lequel l'environnement répond aux requêtes des agents;
- $s_{attente}$: l'état dans lequel l'environnement attend les requêtes des agents.

Les événements pouvant changer l'état comportemental de l'environnement sont les requêtes des agents et les réponses à ces requêtes.

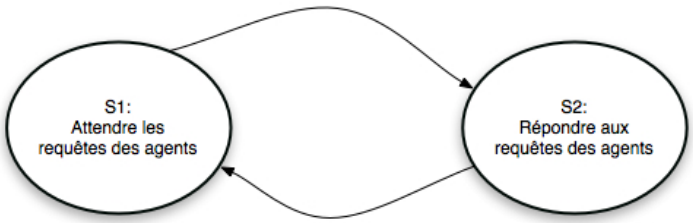

FIG. 3.3 – *Les états comportementaux de l'environnement*

Interaction(I):

Un SMA se distingue d'une collection d'agents indépendants par le fait que les agents *interagissent* pour accomplir une tâche ou atteindre conjointement un but particulier. Les agents peuvent interagir directement entre eux, par l'intermédiaire d'un autre agent ou en agissant sur leur environnement. Dans ce travail, l'interaction est restreinte aux liens de communication pour simplifier l'étude. Les liens de communications entre les différents composants d'un SMA sont modélisés par des événements qui affectent l'état comportemental de l'agent. Les événements de communication sont typés pour refléter

le fait que les communications dans un SMA sont typées. Notons pour la suite $E^{type_i}_{communication}$ l'ensemble des événements de communication pour le $type_i$.

Le cas des communications dans un SMA sera étudié dans le chapitre 4. Cette étude est couplée à la visualisation dans les chapitres suivants.

Organisation(O):

L'*organisation* est l'ensemble des relations entre composants (agents, environnement) du système. L'organisation sous-tend la façon dont sont réparties les tâches, les informations, les ressources et la coordination d'actions en fonction des capacités et des rôles des agents. Dans ce travail, nous utilisons une certaine organisation hiérarchique des agents pour garder la généralité de notre approche (figure 3.4). Le cas des organisations dynamiques dans lesquelles les agents peuvent changer de rôle durant l'exécution n'est pas traité. La représentation d'une telle organisation dynamique pourrait se faire en caractérisant l'état d'un agent par un couple composé de son état comportemental et de son type. Le changement de l'état d'un l'agent se ferait alors soit au niveau de l'état comportemental, soit au niveau du type de mission associé à l'agent.

Dans la figure 3.4 nous avons:

- Deux composants d'organisation représentant le SMA et les entités le constituant;
- Un composant d'interaction placé au niveau des entités (l'interaction se déroule entre deux ou plusieurs entités);
- Des composants pour les types des agents et l'environnement;
- Les agents effectifs représentant l'ensemble des instances des agents de chaque type.

Comportement d'un SMA:

La modélisation du comportement d'un SMA, donc, est représentée par :

$$H_{SMA} = S_t,S_{t(1)},.....,S_{t(i)},S_{t(i+1)},.....,S_{t(n)}$$

Avec:

- $S_{t(i)} = (s_1,s_2,...,s_k,...,s_n,s_{environnement})$ l'état du SMA dans l'intervalle de temps t_i
- s_k l'état du kième agent l'intervalle de temps t_i
- $s_{environnement}$ l'état de l'environnement à l'instant i

Un changement de l'état du SMA est le résultat d'un changement de l'état d'un agent ou de l'environnement. L'abstraction du comportement d'un SMA est illustrée dans la figure 3.5.

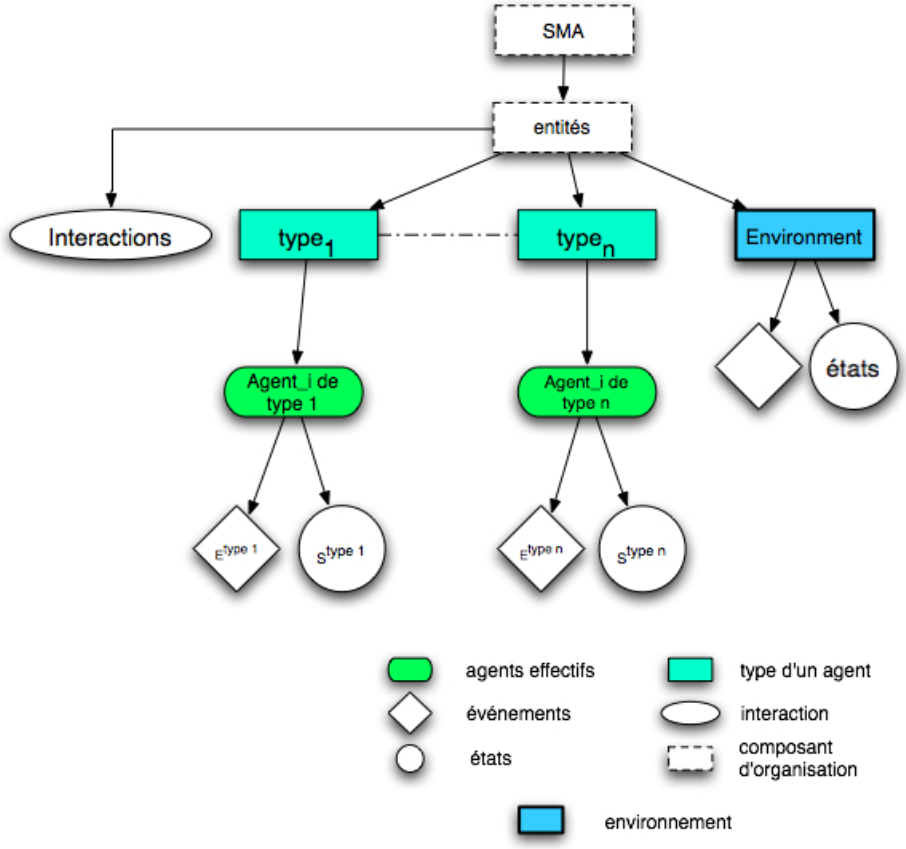

FIG. 3.4 – *Hiérarchie d'organisation*

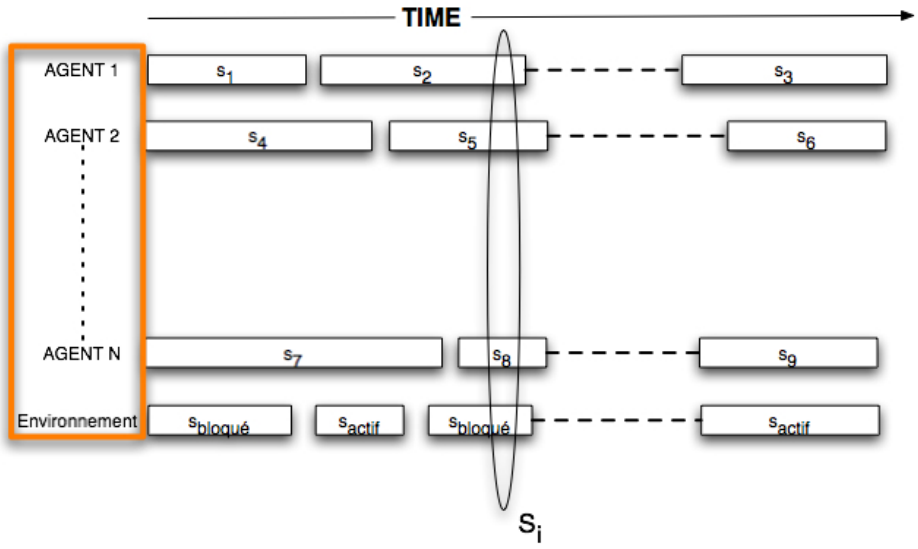

FIG. 3.5 – *Modélisation du comportement d'un SMA*

3.2.2 Processus de visualisation des SMA

Dans cette partie nous traitons les phases de processus de visualisation d'un SMA.

Observation et instrumentation

Concernant la méthode d'observation, la technique qui permet d'obtenir le plus d'informations (en détail et en quantité) sur le comportement d'un SMA est la visualisation fondée sur les traces. Cette méthode est la méthode utilisée dans ce travail.

Les inconvénients de cette méthode se situent au niveau de volume des données engendrées et de la perturbation du programme observé. Malgré ces inconvénients et grâce à son caractère général, le traçage des événements est aujourd'hui à la base de la majorité des outils de visualisation. Les informations fournies par cette technique et la liberté pour le choix des événements intéressants à étudier font donc de cette technique la méthode la plus adaptée pour l'observation des SMA. De plus, la multiplicité des outils de visualisation se fondant sur cette technique nous incite à porter notre choix sur le traçage.

Pour les méthodes d'instrumentation, la première méthode présentée dans le chapitre 1 est **l'instrumentation directe du code source du programme**. Cette méthode à l'avantage d'être facile à implanter et de laisser à l'utilisateur le contrôle complet sur ce qui est instrumenté. Elle peut accéder facilement aux informations d'un SMA et en déduire l'ensemble des états comportementaux des agents. Elle a le désavantage de nécessiter une reinitialisation complète du système.

L'instrumentation pendant la compilation: cette méthode a l'avantage de donner accès aux informations construites par le compilateur. Cette approche facilite l'intégration d'informations sémantiques [Larus (1993)] (changement d'états, variables etc.) dans les traces. Elle a le désavantage de nécessiter l'accès au code source du compilateur. Cette technique est difficile à l'appliquer pour les SMA vue la diversité des plates-formes d'exécution, fondées sur des langages de programmation différents.

L'instrumentation de la bibliothèque de communication: cette méthode ne requiert aucune modification, ni même de recompilation du programme, ce que l'utilisateur appréciera d'autant plus que son programme est constitué de nombreux modules de code source dont la compilation peut durer longtemps. Il suffit, en général, de le relancer en activant le code de traçage compilé dans la bibliothèque. Toutefois, les événements spécifiques à l'application ne sont pas visibles au niveau de la bibliothèque. Ainsi, il est impossible

de détecter les changements d'états et d'évaluer une métrique liée à l'exécution du système à partir de la trace.

L'instrumentation directe de l'image exécutable a également de nombreux avantages. Tout comme l'instrumentation de la bibliothèque de communication, elle est indépendante du langage de programmation et ne requiert pas la recompilation du programme. Cette technique permet d'effectuer l'instrumentation pendant l'exécution du programme. L'utilisateur peut donc intervenir "dynamiquement" sur la localisation de l'instrumentation. L'instrumentation de l'image exécutable a les désavantages de ne pas être facilement portable entre systèmes d'exploitation différents et de ne pas être facile à réaliser. En plus, comme l'instrumentation de la bibliothèque de communication, elle n'a pas accès aux détections des informations concernant les changements d'états dans un système.

Au bilan, l'approche de **l'instrumentation directe du code source du programme** est la seule approche ayant accès aux informations de changement de l'état comportemental de l'agent et qui a en même temps la possibilité d'être appliquée pour les SMA. De plus, pour garder les avantages liés à la détection des communications des agents sans aucune modification du système proposés par **l'instrumentation de la bibliothèque de communication**, nous proposons pour les SMA une approche hybride fondée sur les deux types d'instrumentation : l'instrumentation directe du code et l'instrumentation de la bibliothèque de communication. L'implémentation de cette approche est illustrée dans le chapitre 5.

Collecte et analyse des données

Pour les deux étapes *collecte des données* et *analyse des données*, ces deux étapes traitent les problèmes de qualité et de quantité des événements collectés et le problème de l'horloge. Ces étapes sont considérées plutôt comme des étapes techniques présentées en détails dans le chapitre 5.

Visualisation des traces d'exécution

Une visualisation qu'on cherche pour un système multi-agent doit refléter le modèle proposé pour les SMA et représenter dans la figure 3.5. Une telle visualisation est donc **dynamique** et met sa représentation à jour pour afficher les changements des états au cours du temps. Dans la dimension **ordinalité**, cette visualisation doit avoir des variables non-ordonnées représentant les états des agents en fonction du temps.

La construction d'un outil de visualisation pour les SMA est une tâche complexe qui présente beaucoup de difficultés. De plus, des questions peuvent

être posées sur l'utilité d'un tel outil vue la possibilité d'adapter des outils présents dans des domaines similaires.

Par exemple, Athapascan [Plateau *et al.* (1997)] est un modèle destiné à l'exécution de programmes parallèles portables à destination des systèmes à grand nombre de processeurs. Le principe de "parallélisation" d'un calcul selon Athapascan est son découpage itératif en sous-calculs selon différentes règles. Les sous-parties de calcul sont confiées à un grand nombre de processus qui s'exécutent et se communiquent dans le but de réaliser en haute performance une tâche qu'on peut le réaliser en calcul séquentiel. Le degré de découpage du programme parallèle doit être adapté à celui des machines cibles potentielles étant donné que celles-ci diffèrent soit par leur grain soit par leur nombre de processeurs.

Dans le modèle Athapascan nous pouvons sortir l'existence des mêmes dimensions (processus, communication, machines, découpage) utilisées pour une architecture multi-agents (agents, interaction, environnement, organisation). La différence entre les 2 modèles se situe au niveau de la réalisation des tâches et pas au niveau de l'architecture. Vue la similarité de l'architecture des systèmes parallèles avec l'architecture de notre domaine et la difficulté de création d'un outil de visualisation dédié aux SMA dans le cadre de notre travail, nous nous sommes orientés vers l'utilisation et l'extension d'outils de visualisation développés et utilisés dans les systèmes distribués/parallèles.

Le choix d'un tel outil est lié à ses caractéristiques d'extensibilité et de modularité. Un outil adapté aux SMA doit avoir:

- La possibilité de représenter l'exécution des systèmes durant une longue période;
- La possibilité de traiter la visualisation de l'exécution pour un grand nombre d'entités constituant le système;
- La possibilité d'ajouter des nouvelles fonctionnalités et d'étendre les fonctionnalités existantes sans nécessité de changer l'outil en totalité;
- La possibilité de visualiser des entités qui se créent et se détruisent durant l'exécution du système.

Une présentation détaillée de l'outil de l'évaluation choisi pour les SMA est donnée dans le chapitre 5.

3.3 Visualisation de l'exécution dans un SMA : résultats et discussion

Nous présentons ici brièvement un exemple de visualisation de l'exécution d'une application multi-agents, le système proies/prédateurs, développée sous la plate-forme Core-DMS [Fiorino (2003)] de l'équipe MAGMA. Des expérimentations plus détaillés sur des applications qui présentent plus de complexité figurent dans les chapitres 7 et 8.

Dans le système proies/prédateurs, les prédateurs (predators) arrêtent les proies (preys) situées à l'intérieur de leurs champs de perception. Un champ de perception d'un prédateur est constitué de l'espace de l'environnement qui l'entoure et dans lequel il peut détecter les proies. Si aucune proie n'est détectée, le prédateur explore ses alentours en se déplaçant aléatoirement. Le lecteur intéressé pourra trouver une description plus détaillé de cette application par exemple dans [Meurisse (2004)].

Dans cette application, les agents sont les proies et les prédateurs. Différents types de communications à l'intérieur du système peuvent être distingués. Une première visualisation de l'exécution du système est présentée dans la figure 3.6.

Sur cette figure (3.6) nous distinguons 4 zones:

- Zone 1: la zone de l'échelle de temps;
- zone 2: l'organisation statique des agents dans le SMA;
- zone 3: les entités effectives (les instances des types des agents) dans le système;
- zone 4: les états possibles et les événements associés pour chaque agent.

La figure 3.6 montre donc une vision globale du comportement de l'application proies/prédateurs. De cette figure ressort la masse des informations présentes dans la visualisation, sur le comportement des agents et les interactions entre ces agents. Nous pouvons regarder l'évolution des états des agents durant le temps de l'exécution. Nous pouvons naviguer dans le temps de l'exécution et passer en détail pour voir qu'est ce qui se passe au niveau de chaque agent. Un exemple est présentée dans la figure 3.7.

La figure 3.7 présente le comportement de l'agent proie 1 du serveur 2 entre les instants 60 et 400 durant l'exécution du système. Les différentes couleurs sont utilisées pour représenter les états, les marques représentent les événements et les flèches représentent les liens de communication entre les agents. Dans cette figure, nous pouvons voir que la proie 0 du serveur 2 sort de l'état d'initialisation (bleu) à l'instant 100. Elle entre dans l'état de déplacement

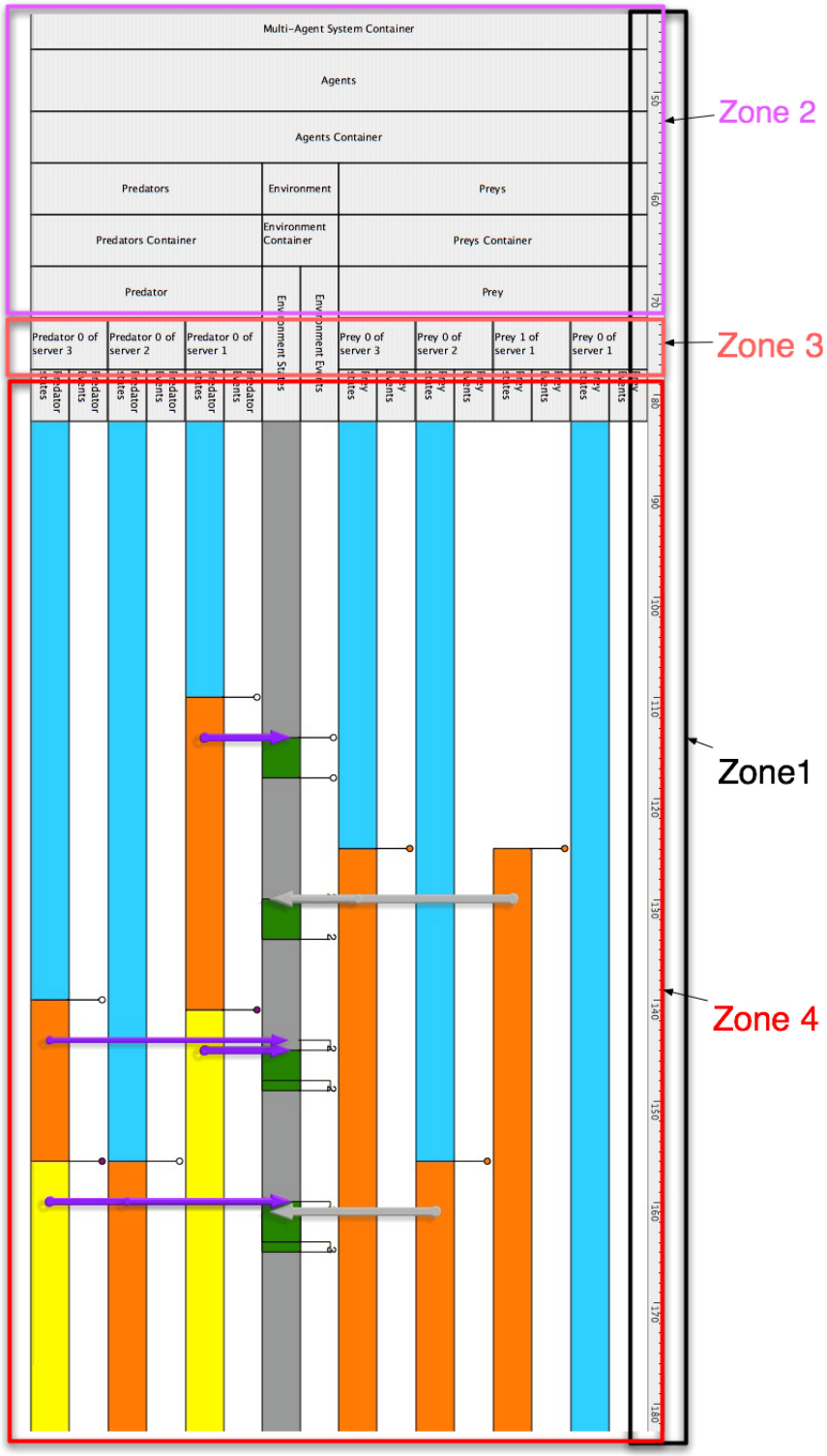

FIG. 3.6 – *Visualisation de l'exécution du système proie/prédateur*

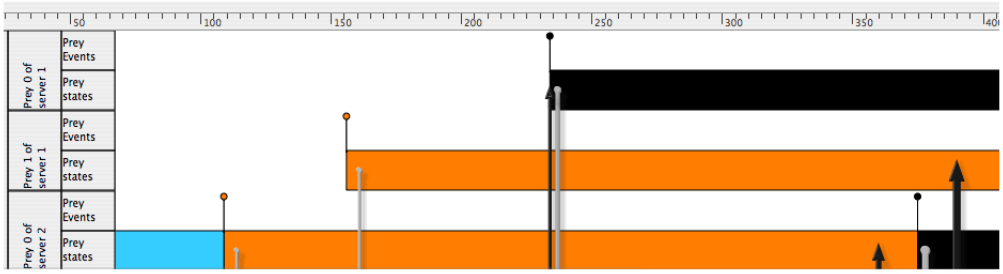

FIG. 3.7 – *Comportement de proie 0 sur serveur 2*

"Move" (rouge) entre les instants 100 et 375. Pendant cette période, la proie communique par envoi et réception de messages.

Les informations présentent dans la figures 3.6 et les analyses faites pour la figure 3.7 montrent combien une visualisation peut être utile pour *la compréhension et l'analyse du comportement des agents à l'intérieur d'un SMA.*

Les figures 3.8, 3.9 et 3.10 présentent le pourcentage de temps passé dans les états par les agents prédateur 0 du serveur 1, prédateur 0 du serveur 3, proie 0 du serveur 3.

Dans la figure 3.8, le prédateur (prédateur 0 du serveur 1) passe 77.6% du temps en essayant de détecter une proie et 22.4% en se déplaçant. Tandis que pour le prédateur de la figure 3.9, la majorité du temps est passé en se déplaçant (78.4%). Dans cette application, nous pouvons remarquer que la distribution des tâches aux agents n'est pas uniforme. Pour cette application particulière, cela n'a pas de sens mais dans d'autres applications l'étude de l'uniformité de la distribution des tâches est très importante (application de transport, etc.).

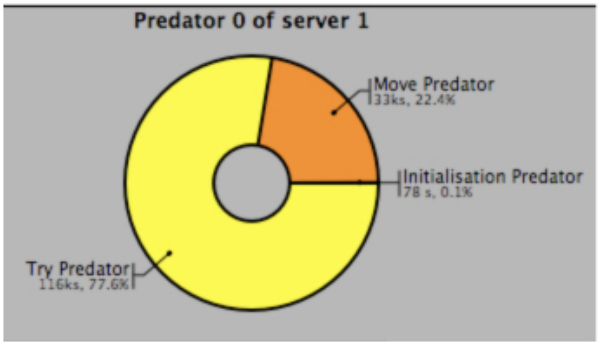

FIG. 3.8 – *Distribution des tâches pour le prédateur 0 du serveur 1*

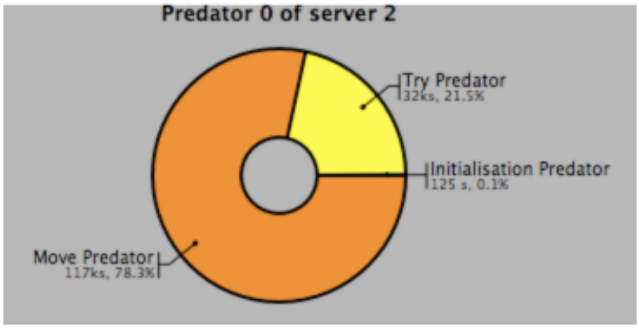

FIG. 3.9 – *Distribution des tâches pour le prédateur 0 du serveur 2*

Dans la figure 3.10, la proie 0 du serveur 3 se déplace seulement 8.7% du temps avant d'être arrêtée. Cela est dû à la distribution initiale qui a placé cette proie dans le champ de perception des prédateurs. De même pour la proie 0 du serveur 1, cette proie est arrêtée au lancement de l'application (figure 3.7).

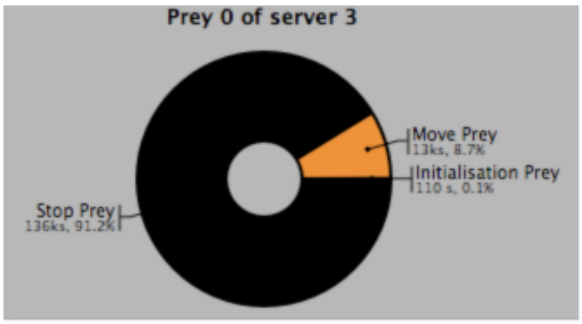

FIG. 3.10 – *Distribution des tâches pour la proie 0 du serveur 3*

Les figures 3.8, 3.9 et 3.10 montrent le rôle de cette technique dans la présentation de *la distribution des tâches sur les différentes entités* et l'utilisation de cette distribution dans la *détection des problèmes* (non-uniformité de distribution des tâches, blocage des agents,etc.) dans les paramètres initiaux du système (emplacement initial des agents par exemple dans le cas des proies/-prédateurs) ou dans la conception de l'application.

La figure 3.11 présente le pourcentage de temps passé par l'environnement en attendant les requêtes des agents et en répondant à ces requêtes. Cette figure représente donc la *consommation des ressources* de l'environnement par les agents du système.

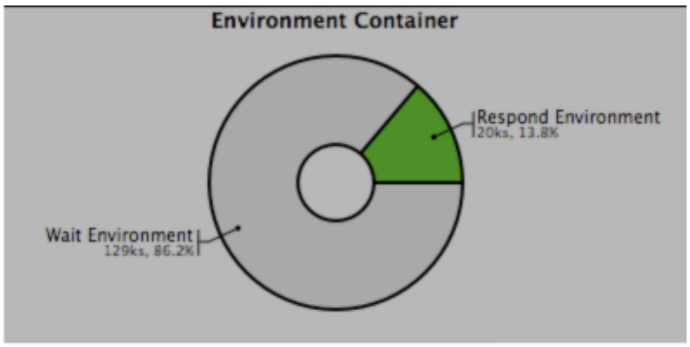

FIG. 3.11 – *Les états de l'environnement*

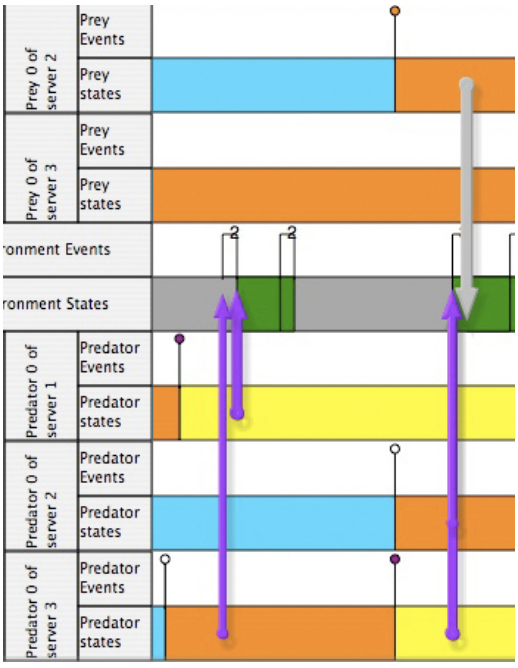

FIG. 3.12 – *Les communications entre les agents*

De plus, la technique de visualisation joue un rôle important dans l'étude des interactions et la vérification de la conformité des protocoles durant l'exécution d'un SMA. Une visualisation des communications dans l'application proie/prédateurs est présentée dans la figure 3.12. Les différents types de communication sont présentés par des différentes couleurs. Le cas de la communication est étudié dans le chapitre 4, puis validé et expérimenté dans les chapitres 6 et 8.

Finalement, la visualisation de l'exécution est fondamentale pour la comparaison des différents SMA résolvant le même problème et implémentés avec des architectures différents (chapitre 7). Des études plus approfondies et qui montrent le rôle de cette technique sont présentées dans les chapitres 7 et 8.

Pour conclure ce chapitre, La modélisation des SMA selon le modèle événement/état et par suite la visualisation de l'exécution des SMA pouvant être modélisés ainsi représente une approche novatrice d'évaluation des SMA qui permet:

- L'analyse du comportement pour un SMA;
- La détection des erreurs du comportement et le débogage;
- La présentation des indices de performances;
- L'utilisation des ressources par les différentes entités;
- La distribution des tâches sur les entités;
- La comparaison entre différents systèmes résolvant le même problème.

Chapitre 4

Étude de la communication dans l'exécution d'un SMA

Dans le chapitre précédent, nous avons traité du problème de la visualisation de l'exécution, afin d'analyser et de comprendre le comportement interne au niveau des entités constituant un SMA. Cette étude nécessite d'être complétée par un travail portant sur la compréhension de l'ensemble des entités et également des liens entre elles. Un tel travail est important pour:

- L'analyse du comportement au niveau de l'ensemble des entités;
- Le débogage pour la performance au niveau de l'ensemble des entités;
- Avoir des indices de performances au niveau de l'ensemble des entités et des liens entre elles;
- La présentation des informations concernant les liens entre les entités.

Dans ce chapitre, nous montrons que l'étude et la modélisation des caractéristiques spécifiques aux SMA sont intéressantes pour élaborer des indices de performances au niveau de l'ensemble des entités et des liens entre elles. Cette étude est une tâche complémentaire à la technique de visualisation pour avoir une vision globale et complète sur le comportement des SMA.

L'étude et l'évaluation de la plupart des caractéristiques, représentant l'ensemble des entités, se fondent sur l'interaction (chapitre 2 et section 4.1). Dans ce chapitre, nous considérons, comme exemple des caractéristiques à modéliser, l'interaction simple fondée sur la communication, et les conversations entre agents. Nous nous concentrons donc sur la dimension *interaction* de l'architecture VOYELLES [Demazeau (1995)]. Cette dimension représente la suite de la dimension *agent* traitée dans le chapitre précédent.

Ce chapitre est composé de deux parties. Dans une première partie, une communication est traitée comme une unité isolée de tout ce qui se passe avant son émission et après son traitement.

Dans la deuxième partie, nous considérons les conversations, l'ensemble des communications conformes à un protocole, pour étudier les séquences des communications plutôt que des communications discrètes.

4.1 La communication comme problématique d'évaluation

La plupart des travaux d'évaluation dans le domaine des SMA traitent de la communication comme critère de comparaison entre les différents systèmes [Mylopoulos *et al.* (2002a); Cernuzzi et Rossi (2002); Davidsson et Johansson (2002); Jurasovic *et al.* (2006a); Bincheng *et al.* (2004), etc.]. La liste des travaux montre l'importance de la communication comme critère d'évaluation. Toutefois, la communication est considérée comme critère de base. Des caractéristiques plus complexes (coopération, coordination, ...) sont évaluées sur la base de sa mesure (en d'autres termes une étude au niveau de coopération entre les agents nécessite une étude de base sur les communications).

Trois problèmes peuvent être soulevés dans les évaluations des SMA fondées sur la communication:

- Le problème de la subjectivité, en associant une valeur comme résultat de la fonction de quantification de la communication par l'évaluateur (valeurs associées aux critères d'évaluation par l'évaluateur);
- Le problème de mesure de la quantité des communications plutôt que mesurer le poids de l'information portée par les unités de communication (mesurer le nombre de messages plutôt que leurs importance);
- La considération de la communication comme un critère lié au domaine d'application et pas comme un composant fondamental des SMA (évaluation des applications spécifiques sans traiter du cas général du SMA).

En conséquence de ces problèmes, l'évaluation des SMA au niveau de leurs communications souffre des lacunes suivantes:

- La subjectivité empêche les évaluations de refléter la réalité de l'application;
- La mesure de la quantité de communication plutôt que leurs poids ne permet pas de comparer différents systèmes entre eux. La quantité de communication varie d'un système à un autre selon les approches de conception utilisées;
- Les approches d'évaluation de la communication sont dépendantes du domaine d'application ce qui rend leurs utilisations difficile ailleurs.

Du coup, nous distinguons certaines situations liées à l'évaluation de la communication, qui sont à la base des problèmes d'évaluations cités ci-dessus. Ces situations sont distinguées dans le but de proposer une approche indépendante du domaine d'application en essayant de minimiser l'effet de subjectivité dans l'évaluation. La première situation traite du problème de variation de l'importance de message d'un système à un autre. La deuxième situation traite du problème de biais possible dans le mesure de l'interaction. Les situations de la communication que nous proposons de distinguer sont:

- L'effet d'une unité de communication dans un système est équivalent à "n" unités de communication dans un autre système. (Situation 1)
- Les unités de communication reçues et s'avérant non pertinentes pour l'agent. (Situation 2)

L'évaluation au niveau des communications ne peut ainsi se faire sans traiter ces situations du point de vue de l'utilisation et de la quantité d'informations portées par les unités de communications.

Spécifications

L'évaluation de la communication nécessite des spécifications du niveau d'abstraction et l'aspect selon lesquels doit être menée.

La communication peut être évaluée selon 2 niveaux d'abstractions:

- *"Micro level"*: l'étude de la communication se fait au niveau des agents sans prendre compte ce qui se passe au niveau global (niveau SMA).
- *"Macro level"*: dans ce cas, l'étude de la communication se fait sur la globalité du système. L'étude est réalisée sur l'ensemble des agents et l'environnement.

Sachant le but de notre recherche d'étudier les liens entre les entités à l'intérieur d'un SMA, nous nous intéressons à l'évaluation au micro niveau. La communication doit être étudiée principalement au niveau des agents pour compléter les travaux réalisés dans le chapitre 3. Il est important de spécifier en plus les aspects évoqués dans notre étude. Il existe différentes aspects selon lesquels la communication dans un système multi-agent peut être évaluée :

- L'*aspect structurel* concernant la topologie et la structure du réseau de communication qui relie les agents et les propriétés qui en découlent;
- L'*aspect statistique* concernant l'étude et la quantification des grandeurs usuelles telles que le nombre de messages échangés et leurs tailles;
- L'*aspect syntaxique* concernant la typologie des messages et la complexité de leurs contenus.

Notre étude tente à couvrir ces trois aspects. Toutefois, nous n'accordons pas d'importance à l'aspect structurel étant la nature de la modélisation de SMA que nous adoptons et le but de proposer une évaluation de la quantité d'informations circulant entre les agents. L'étude des réseaux de communication concerne le SMA et pas les agents isolés et ne s'accorde pas au choix fait d'évaluer la communication au "micro niveau". Ces aspects sont considérés dans le but d'assurer une évaluation de la communication traitant des poids des communications entre les agents plutôt que de leur nombre.

4.1.1 Modélisation de la communication

Dans cette partie, nous modélisons le processus de la communication dans les SMA pour ensuite formaliser les problèmes de l'évaluation des communications et proposer des solutions.

Comme nous l'avons vu antérieurement, un agent est 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 en plus *d'agir et d'interagir* avec les autres agents [Demazeau et Costa (1996); Ferber (1995)]. Selon [Wooldridge et Ciancarini (2000); Wooldridge et Jennings (1995)],

- Un agent a des *connaissances* sur lesquels il possède un contrôle total;
- Ces connaissances sont notamment inaccessibles aux autres agents;
- L'agent *prend des décisions* en se fondant sur ses propres connaissances sans intervention extérieure (humaine ou d'un autre agent).

Avec cette définition, nous pouvons distinguer 3 aspects portés par un agent : les connaissances, l'interaction et la prise de décision.

La structure d'un agent comporte donc:

- Une mémoire contenant les connaissances de l'agent;
- Les capacités d'agir et d'interagir par le moyen d'une interface de réception des messages et d'une interface d'envoi des messages. Les messages reçus sont stockés dans une pile de réception des messages;
- Une unité de traitement, c'est la partie qui traite de l'aspect décision. Les décisions sont prises en se fondant sur les connaissances de l'agent.

Comme nous l'avons déjà précisé dans ce travail, le seul moyen d'agir et d'interagir, entre un agent et les autres entités qui existent (les autres agents et l'environnement), est par des messages qui sont notés, dans la suite, par "CU" (*communication unit*).

Un agent est un processus comportant trois phases successives : réception d'un CU, traitement et action. L'action de communication par un agent est

traduite par l'envoi d'un ou plusieurs CU aux autres entités. Un agent peut aussi avoir des initiatives. Il peut alors envoyer des CU sans être stimulé.

Comme nous l'avons vu, l'environnement est le medium commun partagé par l'ensemble des agents. Un environnement est :

- Accessible : un agent peut, avec ses propres capacités, déterminer la configuration physique de l'environnement et ainsi procéder à une action;
- Déterministe : la configuration physique future de l'environnement est fixée par sa configuration physique courante et par les actions de l'agent;
- Statique : la configuration physique de l'environnement est stable (ne change pas), sauf à évoluer sous l'effet des actions des agents;
- Discret : à tout moment, le nombre des actions faisables et les configurations physiques de l'environnement sont finis.

Dans ce chapitre, conformément au chapitre précédent, l'environnement est traité comme une entité passive ayant une architecture similaire à celle d'un agent. Elle a le rôle de réagir aux actions des agents sur elle et de répondre à leurs requêtes. La différence entre les 2 entités agent et environnement se situe au niveau de leur fonctionnement et pas au niveau de leurs architectures. Cette architecture garantit les propriétés de l'environnement cités ci-dessus.

Communication et Post-communication

Un SMA est un ensemble d'agents évoluant et possédant un environnement en commun. Nous divisons le processus de déroulement au sein d'un SMA en deux étapes : la communication et la post-communication.

La communication

Une communication est l'envoi d'un CU entre deux entités "Sender" et "Receiver". Du point de vue "Sender", la communication est l'envoi d'un CU parmi l'ensemble des CU qu'il peut l'envoyer. Du point de vue "Receiver", la communication est la réception d'un CU parmi les CU qu'il peut comprendre ou un CU incompréhensible. Nous notons:

- CU^A_{Envoye}: l'ensemble des CU pouvant être envoyés par l'agent A et $cu^A_{Envoye,i}$ le CU envoyé par l'agent A à l'instant i.
- CU^A_{Recu}: l'ensemble des CU pouvant être reçus par l'agent A et $cu^A_{Recu,i}$ le CU reçu par l'agent A à l'instant i.
- cu^A_ϕ désigne un CU incompréhensible par l'agent A.

Alors:

$$Communication : CU^{A}_{Envoye} \rightarrow CU^{B}_{Recu} \cup \{cu^{B}_{\phi}\}$$

La communication est une fonction qui associe à un CU envoyé, "$cu^{A}_{Envoye,i}$" une CU reçu "$cu^{B}_{Recu,j}$" ou un CU incompréhensible "cu^{B}_{ϕ}" et $j \succ i$.

Post-communication

Dans l'étape de post-communication, le traitement d'un CU se fait en 2 étapes : la mémorisation traitant de la partie concernant le changement des connaissances causé par la réception du CU, la décision traitant de la partie concernant les actions déclenchées par rapport au CU reçu par l'agent. Nous utilisons les notations suivantes :

- C^A: l'ensemble des connaissances possibles d'un agent A et c^A_i les connaissances de l'agent A à l'instant i.
- AC^A: l'ensemble des actions pouvant être effectuées par l'agent A et ac^A_i l'action effectuée par l'agent A à l'instant i.
- $ac^A_i = \phi$: désigne qu'il n'y a pas d'actions déclenchées par l'agent A à l'instant i.

Le traitement d'un CU par un agent se fait alors en 2 étapes:

- L'étape de mémorisation représentée par une fonction MEM^A qui associe à un CU reçu et aux connaissances courantes un nouvel état des connaissances.

$$MEM^A : CU^A_{Recu} \times C^A \rightarrow C^A$$

- L'étape de décision réalisée par une fonction DEC^A choisissant une action à effectuer en fonction des connaissances courantes et du CU reçu.

$$DEC^A : CU^A_{Recu} \times C^A \rightarrow AC^A \cup \{ac^A_i = \phi\}$$

Dans les sections suivantes, l'évaluation de la communication est étudiée suivant ce modèle.

Modélisation des situations

Comme mentionné ci-dessus, une évaluation de la communication doit faire face à plusieurs situations :

La première situation est que l'effet d'un message dans un système peut être équivalent à "n" messages dans un autre système. Cette situation est modélisée de la façon suivante :

Pour la réception d'un CU, "cu", par l'agent A (les connaissances actuelles etait c_k^A),

$$DEC^A(cu,c_k^A) = ac_{k+1}^A, MEM^A(cu,c_k^A) = c_{k+1}^A$$

Pour la réception d'un ensemble des CU, $\{cu_i\}_{i=l,..,l+n}$, par l'agent B (les connaissances actuelles etaient c_l^B et $c_k^A \cong c_l^B$):

$$DEC^B(cu_i,c_i^B) = ac_{i+1}^B, MEM^B(cu_i,c_i^B) = c_{i+1}^B$$

Avec: $\bigcup ac_i^B \cong ac^A$ and $c_{l+n+1}^B \cong c_{k+1}^A$

L'opérateur $\cong$ est défini comme une équivalence entre un ensemble des variables ou caractéristiques représentant les connaissances. Les caractéristiques représentant ces connaissances dépendent.

Cette situation est présentée dans la figure 4.1, le même résultat nécessite plus de communications. Cela peut constituer un biais dans l'étude et la comparaison de ces systèmes au niveau de leurs communications. La structure des communications dépend fortement de la méthode de conception de ces systèmes. Dans la figure 4.1, le même travail est réalisé par une unité de communication dans le premier système et par "n" unités de communication dans le second.

La deuxième situation concerne les unités des communications reçues et non utilisées. Cette situation est formalisée de la façon suivante: Pour la réception d'un CU "cu" par l'agent A :

$$DEC^A(cu,c_i^A) = \{ac_i^A = \phi\}, MEM^A(cu,c_i^A) = c_{i+1}^A$$

Avec: $c_{i+1}^A \cong c_i^A$. Ces CU, qui sont considérés dans la mesure de la communication, sont aussi un biais dans l'évaluation de la communication et la comparaison entre les différents SMA.

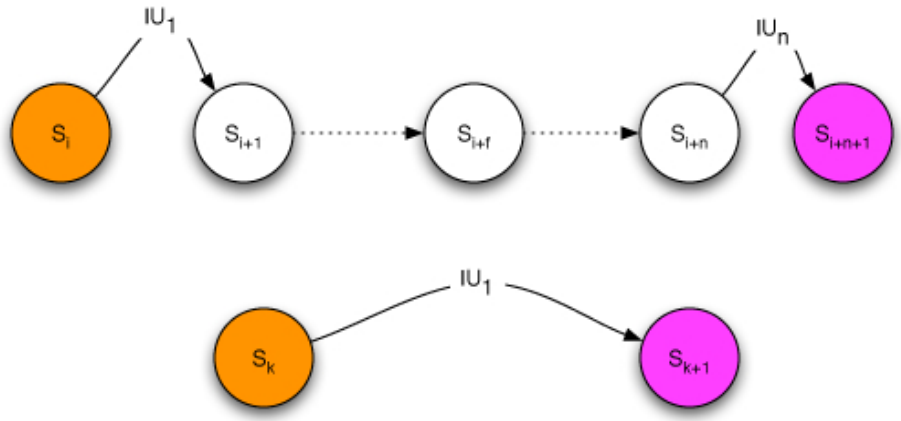

Fig. 4.1 – *La différence entre l'effet des CU dans 2 systèmes ayant des architectures différentes*

4.2 Approches d'évaluation

Approche normale

La première approche traite de la quantité des communications. Cette approche est utilisée dans les évaluations réalisées dans les SMA et présentées dans le chapitre 2. Le but dans cette approche est de comparer l'évolution des communications durant l'exécution, et de mesurer le nombre des messages échangés entre les agents. Cette approche est intéressante pour identifier les liens entre les différents agents et la comparaison entre l'utilisation des différents canaux de communications entre agents.

Le nombre des messages reçus par l'agent A est :

$$\text{Q=NB(M) / } \exists \text{ B et } \exists \text{ Communication: } cu^{B}_{Envoye} \rightarrow cu^{A}_{Recu} \text{ et } cu^{A}_{Recu} = M$$

Cette approche est significative donc pour comparer les implémentations d'une même architecture mais ne peut pas être utilisée pour comparer les différentes architectures. La comparaison entre les différentes implémentations d'une même architecture se fait au niveau quantitatif pur alors qu'on ne dispose pas d'indices représentant la quantité d'information portée par les messages. Cette approche ne propose pas de solution résolvant les problèmes cités ci-dessus.

Approche type

Une première idée, pour proposer une approche traitant la quantité des informations portées par les messages, est de diviser l'ensemble des *CU* reçus

par un agent en des sous-ensembles ayant le même type, et donc le même effet sur l'agent. Ensuite, le poids est associé à un CU selon son type. La distinction entre les types des CU dépend de l'application des SMA à évaluer. L'ensemble des types des CU possibles pour un agent A est noté $TYPE^A$.

Prenons par exemple les primitives proposées par Gaspar pour les types de communication [Gaspar (1991)]. Gaspar a proposé quatre types des messages échangées: present, request, answer, et inform. Ces quatre types sont distingués selon le changement des comportements de l'expéditeur ou du destinataire. "Request" implique un changement de coté de l'expéditeur, dans l'attente de la réponse. "Inform" ne comporte pas de changement à la fois pour l'émetteur et le récepteur. Il pourrait engendrer d'autres "inform", et éventuellement des réponses. "present" inclut un éventuel changement de coté de l'expéditeur et/ou du récepteur. Typiquement, un "present" permet de s'introduire face aux autres agents au début de l'exécution.

Cette solution est "statique" dans la mesure où le poids d'un CU est affecté en se basant sur la conception du SMA. Elle est idéale dans le cas où deux CU du même type ont des effets équivalents sur l'agent indépendamment du temps.

Le problème principal de cette approche est que la quantité des informations portées par une unité de communication est considérée comme statique et fixe selon le type. Dans la majorité des sytèmes multi-agents, un message peut avoir deux effets différents à deux instants différents.

Approche poids des messages

Dans la suite, la notion de *communication pertinente* est définie. Une communication pertinente est une CU qui change les connaissances d'un agent ou déclenche une action. Un cu (reçu par un agent A) est une communication pertinente si:

$$DEC^A(cu,c_i^A) = ac_{i+1}^A, MEM^A(cu,c_i^A) = c_{i+1}^A$$

Avec: $c_{i+1}^A \ncong c_i^A$ or $ac_{i+1}^A \neq \phi$.

Alors, la solution proposée pour la situation 2 (les unités de communication reçues et s'avérant non pertinentes pour l'agent) est de considérer plutôt les communications pertinentes que la quantité totale des communications.

Concernant la première situation, une fonction Φ est définie pour calculer le poids des communications pertinentes (un poids nul est associé aux autres communications). La décision d'un agent est prise selon le CU reçu.

Cette approche consiste à associer le poids au *CU* reçu selon les résultats de son traitement. Selon le modèle présenté, le traitement d'un *CU* est divisé en 2 fonctions: *décision* et *mémorisation*. La solution propose donc de diviser la fonction Φ en deux fonctions Φ_{DEC} et Φ_{MEM} :

La fonction Φ_{MEM} associe une valeur à la variation des connaissances causée par la réception d'un *CU*. Pour qu'on puisse quantifier, un ensemble des caractéristiques mesurables représentant les connaissances doit être défini (figure 4.2). La spécification de ces caractéristiques est liée à l'application. Un poids est associé à la variation de chacun de ces caractéristiques. La fonction Φ_{MEM} est considérée comme la somme de ces poids. Ensuite, pour quantifier la variation des conaissances de l'agent causé par la réception d'un CU "cu" :

$$\Phi^{A}_{MEM} = \sum Weight_i$$

Notons que $Weight_i$ est le poids du caractéristique C^A_i variant après la réception de "cu".

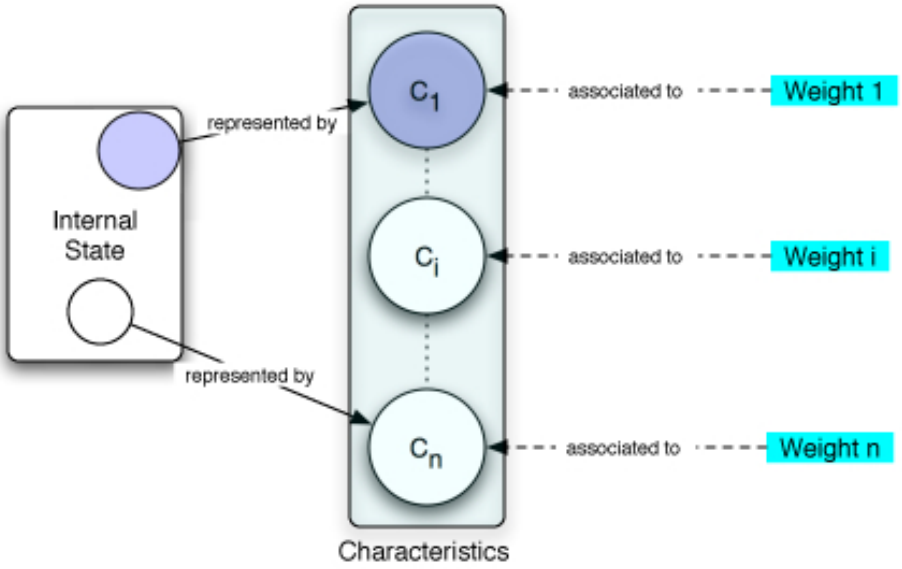

Fig. 4.2 – *La quantification de l'étape de mémorisation*

Concernant Φ_{DEC}, Cette fonction associe une valeur à l'action déclenchée (le résultat de l'étape de décision). Les types des actions possibles sont définies. Un type d'action a un poids. Ensuite, la valeur de la fonction Φ_{DEC} est considérée comme la somme des poids des actions déclenchées.

$$\Phi^{A}_{DEC} = \sum ACT_i$$

Notons que ACT_i est le poids de l'action déclenchée après la réception de "cu_i".

Finalement, la fonction Φ est définie comme la somme de ces deux fonctions Φ_{MEM} et Φ_{DEC}. La spécification des caractéristiques représentant les connaissances et leurs poids, les types d'actions et leurs poids reste à la charge de l'évaluateur.

4.2.1 Étude des communications dans les conversations

Dans ce qui précède, la communication est traitée comme une unité isolée de tout ce qui se passe avant son émission et après son traitement. L'affectation du poids est établie en tenant compte du type de l'action déclenchée et l'effet de l'unité de communication reçue sur les connaissances de l'agent. Par contre, l'effet de l'action déclenchée sur les agents et le SMA ne sera pas pris en compte dans les prochains cycles.

Malgré les problèmes traités par les approches présentées dans les sections précédentes, cette approche ne tient pas compte du fait qu'une unité de communication est impliquée dans une chaîne des communications. Les communications sont conçues pour établir un consensus entre les entités dans le SMA. L'effet de cette unité de communication sur l'agent et le SMA peut être alors nul si le consensus demandé n'est pas établi (figure 4.3).

La figure 4.3 présente deux scénarios possibles de communication. L'entité, dans l'état E_1, reçoit l'unité de communication CU_1. Nous supposons que dans les 2 cas CU_1 a le même effet sur l'agent et que l'agent passe dans les deux scénarios vers le même état E_{i+1}. Après la séquence de communications:

- Le scénario 1 se termine par une réussite et le passage vers l'état E_{i+n+1} avec $E_{i+n+1} \nLeftrightarrow E_1$;
- Le scénario 2 se termine avec un échec et un retour vers l'état E_1.

L'effet de CU_1 du scénario 2 sur l'agent et le SMA est nul si nous considérons le point de vue du séquence des communications et l'effet de ces communications sur l'agent et le SMA.

Pour traiter ce problème, plusieurs questions se posent:

- Comment spécifier une chaîne qui contient l'unité de communication CU_1?
- Quelles sont les limites d'une telle chaîne?

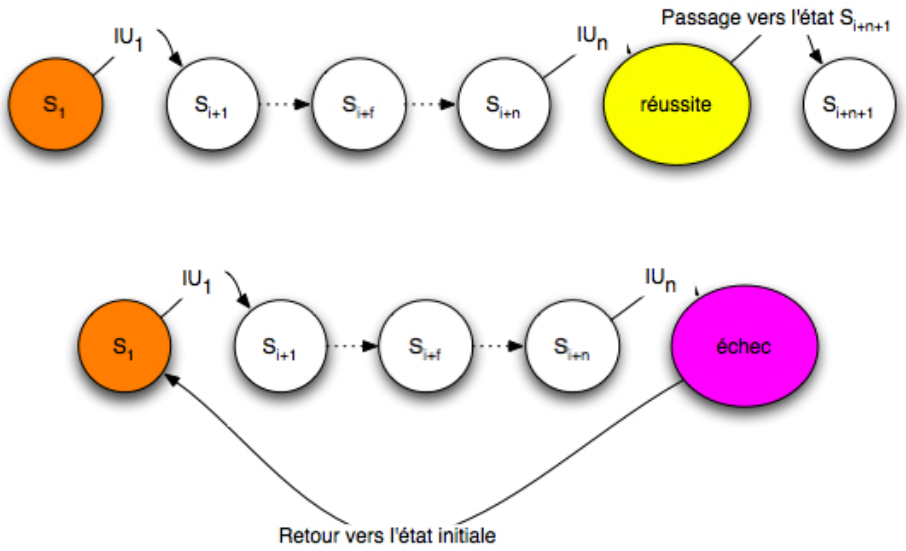

Fig. 4.3 – *Echec et réussite d'une chaine de communication*

– Comment distinguer entre les différents chaînes durant l'exécution d'un SMA ?

Protocoles de communications et conversations

Un des problèmes majeurs dans les méthodes de conception des SMA reste l'ingénierie des protocoles de communications. Les communications nécessitent des protocoles sophistiqués capables de prendre en charge la dynamique du système et la diversité des tâches affectés aux agents. Les agents doivent accommoder leurs échanges d'informations aux situations rencontrées. Les langages de communication inter-agents [Finin *et al.* (1994); Labrou et Finin (1997); FIPA (1997)] sont habituellement associées à un ensemble de protocoles prédéfinis structurant les conversations entre les agents. Les communications sont limitées par les protocoles prédéfinis car les structures des agents ne permettent généralement pas de modifier ces derniers dynamiquement.

La figure 4.4 montre un protocole de communication, "Everybody's request protocol", présenté dans [Demazeau (1995)].

Dans ce protocole, le récepteur "B" d'une requête peut :

– Donner une réponse (2)
– Demander au demandeur "A" plus d'information (3)
– Ne pas répondre (4). Dans ce cas, A demande de B les raison pour cette décision.

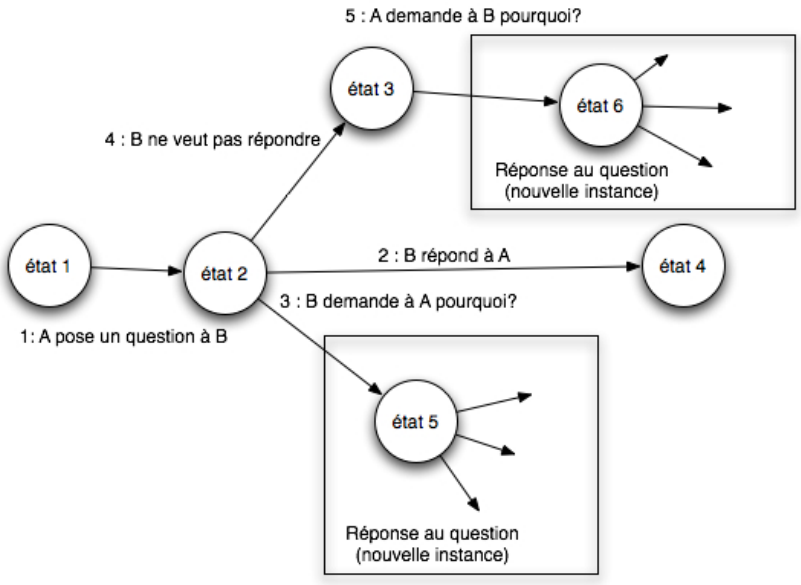

FIG. 4.4 – *Protocole "Everybody's request protocol" [Demazeau (1995)]*

Selon [Demazeau (1995)] un protocole de communication est représenté par un graphe d'états. Les états sont figurés par des cercles. Ces états sont occupés par les agents durant le protocole de communication en fonction du type de communication échangé. Un lien entre deux états représente une transition réalisée suite à une communication.

Dans la suite du travail, nous gardons les informations suivantes concernant un protocole de communication.

- Un protocole de communication est un ensemble de règles génériques qui sont partagées par les agents structurant les échanges entre ces entités.
- Un protocole a un nom;
- Un protocole alloue un rôle à un participant;
- Un protocole décrit la séquence des communications qui peuvent être échangées.

Par la suite, nous considérons une conversation comme une séquence d'actes de communications simples, liées par un sujet, conformes à un protocole et échangées entre deux ou plusieurs agents.

Une solution possible pour délimiter les séquences des messages est de distinguer les différentes conversations se déroulant durant l'exécution d'un SMA. La présence du protocole de communication, qui porte ces conversations, facilite cette tâche. Pour cela, nous proposons, dans la suite, un modèle formel pour

les protocoles des communications et les conversations. Ce modèle se fonde sur les notions présentées dans le modèle de formalisation de la communication.

Modélisation et présentation des problèmes

Dans ce qui suit, nous proposons une modélisation des protocoles par un graphe d'état. Notre modélisation est centrée sur la communication, les états du graphe représentent les communications du protocole. Les transitions entre ces états représentent la relation de précédence entre les CU. Une transition est un passage d'une communication à une autre dans la séquence des communications constituant le protocole.

De plus, nous distinguons dans un protocole :

- Des conditions d'entrée (C_{Entree}) : ces conditions doivent être remplies pour pouvoir entrer dans le protocole.
- Une communication d'entrée (C_{Ent}): c'est la première communication dans un protocole.
- Des conditions de succès (C_{Succes}) : ces conditions représentent le but (finalité désirée d'une séquence de communications) de protocole.
- Des conditions d'échec (C_{Echec}) : ce sont les conditions qui permettent de déduire que le protocole a atteint un état d'échec (finalité désirée n'est pas atteinte).
- Des règles (R) : ce sont les règles permettant au dialogue de progresser vers les conditions de succès ou d'échec.

Une représentation graphique d'un protocole de communication suivant ces spécifications est présenté dans la figure 4.5.

Une communication est définie dans la première partie de ce chapitre comme une fonction qui associe à un CU envoyé "$cu^{A}_{Envoye,i}$" un CU reçu "$cu^{B}_{Recu,j}$" ou un message incompréhensible "cu^{B}_{ϕ}" et $j \succ i$.

$$Communication : CU^{A}_{Envoye} \rightarrow CU^{B}_{Recu} \cup \{cu^{B}_{\phi}\}$$

Soit Com l'ensemble de toutes les communications possibles.

$Com = \{Com_i / \exists Agents$ A, B et $Com_i : cu^{A}_{Envoye} \rightarrow cu^{B}_{Recu}\}$

Alors P_i un protocole de communication:

$$P_i = C_{Entree}, Com_1, ..., Com_n, R_1(Com_{n+1}, ...,$$
$$Com_{n+k}, Echec; Com_{n+k+1}, ..., Com_{n+k+l}, Succes)$$

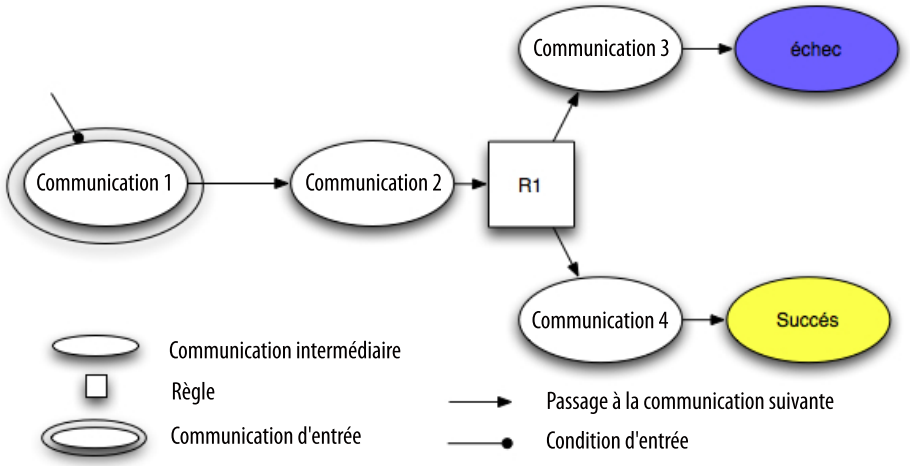

FIG. 4.5 – *Protocole de communication*

Avec:

- C_{Entree} une condition d'entrée;
- Com_i une communication appartenant à l'ensemble Com;
- $R_i(A;B)$ est une règle. La séquence des communications A aura lieu si R_i est validé et la série B aura lieu dans le cas contraire.

Par exemple, le protocole de communication présenté dans la figure 4.6 est formalisé de la façon suivante:

$$Protocole = C_{Entree}, Com_1, Com_2, R_1(Com_3, Succes; Com_4, Echec)$$

Nous notons qu'une communication Com_i est un *successeur* deCom_j dans un protocole de communication P_i si et seulement si:

- Il existe une transition entre Com_i et Com_j alors Com_j suit Com_i directement dans la formalisation ou;
- Il existe une règle R_i qui sépare Com_i et Com_j alors Com_i est avant R_i directement et Com_j suit directement R_i.

Une conversation $Conv_i$ est une séquence de communications alors:

$$Conv_i = com_0; com_1, com_2, ..., com_n, Etat_{final}$$

Avec:

$Etat_{final}$: peut être échec (E) succès (S)

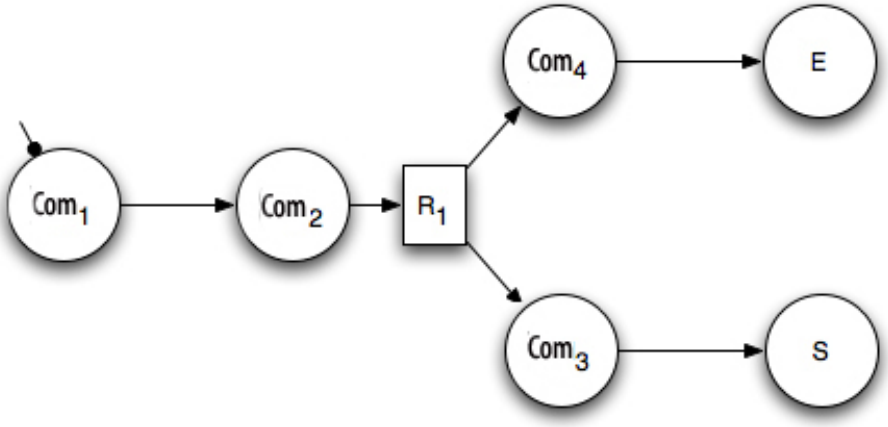

Fig. 4.6 – *exemple d'un protocole de communication*

une exemple d'une conversation qui suit le protocole présenté dans la figure 4.6 est :

$$Conv_i = com_1; com_2, com_4, Echec$$

Une conversation $Conv_i$ suit le protocole P_i si et seulement si:

- La communication d'entrée com_0 de $Conv_i$ est équivalent à la communication d'entrée $Conv_0$ de P_i
- Quelque soit la communication com_k de $Conv_i$ équivalent à la communication Com_k de P_i alors le successeur de com_k dans $Conv_i$ est équivalent au successeur de Com_k de P_i

Traitement des problèmes

Le problème présenté au début de cette section concerne les séquences des communications qui se termine par un échec. Dans ce cas l'effet d'un CU appartenant à cette séquence est nul sur l'agent et le SMA. Les séquences distinguées sont les conversations: l'effet d'un CU, com_k, appartenant à une conversation $Conv_i$ est nul si $Conv_i$ se termine par un échec.

Pour résoudre ce problème, nous définissons une *conversation pertinente* comme une conversation qui se termine par un état de succès. Alors une *communication globale pertinente* est une communication qui appartient à une conversation pertinente. Ce qui signifie que com_k est une *communication globale pertinente* si et seulement si :

- Il existe une conversation $Conv_i$ telque com_k appartient à $Conv_i$;
- $Conv_i$ est pertinente.

La solution du problème présenté revient à la mesure des *communications globales pertinentes* durant l'exécution d'un SMA. Les mêmes fonctions de quantification proposées dans la première partie sont utilisées pour quantifier les *communications globales pertinentes.*

4.3 Analyse des communications : résultats et discussion

Nous présentons ici brièvement une application de notre approche sur le système proies/prédateurs. Des expérimentations plus détaillées sur des applications qui présentent plus de complexité au niveau de l'architecture des communications sont présentées dans le chapitre 8.

Les mesures présentées dans cette partie sont faites sur différents agents prédateurs qui s'exécutent de façon distribuée sur un certain nombre de serveurs. D'où vient les notations utilisées, dans la suite de cette partie, prédateur i sur serveur j pour désigner les agents.

La figure 4.7 présente la variation du nombre des messages reçus par le prédateur 0 du serveur 1 en fonction du temps. Cette figure montre qu'une telle variation est linéaire avec une pente égale à 65.5. La pente de la fonction de variation est obtenue par une régression linéaire des valeurs en fonction du temps. Nous constatons qu'une variation est linéaire si le coefficient de détermination R^2 est proche de 1.

Dans la figure 4.8, la variation du poids des messages reçus par le prédateur 0 du serveur 1, en fonction de temps selon l'approche types des messages, est présentée. Cette variation est aussi une fonction linéaire avec une pente égale à 103.

La différence entre les pentes dans les 2 approches est due au fait que le poids des messages diminue selon le type de message et son effet pour l'agent. Il existe de fait des types des messages qui sont plus ou moins utiles pour l'agent. Une telle affectation de poids de message par type augmente donc la pente de la fonction de variation. La figure 4.8 présente la quantité des informations portées par les messages plutôt que la quantité des messages dans la figure 4.7.

La figure 4.9 présente la variation du poids des messages selon la troisième approche (action et mémorisation). Dans cette approche, la variation n'est pas linéaire, ceci est dû au fait qu'un grand nombre des messages reçus sont considérés non utiles du point de vue de la quantité des informations portées.

La différence entre les 3 figures se situe au niveau de la pente du fonction de variation. Cette différence est expliquée par l'affectation du poids aux messages selon l'importance de ces messages pour l'agent soit en affectant le poids aux

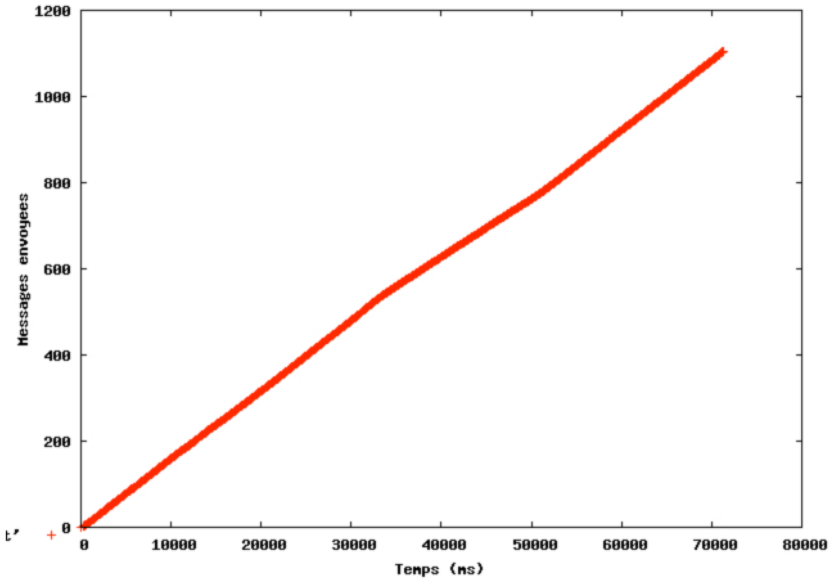

Fig. 4.7 – *Variation de nombre des messages reçus en fonction du temps pour le prédateur 0 du serveur 1*

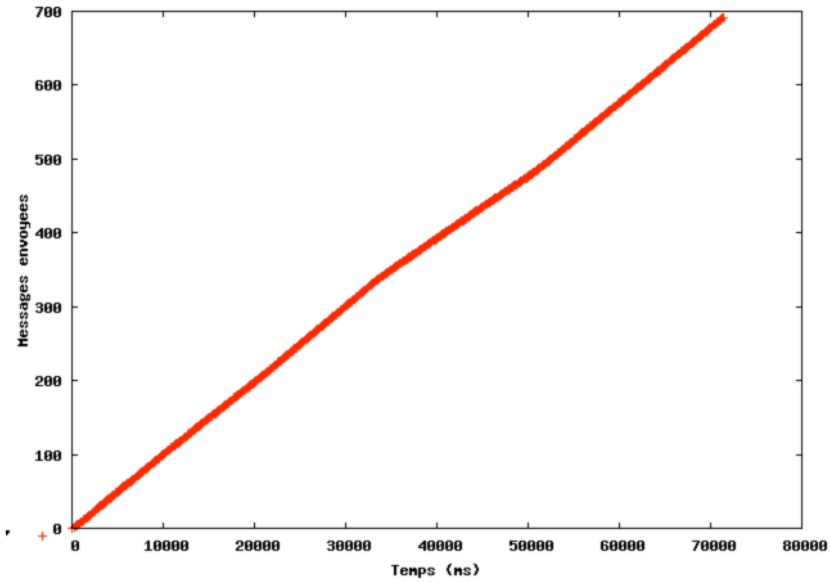

Fig. 4.8 – *Variation du poids des messages (approche type) en fonction du temps pour le prédateur 0 du serveur 1*

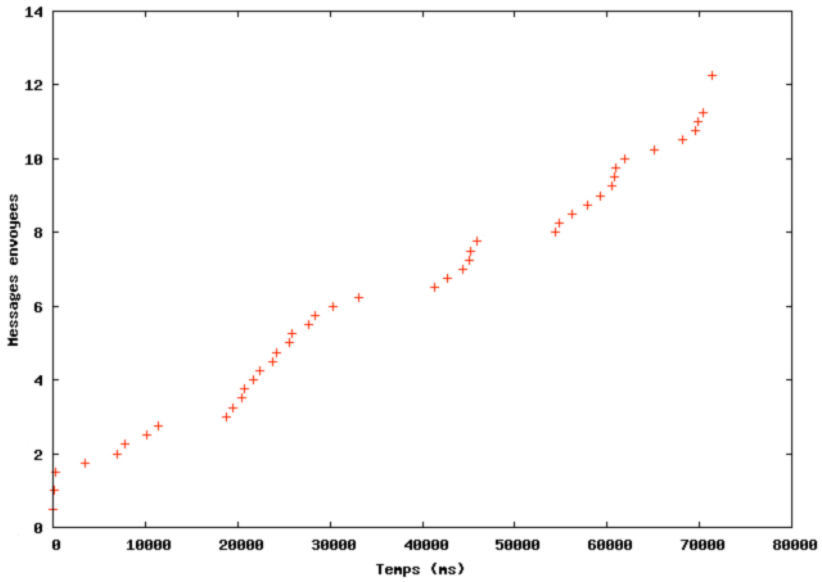

FIG. 4.9 – *Variation du poids des messages (approche 3) en fonction du temps pour le prédateur 0 du serveur 1*

types des messages comme dans l'approche 2, soit en l'affectant aux effets des messages comme dans l'approche 3. Pour plus d'analyse et de compréhension et pour étudier les résultats des différents approches, nous présentons dans la suite de ce rapport des extractions, dans des intervalles de temps précis, de ces 3 figures en les analysant selon l'état de l'agent observé dans la visualisation de l'exécution de ce système.

La figure 4.10 présente un extrait de la visualisation de l'exécution du système. Dans cette figure, les 2 agents prédateur 0 du serveur 2 et prédateur 0 du serveur 3 sont dans l'état de "capturer un proie" (jaune) dans la même période de temps (entre 7437 et 7515). A la fin de cette période, l'agent prédateur 0 du serveur 2 finit par capturer une proie tandis que l'agent prédateur 0 du serveur 3 n'y réussit pas.

La figure 4.11 présente la variation de la quantité de messages dans un intervalle de temps la période de 7437 à 7515 pour l'agent prédateur 0 du serveur 2. La variation du poids des messages (approche 3) dans un intervalle contenant la même période est présentée pour le même agent (prédateur 0 du serveur 2) dans la figure 4.12. Les figures 4.13 et 4.14 présentent les même fonctions de variation pour l'agent prédateur 0 du serveur 3 dans des intervalles de temps similaires. Nous pouvons remarquer que selon l'approche usuelle, nous ne pouvons pas distinguer les communications importantes pour l'agent. Par suite, la distinction entre les résultats (succés ou échec) des communications

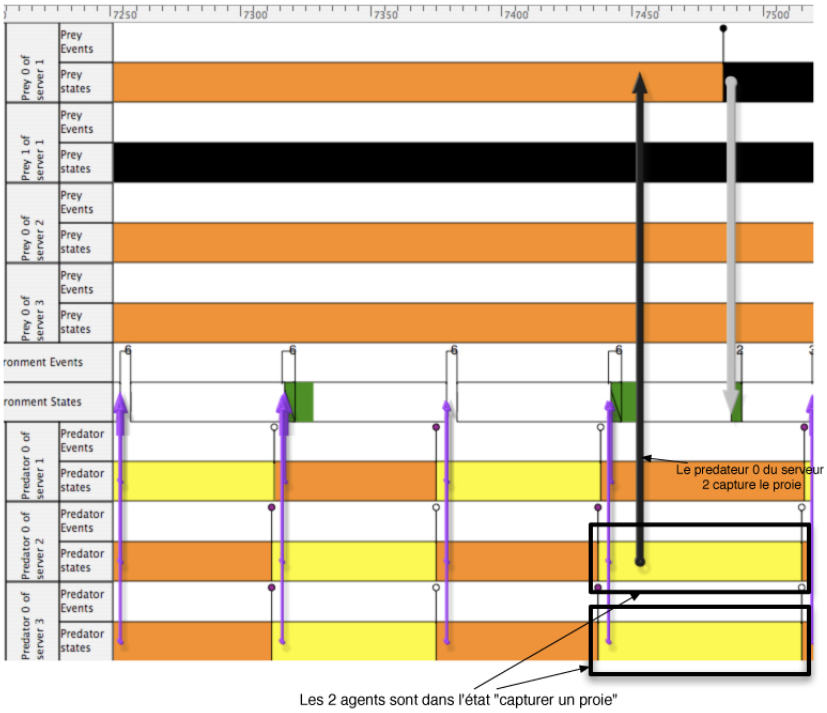

Fig. 4.10 – *Extrait de la visualisation du système*

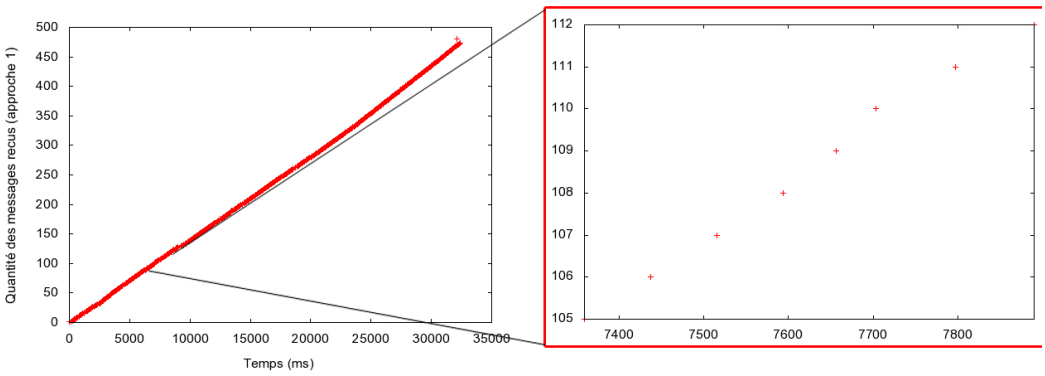

Fig. 4.11 – *Variation de la quantité de messages pour le prédateur 0 du serveur 2 (intervalle 7437, 7515)*

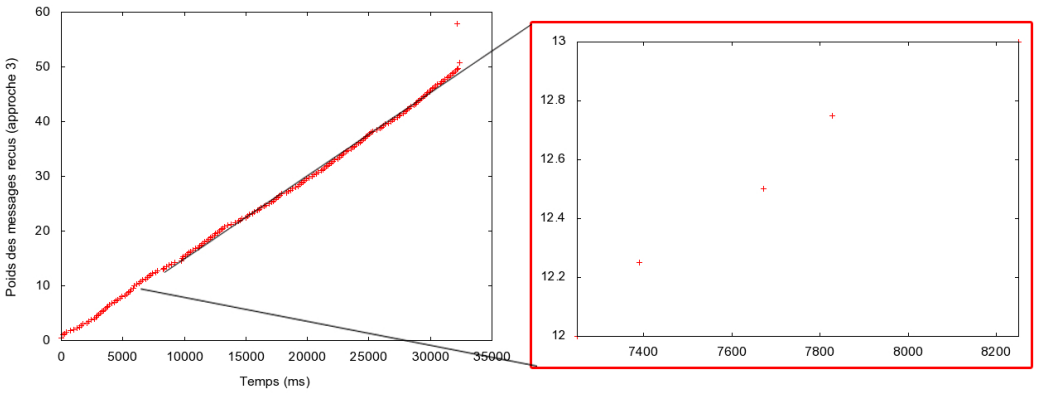

FIG. 4.12 – *Variation du poids des messages pour le prédateur 0 du serveur 2 (intervalle 7437, 7515)*

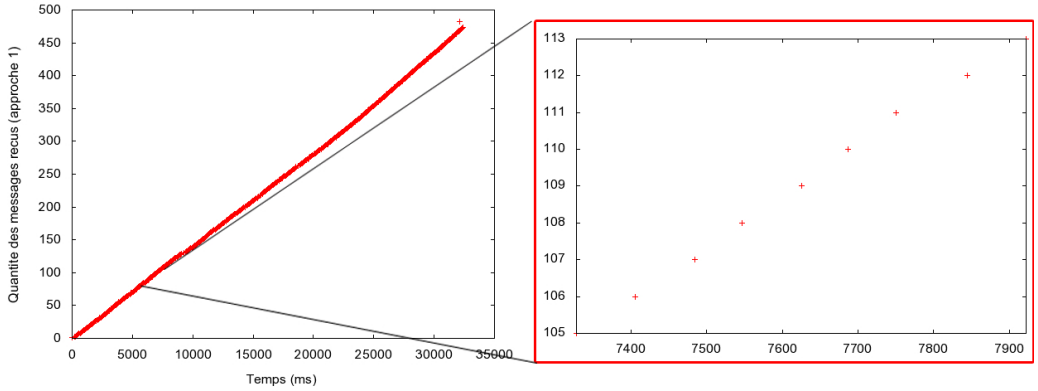

FIG. 4.13 – *Variation de la quantité de messages pour le prédateur 0 du serveur 3 (intervalle 7437, 7515)*

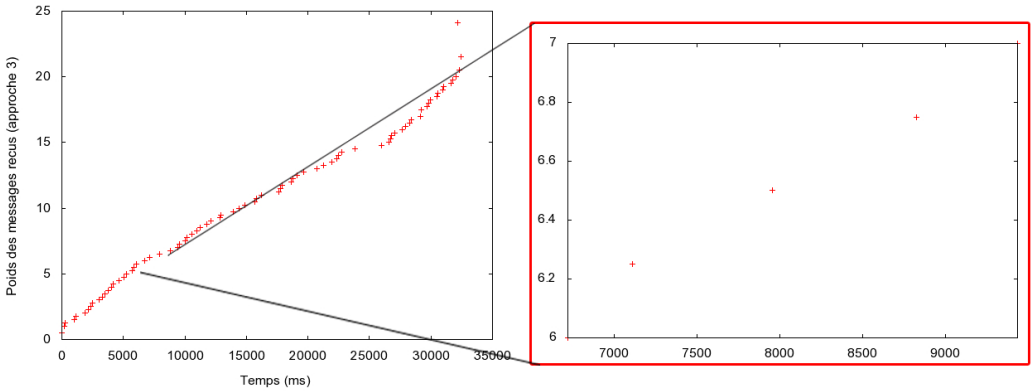

FIG. 4.14 – *Variation du poids de messages pour le prédateur 0 du serveur 3 (intervalle 7437, 7515)*

de l'agent est impossible. La différence est remarquable entre les 2 figures représentant les variations des poids de messages pour les 2 agents.

Dans ce qui suit, nous présentons la variation de poids des messages, selon l'approche 3, pour le prédateur 0 du serveur 2 et pour le prédateur 0 du serveur 3 (figures 4.15 et 4.16). Nous avons antérieurement montré cette variation pour le prédateur 0 du serveur 1 pour cette même approche (figure 4.9).

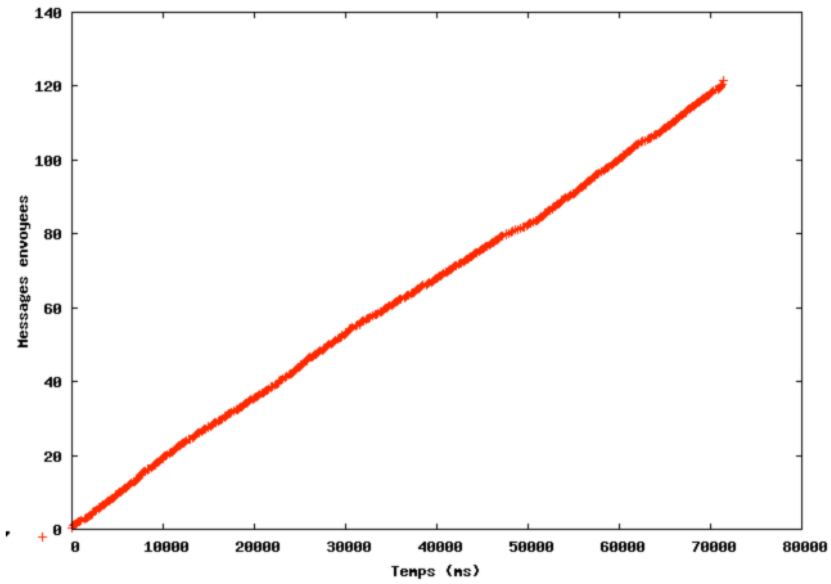

FIG. 4.15 – *Variation du poids des unités des communications (approche 3) en fonction du temps pour le prédateur 1 du serveur 2*

Nous notons que la variation du poids des messages reçus (approche 3) n'est pas uniforme pour ces 3 prédateurs. Pour une meilleure interprétation de cette variation, nous présentons dans les figures 4.17, 4.18 et 4.19 la répartition des tâches pour ces 3 prédateurs.

Nous remarquons qu'une longue période du temps est passée (77,6), par l'agent prédateur 0 du serveur 1, pour essayer d'arrêter les proies. Malgré le temps passé, la quantité de communications pertinentes de cet agent est très petite. Ce qui est expliqué par le fait que cet agent a essayé d'arrêter des proies sans avoir réussir. La majeure partie des communications de cet agent étaient inutiles. L'agent prédateur 0 du serveur 3 passe la même période du temps (76,9) dans cette état mais avec plus d'efficacité ce qui entraîne plus de communications importantes pour l'agent. L'agent prédateur 0 du serveur 2 passe une période de temps beaucoup plus petite dans l'étape de "capturer les proies" et paraît beaucoup plus efficace. Il arrive à arrêter plus des proies. Ces remarques prouvent l'importance de notre approche dans l'étude et la compré-

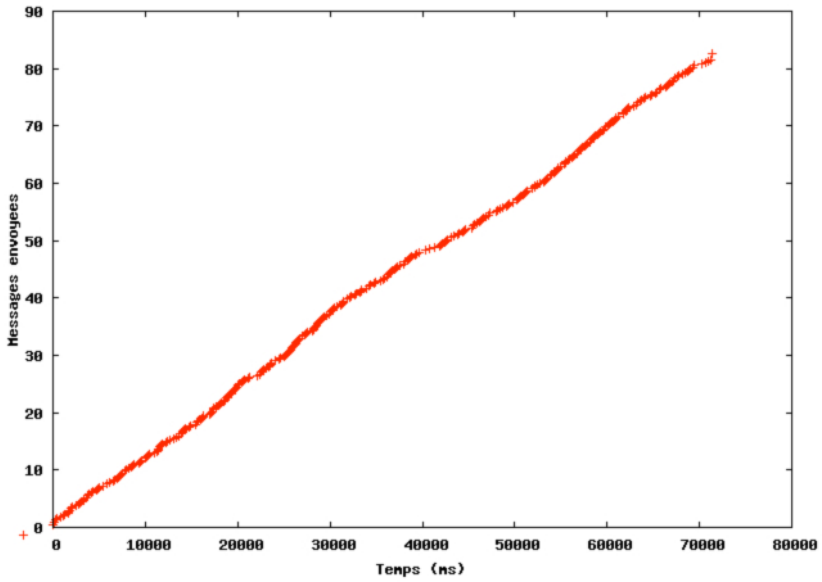

FIG. 4.16 – *Variation du poids des unités des communications (approche 3) en fonction du temps pour le prédateur 1 sur serveur 3*

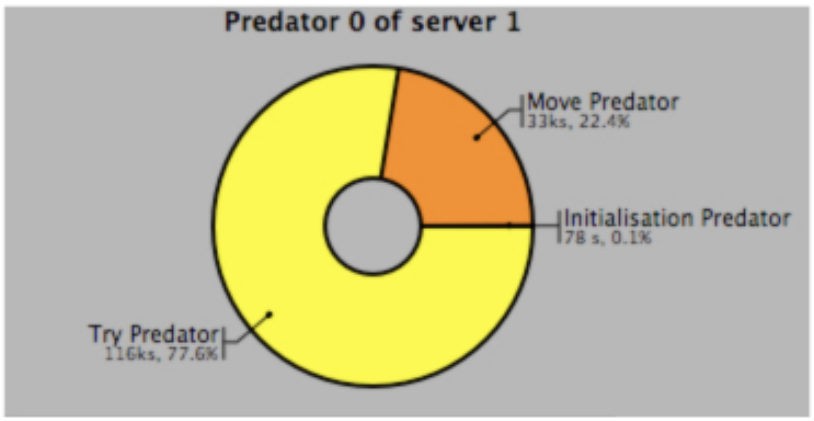

FIG. 4.17 – *Distribution des tâches pour le prédateur 0 du serveur 1*

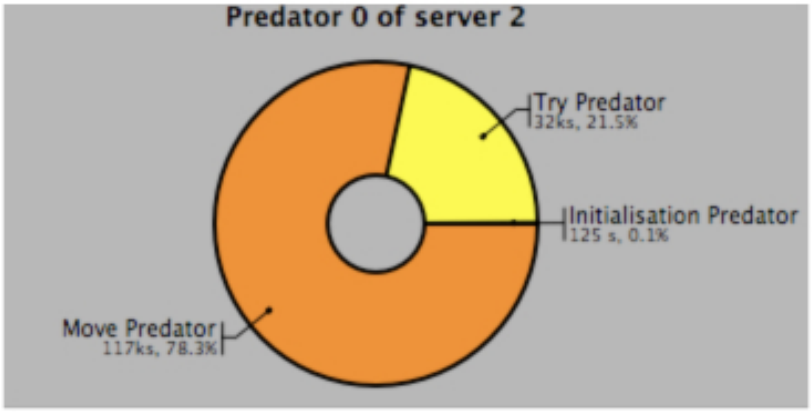

Fig. 4.18 – *Distribution des tâches pour le prédateur 0 du serveur 2*

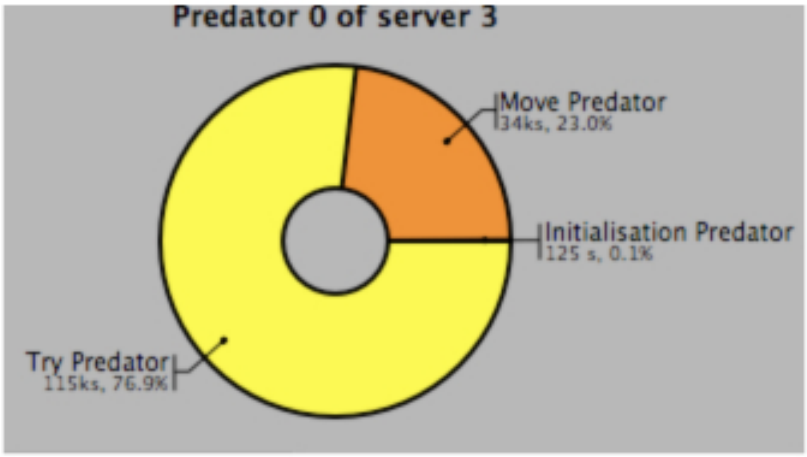

Fig. 4.19 – *Distribution des tâches pour le prédateur 0 du serveur 3*

hension du comportement internes et la détection des sources de défaillance dans l'architecture interne des agents.

Pour conclure, la modélisation et l'analyse de la communication et des conversations dans un SMA permet de :

- Présenter des informations concernant les liens entre les agents ;
- Obtenir des indices de performances au niveau de l'ensemble des entités ;
- Localiser les défaillances et les dysfonctionnements au niveau des liens entre les entités ;
- Permettre de comparer les entités et les différents systèmes au niveau de leurs communications.

Un couplage entre les deux techniques, la visualisation du comportement des entités dans un SMA et la modélisation et l'analyse des caractéristiques représentant les liens entre les agents (chapitres 6 et 8), permet de proposer une approche permettant d'obtenir une analyse complète du comportement d'un SMA.

Chapitre 5

MAS-Paje Voyelles : Architecture et implantation

Dans le chapitre 3, nous avons traité le processus de visualisation de l'exécution des SMA d'un point de vue théorique. L'abstraction et le modèle de visualisation des SMA ont été proposés et les différentes alternatives servant dans ce processus ont été présentées et discutées. Une étude au niveau théorique ne suffit pas pour réaliser une telle visualisation. Le processus de l'étape de collecte de données jusqu'à l'obtention des traces d'exécution est plus ou moins difficile à réaliser et, en même temps, à garantir le fait de ne pas importer d'intrusions dues aux techniques utilisées (collecte de données, ...), selon les choix faits et les solutions proposées. La dimension technique du travail que nous présentons ici est nécessaire pour :

- Réaliser la visualisation des SMA;
- Garantir, dans le processus de visualisation, le fait de minimiser l'effet de l'intrusion selon la technique de collecte utilisée;
- Garantir, dans le processus de visualisation, le fait d'obtenir le maximum d'informations sur le comportement des entités dans un SMA.

Nous démontrons dans ce chapitre l'applicabilité du modèle formel défini dans le chapitre 3 selon 3 objectifs:

- Sélectionner un outil de visualisation qui réponde aux critères de choix discutés dans la partie 3 du chapitre 3 (détaillés dans la section suivante);
- Réaliser la dimension technique de la visualisation des SMA;
- Proposer une architecture d'un système qui gère le processus de visualisation de l'étape de collecte de données jusqu'à l'obtention d'un fichier de traces compatible avec le format accepté par l'outil de visualisation choisi.

Ce chapitre est divisé en 3 parties. Dans la première partie, nous présentons un comparatif entre différents outils de visualisation candidats pour la visualisation des traces des SMA. Aprés avoir choisi, nous présentons l'outil de visualisation Paje. Cet outil est utilisé dans la suite pour visualiser les SMA. Le rôle, l'architecture et le format de données acceptés par cet outil sont ensuite présentés.

Dans la deuxième partie, Nous proposons d'automatiser la tâche de visualisation des SMA en construisant un système de visualisation des SMA (MAS-Paje). Une architecture de MAS-Paje est proposée et les moyens techniques qui la supportent sont choisis et argumentés. Finalement, nous présentons le rôle de MAS-Paje dans la visualisation d'un SMA sur l'application Proies/-Prédateurs.

5.1 Paje : choix et description générale

Le choix d'un outil de visualisation, conformément à ce qui est présenté dans la partie 3 du chapitre 3, doit être lié à ses caractéristiques d'extensibilité et de modularité. Un outil adapté à la visualisation des SMA doit avoir:

- La possibilité de représenter l'exécution des systèmes durant une longue période;
- La possibilité de traiter la visualisation de l'exécution pour un grand nombre d'entités constituant le système;
- La possibilité d'ajouter de nouvelles fonctionnalités et d'étendre les fonctionnalités existantes sans nécessité de changer l'outil en totalité;
- La possibilité de visualiser des entités qui se créent et se détruisent durant l'exécution du système.

En plus des points listés, nous avons distingués deux dimensions importantes de la classification des outils des visualisations proposées par Reed [Reed et Ribler (1998)]:

- **Dynamisme**, au sens où l'outil met sa représentation à jour pour afficher le changement, selon les besoins de l'utilisateur.
- **Ordinalité**, une visualisation peut comporter des variables non-ordonnées représentées en fonction du temps. Au sens où on peut visualiser des données non ordonnées, comme par exemple l'état de l'agent en fonction du temps.

A la fin de la partie 3 du chapitre 3, nous nous sommes orientés pour visualiser les traces d'exécution des SMA vers l'adaptation des outils de visualisation utilisées pour les systèmes parallèles, plutôt que vers la construction des outils de visualisation spécifiques aux SMA.

-	Paragraph	Pablo	Svpablo	Paradyn	Annai	Scope	Gthread	Paje
Exécution longue periode	+	+	+	0	0	+	+	+
Grand nombre des entités	-	+	+	+	+	0	0	+
Extension des fonctionnalités	-	-	+	0	+	0	0	+
Entités construites durant l'exécution	-	-	0	0	0	0	0	+
Dynamisme	0	0	0	+	+	+	+	+
Ordinalité	+	+	+	0	+	+	+	+
Interactivité de l'interface	+	-	0	0	+	+	0	+
Visualisation en fonction du temps	+	-	0	+	+	+	+	+
Diversité des visualisations	0	0	+	+	+	0	0	0

TAB. 5.1 – *Comparatif entre différents outils de visualisation des systèmes parallèles*

De nombreux outils de visualisation des traces existent dans le domaine des systèmes parallèles. Nous avons répertorié : Paragraph [Heath et Etheridge (1991)], Pablo [Reed (1993b)], SvPablo [Zhang et Reed (1998)], Paradyn [Miller *et al.* (1995)] , Annai [Wylie et Endo (1996)], scope[Arrouye (1995)], Gthread [Zhao et Stasko (1995)], Paje [Stein et Kergommeaux (1998)]. Ces outils montrent graphiquement les communications interprocessus, l'activité des processeurs et différents autres indices de performances à partir d'un fichier de traces. Ces outils proposent un très grand nombre de représentations possibles.

Dans le but de choisir l'outil de visualisation le plus adapté à nos critères pour visualiser les SMA, nous présentons, dans ce qui suit, un comparatif entre différents outils de visualisation des systèmes parallèles.

Les comparaisons sont réalisées selon des critères représentant les points cités ci-dessus. Pour chaque outil, une valeur parmi +, 0 et - est affectée à chaque critère. La valeur + signifie que le critère est bien présent dans la visualisation proposée par l'outil. La valeur 0 signifie que le critère est présent mais pas suffisamment traité dans l'outil. Finalement, la valeur - signifie que le critère n'est pas présent dans la visualisation proposée par l'outil. Le tableau 5.1 présente les résultats.

Suite aux résultats présentés dans le tableau 5.1, nous considérons dans ce travail Paje [Stein et Kergommeaux (1998)] comme outil pour visualiser

l'exécution des SMA. Dans ce qui suit, nous listons les avantages apportés par cet outil et nous les discutons en détails dans la suite. Paje permet:

- De représenter un nombre potentiellement grand de fils d'exécution dans une exécution éventuellement longue ("scalabilité" ou aptitude à passer à l'échelle);
- D'ajouter de nouvelles fonctionnalités à l'environnement (extensibilité);
- La visualisation d'autres modèles de programmation, en lui laissant la possibilité de spécifier comment cette visualisation doit être faite;
- De visualiser des entités qui peuvent être créées et détruites dynamiquement durant le déroulement de l'exécution des systèmes.
- Les déplacements dans le temps, l'interrogation de données visualisées (l'interactivité de l'interface de présentation des traces)(interface espace-temps).

"Scalabilité" (aptitude à passer à l'échelle): Paje offre une forme de "scalabilité" en permettant aux utilisateurs de "zoomer" interactivement dans l'espace et dans le temps.

- Dans l'espace : il est possible d'observer une exécution à différents niveaux d'abstraction : groupes de noeuds, noeuds, fils d'exécution.
- Dans le temps : il est possible de changer dynamiquement l'échelle de représentation du temps ce qui permet de passer de la représentation d'une longue période de temps à une représentation d'une période plus courte, où plus de détails seront visibles.

Le fait que le passage d'un niveau d'abstraction à un autre et que le changement d'échelle du temps soient interactifs a une grande importance pour la scalabilité, car si la vision d'ensemble est la seule façon de maîtriser la grande quantité d'informations à traiter, la vision détaillée est la seule façon de mettre en évidence l'origine d'un problème de performances. Donc, pour un environnement non interactif, nous sommes obligés de limiter à des représentations graphiques "scalable" qui sont habituellement calculées à partir de moyennes et ne permettent pas d'analyse détaillée. Donc, la "scalabilité" de Paje permet aux programmeurs des systèmes de grande taille la possibilité de naviguer selon leurs besoins entre différents niveaux d'abstraction.

La présence d'un grand nombre d'entités dans un SMA nécessite la présence de "scalabilité" dans l'outil de visualisation utilisé. De plus, la façon de résoudre le problème de "scalabilité" dans Paje, par passage d'un niveau d'abstraction à un autre, semble idéal pour la compréhension du comportement:

- Un passage dans l'espace est fondamental pour analyser en détail les comportements des agents sur différents niveaux d'abstractions et pour extraire des "patterns" d'exécution. Le passage dans l'espace sert aussi à

tirer des conclusions sur la distribution des états des agents sur le temps de l'exécution.
- Un passage dans le temps est très important pour naviguer dans la visualisation de l'exécution et pour analyser des comportements qui ont eu lieu à des instants précis de l'exécution.

Extensibilité : Paje offre plusieurs caractéristiques destinées à en faciliter l'extension :

- Conception en modules indépendants;
- Indépendance des visualisations relativement à la sémantique du modèle de programmation de l'application visualisée;
- Généricité du module de simulation qui constitue le coeur de Paje.

Cette dernière propriété de Paje donne en fait aux "programmeurs d'applications" la possibilité de spécifier dans les programmes tracés ce qu'ils veulent visualiser et comment la visualisation doit être faite (le modèle en composant de Paje est présenté dans la partie architecture de Paje dans la suite).

L'extensibilité était le critère le plus important dans le choix d'un outil de visualisation pour les SMA. Un outil de visualisation de systèmes parallèles qui ne possède pas la propriété d'extensibilité, au niveau de généricité de module de simulation et de l'indépendance des visualisations proposées de la sémantique du modèle de programmation, rend impossible notre travail d'adaptation d'un tel outil aux SMA.

Interactivité : Outre les propriétés d'interactivité présentées ci-dessus, possibilités de changer le niveau d'abstraction et l'échelle de temps durant la session de visualisation, Paje permet d'inspecter les données visualisées (fils d'exécution, communications, etc.), de se déplacer en avant et en arrière dans le temps, etc. L'interactivité suppose un accès très rapide aux données de visualisation puisqu'un réaffichage peut être provoqué par le moindre mouvement de souris de l'utilisateur. L'interactivité implique donc le maintien en mémoire des données de la trace ainsi que des données produites par le simulateur. Différentes organisations de données ont été conçues dans Paje pour accélérer la recherche des données [Stein et Kergommeaux (1998)].

Diagramme Espace-temps : Ce diagramme représente les états successifs, les communications et les événements de chaque fil d'exécution en fonction du temps. L'espace alloué à chaque noeud s'adapte dynamiquement au nombre de fils d'exécution qui s'y exécutent. L'axe horizontal représente le temps, et l'axe vertical représente les fils d'exécution, groupés par noeud.

La présentation des états des agents dans un diagramme espace/temps est l'un des critères importants pour réaliser l'abstraction proposée dans le chapitre 3 pour les SMA et présentée dans la figure 3.5.

De tout ce qui précède, nous validons le fait que Paje répond aux critères de choix de l'outil de visualisation pour un SMA présentées dans la partie 3 du chapitre 3. Finalement, il est important de noter que l'outil Paje est réalisé par l'équipe MESCAL (l'une de nos deux équipes de recherche), ce qui motive, en plus de ses qualités techniques présentées ci-dessus, son utilisation pour la visualisation des SMA.

5.2 Paje : architecture, format de fichier

La figure 5.1 représente le graphe des tâches effectuées par Paje. Ce graphe est composé d'un ensemble des composants indépendants. Les sommets du graphe sont les composants d'analyse tandis que les arêtes sont les liens de communication entre ces composants. Les données qui parcourent les liens sont des informations représentant les entités, événements, états, communications, etc. L'indépendance des composants de l'outil de visualisation sert à les rendre plus réutilisables.

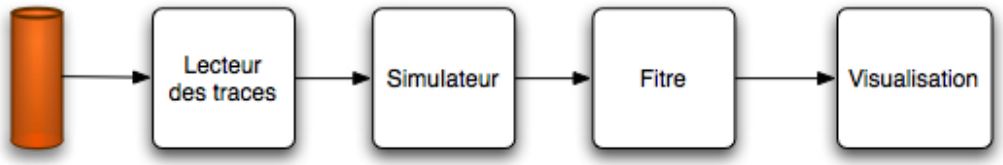

FIG. 5.1 – *tâches réalisées par l'outil de visualisation Paje*

Dans une visualisation réalisée par Paje, le fichier de traces est lu par le composant "lecteur de traces" (il faut que le contenu du fichier soit conforme au format admis par Paje). Le lecteur de traces vérifie la conformité du fichier de traces et engendre des informations représentant les événements produits lors de l'exécution du système analysé. Le simulateur utilise ces informations de bas niveau pour produire des données représentant des informations plus abstraites telles que les états successifs, les communications, etc. Ces informations sont ensuite envoyées à un filtre qui choisit les données à visualiser selon le niveau d'abstraction souhaité par l'utilisateur de l'outil et selon l'intervalle du temps sélectionné par l'utilisateur. Le filtrage à ce niveau est un filtrage de niveau d'abstraction de présentation des données.

Ainsi, l'outil de visualisation prend un fichier de trace pré-traité par l'analyse et réalise la partie de présentation visuelle des traces d'un système.

Format de fichier de traces

La seule contrainte imposée par les composants de Paje sur l'organisation des données est leur structuration hiérarchique, qui doit refléter la hiérarchie du modèle de programmation (un exemple de l'organisation des données en système parallèle est représenté dans la figure 5.2). Les noeuds de cet arbre sont les feuilles et les conteneurs : une feuille, représentée dans la figure 5.2, est une donnée élémentaire telle qu'un événement particulier dans l'exécution du système, un certain état, une communication, etc. Les valeurs des feuilles sont les données représentées graphiquement par l'outil de visualisation.

Le rôle d'une feuille est de représenter une donnée visuelle attachée à un conteneur. Les conteneurs sont les noeuds de plus haut niveau dans la hiérarchie, leur rôle est de rassembler des feuilles ou d'autres conteneurs et de représenter l'organisation des composants à l'intérieur d'une application.

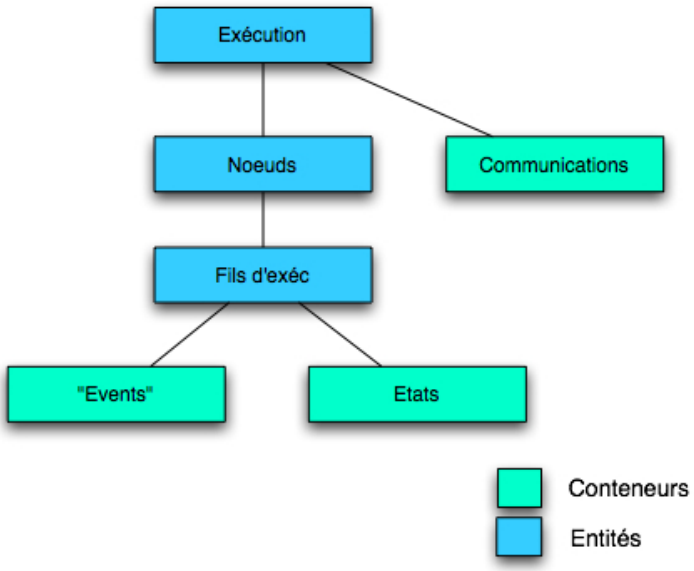

FIG. 5.2 – *Structuration hiérarchique des données pour une application parallèle*

La visualisation fournie par Paje se fonde sur un fichier de traces. Ce fichier est composé d'une suite d'"events[1]". Un "event" est un tableau possédant plusieurs champs: nom, type, valeur, etc. (la figure 5.3 montre un "event" d'envoi d'un message de 320 octets du processus 5 au processus 3). Généralement, il

1. Le terme anglais "event" est utilisé pour distinguer de terme événement. Le terme "event" signifie dans ce document une entrée dans le fichier de traces (création des conteneurs, instanciation des conteneurs, etc.).

existe un grand nombre d'"events" similaires dans un fichier de trace, du fait de la multitude d'envois et de réceptions de messages.

Field Name	Field Type	Field Value
Event Name	string	SendMessage
Time	timestamp	3.233222
ProcessId	integer	5
Receiver	integer	3
Size	integer	320

FIG. 5.3 – *Exemple d'un "events" d'envoi d'un message*

Afin de réduire la taille du fichier de trace, une solution introduite par Paje consiste à ajouter la définition des "events" au début du fichier de trace et de référencer par la suite un "event" d'un type donné par un identifiant. Par conséquence, un fichier de trace Paje est "self-defined", ce qui veut dire que les informations de définition des types d' "events" sont incluses dans le fichier de trace. En résumé, le fichier de trace est donc constitué de deux parties. La première constitue l'ensemble des définitions des "events" et la deuxième contient les identifiants associés à chaque "event" avec les valeurs associées.

Les "events" utilisés dans Paje peuvent être divisés en 4 classes :

- Les "events" servant à définir les types des conteneurs (ces "events" sont de type PajeDefineContainerType);
- Les "events" servant à définir les types des feuilles et les valeurs possibles pour ces feuilles;
- Les "events" qui instancient et détruisent les conteneurs (ces "events" sont de type PajeCreateContainer et PajeDestroyContainer);
- Les "events" qui créent les données visuelles (changements des valeurs des feuilles).

Typiquement, les "events" des deux premières classes sont au début du fichier de trace. Viennent ensuite les "events" qui instancient les conteneurs, puis le fichier se termine par un grand nombre d'"events" qui créent les données visuelles. Il existe 4 types des données visuelles (types des feuilles) :

- Les événements : ils sont utilisés pour représenter les faits importants qui ont eu lieu à un instant donné;
- Les états : ils sont utilisés pour représenter le fait qu'un certain conteneur est dans un état précis durant un certain temps. Ils sont visualisés par des rectangles dans le diagramme d'espace-temps de Paje;

- Les liens: ils représentent les liens entre deux conteneurs. Ces liens commencent à un instant donné et se terminent à un autre instant différent (comme par exemple la communication entre deux agents). Ces liens sont visualisés par des flèches;
- Les variables: elles sont utilisées pour représenter l'évolution dans le temps d'une valeur associée à un conteneur. elles sont représentées par un graphe dans le diagramme espace-temps de Paje.

Les différentes feuilles sont définies par les "events": PajeDefineEventType, PajeDefineStateType, PajeDefineLinkType et PajeDefineVariableType). La création des données visuelles (changements des valeurs des feuilles) se fait par: PajeSetState, PajePushState, PajeNewEvent, PajeSetVariable, PajeStartLink et PajeEndLink). Nous aborderons avec plus des détails tous ces événements dans la partie application de ce chapitre (Section 3).

Un exemple simple d'un fichier de traces est présenté dans la figure 5.4.

5.3 MAS-Paje : architecture et détails techniques

Ayant choisi et présenté l'outil de visualisation, nous proposons d'automatiser la visualisation des SMA, en présentant une architecture d'un système de visualisation des SMA par Paje (MAS-Paje).

MAS-Paje : rôle général

Le rôle plus général de MAS-Paje, comme système de visualisation des SMA, est de faire la liaison entre l'exécution d'un tel système et l'outil de visualisation de traces "Paje" (figure 5.5).

Comme déjà mentionné, l'outil de visualisation prend un fichier de traces prétraité par l'analyse et traite, ensuite, la partie de présentation visuelle des traces d'un système. MAS-Paje, a donc pour rôle, de réaliser l'étape de collecte des données, l'analyse des données et ensuite la conversion du format de fichier de traces, le tout afin d'engendrer un fichier de traces conforme au format accepté par Paje.

5.3.1 MAS-Paje : tâches et discussion

Le graphe des tâches à effectuer par MAS-Paje est montré dans la figure 5.6. La première étape consiste à recevoir des informations statiques de l'utilisateur

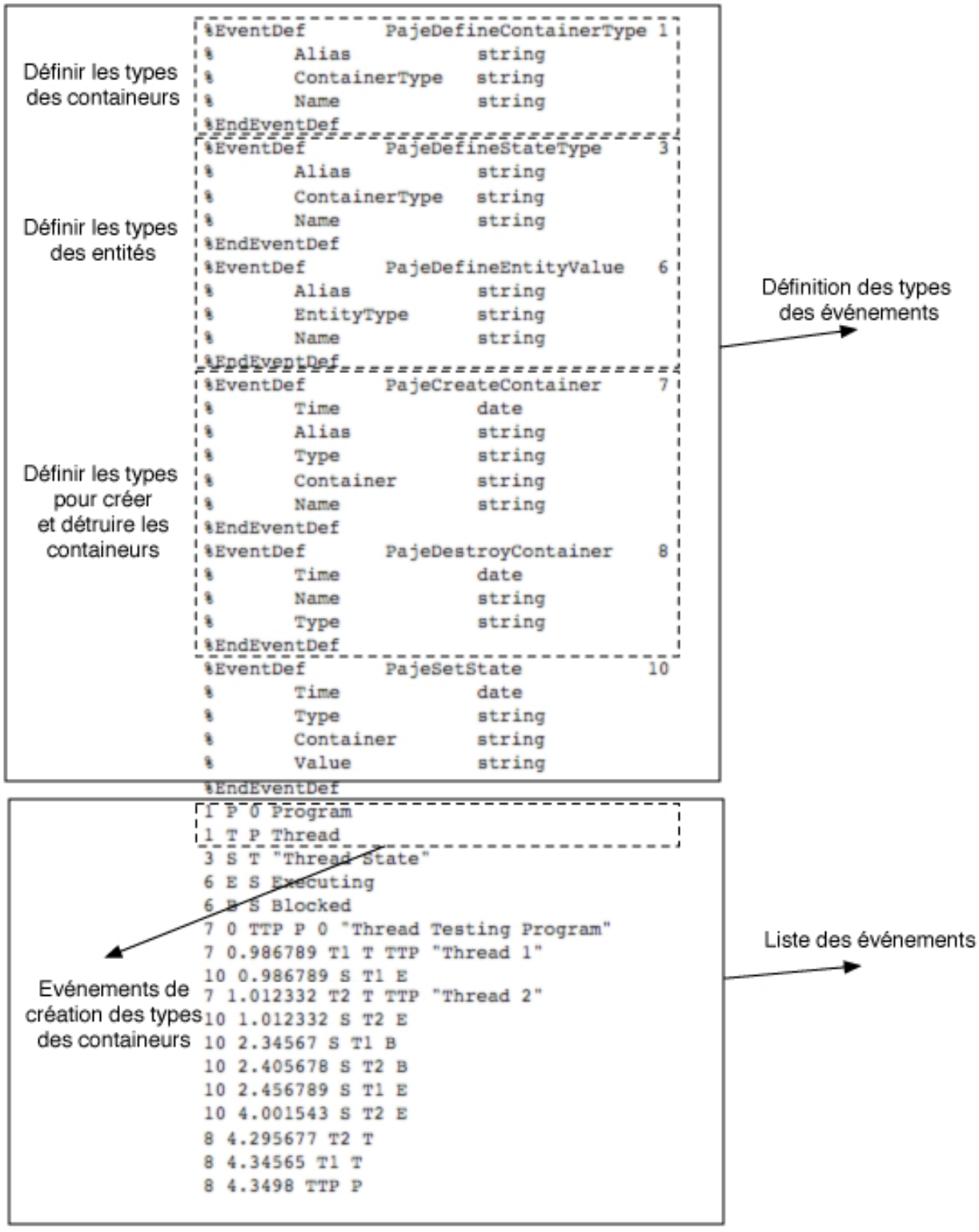

FIG. 5.4 – *Exemple simple d'un fichier de traces*

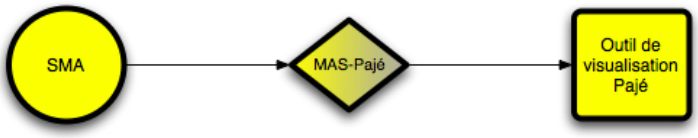

FIG. 5.5 – *Positionnement de MAS-Paje*

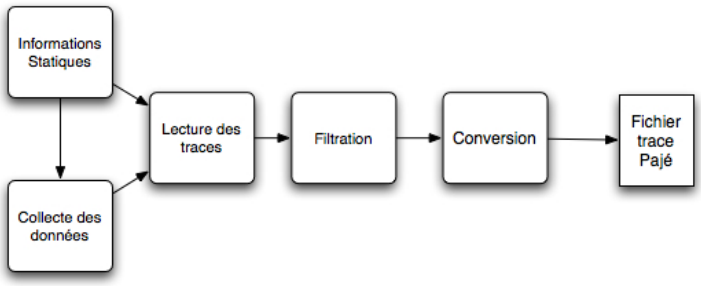

FIG. 5.6 – *Fonctionnalités et tâches effectuées par MAS-Paje*

de MAS-Paje (en général le concepteur du système ou l'évaluateur) concernant les données à visualiser (Section 3.2). Quelles sont les informations statiques ?

Dans le chapitre 3, nous avons validé le fait que pour un SMA, la visualisation se fait au niveau des comportements des agents (état des agents, communications entre les agents, etc.) à condition que ces systèmes soient capables de décrire tous leurs états. Les données à visualiser et les rôles des agents à étudier varient d'un système à un autre (différents rôles des agents, ensemble d'états pour chaque type des agents, différents types des messages, etc.) selon le problème et le domaine d'application. Cette variation affecte les informations et les données à collecter durant l'étape de collecte de données. Les informations nécessaires pour la collecte de données sont désignées par les informations statiques d'un SMA.

Les autres modules nécessaires pour MAS-Paje sont :

- La **lecture des traces**: le rôle de ce composant est de lire les traces collectées afin de construire un fichier contenant ces traces;
- La **filtrage**, syntaxique à ce niveau contrairement à la filtrage sémantique réalisée au niveau de Paje. Cette filtrage a pour rôle d'éliminer les informations inutiles (redondances, informations parasites) pour la visualisation, de distinguer les informations obtenues selon leurs types (communications, états, etc.) et de les ordonner selon la date de leurs occurrences. Pourquoi distinguer les informations obtenues selon leurs types est discuté dans la suite.
- La **conversion** en fichier de trace conforme au format de Paje.

Après collecte des informations sur les traces, ces informations sont reçues avec les informations statiques par un composant de lecture. Son rôle est d'assurer la cohérence entre les informations statiques et les données collectées. Parmi les informations statiques données par l'utilisateur, nous avons la définition des rôles que peut jouer les agents. Le composant de lecture associe les noms des agents effectifs aux rôles des agents (selon les états spécifiés aux agents par les informations statiques). Par agents *effectifs*, nous désignons les instances des rôles des agents. Un agent effectif est une instance d'un rôle, parmi les rôles définis dans les informations statiques.

Ensuite, les informations sont filtrées pour supprimer les redondances, distinguer entre les différents types d'informations et organiser les événements selon la date de leurs occurrences. Le filtrage au niveau de MAS-Paje est un filtrage syntaxique. Le filtrage au niveau de Paje, déjà présenté, se fait lui au niveau sémantique: il permet de sélectionner des informations selon leurs sémantiques et le niveau d'abstraction demandé par l'utilisateur. Finalement, le composant de conversion assure l'obtention d'un fichier de traces qui soit conforme au format de Paje.

Dans ce qui suit, nous présentons en détails ces étapes pour proposer l'architecture de MAS-Paje :

Informations statiques

Les informations nécessaires pour la collecte des données vont être listées selon les 4 dimensions de l'approche VOYELLES [Demazeau (1995)]:

Agents(A) :

- Les rôles des agents selon la mission attribuée;
- Les différents comportements que peut subir un agent durant l'exécution du système et les transitions possibles entre ces différents comportements (un graphe d'états finis);
- Les variables marquant l'exécution des agents.

Environnement(E) : au niveau de l'environnement, les états sont connus (2 états: répondre aux requêtes et attendre) et les transitions entre les états sont fixes. Nous n'avons donc pas besoin d'informations supplémentaires concernant les états comportementaux et les transitions entre ces états. Les informations statiques nécessaires sont les informations concernant les:

- Variables représentant les objets passifs manipulées par les agents dans l'environnement.

Interaction(I): A ce niveau, les informations statiques nécessaires sont les différents types de messages possibles entre les différents types d'agents d'un SMA. Des améliorations à ce niveau seront présentées dans le chapitre suivant, qui traite en détail le cas des communications.

Organisation(O): Au niveau de l'organisation, nous présentons les relations statiques entre les composants d'un SMA. La hiérarchie de l'organisation présentée respecte la hiérarchie imposée par Paje. Cette organisation a été présentée dans la figure 3.5 du chapitre 3, tandis qu'une présentation de l'organisation du système proies/prédateurs est présentée dans la partie 3.3.

Collecte des données

La collecte de données se fait par ajout des points d'instrumentation dans le code principal de l'application. Ces points d'instrumentation se font au niveau des entrées et des sorties des états. Les entrées et les sorties des états sont des instructions caractérisant le changement de l'état comportemental de l'agent, l'envoi et la réception des messages (par exemple, l'envoi d'un message de confirmation de prise en charge d'une mission, est une instruction de début d'un état comportemental), les événements et les variables. MAS-Paje se charge ensuite de l'ajout des informations nécessaires pour collecter les informations. Au niveau de la collecte de données, il faut traiter les problèmes suivants :

- Problème de l'horloge;
- Problème de la quantité des informations collectées.

Horloge : Dans notre architecture nous proposons une solution logicielle pour l'horloge en essayant d'éviter les changements sur l'architecture du système étudié, dans ce travail. Des algorithmes de corrections peuvent être ensuite utilisés pour la correction des traces [Maillet (1990)] .

L'horloge globale est considérée dans notre cas comme un objet de l'environnement qui envoie d'une façon périodique des données de synchronisation aux horloges locales des composants. Ce choix est fait pour des différents raisons :

- Pour ne pas changer l'architecture du système étudié, ajouter un objet à l'environnement ne change pas l'architecture du système;
- L'environnement où les agents sont situés est créé avant tous les agents. Ce qui est nécessaire pour le lancement de l'horloge et que l'horloge soit un objet dans l'environnement.

Quantité de données : Le problème de la quantité de données collectée est traité au niveau de composant de filtrage qui filtre les informations inutiles pour la visualisation demandée. L'architecture détaillée de MAS-Paje contribue à la reduction de la quantité de données en distinguant différents types des fichiers résultants du processus, selon les informations demandées par l'utilisateur. Cela évite d'avoir un seul fichier contenant toutes les informations, dans le cas où l'évaluateur s'intéresse à une information précise parmi les informations collectées. Dans ce cas, il peut se servir des données désignées sans garder toutes les informations inutiles.

5.3.2 MAS-Paje : architecture

Dans cette partie, nous présentons une architecture détaillée de MAS-Paje qui prend en compte les tâches présentées ci-dessus et traite en détails ces différentes tâches. Pour chaque composant de l'architecture de MAS-Paje, les tâches réalisées, visualisation d'un SMA et génération d'un fichier de trace compatible avec le format de Paje, sont illustrées sur l'application proies/prédateurs.

Le fichier de traces de Paje est organisé en 2 parties :

- La partie de définition des types des "events";
- La partie qui contient les occurrences des "events" constituant les traces.

La distinction entre partie définition (statique) et partie occurrence des "events" (dynamique) ainsi que la nécessité d'ajout d'un composant qui traite des informations statiques (figure 5.6) propres aux SMA, nous ont fait penser à reproduire cette séparation au niveau de l'architecture de MAS-Paje. L'architecture de système de visualisation MAS-Paje est ainsi composée de deux sous-systèmes "MAS-Paje Statique" et "MAS-Paje Dynamique". Ces deux sous-systèmes ne s'exécutent pas séparément. Donc, en plus de ces sous-systèmes, nous désignons par "MAS-Paje Liaison" le sous-système qui assure la liaison entre "MAS-Paje Statique" et "MAS-Paje Dynamique" durant le processus de visualisation. Dans ce qui suit, les trois sous-systèmes et leurs fonctionnalités sont décrits.

MAS-Paje Statique

Le but de MAS-Paje statique est de définir la partie statique du fichier de trace. Les fonctionnalités de la partie statique de MAS-Paje sont présentées dans la figure 5.7. La première tâche à réaliser consiste à ajouter l'en-tête du fichier des traces. L'entête contient les définitions des types des "events" qui

sert à créer, détruire et enregistrer les changements (états, événements, communications, variables) sur les conteneurs et les entités. Ensuite, les différents rôles des agents qui sont reçus de l'utilisateur sont enregistrés et déclarés dans le fichier de trace. Puis, les conteneurs sont créés selon une hiérarchie conforme à l'organisation présentée dans la partie information statique de cette section. Les états associés aux agents selon leurs types sont ensuite enregistrés et déclarés successivement dans le fichier de trace. Puis, toutes les transitions entre les états sont définies comme des entités de type "événement". Finalement, un type de message est déclaré pour chaque relation entre deux types des agents. Une description du rôle de MAS-Paje Statique sur l'application Proies/Prédateurs est détaillée par la suite.

Les entrées de MAS-Paje statique : informations statiques de l'application

Les informations statiques nécessaires pour la visualisation de cette application sont :

- Les rôles des agents : deux rôles d'agents existent dans cette application, les agents ayant le rôle "proies" et ceux ayant le rôle "prédateurs";
- Les états de chaque type des agents : les diagrammes à états finis de ces deux agents : ces diagrammes sont présentés dans les figures 5.8 et 5.9. Ces diagrammes montrent les états comportementaux possibles de ces agents;
- Les événements possibles;
- Les types de messages possibles.

Nous ne traitons pas les types des communications dans ce chapitre. Nous laissons ce traitement pour le chapitre suivant. Nous distinguons dans cette partie les types des communications selon leurs émetteurs et leurs récepteurs uniquement. Les communications distinguées sont ici de type proie - prédateur, prédateur - proie, prédateur - environnement, environnement - prédateur, proie - environnement et environnement - proie.

Les relations entre les différentes parties du système sont présentées dans le diagramme d'organisation de la figure 5.10. L'organisation statique qui y est présentée respecte les relations existantes dans le système en garantissant la structure hiérarchique imposée par Paje.

Informations délivrées par MAS-Paje statique

Le traitement au niveau de MAS-Paje Statique délivre la partie contenant : l'entête, les définitions des conteneurs et les "events" de leurs créations. Nous présentons les résultats dans ce qui suit :

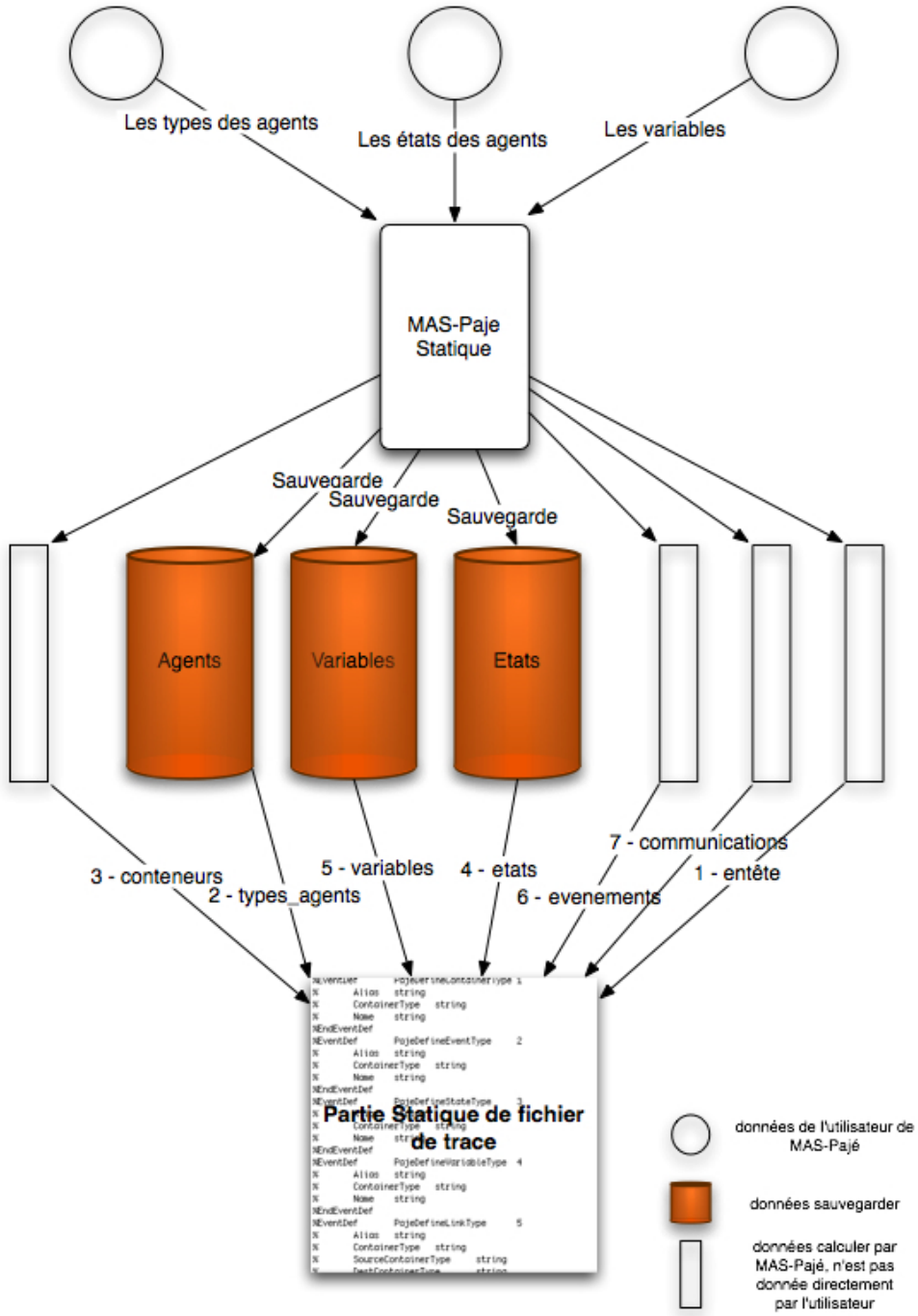

FIG. 5.7 – *Fonctionalités et tâches réalisées par MAS-Paje Statique*

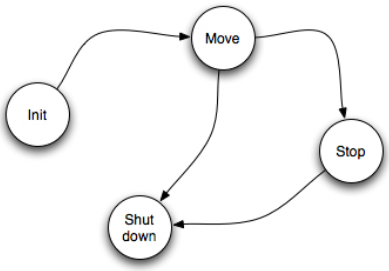

FIG. 5.8 – *États comportementaux des proies*

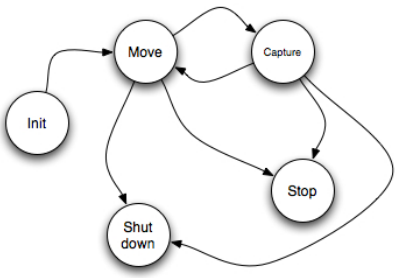

FIG. 5.9 – *États comportementaux des prédateurs*

L'en-tête

L'en-tête contient la définition des types des "events". L'en-tête standard pour les fichiers de trace de format Paje est présentée dans le partie 5.2. Dans cette partie, nous nous servons des définitions des "events" présentées et contenues dans l'entête d'un fichier pour expliquer les occurrences des événements.

Conteneurs

Pour l'application proies-prédateurs les types des conteneurs sont (en se référant sur la figure 5.10) :

- Le conteneur représentant le SMA
- Le conteneur représentant l'ensemble des entités
- Les conteneurs représentant les 2 ensembles d'agents proies et prédateurs ("Preys" et "Predators")
- Les deux conteneurs servant de type pour les agents effectifs ("Prey" et "Predator")

Dans ce qui suit nous présentons les événements de déclaration de ces conteneurs engendrées, en listant les définitions des types d'événements suivies par ces événements :

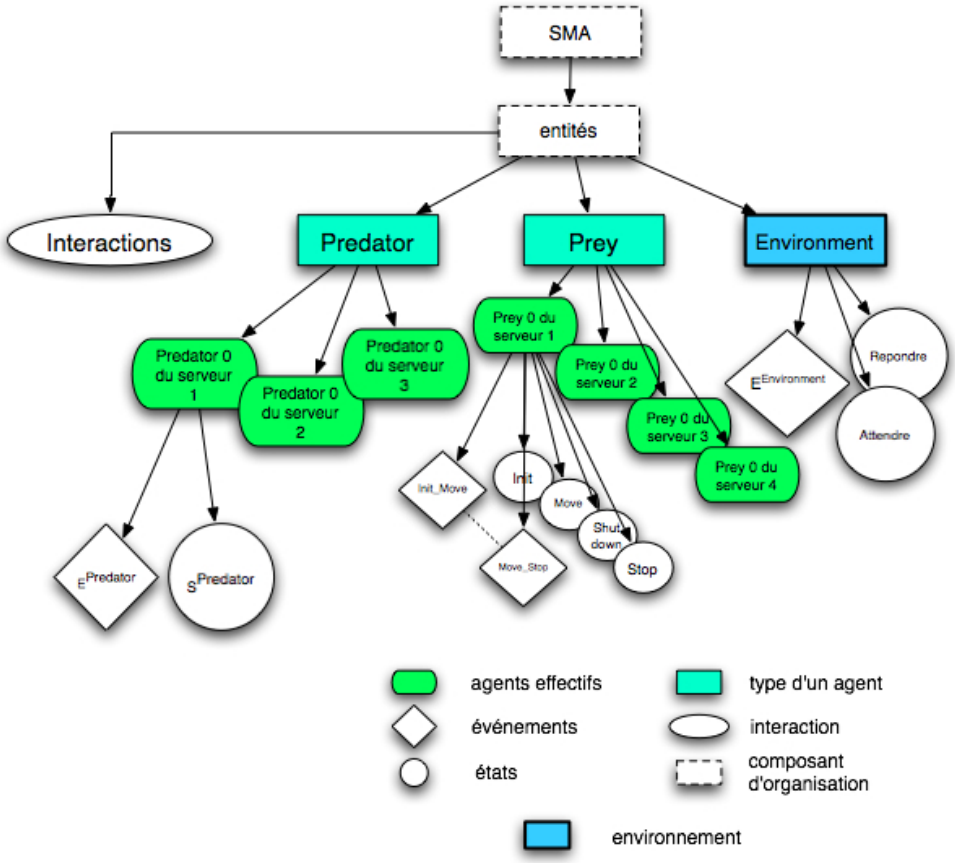

FIG. 5.10 – *Diagramme d'organisation du système proies/prédateurs*

Pour définir un conteneur, la déclaration de type d'"event" est la suivante :

```
\%EventDef  PajeDefineContainerType 1
\%  Alias     string
\%  ContainerType    string
\%  Name      string
\%EndEventDef
```

Pour créer un conteneur :

```
\%EventDef  PajeCreateContainer 7
\%  Time      date
\%  Alias     string
\%  Type      string
\%  Container    string
\%  Name      string
\%EndEventDef
```

L'ensemble des "events" de définition des types des conteneurs et leurs créations est alors la suivante :

```
1 MAS 0 "Multi-Agent System"
1 Ags MAS "Agents"
1 Pys Ags "Preys"
1 Pts Ags "Predators"
1 En Ags "Environment"
1 Py Pys "Prey"
1 Pt Pts "Predator"
7 0 MASC MAS 0 "Multi-Agent System Container"
7 0 AgsC Ags MASC "Agents Container"
7 0 PysC Pys AgsC "Preys Container"
7 0 PtsC Pts AgsC "Predators Container"
7 0 EnC En AgsC "Environment Container"
```

A la première ligne, nous avons l'"event" de définition de conteneur "Multi-Agent System". Cet "event" est de type PajeDefineContainerType (identifiant 1) avec l'alias MAS (l'alias est l'identifiant du conteneur créé). Le 0 fait référence au fait qu'il n'y a pas de précédence de ce conteneur dans la hiérarchie du diagramme d'organisation. Le nom de ce conteneur est "Multi-Agent System".

La création des conteneurs représentant les événements pour les agents et l'environnement s'effectue selon la définition du type d'"events" suivant :

```
\%EventDef  PajeDefineEventType 2
\%  Alias    string
\%  ContainerType    string
\%  Name    string
\%EndEventDef
```

Les "events" de création des types d'événements engendrées pour l'application particulière "proies-prédateurs" sont:

```
2 MPy Py "Prey Events"
2 MPt Pt "Predator Events"
2 MEn En "Environment Events"
```

La déclaration des valeurs possibles d'événements suivant la définition du type d'"events" suivants:

```
\%EventDef  PajeDefineEntityValue    6
\%  Alias    string
\%  EntityType   string
\%  Name    string
\%  Color    color
\%EndEventDef
```

Les "events" de déclaration qui correspondent à la déclaration des valeurs possibles d'événements sont:

```
6 InitPyE MPy "Initialisation Prey Event" ".9 .7 .9"
6 MovePyE MPy "Move Prey Event" ".0 .0 .8"
6 StopPyE MPy "Stop Prey Event" ".0 .0 .3"
6 InitPtE MPt "Initialisation Predator Event" ".0 .7 .7"
6 EssaiePtE MPt "Essaie Predator Event" ".7 .7 .7"
6 MovePtE MPt "Move Predator Event" ".3 .7 .7"
6 StopPtE MPt "Stop Predator Event" ".4 .7 .4"
6 WaiEnE MEn "Wait Environment Event" ".0 .9 .9"
6 ShaEnE MEn "Share plans Event" ".9 .9 .9"
```

La création des conteneurs des états des agents et de l'environnement suit la définition du type d'"events" suivant:

```
\%EventDef  PajeDefineStateType 3
\%  Alias    string
\%  ContainerType    string
\%  Name    string
\%EndEventDef
```

Les "events" de création de états de proies et de prédateurs sont les suivants :

```
3 SPy Py "Prey states"
3 SPt Pt "Predator states"
3 SEn En "Environment states"
```

Les "events" definissant les valeurs possibles des états sont :

```
6 InitPy SPy "Initialisation Prey" ".0 .8 .2"
6 MovePy SPy "Move Prey" ".0 .1 .2"
6 StopPy SPy "Stop Prey" ".1 .1 .1"
6 InitPt SPt "Initialisation Predator" ".8 .8 .2"
6 EssaiePt SPt "Try Predator" ".9 .9 .2"
6 MovePt SPt "Move Predator" ".0 .7 .4"
6 StopPt SPt "Stop Predator" ".2 .2 .4"
6 WaiEn SEn "Wait Environment" "1 .2 .4"
6 ShaEn SEn "Respond Environment" "1 .3 .4"
```

La création des conteneurs représentant les communications entre les agents et entre les agents et l'environnement suit la définition du type d'"events" suivant :

```
\%EventDef PajeDefineLinkType 5
\% Alias string
\% ContainerType string
\% SourceContainerType string
\% DestContainerType string
\% Name string
\%EndEventDef
```

Les "events" qui créent les conteneurs des communications sont :

```
5 CPyPt Ags Py Pt "Prey Predator Communication"
5 CPtPy Ags Pt Py "Predator Prey Communication"
5 CEnPy Ags En Py "Environment Prey Communication"
5 CPyEn Ags Py En "Prey Environment Communication"
5 CPtEn Ags Pt En "Predator Environment Communication"
5 CEnPt Ags En Pt "Environment Predator Communication"
```

Finalement, Les valeurs possibles des différentes valeurs de communication sont définies dans la suite :

```
6 sentPyPt CPyPt "sentPyPt" ".0 .8 .2"
6 sentPtPy CPtPy "sentPtPy" ".0 .8 .2"
6 sentPyEn CPyEn "sentPyEn" ".0 .8 .2"
```

```
6 sentEnPy CEnPy "sentEnPy" ".0 .8 .2"
6 sentPtEn CPtEn "sentPtEn" ".0 .8 .2"
6 sentEnPt CEnPt "sentEnPt" ".0 .8 .2"
```

MAS-Paje Liaison

Le rôle principal de MAS-Paje Liaison est d'enregistrer les identifiants des agents effectifs pris durant l'exécution du système et de les différencier selon leurs rôles déclarés dans les informations statiques obtenus au début de processus de visualisation, plusieurs agents effectifs peuvent jouer le même rôle dans l'exécution obtenus dans les informations statiques. Le rôle de MAS-Paje statique est ainsi de faire la liaison entre les traces collectées par MAS-Paje Dynamique et les déclarations statiques réalisées par MAS-Paje Statique. Le Comportement de MAS-Paje est représenté dans la figure 5.11.

Le traitement au niveau de MAS-Paje Liaison délivre la partie contenant la création des conteneurs des agents effectifs. Cette partie est présentée dans ce qui suit :

```
7 0 run1predator0 Pt PtsC "Predator 0 of server 1"
7 0 run2predator0 Pt PtsC "Predator 0 of server 2"
7 0 run3predator0 Pt PtsC "Predator 0 of server 3"
7 0 run1prey0 Py PysC "Prey 0 of server 1"
7 0 run1prey1 Py PysC "Prey 1 of server 1"
7 0 run2prey0 Py PysC "Prey 0 of server 2"
7 0 run3prey0 Py PysC "Prey 0 of server 3"
```

MAS-Paje Dynamique

Les fonctionnalités de MAS-Paje dynamique sont décrites dans la figure 5.12. Le scénario commence par un événement reçu par MAS-Paje Dynamique de la partie collecte des données. Un événement est de la forme : temps, agents effectifs et action. Ces informations seront communiquées à MAS-Paje Liaison qui vérifie si l'agent effectif existe dans sa base d'agents effectifs. Sinon, il l'ajoute après vérification de son rôle selon l'action effectuée. Des vérifications sont faites pour valider l'événement en se fondant sur les informations statiques. Après validation, l'événement est inséré dans un fichier selon le type de l'action contenu (communication, changement d'états, évents, variables). Une telle distinction entre les différents types des fichiers est faite pour :

- Permettre des évaluations plus détaillées sur différents types pris séparément (le cas de communication est présenté dans le chapitre suivant)

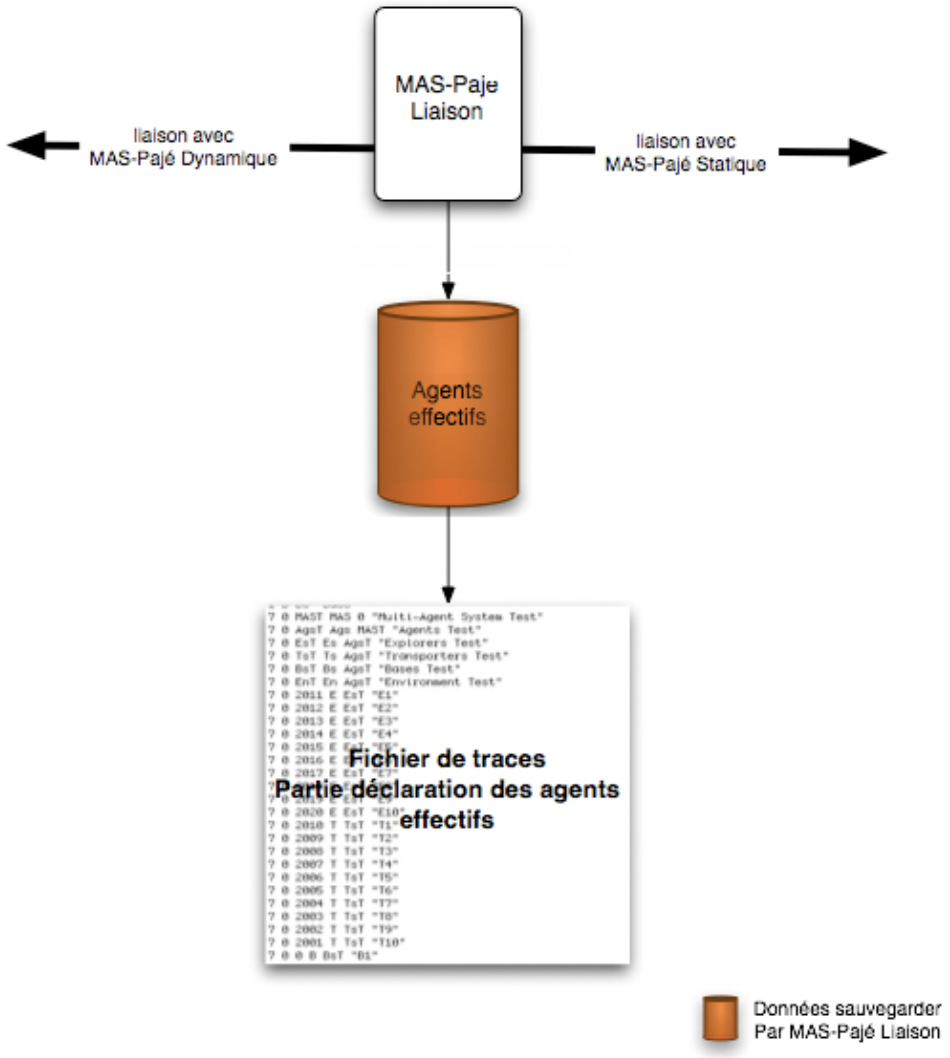

FIG. 5.11 – *Fonctionnalités et tâches réalisées par MAS-Paje Liaison*

– Diminuer la quantité d'informations collectées pour un évaluateur non intéressé par toutes ces informations.

Ces fichiers sont ensuite assemblés dans un seul fichier et filtrés pour éliminer les redondances et les entrées inutiles et organiser les événements selon l'ordre temporel synchronisé. Finalement, le fichier est converti au format de Paje des fichiers des traces (en ajoutant les parties manquantes et en assurant la conformité du format).

La partie dynamique de MAS-Paje délivre le traçage d'"events" selon la définition du type d'événements suivante :

```
\%EventDef  PajeNewEvent     9
\%  Time     date
\%  EntityType  string
\%  Container   string
\%  Value   string
\%EndEventDef
```

Le traçage des états (changement des valeurs des états) suit, lui, la définition du type d'événement suivante :

```
\%EventDef  PajeSetState     10
\%  Time     date
\%  EntityType  string
\%  Container   string
\%  Value   string
\%EndEventDef
```

Le traçage des communications suit les définitions du type d'événements suivantes, la première (16) pour déclarer le début d'un lien de communication et la seconde (17) pour déclarer la fin d'un lien de communication :

```
\%EventDef  PajeStartLink    16
\%  Time     date
\%  EntityType  string
\%  Container   string
\%  Value   string
\%  SourceContainer string
\%  Key string
\%  Size    int
\%EndEventDef

\%EventDef  PajeEndLink 17
\%  Time     date
```

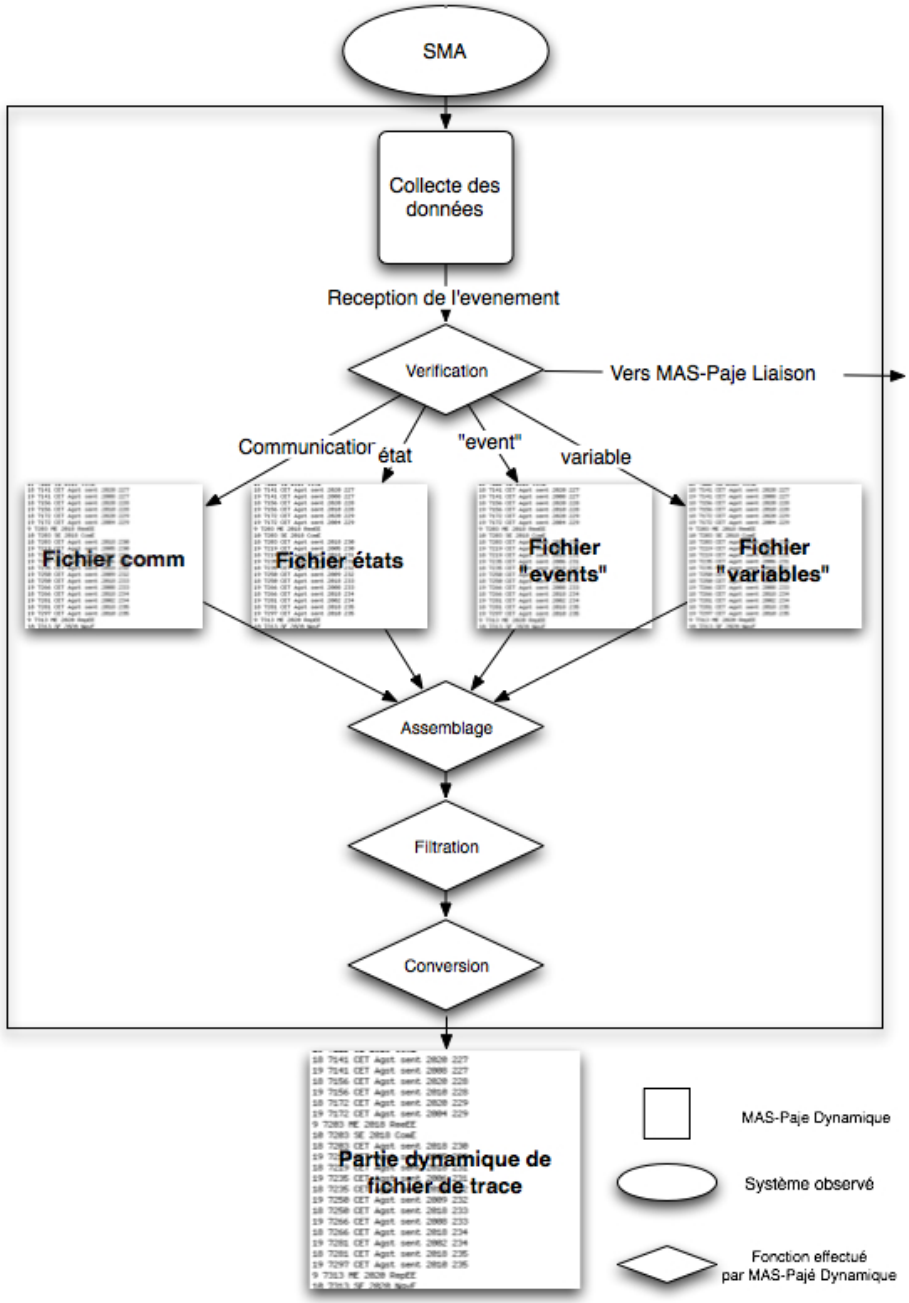

FIG. 5.12 – *Fonctionnalités et tâches réalisées par MAS-Paje Dynamique*

```
\% EntityType string
\% Container string
\% Value string
\% DestContainer string
\% Key string
\% Size int
\%EndEventDef
```

Nous présentons dans la suite un exemple des événements tracées contenant :

- Un changement d'état;
- Une événement;
- Un lien de communication.

```
9 32 MPt run1predator0 InitPtE
10 32 SPt run1predator0 InitPt
16 35 CPtEn AgsC sentPtEn run1predator0 32 10
17 35 CPtEn AgsC sentPtEn EnC 32 10
```

En conclusion de ce chapitre, nous avons proposé une architecture d'un système de visualisation pour les SMA pouvant être décrits par des graphes d'états. Cette architecture est importante du fait qu'elle propose un processus de réalisation de visualisation des SMA par Paje. Une évaluation de ce travail à ces niveaux techniques, théoriques et les choix faites est réalisée dans le chapitre 9.

Chapitre 6

La mise en oeuvre de la solution proposée pour évaluer la communication

Une formalisation de la communication dans les SMA est proposée dans le chapitre 4. Différentes approches sont mentionnées pour assurer la comparaison des communications dans des différentes implémentations, d'un SMA, résolvant le même problème. Les mesures et les analyses faites sur les communications dans les SMA constituent une dimension supplémentaire à la visualisation pour la compréhension et l'évaluation du comportement des entités dans un SMA. La partie formelle du travail, ayant été réalisée sachant les aspects techniques de MAS-Paje présentés dans le chapitre précédent, nous montrons ici comment mettre en oeuvre les solutions proposées. Cette partie technique a donc pour but de :

- Proposer une architecture d'un système d'évaluation des communications en se fondant sur les approches présentées dans le chapitre 4 ;
- Se servir de la solution technique proposée pour la visualisation de l'exécution des SMA dans le chapitre 5.

Dans ce chapitre, nous présentons les étapes de la réalisation de la plate-forme d'évaluation de la communication dans un SMA dont les états peuvent être listés. Cette plate-forme est un raffinement de MAS-Paje. Nous nous appuyons sur les informations collectées par MAS-Paje et obtenues sous forme d'un fichier (sous le format SDDF accepté par Paje) et nous étendons les modules de MAS-Paje pour obtenir et traiter des informations supplémentaires (les informations concernant l'effet de la communication sur l'agent et les conversations dans les SMA). Puis, nous traitons le cas de l'évaluation des conversations.

6.1 Plate-forme d'évaluation de la communication

Dans cette partie, nous traitons du problème de la mise en oeuvre de la plate-forme de l'évaluation de la communication, comme extension des travaux faits au niveau de MAS-Paje dans le chapitre précédent. Nous précisons les tâches à réaliser par la plate-forme, assurant une évaluation de communication selon les approches : approche classique, approche type, approche poids et approche conversation en se fondant sur l'étude théorique du chapitre 4. Des traitements supplémentaires dans le processus de visualisation sont nécessaires. Ces traitements se situent au niveau des étapes de la collecte des données, du traitement des données et de l'analyse de ces données.

Approche classique

La première approche traite la quantité de communications. Les informations obtenues dans le fichier communication sont suffisantes pour réaliser la tâche de l'évaluation selon cette approche, sans nécessité d'informations supplémentaires. Il suffit de sortir les traces de tous les messages envoyés et reçus par un agent pour savoir la quantité de messages échangées.

Dans le modèle de communication présenté dans le chapitre 4 :

$$Communication : IU^{A}_{Transmit} \rightarrow IU^{B}_{Received} \cup \{m^{B}_{\phi}\}$$

La quantité d'informations reçues par l'agent A est :

Q=NB(M) / $\exists$ B et $M : IU^{A}_{Transmit} \rightarrow IU_{Received}$

La transmission d'un message dans le format de Paje suit la déclaration du type d'"event" :

%EventDef PajeStartLink 16
% Time date
% EntityType string
% Container string
% Value string
% SourceContainer string
% Key string
% Size int
%EndEventDef

Et la réception suit la déclaration de type d'"event" :

%EventDef PajeEndLink 17

% Time date

% EntityType string

% Container string

% Value string

% DestContainer string

% Key string

% Size int

%EndEventDef

L'envoi d'un message M de l'agent A à l'agent B est représenté dans les traces engendrées par MAS-Paje Dynamique de la façon suivante :

16 t CommAB AgsC typeM A idc 10

17 t1 CommAB AgsC typeM B idc 10

Avec :

- 16 : L'identifiant de l'événement d'envoi ;
- 17 : L'identifiant de l'événement de réception ;
- t : La date d'envoi ;
- t1 : La date de réception ;
- CommAB : La classe de la communication (ici, une communication entre les deux types d'agents A et B) ;
- typeM : La valeur de la communication (plusieurs types de communications peuvent avoir lieu entre les deux agents A et B).
- idc : L'identifiant de la communication. (son rôle est de faire la relation entre l'envoi et la réception) ;
- 10 : La taille de la flêche dans la fenêtre de visualisation.

En se fondant sur les "events" présentés ci-dessus et engendrés par MAS-Paje, la plate-forme d'évaluation de la quantité de l'information est présentée dans la figure 1.1. Cette plate-forme prend le fichier des communications engendrées par MAS-Paje dynamique et compte le nombre des messages reçus par chaque agent (événements de réception numéro 17) selon les champs numéro 6 de l'événement qui contient l'identifiant de l'agent effectif qui a reçu le message. La sortie de cette plate-forme est un ensemble des fichiers associés aux agents effectifs. Dans chaque fichier, nous avons un tableau de deux champs. Dans le premier champ, nous avons le temps de réception. Dans le

deuxième, la quantité des messages reçus (exemple d'un graphe représentant les quantités obtenus dans la partie 4.3 du chapitre 3).

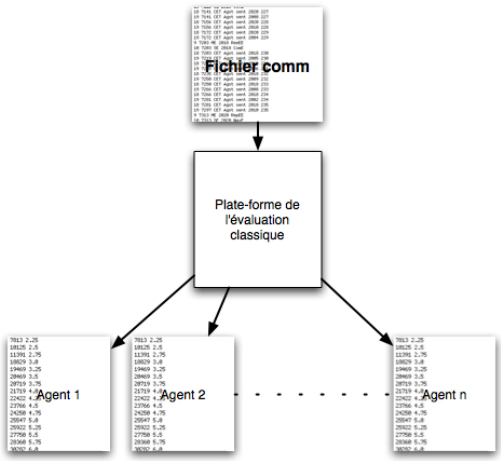

Fig. 6.1 – *Entrées et sorties de la plate-forme de l'évaluation classique des communications*

Approche type

L'approche type consiste à distinguer entre les différents types de messages. A ce niveau, les informations sur l'émetteur et le récepteur d'un message ne sont pas suffisantes. Il faut de plus analyser les types des messages envoyés et reçus par les agents. Un poids est affecté à chaque type des messages.

Pour déclarer une communication entre deux types des agents A et B, la déclaration de type d'"event" est ainsi la suivante :

%EventDef PajeDefineLinkType 5

% Alias string

% ContainerType string

% SourceContainerType string

% DestContainerType string

% Name string

%EndEventDef

Tandis que la communication est crée de la façon suivante :

5 CAB Ags A B "Comm A B"

Une communication peut prendre plusieurs valeurs. Ici, ces valeurs correspondent aux différents types des messages échangées entre deux types d'agents.

Une déclaration d'une valeur de communication suit la définition de type d'"event" suivante :

%EventDef PajeDefineEntityValue 6

% Alias string

% EntityType string

% Name string

% Color color

%EndEventDef

Par exemple, si on a deux types de messages T1 et T2 qui peuvent être échangés entre A et B alors les "events" représentant la situation sont :

6 T1 ComAB "Type communication 1" ".9 .7 .9"

6 T2 CommAB "Type communication 2" ".0 .0 .8"

Une communication de type T1 suit la déclaration de type d'"event" avec les identifiants 16 et 17 présentés dans la première partie. Donc, une communication entre deux agents effectifs A et B suivant le type T1 est représentée par :

16 t CommAB AgsC T1 A idc 10

17 t1 CommAB AgsC T1 B idc 10

En se fondant sur les "events" présentés, la plate-forme d'évaluation (figure 6.2) de la communication selon l'approche type de messages prend en entrée :

- Le fichier de communications engendré par MAS-Paje ;
- Un fichier contenant un tableau affectant un poids à chaque type des messages. Ce poids est un valeur entre 0 et 1. La valeur 0 est affectée à un type de messages non pertinent.

La plate-forme engendre ensuite des fichiers contenant l'évolution du poids des communications en fonction du temps pour chaque agent. Une représentation des informations engendrées par la plate-forme est décrite dans la section 3 du chapitre 4.

Approche poids des messages

L'évaluation de l'approche quantité des informations collectées peut aussi se faire au niveau du fichier de traces. Pour être capable de réaliser une telle évaluation, nous ajoutons pour chaque agent un certain nombre d'informations à collecter par MAS-Paje.

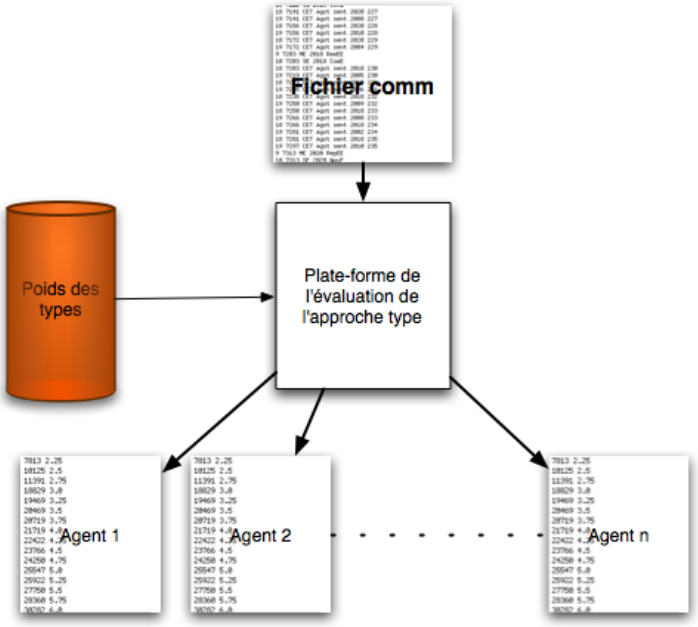

Fig. 6.2 – *Entrées et sorties de la plate-forme de l'évaluation de l'approche type*

A chaque réception d'un message, un certain nombre de caractéristiques représentant les changements possibles de l'état mental de l'agent sont représentés par des variables. Chaque caractéristique peut avoir plusieurs valeurs. L'évaluation du changement de l'état mental de l'agent suite à la réception des messages se fait par l'intermédiaire des valeurs de ces caractéristiques.

Pour la décision, nous affectons une valeur pour chaque action déclenchée par l'agent suite à la réception d'un message.

Pour chaque caractéristique de mémoire et pour l'action nous définissons une variable en se référant sur la déclaration des types de données suivante :

%EventDef PajeDefineVariableType 4

% Alias string

% ContainerType string

% Name string

%EndEventDef

Les valeurs pour chaque caractéristique sont définies donc avec la déclaration PajeDefineEntityValue présentée ci-dessus pour l'approche type.

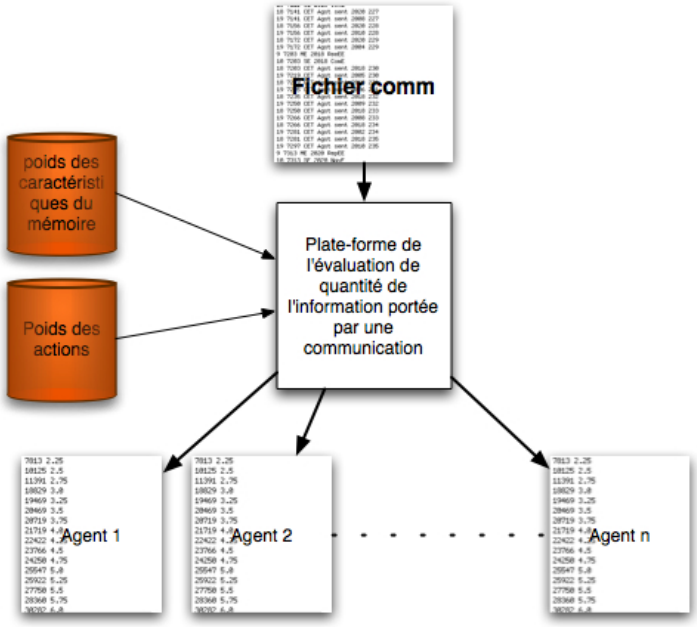

FIG. 6.3 – *Entrées et sorties de la plate-forme dévaluation de l'approche poids*

Les entrées de la plate-forme d'évaluation (figure 8.10, selon la quantité des informations portées par les messages sont :

- Le fichier de communications engendré par MAS-Paje après l'ajout de variables de mémorisation et de l'action ;
- Un fichier contenant un tableau affectant un poids à chaque valeur d'une caractéristique ;
- Un fichier contenant un tableau affectant un poids à chaque action déclenchée.

La plate-forme engendre ensuite des fichiers contenant l'évolution de la quantité d'informations contenues dans les communications en fonction du temps pour chaque agent. Une représentation des informations engendrées par cette plate-forme est dans la partie 4.3.

Plate-forme d'évaluation des conversations

Dans les parties précédentes, la plate-forme d'évaluation délivre des fichiers associés aux agents. Ces fichiers contiennent la liste des messages envoyés et reçus par chaque agent effectif. Cette plate-forme reçoit de plus les définitions des protocoles de communications entre deux types des agents. Le protocole de communication entre deux agents est distingué parmi la liste des "events"

du fichier engendré par MAS-Paje, en suivant la séquence des communications conformes à ce protocole.

Les informations délivrées sont les informations sur l'occurrence des protocoles pour tous les agents et le nombre d'échecs et de réussites d'un protocole.

Dans l'implémentation de cette plate-forme nous avons rencontré le problème de reconnaissance des conversations à partir du fichier de trace. Ce problème est complexe et sera discuté dans le chapitre d'évaluation du travail.

Finalement, une plate-forme générale combinant les trois approches est présentée dans la figure 6.4 :

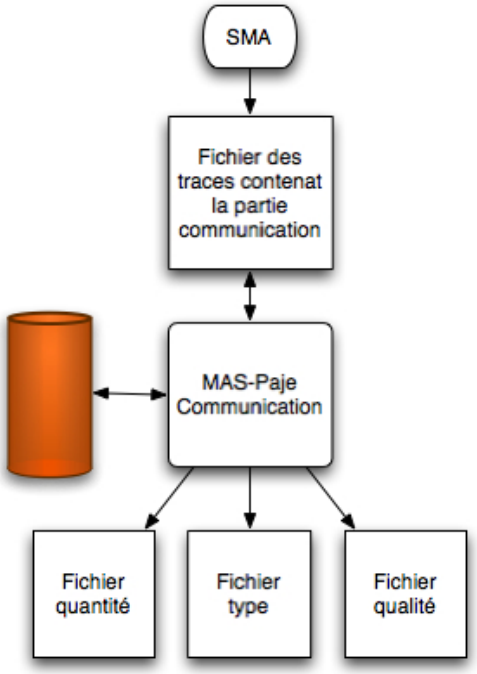

Fig. 6.4 – *Plate-forme d'évaluation de la communication*

Conclusion

Dans ce chapitre, nous avons montré l'applicabilité des solutions présentées pour modéliser et analyser la communication et les conversations dans un SMA. Nous avons proposé des plates-formes qui sont des raffinements de MAS-Paje pour réaliser l'analyse de ces caractéristiques. Ce travail a complété la tâche réalisée par MAS-Paje dans le chapitre précédent. En plus de l'architecture interne des agents, nous avons proposé l'analyse de certaines des caractéristiques représentant les liens (communications simples et conversations) entre

les agents. Ce travail nécessite d'être enrichi par l'étude et l'analyse des autres caractéristiques SMA [Demazeau (2004b)], utiles pour l'évaluation des SMA.

Chapitre 7

Expérimentation de MAS-Paje sur l'application de collecte de minerai

Nous partons dans ce travail d'une hypothèse initiale qu'une vision boîte noire est insuffisante pour analyser le comportement d'un SMA. Nous défendons l'idée principale que la visualisation des SMA est une solution qui facilite la compréhension du comportement interne des agents dans un SMA et par suite une évaluation du SMA complet.

Nous avons proposé dans les chapitres 3 et 5 un modèle formel de visualisation des SMA à base de graphes d'état et la réalisation d'une plate-forme de visualisation des SMA (MAS-Paje). Le but des chapitres 7 et 8 est d'illustrer et de valider l'idée principale défendue pour laquelle les parties techniques et théoriques ont été réalisées.

Dans le chapitre 7, nous présentons dans un premier temps l'utilité de la visualisation pour obtenir des informations sur les comportements des agents et l'utilisation des ressources par les différentes entités du SMA. Ces informations servent à détecter des problèmes dans l'architecture interne des SMA et à extraire des schémas significatifs de l'exécution. Dans un deuxième temps, nous utilisons les informations de visualisation pour différents SMA résolvant le même problème, en vue de pouvoir comparer ces systèmes entre eux.

Ce chapitre est constitué de quatre parties. La première partie présente les critères pour choisir l'application SMA à expérimenter, l'application de collecte de minerai choisie en fonction de ces critères, les arguments de choix de cette application et la présentation d'un premier système multi-agents résolvant le problème. Le protocole expérimental et le modèle de visualisation de cette application sont détaillés dans une deuxième partie. Dans la troisième partie

nous présentons les résultats de visualisation de l'exécution de ce premier système en analysant ces résultats. Finalement, nous présentons les visualisations sur deux autres systèmes multi-agents résolvant le même problème et nous proposons un comparatif des trois applications présentées.

7.1 Choix de l'application

7.1.1 Critères pour choisir le problème

Pour illustrer l'importance de la visualisation des SMA, le choix du problème doit reposer sur un certain nombre des critères. Parmi ces critères, nous listons:

- La simplicité du problème et l'imprévisibilité du comportement dans la résolution: le problème doit être simple à décrire pour servir à illustrer la visualisation mais il s'agit de ne pas passer trop de temps en le décrivant. La simplicité du problème ne doit pas empêcher l'imprévisibilité de comportement de la solution ;
- La présence de plusieurs types d'agents jouant des rôles différents dans la résolution du problème ;
- La présence de plusieurs agents de même type ;
- Le nombre limité d'états comportementaux possibles des agents pour répondre à la modélisation présentée dans le chapitre 3 ;
- La nature hybride (cognitive et réactive) des fonctionnalités des agents pour permettre d'étudier ces différents cas ;
- La facilité de caractériser les ressources consommées ;
- La présence de communications entre agents de même type et entre agents de types différents ;
- La présence de certaines caractéristiques SMA comme la coordination, la coopération et la compétition entre les agents.

7.1.2 L'application à expérimenter

Présentation de l'application de "collecte de minerai"

Le problème consiste à simuler une société de robots ayant pour but de collecter du minerai. Le scénario principal se déroule sur une planète connue comme ayant une certaine quantité de minerai, répartie sur la surface. Ce problème est divisé en deux sous-problèmes. Le premier sous-problème est de trouver le minerai distribué en certains points sur la surface de la planète, le deuxième est de collecter ce minerai, une fois localisé. Deux types de robots

sont utilisés, des explorateurs et des transporteurs. Les robots sont associés à une base d'où les robots partent initialement et où doit être déposé le minerai collecté. Dans la plupart des modélisations SMA de ce problème, ces robots correspondent à des agents.

Agents

Les explorateurs sont affectés à la recherche de minerai. Ils sont équipés d'un composant de perception de l'espace, limité à un certain rayon de perception. Les explorateurs peuvent trouver les coordonnées de minerai sur la surface de la planète, dans la limite de leurs rayons de perception.

Les transporteurs sont en mesure de transporter une quantité limitée de minerai et de la déposer dans une base. Une fois le minerai situé par les explorateurs, les transporteurs peuvent en obtenir les coordonnées exactes par communication. Ils se déplacent dans l'espace pour les collecter. Le travail des transporteurs est donc de ramasser le minerai détecté par les explorateurs.

La base constitue le centre de recharge d'énergie pour les robots. Il constitue en plus le centre de dépôt du minerai transporté par les robots transporteurs. La base est un intermédiaire permettant une certaine de communication indirecte entre les agents, même si la modélisation du problème permet des interactions en dehors de la base, grâce aux communications inter agents.

Communication

Les communications basique se déroulant entre les robots et/ou les bases sont possibles à l'intérieur d'un rayon de communication. Ce rayon de communication ne peut pas être illimité. Le seul moyen de contact direct entre les différentes entités du système s'effectue par l'intermédiaire de la communication.

La figure 7.1 présente plusieurs agents communiquants. Chaque agent est représenté avec son rayon de communication. Cette figure montre le transporteur T et l'explorateur E1 pouvant communiquer directement avec la base B. De même, l'explorateur E2 peut communiquer directement avec E3. Par contre, E3 ne peut communiquer que indirectement avec la base. Les informations reçues par un agent sont soit traitées par l'agent soit routées vers d'autres agents. Le routage n'est pas réalisé dans le sens "sender/receiver" mais plutôt dans le but de partager des informations entre les agents.

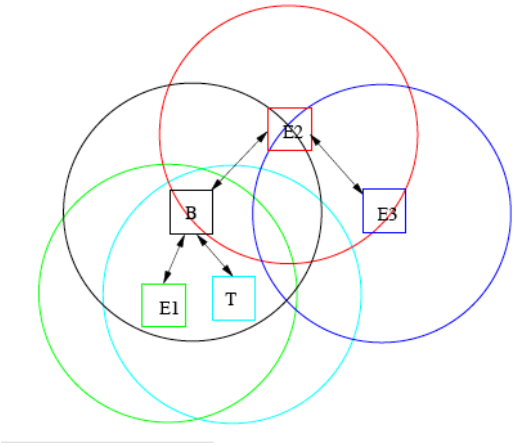

Fig. 7.1 – *Rayons de communication des agents*

Environnement

L'environnement global de cette application est un environnement statique, représenté par une grille contenant les minerais.

En général, l'environnement est représenté par :

- Une grille en 2 dimensions (figure 7.3) ;
- Les côtés opposés de la planète sont connectés, afin de simuler une planète en forme de tore ;
- Sur chaque position sur la grille, il ne peut y avoir qu'un seul agent ;

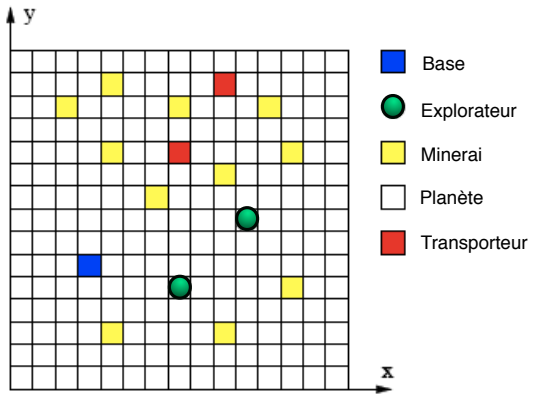

Fig. 7.2 – *Représentation de l'environnement dans l'application*

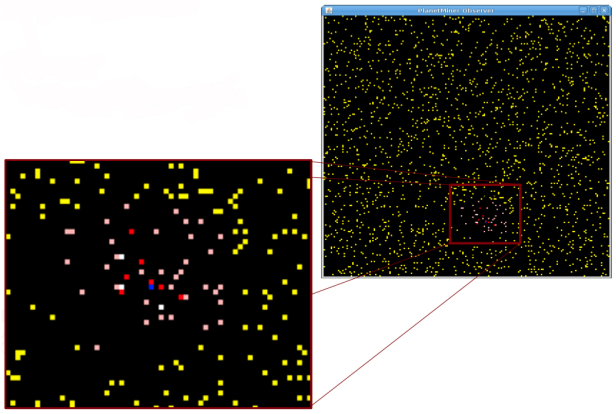

FIG. 7.3 – *Simulation de l'application de collecte de minerai*

Nous complétons ici la description de l'application par d'autres spécifications :

- Le minerai est distribué uniformement sur la surface de la planète avec une densité α ;
- Les batteries des robots sont limitées. Les robots ont besoin de les recharger à la base ;
- À chaque pas de temps, un agent peut effectuer au plus une action, comme l'envoi d'un message, un mouvement d'une case, une perception ou une collecte d'échantillons de minerai ;
- Chacune de ces actions a un coût en énergie ;
- Une communication consomme moins d'énergie qu'un mouvement ;
- La situation où deux bases se trouvent sur la même planète peut se produire. Dans ce cas, deux scénarios sont possibles : les bases coopérent dans la recherche de minerai ou au contraire sont en en compétition.
- Le scénario se termine lorsque toutes les bases atteignent leurs capacités maximales de minerai ou lorsqu'une limite de temps est dépassée (figure 7.4).
- Le but des bases est d'une part de minimiser la consommation d'énergie, et d'autre part de minimiser le temps de la mission.

Discussion du choix de l'application de "collecte de minerai"

L'application est simple à modéliser et permet facilement de distinguer les sous-problèmes et par suite les tâches à assurer par les agents et le modèle de l'environnement. Malgré cette simplicité, le comportement global dans la résolution du problème est difficile à prévoir. L'imprévisibilité n'est pas seulement

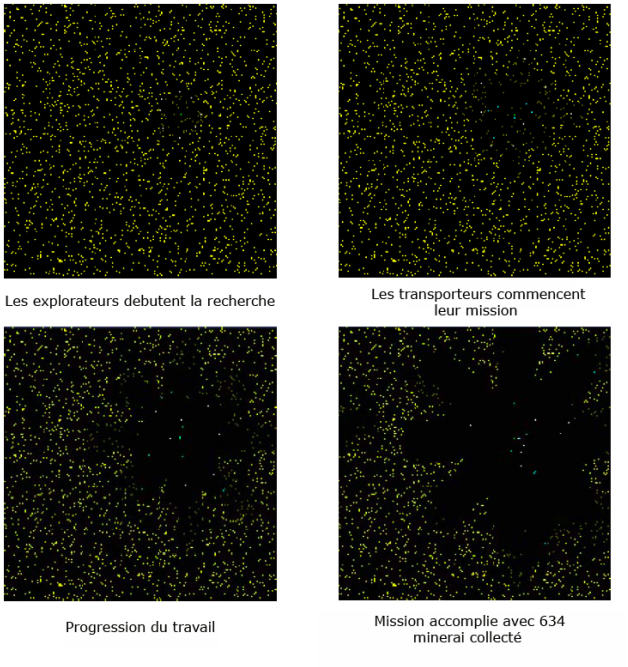

Fig. 7.4 – *Déroulement de la simulation*

due à la distribution du minerai sur la planète, il est aussi la conséquence de réactions individuelles des agents de même type et de types différents qui se comportent de façon autonome.

La résolution de ce problème comporte plusieurs types d'agents (transporteurs, explorateurs et bases) et plusieurs agents de chaque type. L'énergie consommée par les agents et le temps passé pour la collecte de minerai sont les ressources consommées par ces agents et par les bases.

En général, la résolution du problème fait appel à des communications entre les agents pour partager les informations utiles. Ces communications peuvent être entre les agents de même type ou entre agents de types différents selon la solution multi-agent proposée pour résoudre ce problème.

Malgré tous ces points positifs qui nous ont amené à choisir cette application, des limites restent, comme :

La diversité des paramètres : l'idée de l'application est de montrer comment des agents peuvent collaborer pour explorer une planète et collecter du minerai. La tâche doit être effectuée en respectant un certain nombre de contraintes, tels les rayons de perception et de communication, la quantité limitée d'énergie et le nombre de robots. L'ensemble de ces paramètres sera présenté dans le cadre du protocole d'expérimentation. Cette diversité des paramètres rend le travail plus compliqué au niveau de l'évaluation.

Présentation d'un premier système multi-agents

Dans cette partie, nous présentons un premier système multi-agents résolvant le problème de recolte de minerai. Dans ce système, plusieurs spécifications sont ajoutées aux spécifications initiales du problème. En ce qui concerne les agents, chaque agent posséde son propre GPS. Ainsi, il est conscient de sa situation dans l'espace à tout instant (coordonnées de l'agent dans l'espace). Les bases, coopérantes, ne se connaissent pas au lancement de l'application. Chaque base peut détecter les autres par la perception durant le temps de la mission.

En ce qui concerne la communication, le rayon de communication de la base est plusieurs fois plus grand que le rayon de communication d'un autre agent. En plus, les bases peuvent envoyer plusieurs messages en une seule action. La réception et l'interprétation d'un message par un agent ne prend pas de temps et ne coûte pas d'énergie. Il faut noter en fin que les transporteurs dans cette application ne sont pas en mesure de détecter les autres agents dans leurs rayons de perception.

Les agents utilisent la coordination pour réduire la consommation d'énergie. La centralisation de la planification et de la coordination est limitée à un minimum pour assurer l'extensibilité et la tolérance aux pannes. La base doit assister en évitant au maximum le fait d'intervenir durant la planification et/ou la coordination entre les agents. Tous les agents retournent à leurs bases avant un temps t. L'ensemble des agents a deux buts principaux dans cette application, classés selon leurs priorités :

- Le but primaire est de minimiser le temps de collecte du minerai.
- Le but secondaire est de minimiser l'énergie consommée par les agents.

Voici, dans ce qui suit, le diagramme d'états du comportement des différents agents :

Explorateur : le diagramme d'état d'un explorateur est présenté dans la figure 7.5. L'état initial des explorateurs est "se déplacer". Dans cet état, les explorateurs se déplacent en faisant des perceptions dans leurs entourages pour détecter le minerai. Lorsqu'un explorateur décide d'envoyer aux autres agents des informations sur le minerai collecté, il passe dans l'état "envoyer les coordonnés" dans lequel il communique avec les autres agents. À tout instant, l'explorateur peut découvrir qu'il a besoin de recharger sa batterie, ce qui est présenté par l'état "retourner vers la base". En outre, il peut passer dans l'état "pas d'énergie" et mourir. Une représentation plus détaillée du comportement d'un explorateur est présentée par un diagramme d'activité dans la figure 7.6.

Transporteur : le diagramme d'états pour les transporteurs dans cette implémentation est explicité par la figure 7.7. Les transporteurs commencent leur activité par attendre des coordonnées depuis la base. Une fois qu'un transpor-

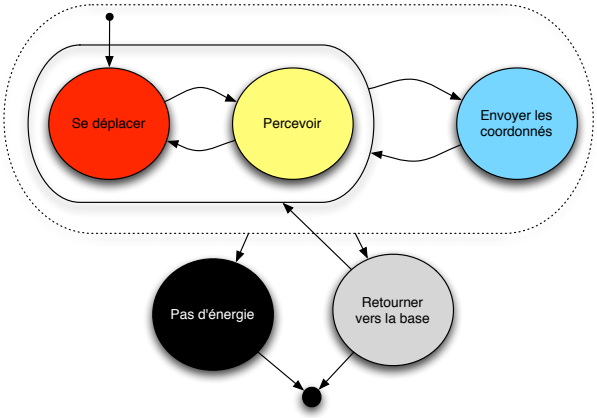

Fig. 7.5 – *Diagramme d'état du comportement d'un explorateur.*

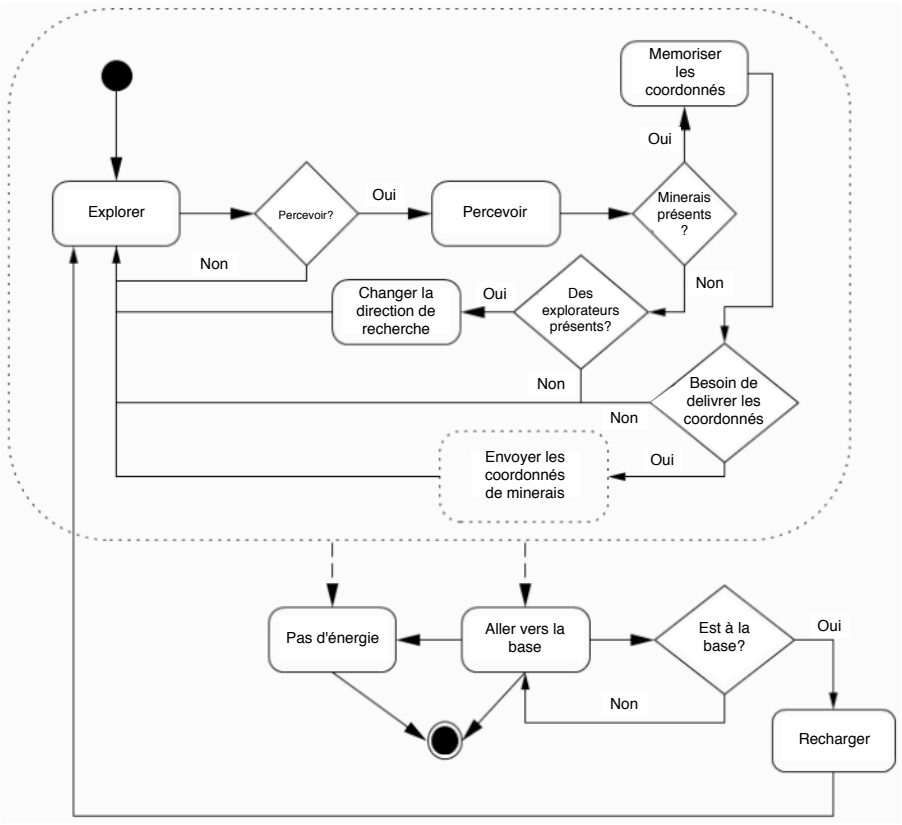

Fig. 7.6 – *Diagramme d'activité d'un explorateur.*

teur en a reçu, il choisit de partir les collecter ou de partager les informations avec les autres transporteurs afin de se coordonner. De la même façon que les explorateurs, les transporteurs ont besoin de se recharger à la base et peuvent mourir s'ils manquent d'énergie. Le comportement d'un transporteur est représenté en détail par le diagramme d'activité de figure 7.8.

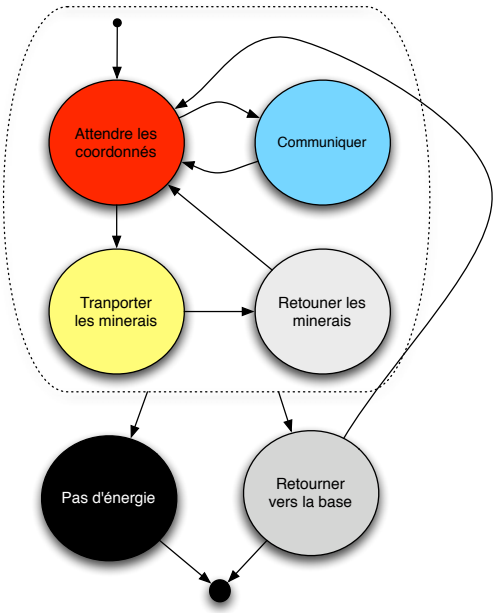

FIG. 7.7 – *Diagramme d'état d'un transporteur.*

Base : les bases (figure 7.9) jouent le rôle d'intermédiaires de communication entre les explorateurs et les transporteurs. Lorsque les explorateurs, pour une raison ou une autre, ne sont pas capables d'atteindre un transporteur afin de fournir les coordonnées du minerai, une base prend en charge ces coordonnées et l'envoie elle-même à un transporteur.

Une approche distribuée par appel d'offre est utilisée dans cette implémentation lorsqu'un explorateur veut livrer des coordonnées de minerai à des transporteurs. Une stratégie de raisonnement chez les explorateurs est appliquée pour choisir la meilleure offre. Les transporteurs utilisent de plus certaines stratégies afin de coordonner et de partager leurs connaissances avec les autres agents dans le but de minimiser l'énergie dépensée pour la collecte de minerai.

7.2 Le protocole expérimental

Les étapes de l'expérimentation présentées ici suivent les étapes proposées par [Jain (1991)] dans sa méthode d'évaluation. Nous présentons uniquement

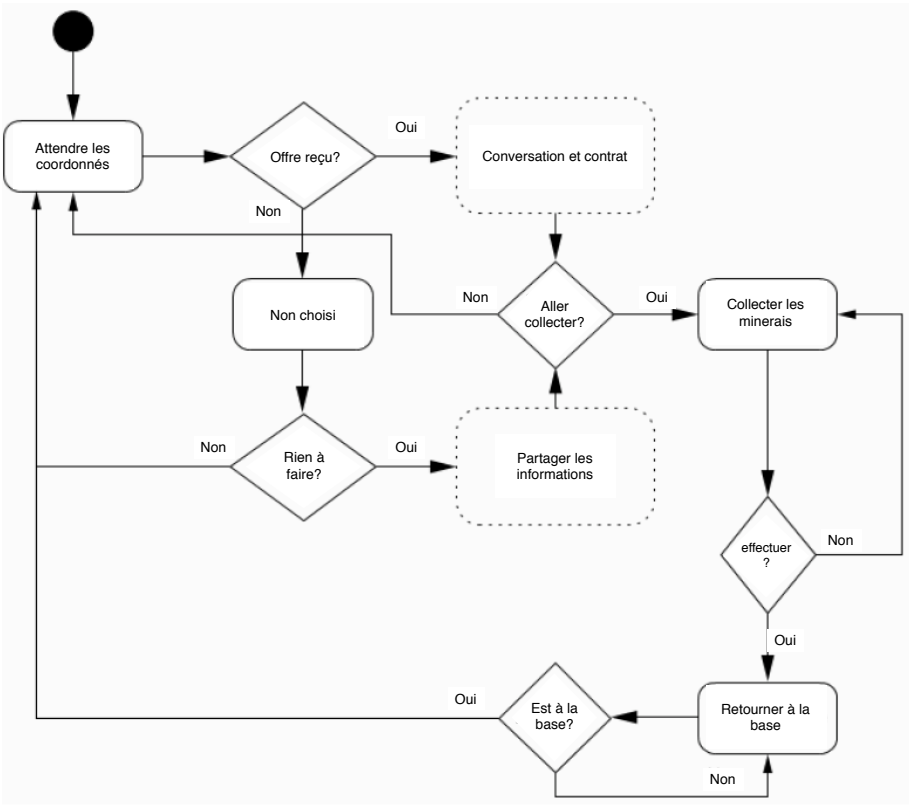

Fig. 7.8 – *Diagramme d'activité d'un transporteur.*

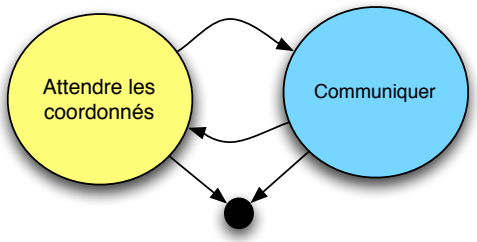

Fig. 7.9 – *Diagramme d'état du comportement d'une base.*

les étapes significatives dans nos expériences. Par exemple, la deuxième étape consiste à *lister les services fournis par le système et les résultats possibles suite à la sollicitation de chacun de ces services.* Notre premier système ne propose aucun service pour l'utilisateur. Cette étape ne sera pas donc présentée.

Établir les buts et définir le système considéré

Selon [Jain (1991)], la première étape dans un travail d'évaluation consiste à *établir les buts et définir le système considéré.* Le but de cette évaluation est d'étudier le comportement des agents dans un SMA, en utilisant la technique de visualisation, puis de comparer deux SMA résolvant le même problème avec des techniques de résolution différentes. Les parties du système à prendre en compte dans l'évaluation de notre applications sont les agents, les canaux de communication entre les agents, et l'environnement. L'organisation elle, est fixe.

Choisir les critères

Les critères qui vont servir pour comparer les systèmes sont :

Les critères du système multi-agents :

- l'énergie totale consommée par les bases et les agents ;
- le temps total passé à collecter une certaine quantité de minerai.

Les critères associés aux agents :

- l'énergie consommée par chaque agent ;
- la quantité de minerai détectée par chaque explorateur ;
- la quantité de minerai collectée par chaque transporteur.

Les critères associés aux communications (ce problème sera traité dans le chapitre suivant) :

- les communications entrantes de chaque agent ;
- les communications sortantes de chaque agent.

Les critères liés à la visualisation du comportement des agents dans le système.

Lister les paramètres et les facteurs

Cette étape consiste à établir une liste de caractéristiques influençant les performances du système multi-agents implémentant l'application. Ces caractéristiques sont appelées paramètres. Parmi la liste des paramètres, nous choisissons les facteurs à faire varier durant les expériences. La liste des paramètres de l'application sont dans le tableau tab 7.1.

N	Nombre de bases
M	Mode coopératif ou compétitif
D	Densité de minerai dans l'environnement
X	Nombre d'explorateurs
Y	Nombre de transporteurs
P	Rayon de perception
Q	Rayon de communication
S	Taille de la mémoire des robots
CM	Coût de l'envoi d'un message
CP	Coût d'un acte de perception
G	Taille de l'environnement
C	Capacité de la base
T	Temps de la mission

TAB. 7.1 – *Les paramètres de l'application de minerai.*

CM	Coût de l'envoi d'un message	1
CP	Coût d'une perception	1
G	Taille de l'environnement	200X200
C	Capacité de la base	80
T	Temps de la mission	T

TAB. 7.2 – *Les paramètres avec des valeurs fixes.*

Les paramètres parmi cette liste qui ont des valeurs fixes selon les spécifications dans le sujet (en annexe) du projet [Demazeau (2004a)], que nous adoptons pour la suite, sont présentés dans le tableau tab 7.2.

Nous considérons toutes les autres caractéristiques comme des facteurs ayant plusieurs valeurs et variant durant l'expérimentation. Le tableau 7.3 donne la liste des facteurs et les valeurs choisies pour ces facteurs.

Les paramètres figurant dans le tableau 7.4 sont désignés comme paramètres par défaut.

Sélectionner la technique d'évaluation

L'évaluation des performances d'un système informatique peut être procédée selon trois différentes techniques : la modélisation analytique, la simulation et les mesures. La première consiste à représenter le système par un modèle abstrait basé sur des notions mathématiques. La simulation consiste à implé-

N	Nombre de bases	1,2,5,8, 9
M	Mode coopératif ou compétitif	coop et comp
D	Densité de minerai dans l'environnement	2%, 5% et 10%
X	Nombre d'explorateurs	De 1 à 9
Y	Nombre de transporteurs	De 1 à 9
P	Rayon de perception	De 1 à 15
Q	Rayon de communication	De 1 à 60
S	Taille de la mémoire des robots	De 1 à 50

TAB. 7.3 – *Les facteurs.*

N	Nombre de bases	1
D	Densité de minerai dans l'environnement	5%
X	Nombre d'explorateurs	9
Y	Nombre de transporteurs	9
P	Rayon de perception	1
Q	Rayon de communication	1
S	Taille de la mémoire des robots	1

TAB. 7.4 – *Les paramètres par défaut.*

menter un modèle logiciel qui imite de manière simplifiée le comportement du système réel à évaluer. La dernière technique, dite de mesures expérimentales, consiste à mesurer directement certaines caractéristiques du système réel et à analyser les résultats de ces mesures. Comme présenté dans la partie technique de ce travail (chapitre 5 et 6), nous appliquons dans ce travail la technique de mesure expérimentale directe sur le SMA de façon à obtenir les informations sur le comportement de ses agents et ainsi évaluer ses performances.

Définir l'expérimentation

Nous avons à disposition une liste de facteurs avec pour chacun de ces facteurs une liste de ses valeurs possibles. Dans cette partie, nous présentons la liste des expériences à réaliser en combinant ces valeurs.

La figure 7.10 présente toutes les combinaisons possibles des différentes valeurs des facteurs pour nombre de bases N égal à 1. Cette combinaison est présentée sous la forme d'un arbre. Pour N égal à 1, le mode de coopérativité entre les bases, le facteur M, n'a aucun sens. Sur cette figure 7.10, chaque chemin allant de la racine vers une feuille représente une expérience possible avec une combinaison de valeurs différentes, par exemple, le chemin jaune de la figure.

Les figures 7.11 et 7.12 présentent les combinaisons possibles des valeurs pour le nombre de bases N égal à 2 et pour deux valeurs possibles du facteur M : "coop" pour la coopérativité entre les deux bases et "comp" pour la compétitivité entre les deux bases.

Le but de l'évaluation réalisée dans ce chapitre est de démontrer l'importance et l'utilité de la visualisation dans l'étude et l'évaluation des comportements des agents dans un SMA. Pour cela, nous choisissons de présenter un certain nombre d'expériences réalisées servant à expliquer notre travail, parmi toutes celles présentées dans les figures 7.10,7.11 et 7.12.

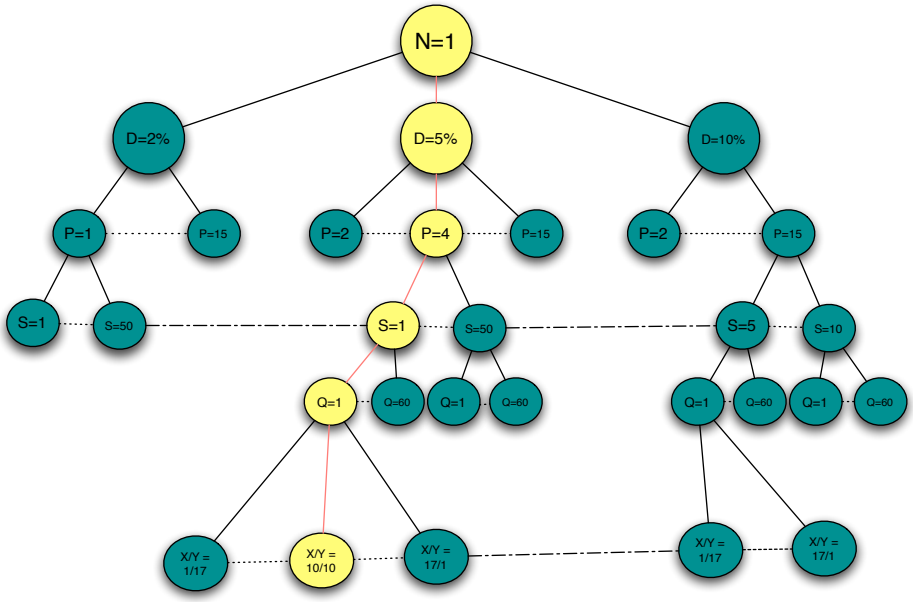

FIG. 7.10 – *Combinaison des valeurs possibles des facteurs pour N=1.*

Analyse et présentation des résultats

Les deux étapes, *analyse et interprétation des données* et *présentation des résultats*, seront présentées dans les sections suivantes.

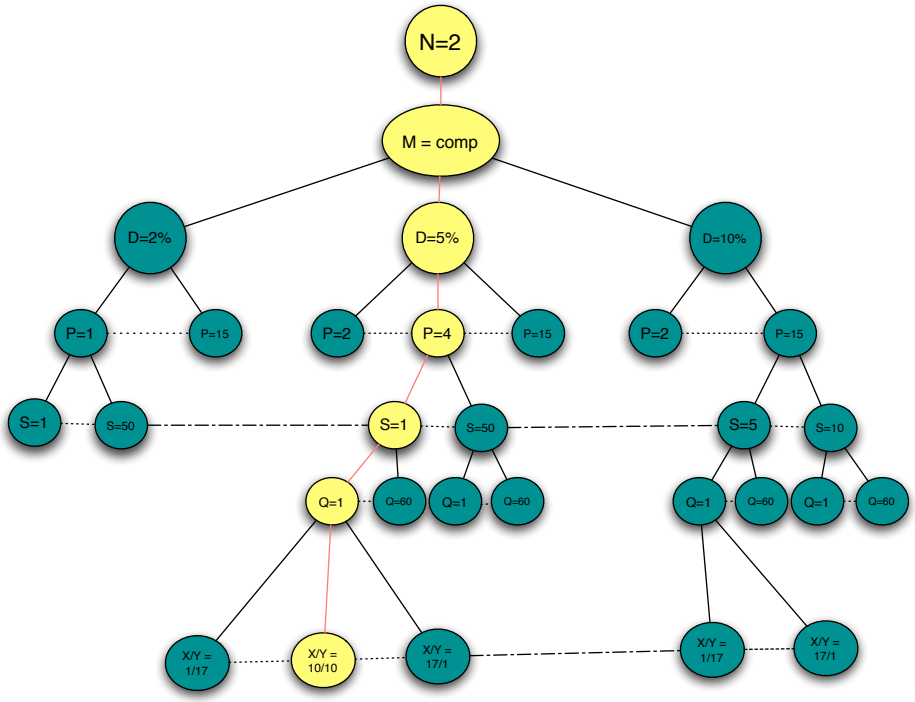

FIG. 7.11 – *Combinaison des valeurs possibles des facteurs pour N=2 et M=comp.*

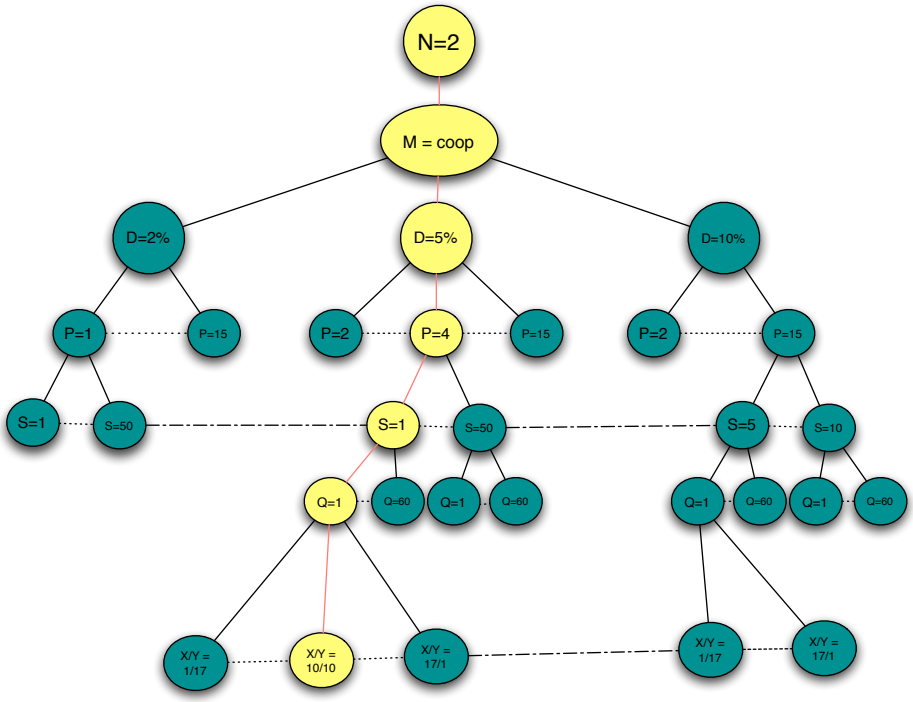

Fig. 7.12 – *Combinaison des valeurs possibles des facteurs pour N=2 et M=coop.*

7.3 Visualisation des traces d'un premier système implémentant l'application de "collecte de minerai"

Visualisation du système avec les paramètres définis par défaut

Dans cette partie nous détaillons les résultats de la visualisation des traces d'un premier système implémentant l'application de récolte de minerai avec les paramètres définis par défaut. Sur cette visualisation, nous allons présenter dans un premier temps l'utilité et les moyens apportés par la visualisation de l'exécution à l'analyse de comportement des agents. Dans un deuxième temps, nous présentons une analyse de l'exécution de l'application fondée sur la visualisation. Notons que les couleurs utilisées dans les visualisations des états des agents sont les mêmes utilisées pour désigner les états dans les figures 7.5, 7.7 et 7.9. La figure 7.13 montre les traces de l'exécution des explorateurs dans l'application. Sur cette figure, nous pouvons identifier tous les états de comportement d'un explorateur dans un diagramme espace-temps durant l'exécution du lancement de l'application à l'instant 0 jusqu'à la fin de l'exécution à l'instant 930. Sur cette figure, nous pouvons remarquer l'importance de l'utilisation de couleur dans la présentation des états de comportement des agents.

Sur la figure 7.14, nous remarquons la possibilité de visualiser le comportement des agents à différents niveaux d'abstraction. Sur la figure 7.14-(a), nous présentons une visualisation d'un niveau élevé de l'exécution de l'agent explorateur E1 entre les instant 0 et 500. Nous passons à plus de détails sur le comportement entre les instants 0 et 250 (figure 7.14-(b)) et finalement avec des détails encore plus fins sur la dernière figure 7.14-(c).

La figure 7.15 présente des indices statistiques sur les états de comportement des explorateurs. Sur cette figure, nous pouvons tirer des informations sur la répartition des tâches sur les explorateurs et le pourcentage de temps passé par chaque explorateur dans chaque état.

La distribution observée des tâches pour les explorateurs est uniforme. Les explorateurs passent plus de temps dans l'état "*se déplacer*" que dans l'état "*percevoir*". En moyenne, La moitié du temps total est passé dans l'état "*Envoyer les coordonnées*".

La figure 7.16 présente la visualisation du comportement des transporteurs dans le système. Les transporteurs (figure 7.16) passent une période en attendant l'ordre de collecte au début. Le temps d'attente des coordonnées au début de l'exécution vaut en moyenne la moitié du temps total de la mission pour les transporteurs. Puis, ils entrent dans des séquences d'états de collecte des minerais et de retour vers la base.

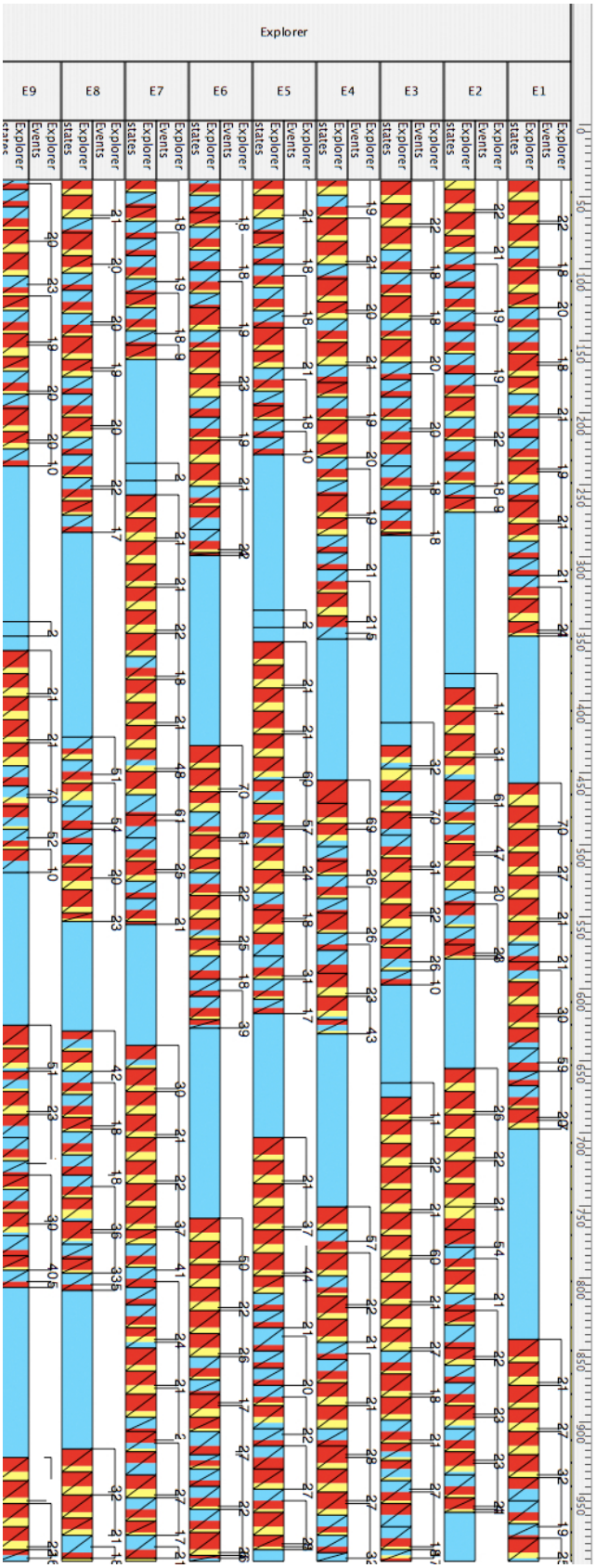

Fig. 7.13 – *Visualisation des comportements des explorateurs.*

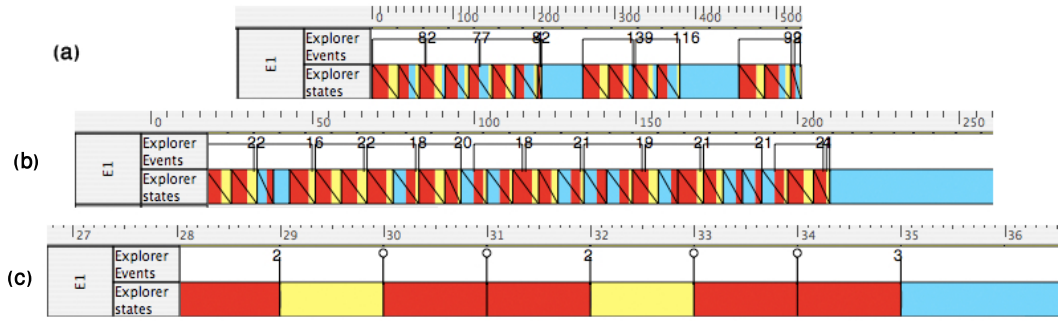

FIG. 7.14 – *Niveaux d'abstraction de la visualisation.*

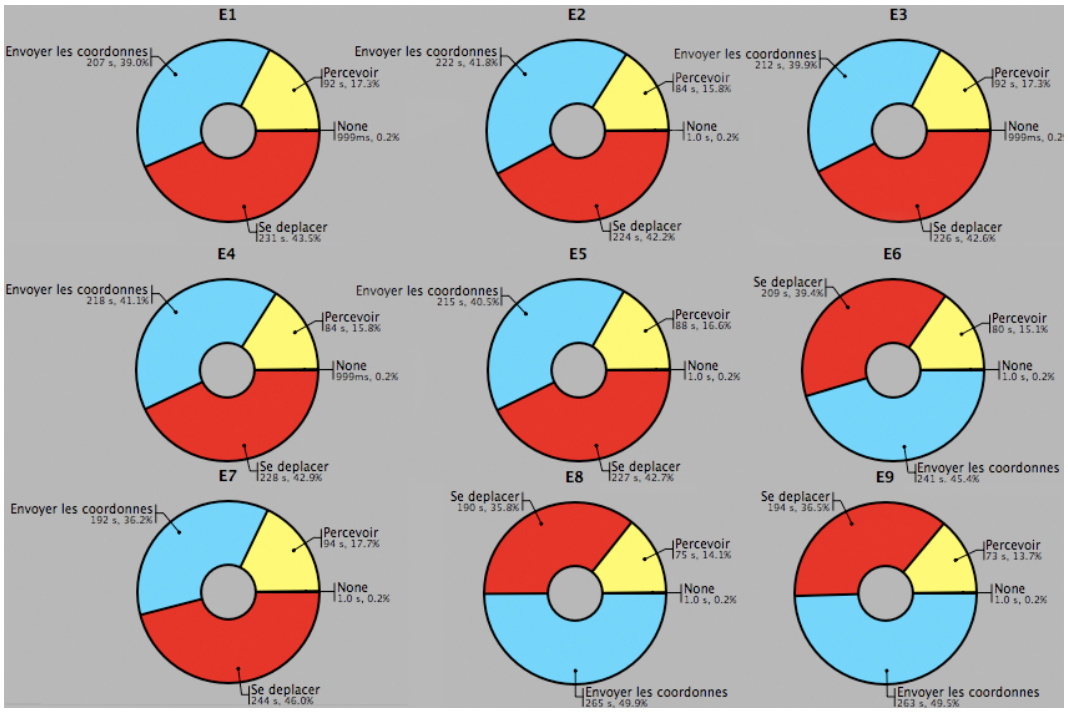

FIG. 7.15 – *Pourcentage de temps passé dans les états par les explorateurs.*

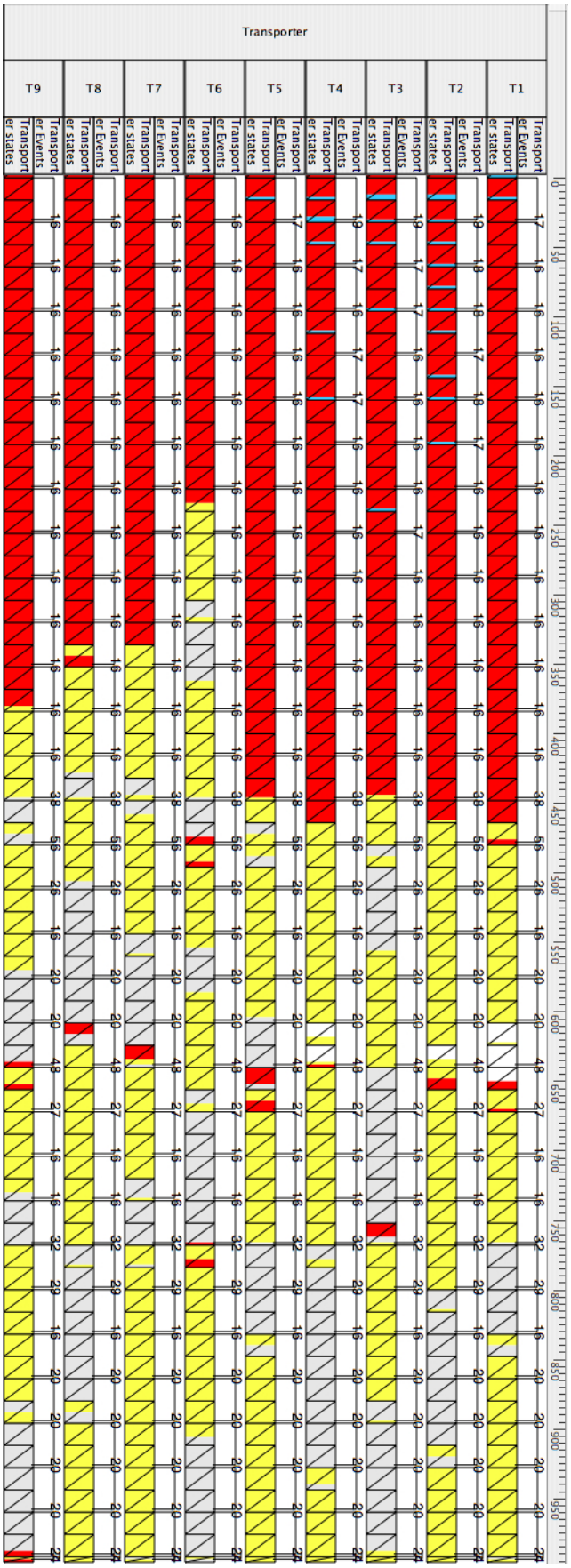

Fig. 7.16 – *Visualisation des comportements des transporteurs.*

À partir de ce qui est présenté dans les 2 figures des traces d'exécution des transporteurs et des explorateurs (figures 7.14 et 7.16), nous remarquons que nous n'avons des communications entre transporteurs et explorateurs qu'au début de l'exécution. Il n y a pas de l'état de communication dans la visualisation des traces des transporteurs dans la deuxième partie de l'exécution. Pour les explorateurs, nous avons des communications distribuées sur toute la période de l'exécution. La figure 7.17 présente le pourcentage de temps passé par la base en attendant les requêtes de coordonnées et à en communiquer.

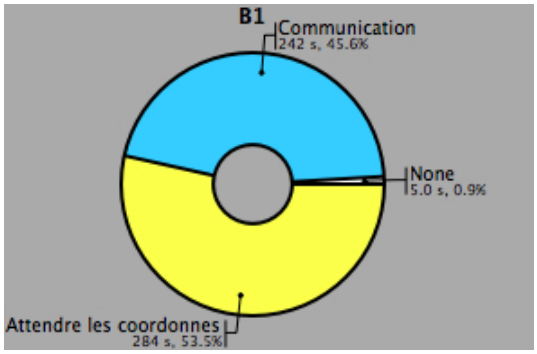

FIG. 7.17 – *Pourcentage de temps passé dans les états par la base.*

La figure 7.18 présente l'exécution de la base. Nous remarquons que le comportement de la base commence par l'état d'attente des requêtes et terminent par des communications dans la deuxième moitié de l'exécution. Comme il n'y a pas de communication entre les transporteurs et les explorateurs dans la deuxième moitié de l'exécution, les communications présentes dans les traces des explorateurs sont des communications avec la base. Toutes les coordonnés détectés par les explorateurs sont proposées par appel d'offre (protocole d'envoi des coordonnés entre explorateurs et transporteurs défini dans les spécifications de ce premier système multi-agents) aux transporteurs. Comme les transporteurs ne sont pas disponibles pour répondre, les coordonnées sont ensuite envoyées à la base. Ce qui se traduit par le fait que la base devient avec le temps le moteur de l'application. Ce premier système se comporte d'abord conformément aux spécifications une application décentralisée où la base doit assister en évitant le maximum d'intervenir. Il devient ensuite de plus en plus centralisée dans la suite de l'exécution.

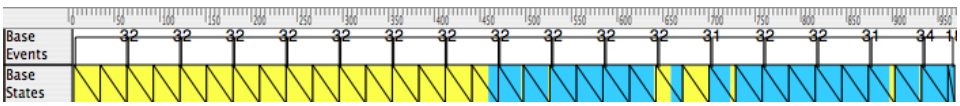

FIG. 7.18 – *Visualisation du comportement de la base.*

Visualisation du système avec Q=60

Dans cette partie, nous présentons la visualisation des traces d'exécution de l'application de minerais avec comme paramètres le rayon de communication "Q" égal à 60.

Les transporteurs (figure 7.19) ont toujours un temps d'attente au début de leurs missions. Dès qu'ils reçoivent de nouvelles coordonnées, ils partent collecter, puis après une période de collecte, ils retournent à la base. Ensuite, ils passent dans un état d'attente de nouvelles coordonnées. Ce scénario est diffèrent de celui vu dans la première visualisation où le temps d'attente vaut la moitié du temps total d'exécution et les transporteurs n'ont plus beaucoup d'attente dans la suite. Dans ce scénario, les transporteurs passent moins de temps au début mais attendent plus longtemps après chaque mission de collecte. Donc, nous avons une distribution du temps d'attente sur le temps total de l'exécution. La distribution des tâches sur les transporteurs (figure 7.21) n'est pas uniforme.

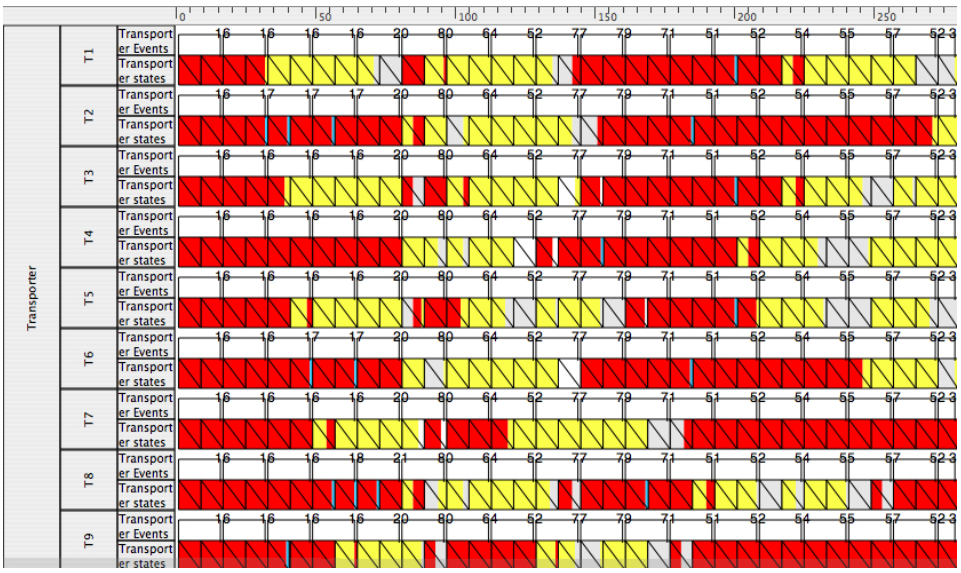

FIG. 7.19 – *Visualisation de l'exécution des transporteurs dans l'application de minerais avec Q=60.*

Pour les explorateurs (figure 7.20), comme pour le premièr jeu de paramètre un ensemble des états d'"*envoyer les coordonnées*" est entouré par des "*déplacer*" et des "*percevoir*". La distribution des tâches sur les explorateurs reste uniforme avec des plus longues périodes de communication (figure 7.22). Avec l'augmentation de rayon de communication, le temps passé dans l'état de communication augmente par rapport au temps de déplacement. Ces explorateurs ont plus de communications avec les transporteurs mais maintiennent beaucoup de communications avec la base.

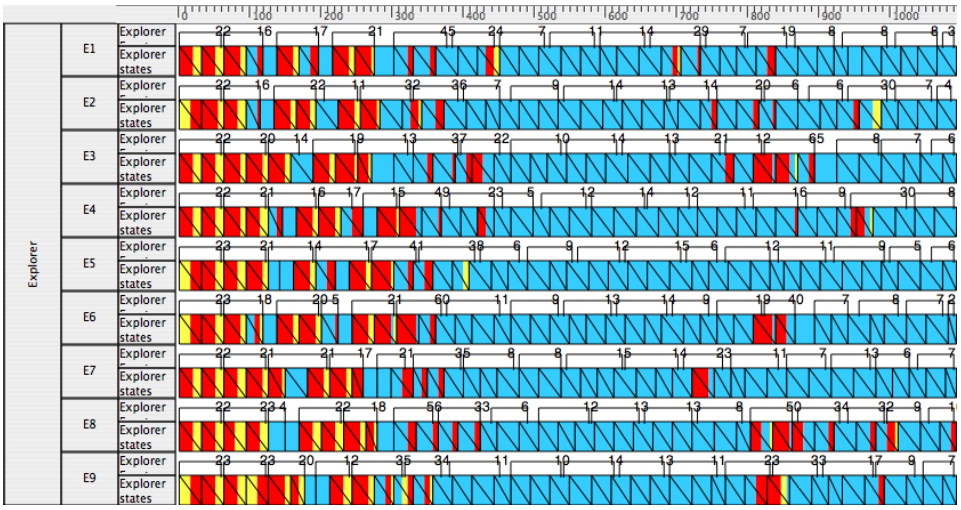

FIG. 7.20 – *Visualisation de l'exécution des explorateurs dans l'application de minerais avec Q=60.*

Jeu de paramètres	Temps (ms)	Énergie
Defaut	930	101240
Q=60	1070	162786
S=5	765	108098

TAB. 7.5 – *L'énergie consommée et le temps passé pour les jeux de paramètres présentés*

Avec le protocole d'appel d'offre utilisé pour l'envoi des coordonnées entre agents, l'augmentation du rayon de communication fait augmenter le nombre de communications entre agents et par suite fait augmenter (tab 7.6) :

- La quantité d'énergie consommée (plus de communications avec plus d'agents dans le rayon de communication) ;
- Le temps de la mission à cause du temps passé dans la communication avec plus d'agents et le temps passé dans choix du transporteur le plus adapté parmi les transporteurs répondant à l'offre.

Visualisation du système avec S=10

Dans cette partie, nous présentons la visualisation des traces d'exécution de l'implémentation 1 avec comme paramètres la mémoire des robots égale à 10.

Les transporteurs restent moins de temps au début dans l'état d'attente de minerai (figure 7.23). Les transporteurs font plus de collecte de minerais et moins de retours à la base pour déposer les minerais. Le retour à la base sert

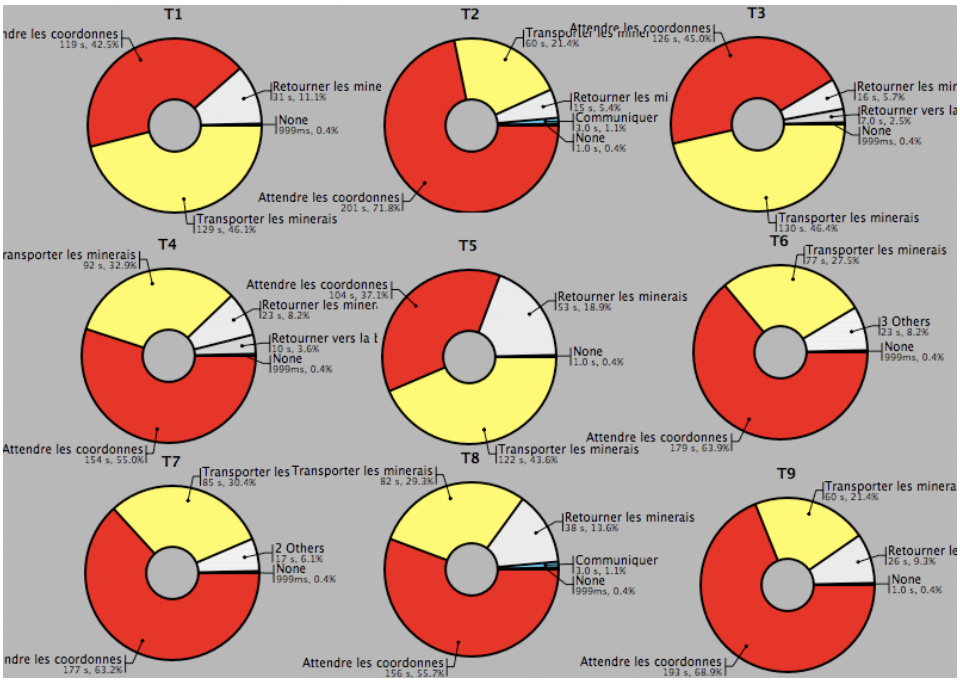

Fig. 7.21 – *Distribution des tâches sur les transporteurs pour l'application de minerais avec Q=60.*

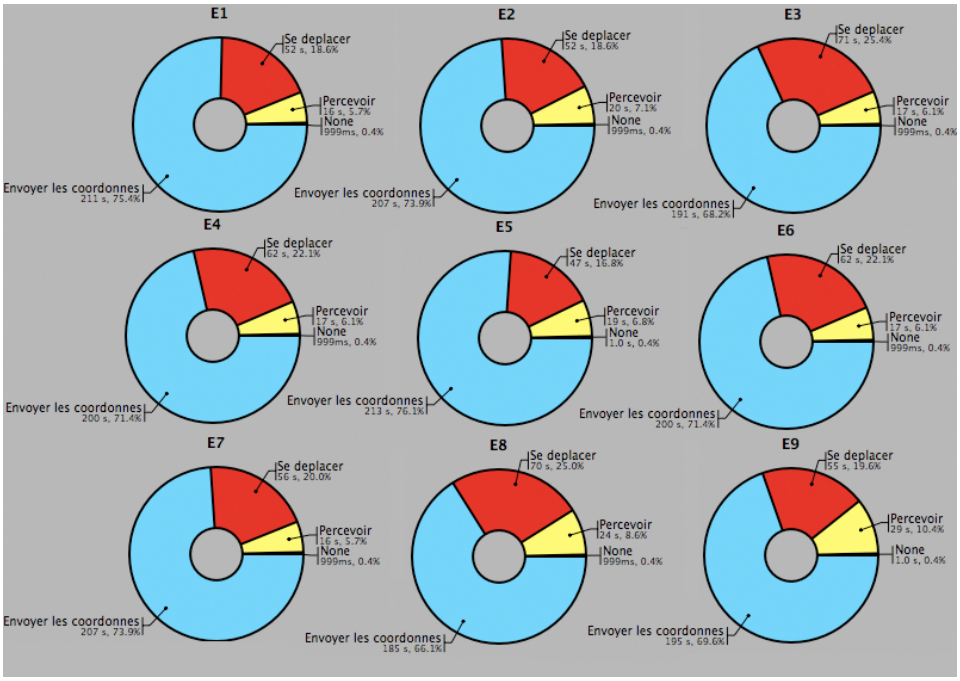

Fig. 7.22 – *Distribution des tâches sur les explorateurs pour l'application de minerais avec Q=60.*

plutôt pour recharger la batterie du robot et c'est ce qu'on peut remarquer dans la figure 7.25, nous avons juste des attentes. La distribution des tâches sur les transporteurs n'est pas uniforme (figure 7.25).

Pour les explorateurs (figure 7.24), l'état "*envoyer les coordonnées*" prend moins de temps. Les explorateurs font plus d'explorations avant de décider d'envoyer les minerais. Les tâches sont uniformement distribuées sur les explorateurs (figure 7.26).

L'augmentation de la taille de la mémoire diminue le temps dexécution du système par rapport aux autres jeux de paramètres présentés (tab 7.6). Cette diminution résulte du fait que les transporteurs n'ont pas besoin de retourner à la base à chaque fois qu'il y a un minerai à collecter (figure 7.23). Par contre, l'augmentation de la mémoire fait augmenter l'énergie consommée par rapport à la première visualisation puisque les explorateurs font plus de déplacement avant d'envoyer les coordonnées ce qui consomme plus d'énergie.

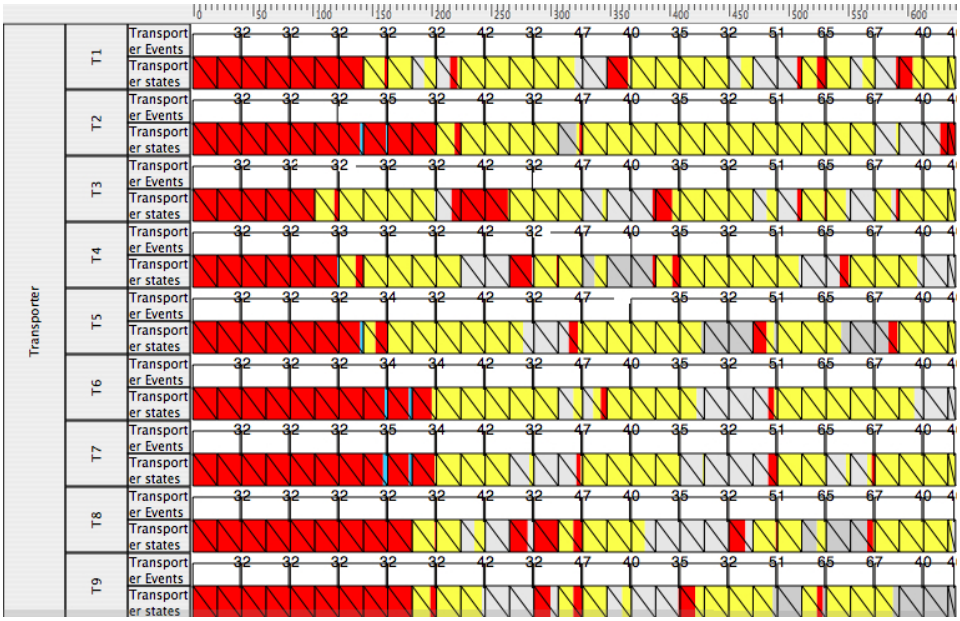

FIG. 7.23 – *Visualisation de l'exécution des transporteurs pour l'application de minerais avec S=10.*

Conclusion

Pour conclure cette partie, nous avons présenté l'utilité de la visualisation de l'exécution pour comprendre les comportements des agents dans l'exécution d'un système multi-agents. Plusieurs gains sont apportés par l'utilisation de cette technique dans le domaine des SMA. Nous avons présenté la possibilité de détecter des problèmes dans l'architecture interne du système comme par

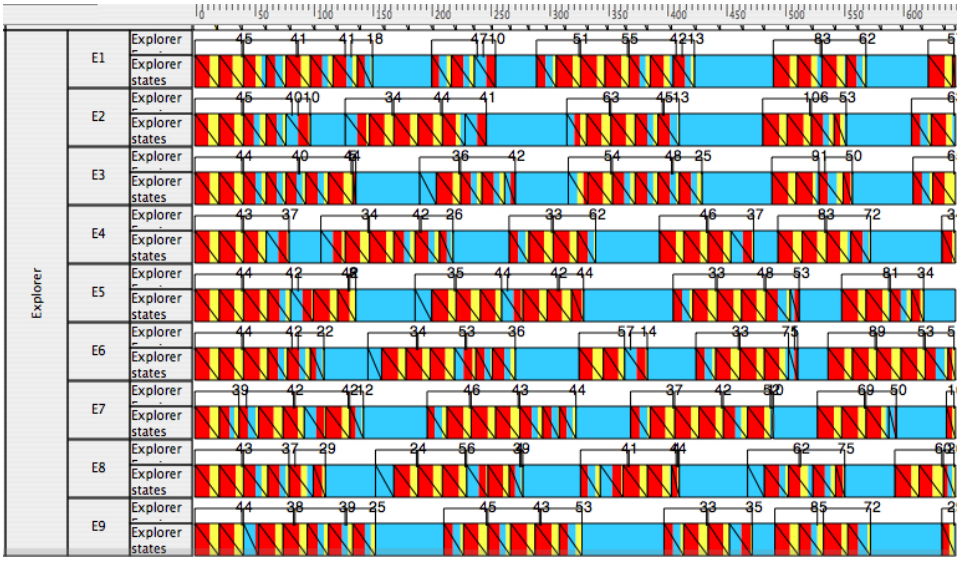

Fig. 7.24 – *Visualisation de l'exécution des explorateurs pour l'application de minerais avec S=10.*

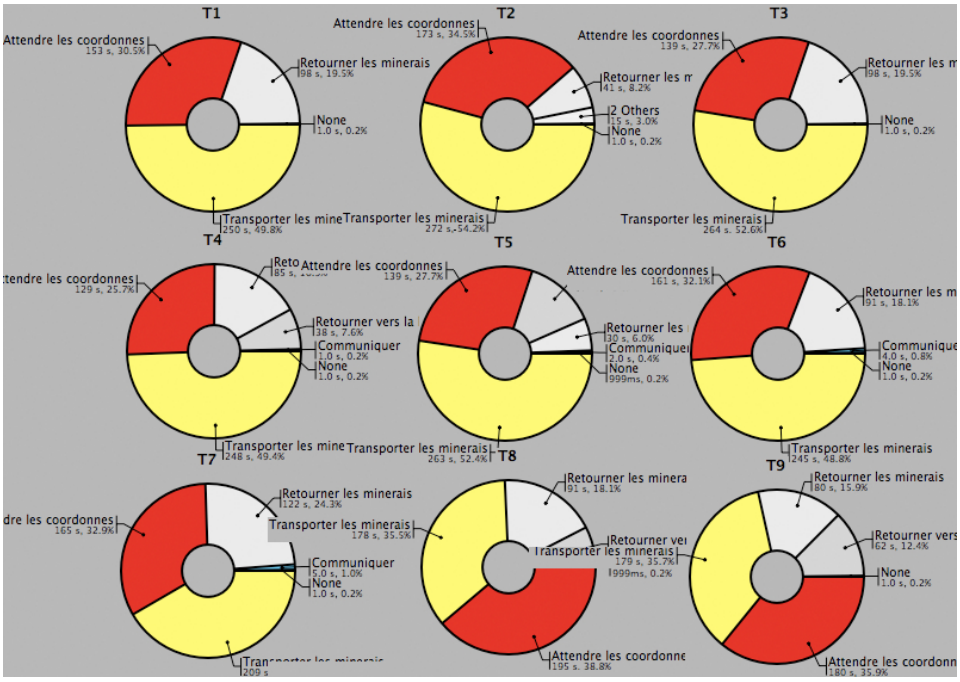

Fig. 7.25 – *Distribution des tâches sur les transporteurs pour l'application de minerais avec S=10.*

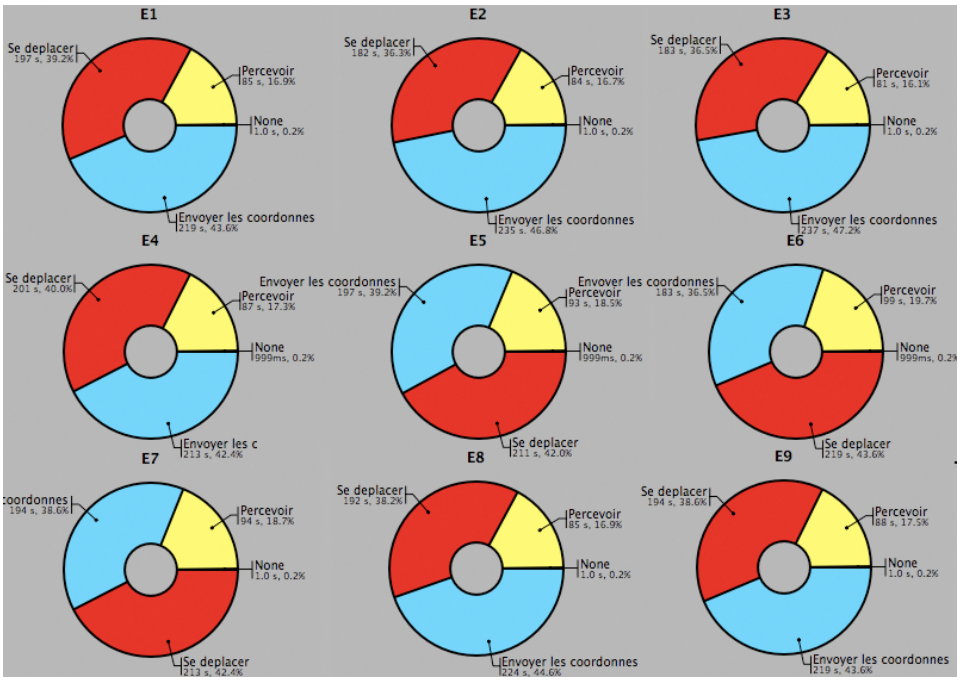

FIG. 7.26 – *Distribution des tâches sur les explorateurs pour l'application de minerais avec S=10.*

exemple dans le premier cas la centralisation d'un système conçu au début pour être complétement décentralisé. Cette technique a permis de comprendre et d'expliquer la consommation des ressources par les agents et de comparer le comportement des agents dans le système avec des jeux de paramètres différents. Ces résultats sont obtenus en utilisant des moyens proposés par l'outil de visualisation. Parmi ces moyens, nous avons présenté la possibilité de passage dans les niveaux d'abstraction de visualisation. Dans cette partie, nous avons présenté la visualisation et les résultats pour 3 jeux de paramètre.

7.4 Visualisation de l'exécution d'autres systèmes implémentant l'application de minerai

Un deuxième système

Dans ce deuxième système, les explorateurs prennent certains facteurs en compte dans leur mécanisme de perception. Est-ce que l'agent a déjà exploré le même espace? Quelle est la probabilité que l'espace a déjà été exploré par un autre explorateur? Le plan de la collecte des transporteurs se modifie lorsque des connaissances sur de nouveaux échantillons de minerai sont acquises. Les communications auront uniquement lieu lorsque les agents se trouvent dans la

base ou dans le rayon de perception de la base. Une perte d'informations sur les minerais par des transporteurs peut avoir lieu dans cette implémentation. Ce scénario peut se passer lorsqu'il n'y a plus d'espace mémoire pour stocker les coordonnées. Tous les robots restent à la base et ne commencent leur mission qu'après un message d'ordre reçu.

Deux stratégies d'exploration sont définies dans cette application : "star" et "fish-bone". La stratégie d'exploration représente les règles utilisées par l'agent explorateur pour se déplacer durant son état d'exploration de l'espace. La première stratégie est la stratégie "star". L'idée de la stratégie "star" est qu'un explorateur choisit un angle aléatoire pour se déplacer en quittant la base. L'explorateur se déplace tout simplement dans ce sens dans la mesure du possible. Quand cet explorateur souffre de manque d'énergie, quand sa mémoire est remplie par des emplacements de minerai, ou si un délai d'attente est écoulé, il retourne à la base. L'autre stratégie d'exploration est la stratégie "fishbone". Dans cette stratégie, la base contrôle la zone de recherche en fournissant une distance (en fonction du temps de voyage avant le retour) et les orientations (haut, bas, gauche, droite) de l'explorateur pour réaliser l'exploration. L'exploration aura donc la forme d'un squelette d'un poisson. La distance entre les "os" est liée au rayon de perception des agents.

Un troisième système

Dans cette implémentation, un transporteur et un explorateur communiquent uniquement et indirectement à travers la base. Les explorateurs échangent périodiquement leurs connaissances et synchronisent les informations dans le but d'assurer l'envoi des données le plus vite possible à la base et d'économiser l'énergie.

7.4.1 Modèle de visualisation

Dans cette partie, nous présentons le modèle de visualisation. Ce modèle doit couvrir les 3 modèles des 3 systèmes multi-agents présentés pour permettre de les comparer. Un travail est nécessaire à ce niveau pour donner des spécifications concernant la construction d'un modèle de visualisation commun entre plusieurs applications résolvant le même problème. Le modèle de visualisation de l'application est présenté dans la suite selon les 2 dimensions :

- Les agents (les états et les événements d'observation)
- L'organisation

L'étude des communications dans la visualisation (les types des communications et les événements liés aux communications) sera présentée dans le chapitre suivant.

Les agents

Les diagrammes d'états des comportements des agents dans les 3 implémentations ne sont pas traités avec le même niveau d'abstraction des états. Une comparaison entre les différentes implémentations nécessite de proposer un modèle de visualisation unique pour l'application de minerai. Un état dans ce modèle est obtenu, soit par raffinement des états des diagrammes des implémentations soit par l'association des plusieurs états des modèles des implémentations pour construire un état plus générique. En plus des états, il faut préciser les conditions d'entrées et de sorties des états. Dans le tableau suivant, nous présentons les états d'un explorateur et les conditions d'entrées et de sorties associées.

Etat	conditions d'entrée	conditions de sortie
Naviguer	fin de recharge, réponse negative de communication, commande base	minerai detecté, faible énergie, fin de la mission, énergie<0
Communiquer	minerai detecté, déposer minerai	réponse negative, mission affectée, fin de la mission, low énergie, énergie<0
Aller vers base	faible énergie, déposer minerai	énergie <0, fin temps, atteindre base
Terminer	énergie<0, fin de la mission	-

Ce tableau se traduit par le diagramme d'état de la figure 7.27. Ce diagramme représente le modèle de visualisation de l'agent explorateur. Les états sur ce diagramme sont les éllipses et les flèches représentent les transitions entre les états suite à la réalisation des conditions d'entrées et de sortie.

Les diagrammes d'états pour les agents transporteurs et bases sont présentés dans les figures 7.28 et 7.29.

L'organisation

L'organisation est présente dans la figure 7.30 : quatre types d'entités (explorateurs, transporteurs, bases, environements) avec des instances des types des agents, pour chaque type d'agents un ensemble d'états et d'événements sont associés. Les communications sont traitées dans le chapitre suivant.

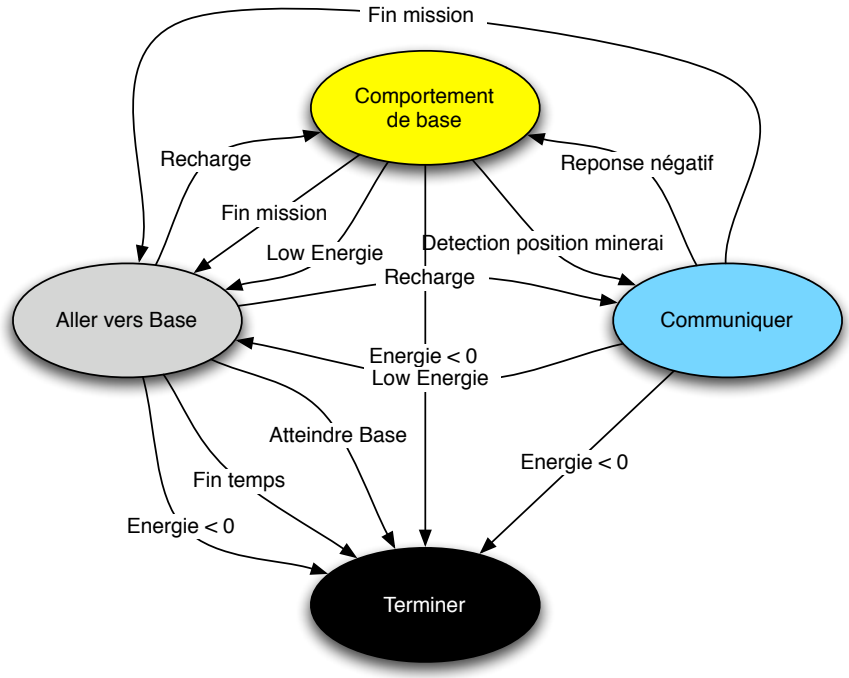

FIG. 7.27 – *Modèle de visualisation de l'explorateur de l'application de minerai*

7.4.2 Deuxième système avec les paramètres par défaut

Dans cette partie, nous présentons la visualisation des traces d'exécution du deuxième système avec les paramètres définis par défaut pour l'application.

Pour les transporteurs (figure 7.31), nous remarquons que nous avons plusieurs transporteurs "inactifs". Ces transporteurs ont passé leurs temps à attendre à la base. La charge du travail n'est pas bien distribuée sur les transporteurs. Par suite, la distribution des tâches sur les agents n'est pas uniforme (figure 7.33).

Malgré le comportement déséquilibré des transporteurs, les explorateurs sont eux uniformes dans leurs comportement (figure 7.32) et la distribution des tâches est uniforme (figure 7.34). Les comportements des explorateurs sont périodiques. Nous avons une période de collecte suivie d'un période de communication et d'envoi des coordonnés. La périodicité du comportement des explorateurs est due au faite que la stratégie d'exploration adoptée par cette implémentation assure la collecte de minerai avec une distance fixe de déplacement des explorateurs. Les politiques d'exploration et les mécanismes de perception utilisées ont beaucoup d'importance pour le système pour minimiser le temps total de collecte de minerai.

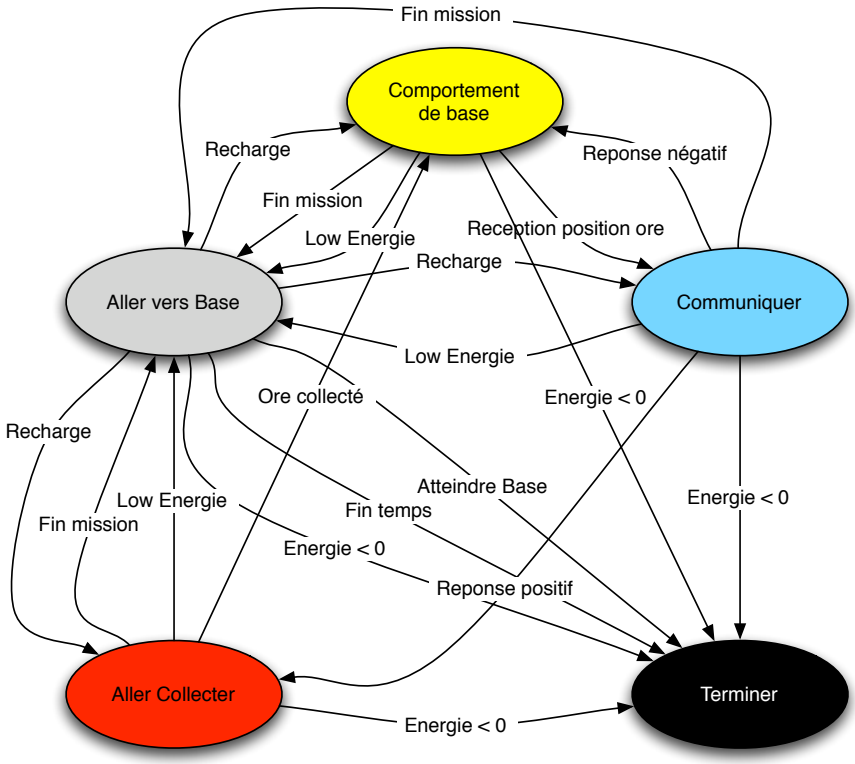

FIG. 7.28 – *Modèle de visualisation du transporteur de l'application de minerai*

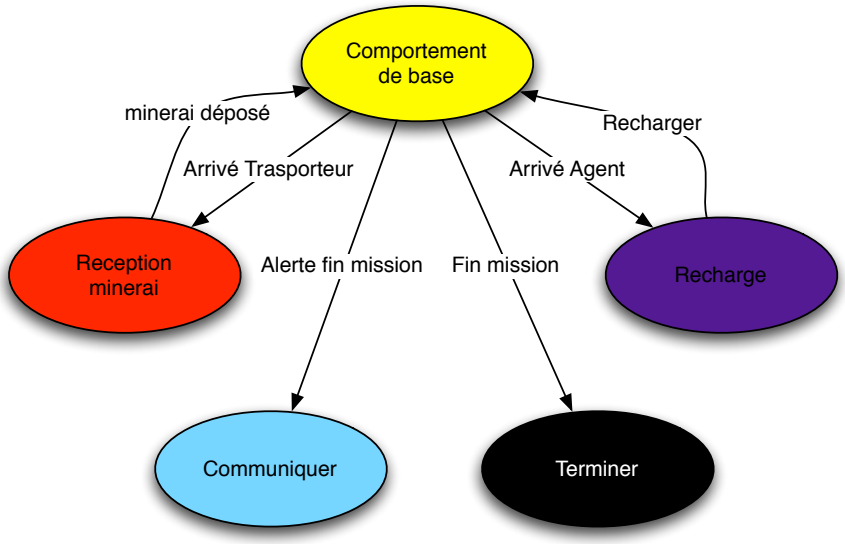

FIG. 7.29 – *Modèle de visualisation de la base de l'application de minerai*

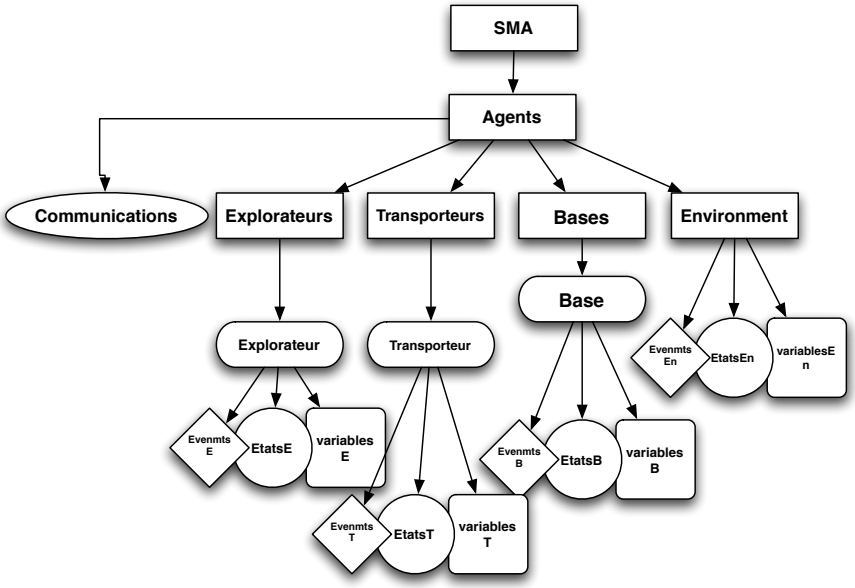

Fig. 7.30 – *Diagramme d'organisation*

Application	Temps (ms)	Énergie
Application 1	930	101240
Application 2	350	748800
Application 3	2270	1153440

Tab. 7.6 – *L'énergie consommée et le temps passé en fonction pour les 3 systèmes avec les paramètres par défaut*

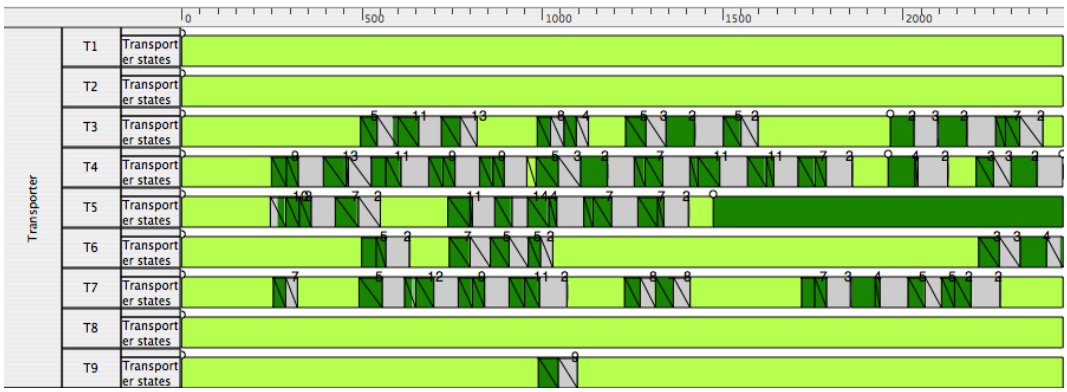

Fig. 7.31 – *Visualisation de l'exécution des transporteurs du deuxième système avec les paramètres par défaut*

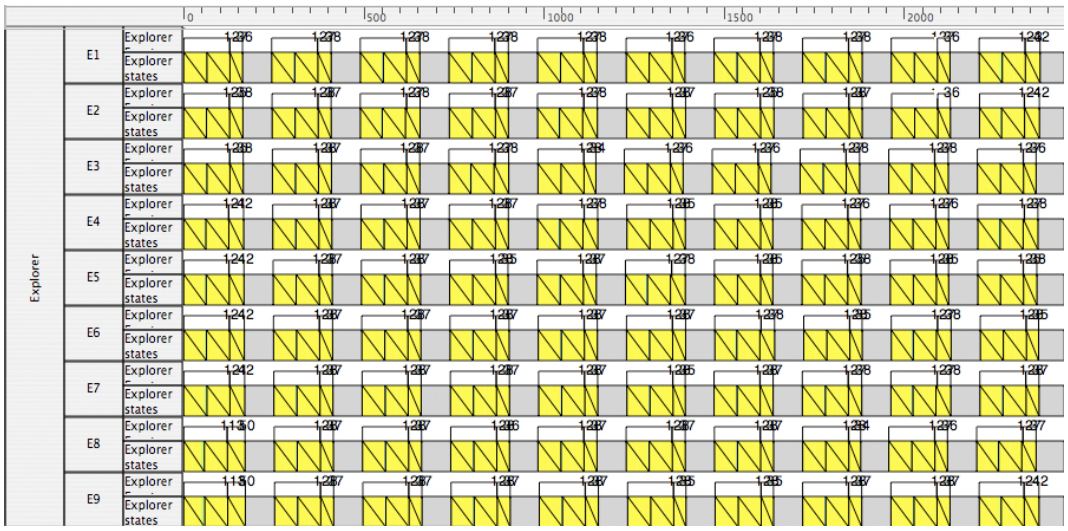

FIG. 7.32 – *Visualisation de l'exécution des explorateurs du deuxième système avec les paramètres par défaut*

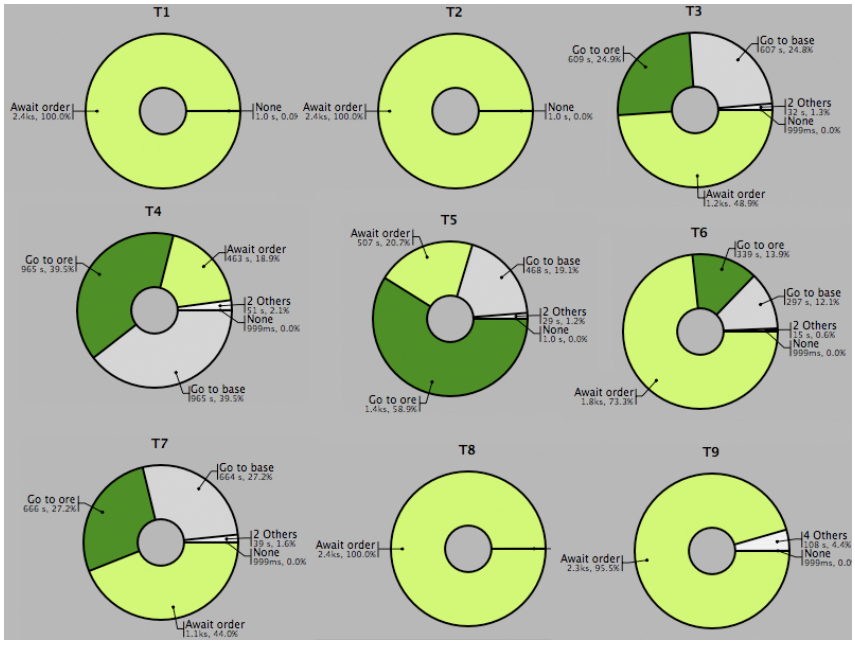

FIG. 7.33 – *Distribution des tâches sur les transporteurs du deuxième système avec les paramètres par défaut*

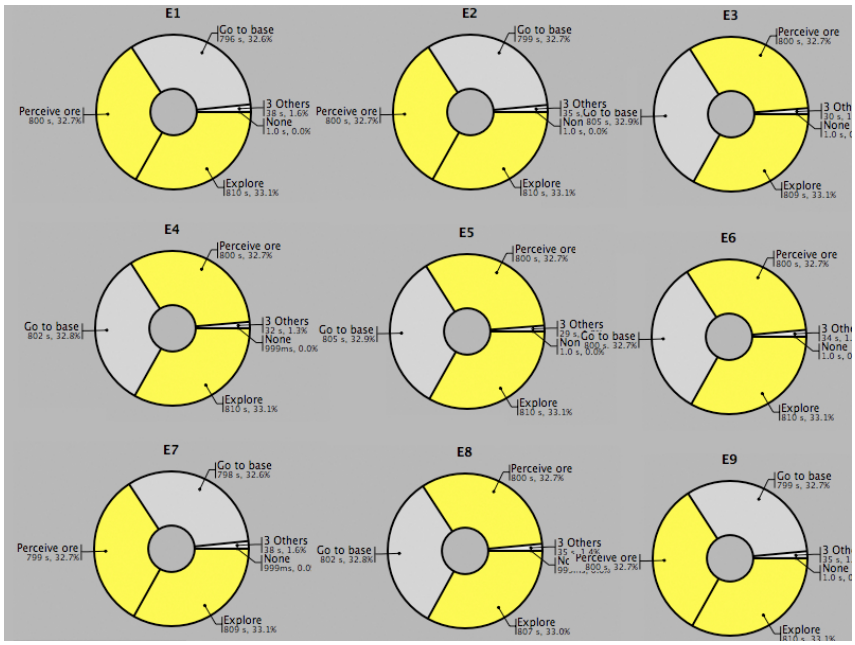

Fig. 7.34 – *Distribution des tâches sur les explorateurs du deuxième système avec les paramètres par défaut*

7.4.3 Troisième système avec les paramètres par défaut

Dans cette partie, nous présentons la visualisation des traces d'exécution du troisième système avec les paramètres définis par défaut pour l'application. Pour les transporteurs (figure 7.35), le comportement est constitué par des états de collecte suivis par des retours à la base. La distribution des tâches sur les différents transporteurs est uniforme. (figure 7.37). Pour les explorateurs (figure 7.36), le temps passé dans l'état "*percevoir*" est petit par rapport au temps passé dans l'état de retour vers la base pour communiquer avec les transporteurs (figure 7.38). La distribution des tâches sur les explorateurs est uniforme.

Pour cette implémentation :

- Le temps passé par un explorateur à retourner à la base et la communication est très grand par rapport au temps passé dans la collecte et à la perception des minerais. Le rôle principal d'un explorateur est donc perdu dans cette implémentation.
- Le temps de la mission et la quantité d'énergie sont invariante avec la variation du rayon de communication et du rayon de perception. Ceci est dû au faite que les explorateurs sont obligés de retourner à la base pour déposer les coordonnées des minerais détectés. Un changement dans l'une de ces rayons a donc un effet négligeable.

Cette implémentation présente donc le maximum d'énergie consommé et de temps nécessaire pour terminer la mission par rapport aux autres systèmes implémentants l'application.

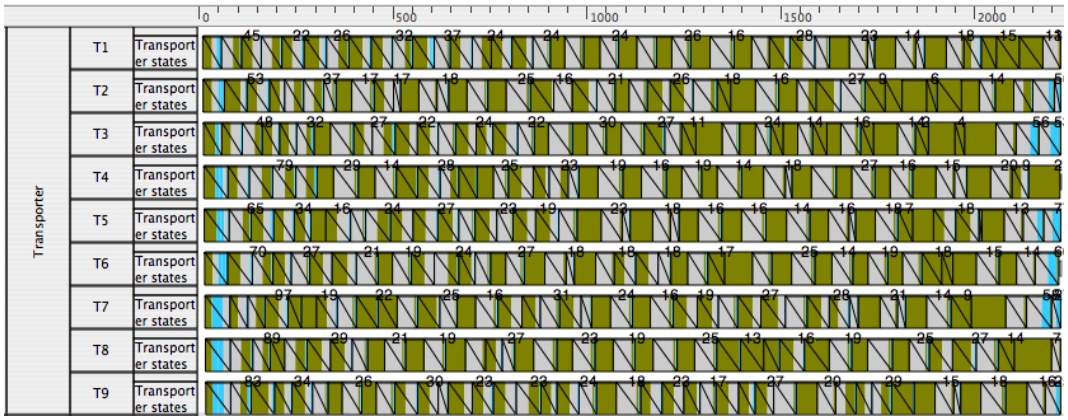

FIG. 7.35 – *Visualisation de l'exécution des transporteurs du troisième système avec les paramètres par défaut*

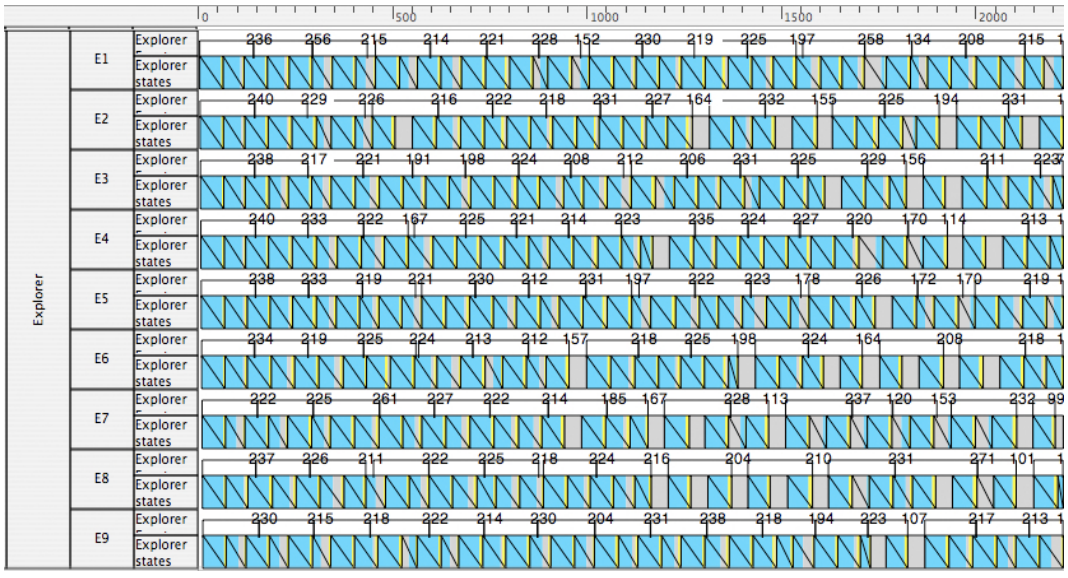

FIG. 7.36 – *Visualisation de l'exécution des explorateurs du troisième système avec les paramètres par défaut*

Conclusion

Pour conclure cette partie, la visualisation réalisée de l'exécution de ces trois implémentations nous permet de réaliser un comparatif selon le comportement

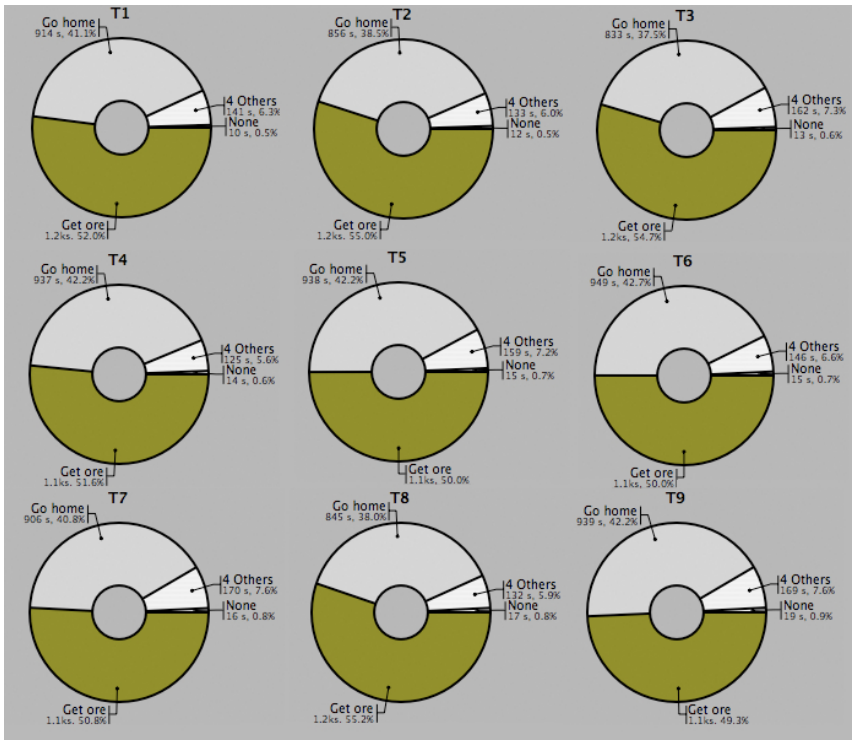

Fig. 7.37 – *Distribution des tâches sur les transporteurs du troisième système avec les paramètres par défaut*

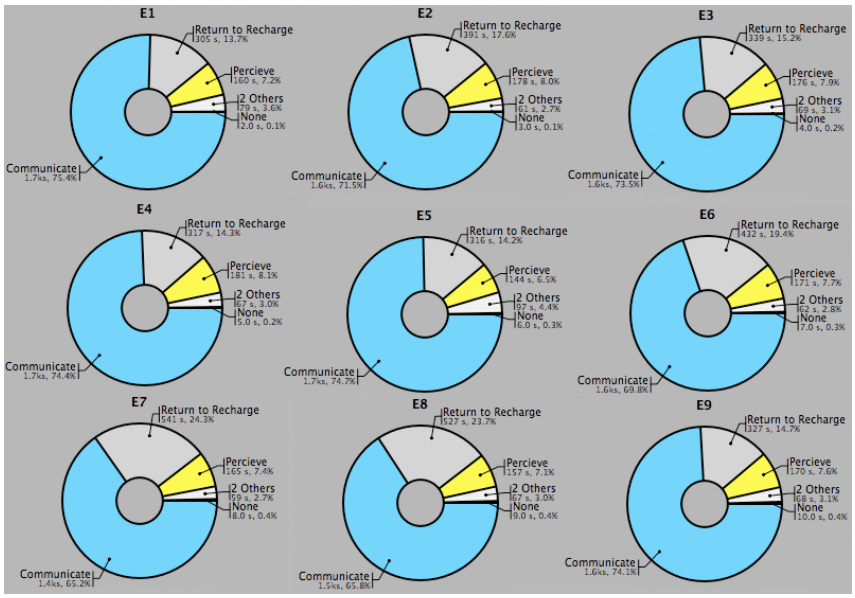

Fig. 7.38 – *Distribution des tâches sur les explorateurs du troisième système avec les paramètres par défaut*

interne des agents, la répartition des tâches et la conformité des exécutions aux spécifications pour ces trois implémentations.

Coordination : dans les spécifications de l'implémentation 1 et 2, les informations partagé par les transporteurs dans la mission de collecte. Une forte coordination est présente entre les agents dans les spécifications de la première implémentation. Par contre, nous remarquons que la coordination de type explorateur/transporteur est rare dans la visualisation de l'exécution de la première implémentation avec les différents paramètres. Pour les autres implémentations les coordinations s'effectuent entre des agents de même type ou avec la base.

Décentralisation : la première implémentation est complètement décentralisée selon les spécifications. Une centralisation cachée se présente dans l'exécution. La majorité des communications est de type explorateur/explorateur ou explorateur/base. Dans la deuxième implémentation, les communications se font conformément aux spécifications à la base et à travers la base dans la troisième implémentation.

Lien explorateurs/transporteurs : l'échange de connaissances se fait entre tous les robots dans la première implémentation. Cet échange est rarement présent dans l'exécution de la première implémentation entre les explorateurs et les transporteurs. Tandis qu'il a lieu juste entre les robots de même type dans les deux autres implémentations.

GPS : Le GPS dans les robots présent dans la première implémentation nécessite d'être complété par une stratégie d'exploration et de navigation pour les explorateurs.

Stratégie d'exploration : deux stratégies d'exploration existent dans la deuxième implémentation. Ces stratégies, en plus du mécanisme de perception présenté dans cette implémentation, sont à la base de la périodicité de comportement des explorateurs. Dans les autres implémentations, l'exploration est faite de manière aléatoire.

Chapitre 8

Expérimentation et Évaluation de l'application de collecte de minerai au niveau de la communication

Nous avons proposé dans les chapitres 4 et 6 une approche pour modéliser la communication. Ce travail complète celui réalisé au niveau de la visualisation de comportements internes des entités (chapitres 3 et 5). Ce chapitre complète le travail réalisé sur la visualisation pour les commuications et facilite l'étude et la compréhension de comportements des entités dans un SMA. Dans ce chapitre, nous démontrons l'intérêt de l'évaluation de la communication et des conversations pour :

- La détection des problèmes au niveau des communications (communications non pertinentes, poids des communications) ;
- L'importance de la visualisation des communications dans un SMA et le couplage entre l'évaluation de la communication et la visualisation ;

Puis nous traitons la modélisation des protocoles de communication et l'étude des communications selon leurs rôles durant les conversations.

Ce chapitre est constitué de 2 parties. Dans la première partie, nous présentons la dimension communication pour l'application de collecte de minerai. La deuxième partie traite la visualisation d'un système implémentant l'application et l'étude de la communication et de conversations pour ce système.

8.1 Les communications dans l'application de minerai

Les communications de base se déroulant entre les agents se produisent dans un rayon de communication. Le seul moyen de contact entre les différentes entités du système se fait par l'intermédiaire de la communication. La communication dans le système est instantanée et asynchrone et se traduit par l'envoi d'un message (*point-to-point*) entre deux agents. Les communications se présentent sous différents formes :

- Les communications entre un explorateur et un transporteur pour envoyer les coordonnées des minerais et affecter une mission de collecte à un transporteur.
- Les communications entre transporteurs dans le but d'échanger des informations concernant les minerais à collecter.
- Les communications entre un transporteur et une base pour déposer les minerais collectés et pour recharger.
- Les communications entre un explorateur et une base pour envoyer les coordonnées des minerais détectés et pour recharger.

Un premier scénario de communication se déroule lorsqu'un explorateur essaye de délivrer les coordonnées des minerais détectés à un transporteur ou à la base (figure 8.1). Dans ce scénario, l'explorateur cherche les transporteurs présents dans son rayon de communication :

- Si l'explorateur trouve des transporteurs, il envoie un message contenant les coordonnées des minerais détectés à chacun de ces transporteurs. Le transporteur répond par un message contenant son identifiant pour manifester l'acceptation de l'offre de l'explorateur. Autrement, un transporteur peut envoyer un message contenant une notification à l'explorateur que les coordonnées sont déjà collectées. Ce message exprime donc le fait que les minerais sont déjà collectés par ce transporteur. Après la réception de l'ensemble des identifiants des transporteurs, l'explorateur sélectionne le transporteur le plus adapté à la mission de collecte. Un message est envoyé à ce transporteur. Si le transporteur accepte la mission, il envoie un accusé de réception à l'explorateur. Le transporteur peut ignorer l'offre s'il est engagé dans une autre mission.
- Si l'explorateur ne trouve aucun transporteur, il envoie les coordonnées de minerais détectés à la base qui se charge de l'envoyer aux transporteurs. La base envoie un accusé de réception à l'explorateur.

Un autre scénario de communications possible entre deux transporteurs est présenté dans la figure 8.2. Dans cette application, les transporteurs essayent

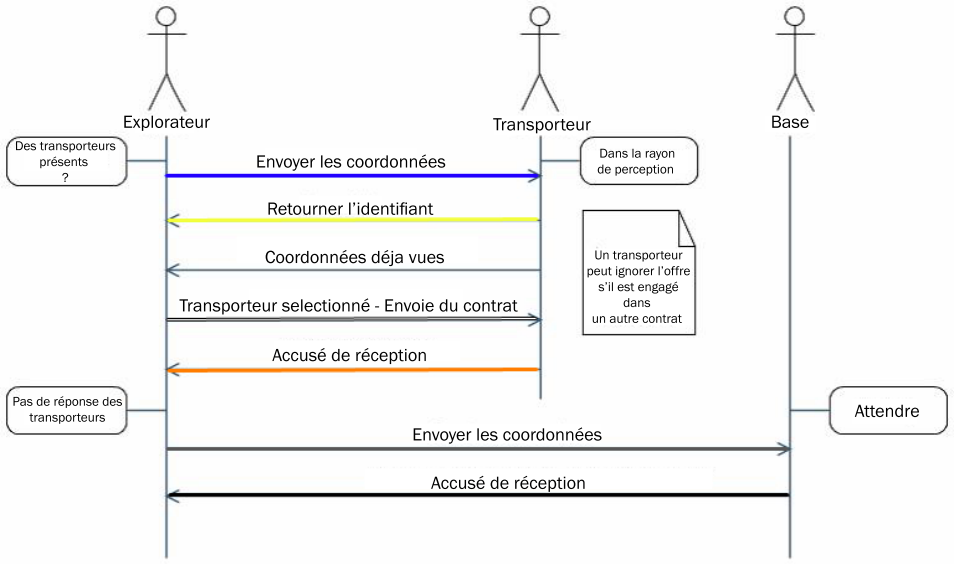

FIG. 8.1 – *Scénario de communications d'un explorateur essayant de délivrer les coordonnées detectées*

de partager les coordonnés avec leurs homologues. Ces communications ont pour but de faciliter la mission en partageant les coordonnées des minerais et de supprimer les coordonnées déjà collectées pour éviter la perte d'énergie. Dans ce scénario, un transporteur émetteur envoie un ensemble de coordonnés de minerais à partager avec les transporteurs présents dans son rayon de communication. Un transporteur récepteur répond par l'ensemble de coordonnées des minerais déjà collectés, s'il existe, pour les supprimer. Un transporteur récepteur peut inverser les rôles et envoyer des coordonnées au transporteur qui a initié l'envoi.

Dans la suite de ce chapitre, nous appliquons les approches d'évaluation présentées dans le chapitre 4. Dans ce but, nous commençons par lister les types de communications possibles pour les différents types des agents.

Les types des messages pouvant être reçus par un explorateur sont :

- "Accepter l'offre" : est un message d'acceptation de l'offre de l'explorateur pour collecter les minerais. Ce message est reçu d'un transporteur qui se trouve dans le rayon de communication et qui a reçu l'offre de l'explorateur.
- "Acquittement d'un transporteur" : est un accusé de réception d'une offre. Ce message est envoyé par un transporteur et reçu par un explorateur.
- "Coordonnées à supprimer" : est une réponse à une offre signalant qu'une partie des coordonnées envoyées est déjà collectée.

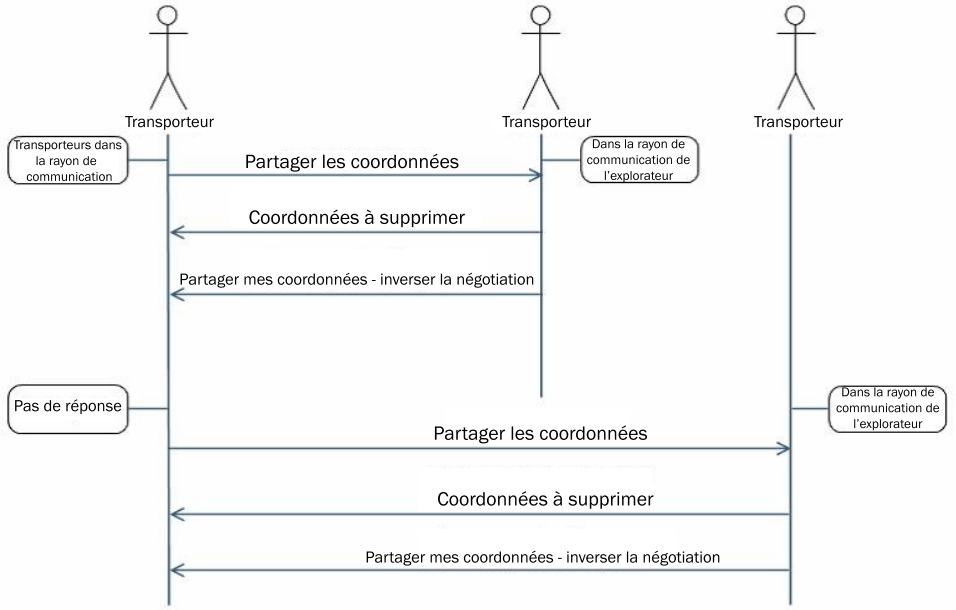

FIG. 8.2 – *Scénario de communications de partage des informations entre les transporteurs*

- "Acquittement d'une base" : traduit que la base a pris en charge l'envoi d'un ensemble des coordonnées de minerais aux transporteurs.
- "Recharge explorateur" : est un message reçu de la base après une demande d'un explorateur situé à la base.

Ces types de messages constituent l'ensemble $TYPE^{Explorateur}$ désigné dans le chapitre 4. Cet ensemble est l'ensemble des types de messages qui peuvent être reçus par l'agent explorateur.

Les types des messages pouvant être reçus par un transporteur sont :

- "Offre de collecte" : est une offre de collecte de minerai reçu d'un explorateur.
- "Sélection" : désigne la sélection du transporteur par l'explorateur. Ensuite, le transporteur va collecter les minerais envoyés.
- "Coordonnées reçues d'un transporteur" : désigne la réception des coordonnées envoyées par un transporteur qui partage les informations avec ses homologues figurant dans son rayon de communication.
- "Coordonnées reçues d'une base" : désigne la réception des coordonnées de minerais envoyées par la base.
- "Recharge transporteur" : est un message de recharge d'énergie reçu de la base après une demande d'un transporteur situé à la base.

Ces types de messages constituent l'ensemble $TYPE^{Transporteur}$ désigné dans le chapitre 4. Cet ensemble est l'ensemble des types des messages qui peuvent être reçus par l'agent transporteur.

Les types des messages pouvant être reçus par une base sont :

- "Demande de recharge explorateur" : est une demande de recharge d'énergie reçue d'un explorateur.
- "Demande de recharge transporteur" : est une demande de recharge d'énergie reçue d'un transporteur.
- "Dépôt de minerai" : est reçu d'un transporteur situé à la base pour déposer un ensemble de minerais collectés.
- "Coordonnées reçues d'un explorateur" : désigne la réception des coordonnées de minerais d'un explorateur.

Ces types de messages constituent l'ensemble $TYPE^{Base}$, désigné dans le chapitre 4. Cet ensemble est l'ensemble des types des messages qui peuvent être reçus par l'agent base.

Pour continuer avec l'approche poids de message présentée dans le chapitre 4, il faut lister les connaissances et les actions possibles pour chaque type d'agents selon les spécifications du problème.

La liste des connaissances possibles d'un explorateur sont :

- Les positions des minerais détectés.
- Les positions des bases.
- La liste des identifiants des transporteurs répondant à une offre.
- La quantité d'énergie restante.

Ces connaissances constituent l'ensemble $C^{Explorateur}$

La liste des connaissances possibles d'un transporteur sont :

- La liste des minerais à collecter.
- La liste des minerais collectés.
- La les positions des bases.
- La quantité d'énergie restante.

Ces connaissances constituent l'ensemble $C^{Transporteur}$

La liste des connaissances possibles d'une base :

- La quantité de minerai collecté.
- La quantité d'énergie restante.
- La liste des transporteurs à disposition.
- La liste des minerais détectés en attendant l'affectation.

– La liste des minerais collectés.
– La position d'autres bases.

Ces connaissances constituent l'ensemble C^{Base}

Les actions possibles d'un explorateur sont :

– Le mouvement dans l'environnement.
– La communication avec un agent.
– La perception de l'environnement.

Ces actions constituent l'ensemble $AC^{Explorateur}$

Les actions possibles d'un transporteur sont :

– Le mouvement dans l'environnement.
– La communication avec un agent.
– La collecte de minerai de l'environnement.

Ces actions constituent l'ensemble $AC^{Transporteur}$

L'action possible d'une base est de communiquer avec les agents. L'ensemble AC^{Base} est ainsi composé uniquement de l'action "communiquer".

8.2 Étude et visualisation de la communication dans l'application de minerai

Dans le chapitre 4 nous avons différencié 3 types d'approches pour étudier les communications dans un SMA :

– L'approche classique ;
– L'approche type ;
– L'approche poids de message.

Dans le chapitre 4 nous avons montré des mesures relatives à ces approches. Dans cette partie, nous nous intéressons plus sur le couplage de ces approches avec les visualisations de MAS-Paje. Nous présentons les résultats de la visualisation de la communication selon ces 3 méthodes, puis nous analysons les résultats.

Visualisation des communications

La figure 8.3 présente la visualisation des communications entre les agents sur une période de temps durant l'exécution du système. Nous pouvons distinguer sur cette figure les communications entre les agents représentées par des

flèches reliant les différents agents. Nous ne présentons pas toutes les données numériques associés aux agents. Nous notons uniquement que le nombre des messages reçus par le transporteur T6 durant l'exécution du système de minerai est Q(T6)=146 messages. Le nombre des messages reçus par l'agent T4 est Q(T6)=84 messages.

Cette approche et la visualisation associée peuvent donner des informations sur la quantité des messages Q=NB(M). Elles donnent ainsi des informations sur l'utilisation des canaux de communication entre les agents.

Mais suite à ces résultats, nous ne disposons pas des informations sur la quantité et la nature des informations portées par les messages. De plus, la présentation des communications sur la visualisation de l'exécution rend cette visualisation incompréhensible et difficile à utiliser pour en tirer des conclusions. Pour cette raison nous passons à une autre présentation proposée par l'approche type.

Visualisation avec l'approche type

Pour appliquer l'approche de type, nous distinguons entre les différents types de messages existants. Les messages distingués pour cette application sont présenté dans la première partie de ce chapitre par les ensembles :

$$TYPE^{Explorateur}, TYPE^{Transporteur} \text{ et } TYPE^{Base}.$$

Pour chaque type des messages nous associons un poids et une couleur.

La figure 8.5 présente la visualisation de l'exécution dans la même période de temps que la figure 8.3 en tenant compte de la distinction entre les différents types des messages. Cette distinction rend la visualisation plus compréhensible et permet de retirer des patterns d'exécution afin de comprendre le comportement des agents et d'en tirer des conclusions. Dans la suite, nous présentons quelques patterns d'exécution.

La figure 8.6 présente des communications entre la base et l'explorateur. L'explorateur envoie des données à la base qui répond avec un accusé de réception. Ce scénario se déroule lorsque le système devient centralisé sur la base (chapitre 7).

La figure 8.7 présente l'explorateur E1 essayant de délivrer les minerais au transporteur T8 qui se trouve dans son entourage. Le transporteur a ignoré l'offre puisqu'il est engagé dans une autre mission de collecte. Après une période d'attente, l'explorateur recommence à chercher de s'il y a des nouveaux transporteurs dans son entourage. E1 a trouvé T9 et T6. L'explorateur envoie un message à chacun de ces transporteurs. Ces transporteurs également ignorent l'offre. Après plusieurs essais, le transporteur envoie les données à la base qui prend en charge l'envoi de ces données à un transporteur.

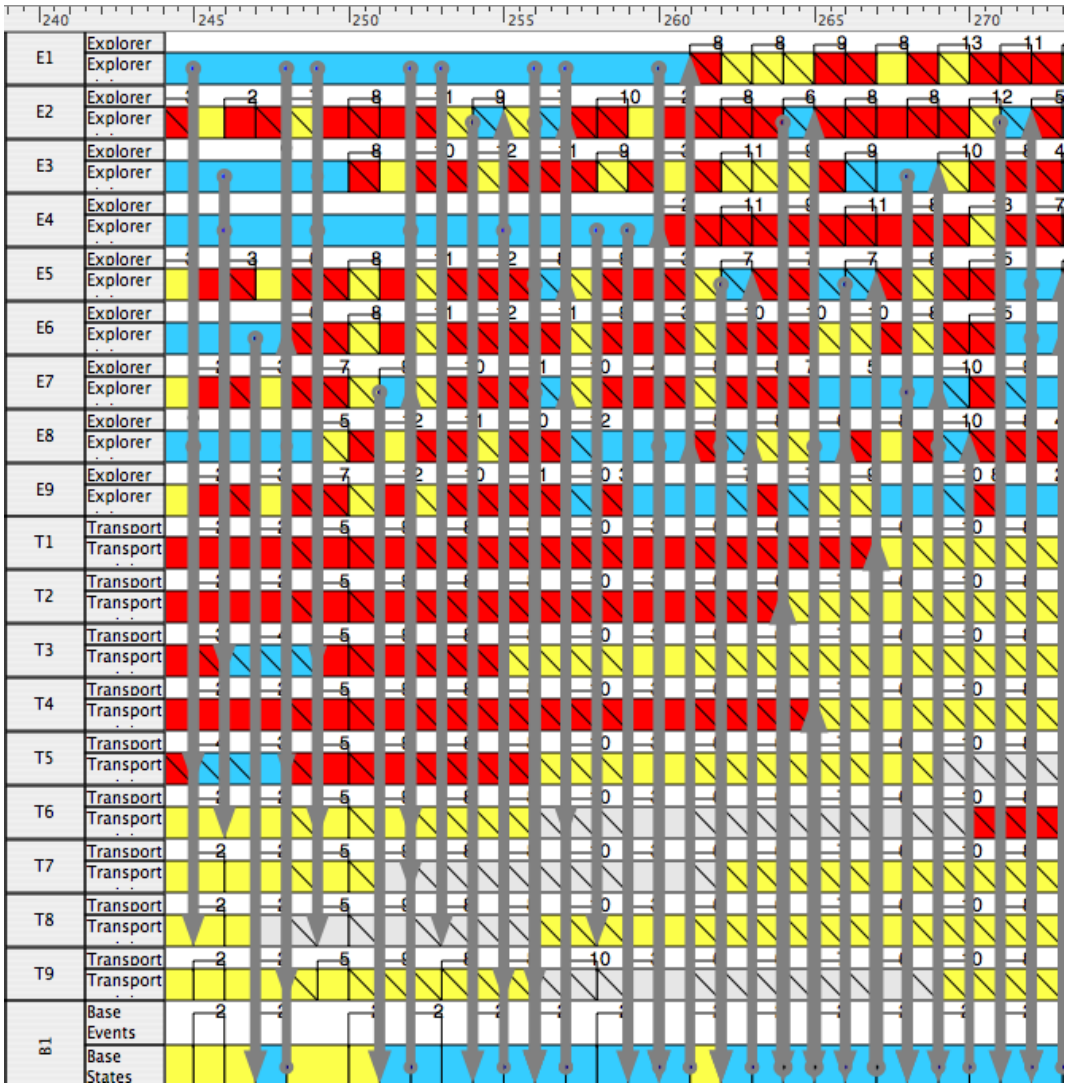

FIG. 8.3 – *Visualisation avec l'approche classique des communication dans l'application de minerai*

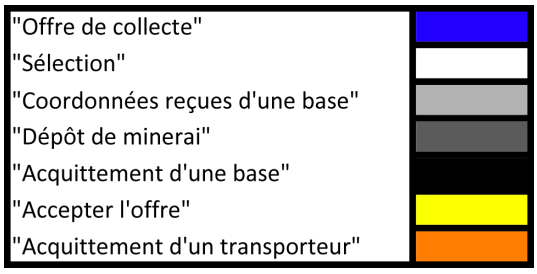

FIG. 8.4 – *Les couleurs associées aux types de messages*

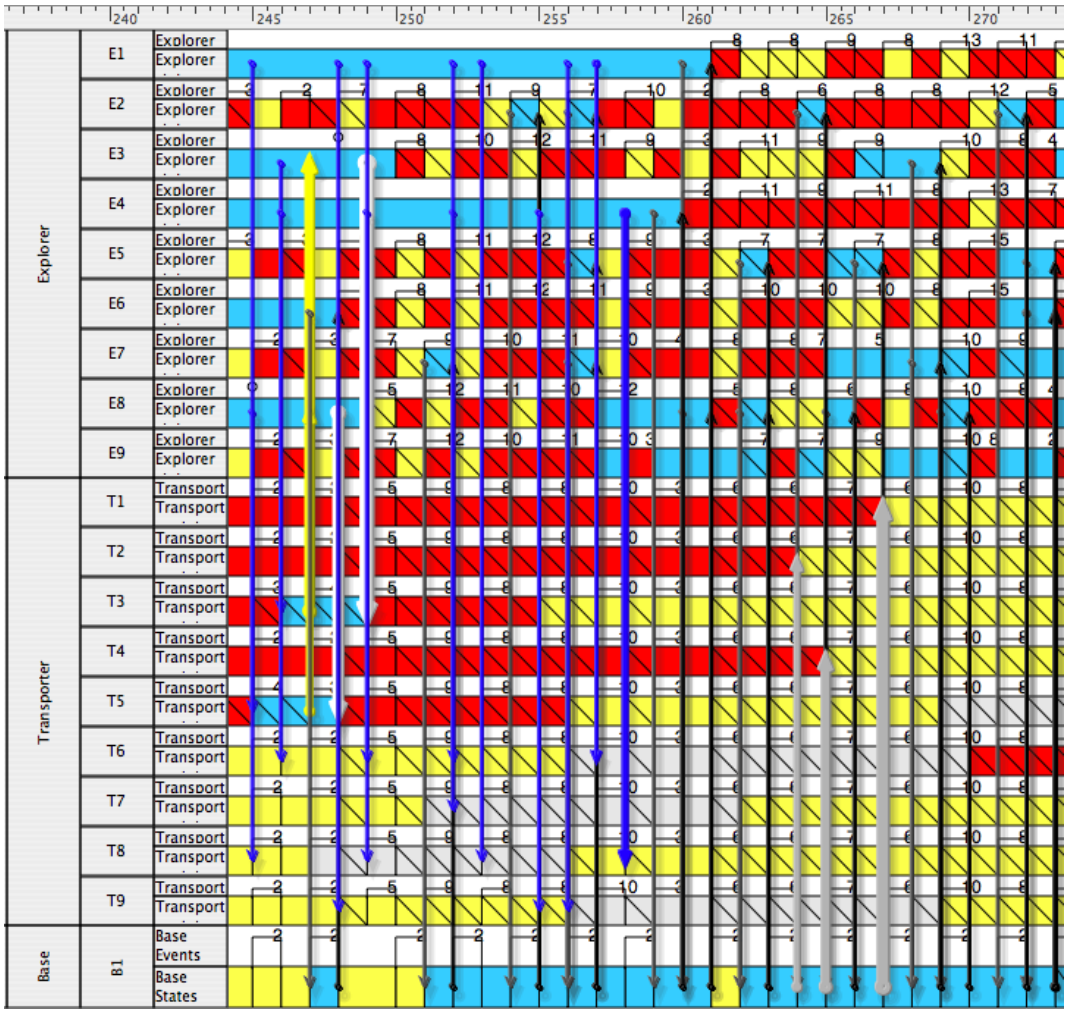

FIG. 8.5 – *Visualisation des communications dans l'application de minerai avec l'approche type*

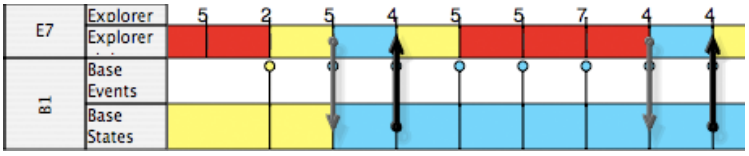

FIG. 8.6 – *Scénario de communications entre l'explorateur E1 et la base*

Le scénario présenté explique le résultat obtenu dans le chapitre précédent. Les transporteurs après leurs lancements ignorent tous les messages des explorateurs. Les explorateurs seront obligés d'envoyer les données à la base. L'application devient de plus en plus centralisée.

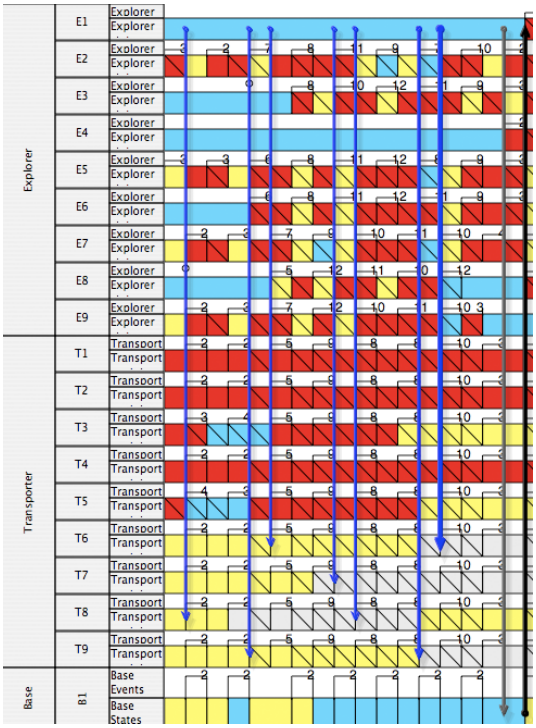

Fig. 8.7 – *Scenario de communication entre un explorateur essayant de delivrer ses minerais à un transporteur*

Sur la figure 8.8, l'explorateur E3 envoie un offre au transporteur T3 qui répond positivement. L'explorateur attend une période prédéfinie pour savoir s'il y a d'autres agents intéressés par l'offre. Après cette période, l'explorateur affecte la mission au transporteur T3. Sur ce scénario, nous pouvons remarquer l'importance des couleurs pour distinguer les messages envoyés par les agents.

Sur l'approche type, le poids associé à deux messages "*offre de collecte*" de l'ensemble $TYPE^{Transporteur}$ à deux instants différents reste pareil, même si on a pu démontrer sur la visualisation de l'exécution qu'ils n'ont pas le même effet sur l'agent.

Visualisation avec l'approche poids

Nous appliquons ici l'approche qui associe le poids du message aux résultats de traitement, en associant des caractéristiques aux connaissances des agents et aux actions déclenchées. L'ensemble des connaissances possibles des agents

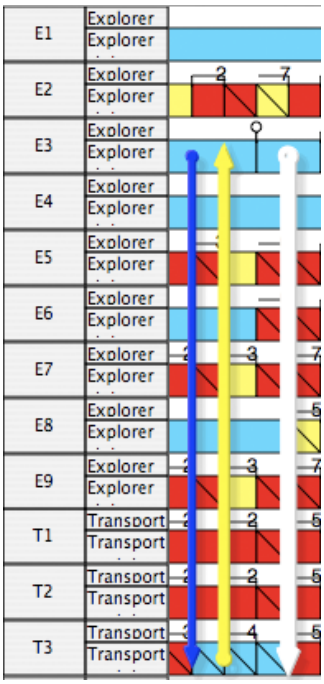

FIG. 8.8 – *Scénario des communications entre un explorateur et un transporteur*

présentés dans ce chapitre sont : $C^{Explorateur}$, $C^{Transporteur}$ et C^{Base}. L'ensemble des actions possibles des agents sont : $AC^{Explorateur}$, $AC^{Transporteur}$ et AC^{Base}

La figure 8.9 présente la visualisation des poids des informations portées par les messages. Le poids est désigné sur la figure par l'épaisseur de la flèche. L'ajout de poids permet d'identifier visuellement la quantité d'informations portée par un message. Sur cette figure nous remarquons que les messages contenant l'information de la base aux transporteurs bloqués (T4, T2, T1) contiennent un maximum d'information. Ce qui est expliqué par le fait que les coordonnées de minerais reçues par la base sont envoyées à ces transporteurs.

La figure 8.10 présente la variation du nombre de messages reçus par l'agent explorateur E1 durant sa vie. Cette figure montre la possibilité d'associer des variables au comportement des agents pour les utiliser dans la compréhension des comportements des agents. Pour plus des détails, nous présentons cette même quantité pour les agents explorateurs E1 et E3 et l'agent transporteur T6 durant la même période de visualisation des figures précédentes 8.11.

Conversations

La figure 8.12 présente une conversation extrait de la visualisation de l'exécution. Cette conversation suit le protocole présenté dans la figure 8.1. Les couleurs des flèches de la visualisation réfèrent au même couleur utilisés dans

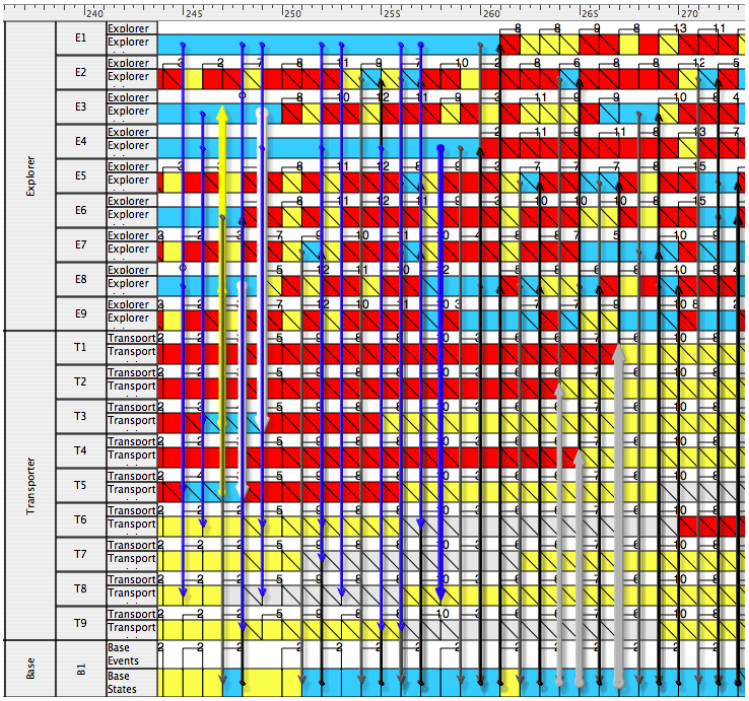

FIG. 8.9 – *Visualisation des communications dans l'application de minerai avec l'approche poids des informations portées par les messages*

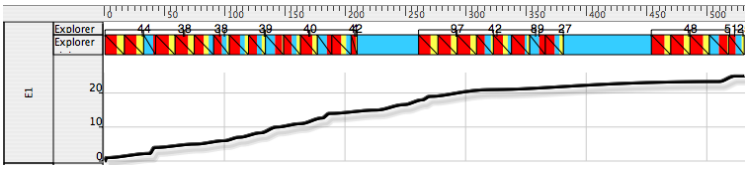

FIG. 8.10 – *Scénario des communication de partage des informations entre les transporteurs*

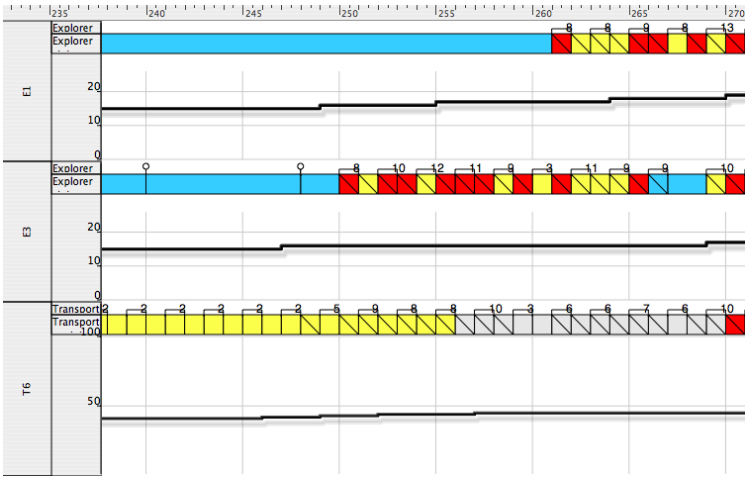

FIG. 8.11 – *Scénario des communication de partage des informations entre les transporteurs*

la figure 8.1. D'autre conversations possibles suivant le même protocole sont dans les figure 8.7, 8.6 présentés précédemment. La première figure présente le cas où le transporteur ignore l'offre. La deuxième présente le cas de l'envoi des données à la base.

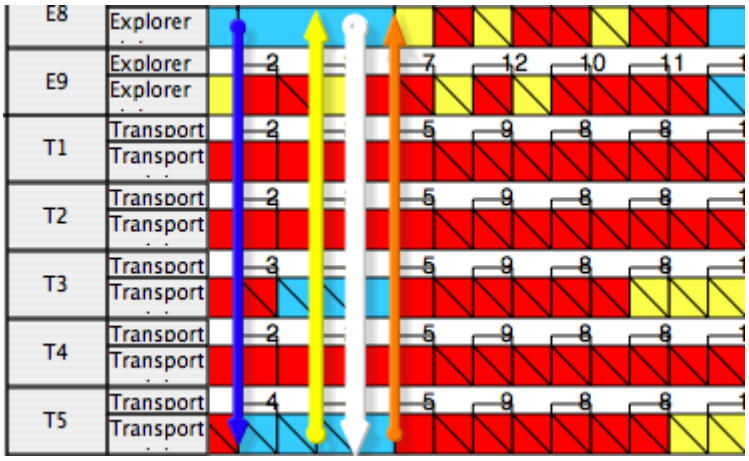

FIG. 8.12 – *Conversation entre un explorateur et un transporteur*

Pour conclure ce chapitre, nous avons présenté le rôle des approches présentés dans le chapitre 4 dans la détection et la compréhension des problèmes dans l'exécution d'un système multi-agents. Nous avons montré l'importance du couplage entre la modélisation de la communication et la visualisation de l'exécution.

Dans ce chapitre, la quantification et la mesure numérique de la communication n'est pas détaillé. Une partie de ce travail a été présenté dans le chapitre

4. Le chapitre est destiné à la présentation de l'importance de nos approches de modélisation de la communication dans la visualisation de l'exécution des SMA.

Chapitre 9

Évaluation du travail

Dans ce travail, nous avons présenté un modèle de visualisation de comportement à l'interieur des SMA. Le modèle a été complété par une dimension traitant des communications entre les agents. L'implémentation de ce modèle et l'application sur les SMA sont réalisées par l'intermédiaire de notre système de visualisation MAS-Paje.

Dans ce chapitre, nous présentons une évaluation du travail réalisé. Nous présentons les choix adoptés durant le travail et nous discutons les extensions possibles au niveau de la théorie et de la technique.

Ce chapitre est divisé en 4 parties. Dans la première partie, nous exposons une évaluation du modèle de visualisation présenté dans le chapitre 3. Dans la deuxième partie, nous présentons une évaluation du modèle de l'évaluation de communication présente dans le chapitre 4. Dans la troisième partie, nous présentons une évaluation de l'architecture de MAS-Paje présentée dans les chapitres 5 et 6. Finalement, nous concluons ce chapitre.

9.1 Évaluation du modèle de visualisation

Dans cette partie, nous présentons une évaluation du modèle de visualisation présenté dans le chapitre 3. Au niveau de l'abstraction des SMA pour la visualisation de l'exécution, nous considérons que l'ensemble des états comportementaux de l'agent est un ensemble d'états finis. Cette restriction etait faite pour avoir un modèle de visualisation fonctionnel et réalisable dans le cadre éventuelle d'une thèse. Clairement, ce modèle ne couvre qu'une classe des SMA. Une partie des systèmes multi-agents, par exemple qui considère l'apprentissage des agents durant l'exécution d'un SMA, n'est pas couverte. Une extension du modèle pour que le graphe d'état ne soit pas construit statiquement au début mais durant l'exécution de l'agent est nécessaire. Cette considération permettant d'avoir un ensemble infini d'états possibles des agents. La

réalisation d'un graphe infini d'états dans le modèle de l'agent rend difficile l'analyse des résultats de la visualisation de l'exécution d'un SMA. Un travail supplémentaire dans ce cas est nécessaire pour l'analyse des résultats de visualisation.

Dans ce travail, nous avons utilisé un outil de visualisation utilisé pour les systèmes distribués/parallèles et nous l'avons adapté aux SMA. Nous nous sommes orientés vers l'utilisation des outils de visualisation développés et utilisés dans les systèmes distribués/parallèles. Cette orientation a été argumentée par le fait que la construction d'un outil de visualisation pour les SMA est une tâche complexe qui présente beaucoup de difficulté et rend le travail non réalisable dans la cadre d'une thèse d'évaluation des SMA en général. De plus, des questions peuvent être posées sur l'utilité d'un tel outil vu la possibilité d'adapter des outils présents dans des domaines similaires, vu la similarité de l'architecture des systèmes parallèles avec l'architecture du sous-domaine SMA, et sachant la difficulté de création d'un outil de visualisation dédié aux SMA dans le cadre de notre travail. une perspective générale de notre travail est de construire un outil de visualisation orienté SMA.

9.2 Évaluation du modèle de communication

Dans cette partie nous présentons une évaluation du modèle de l'évaluation de communication présenté dans le chapitre 4.

Dans ce travail, nous avons pris pour hypothèse que le seul moyen d'échange d'information entre les agents etait par l'intermédiaire des messages. Nous ne considérons pas les autres types d'interactions entre agents. Ce qui est encore une restriction de notre modèle. Cette restriction a pour but de mettre place le modèle avec les éléments initiaux de la visualisation. Une amélioration reste possible par l'integration des autres moyens d'échange d'informations dans le modèle.

Nous ciblons dans la thèse une approche indépendante du domaine d'application en essayant de minimiser l'effet de subjectivité dans l'évaluation. La subjectivité reste présente dans la façon d'affecter des poids aux caractéristiques pour les approches types et l'approche poids. Pour l'approche protocole de communication, l'application de cette approche reste la plus compliquée du fait de la difficulté de reconnaissance des protocoles durant l'exécution des SMA. Ce qui reste le problème majeur et ce qui est présenté dans la partie suivante.

9.3 Évaluation du modèle de l'architecture de MAS-Paje

Dans cette partie nous présentons une évaluation de l'architecture de MAS-Paje présentée dans les chapitres 5 et 6. Dans ce travail, nous avons proposée une solution logicielle pour l'horloge en essayant d'éviter les changements sur l'architecture du système étudié. Des algorithmes de corrections peuvent être ensuite utilisés pour la correction des traces [Maillet (1990)]. L'horloge globale est considérée dans notre cas comme un objet de l'environnement qui envoie d'une façon périodique des données de synchronisation aux horloges locales des composants. Ce choix est fait ainsi pour différentes raisons :

- Pour ne pas changer l'architecture du système étudié, ajouter un objet à l'environnement ne change pas l'architecture système ;
- L'environnement, où les agents sont situés, est créé avant tous les agents. Ce qui est nécessaire pour le lancement de l'horloge et que ça soit un objet dans l'environnement.

Ce choix n'est pourtant pas l'idéal. Des méthodes de correction sont nécessaires pour améliorer le travail. Des travaux à ce niveau sont réalisés par [Maillet (1996)] qui propose une évaluation statistique du surcoût dû à l'enregistrement des traces d'exécution et un algorithme pour parcourir ensuite le fichier des traces d'exécution afin de retrancher des valeurs d'horloge enregistrées ce surcoût.

Un autre problème est la reconnaissance des protocoles à partir des traces d'exécution. Des travaux ont traités ce problème pour les SMA comme dans [Fallah-Seghrouchni *et al.* (1998, 1999, 2000); Mazouzi *et al.* (2002)]. Ces travaux ont proposé une méthode formelle de reconnaissance des protocoles à partir de l'observation des SMA. Le travail réalisé dans la thèse doit être ajouté à ces travaux pour avoir une reconnaissance des protocoles dans les traces d'exécution.

9.4 Conclusion et discussion

Après avoir évalué et discuté le travail réalisé dans cette thèse, il est important de noter quelque points :

- Il faut penser dans la suite de ce travail à spécifier les entrées et les sorties des états et trouver une abstraction indépendante du domaine d'application pour les modéliser. Cette modélisation peut être faite avec un langage de description des systèmes comme les réseaux de Petri par exemple. Ce travail peut être le sujet d'une autre thèse.

- Le passage du niveau micro présenté dans la visualisation dans ce travail à une synthèse niveau macro est important pour tirer l'utilité globale du SMA. Ce passage peut être réalisé par la définition d'une fonction de mesure quantitative au niveau macro à partir des indices de niveau micro.
- Une idée fondamentale pour utiliser les résultats de la visualisation est d'élaborer une métrique représentant l'utilité globale de comportement de l'agent et du SMA en générale à partir de la visualisation.
- Le travail de visualisation qu'on a fait, pourrait être complété par une partie vérification permettant d'étudier comment utiliser la visualisation pour verifier les SMA.
- Une visualisation 3D des SMA est une tâche importante à réaliser surtout que des outils de visualisation comme Paje ont édité une version 3D. Une tel visualisation permet de représenter plus de caractéristiques d'un SMA.

Chapitre 10

Conclusion générale

Dans ce manuscrit, nous avons présenté nos travaux relatifs à l'évaluation des systèmes multi-agents par l'utilisation de la technique de visualisation.

Il s'agissait d'utiliser des techniques d'évaluation utilisées pour des systèmes possédant des architectures similaires aux SMA et d'adapter ces techniques aux SMA. La technique de visualisation largement utilisée pour les systèmes parallèles/distribués a été adaptée aux SMA et nous avons démontré l'importance de la visualisation pour la compréhension des comportements internes des entités dans les SMA et pour révéler des indices servant à l'évaluation de ces systèmes.

La technique de visualisation comporte un certain nombre d'étapes dans sa réalisation : la collecte des données, l'analyse des données, la présentation des données. La présentation des données nécessite la présence d'un outil de visualisation. Cet outil doit posséder un certain nombre de caractéristiques pour être candidat à la visualisation des SMA.

Du fait du grand nombre d'outils de visualisation présents pour visualiser d'autres systèmes informatiques et la difficulté de création d'un outil spécifique pour les SMA, nous nous sommes orientés vers l'utilisation des outils utilisés pour les systèmes parallèles/distribuées. Ce choix a été fait pour la raison principale que ces systèmes possèdent une architecture similaire au SMA (système ATHASPAN par exemple). Parmi ces outils, nous avons choisi l'outil de visualisation Paje. Après l'avoir comparé à d'autres outils candidats, il apparaît que cet outil répond le mieux aux critères demandés pour la visualisation des SMA.

Le choix de l'outil de visualisation ayant été fait, nous avons présenté un système de visualisation des SMA, MAS-Paje. Ce système traite des étapes de visualisation des SMA fondées sur un modèle prédéfini à base d'états connus. MAS-Paje se charge des étapes de collecte des données et de l'analyse des

données dans le but d'obtenir un fichier de trace de l'exécution conforme au format de trace (SDDF) exigé par l'outil de visualisation Paje.

Le comportement individuel des agents ne suffit pas pour comprendre et évaluer le comportement global du système. Une dimension communication est ajoutée au travail de visualisation dans le but d'étudier le travail collectif des agents. Plusieurs approches d'étude et de visualisation des communications sont présentées dans notre travail.

Les travaux sont finalement validés par des expérimentations sur l'application de collecte du minerai. Dans ces expérimentations, nous avons présenté l'utilité de la visualisation pour obtenir des informations sur le comportement des agents, la communication entre agents et l'utilisation des ressources par les différentes entités du SMA. Ces informations ont servi à détecter des problèmes dans l'architecture interne des SMA et ont permis d'extraire des schémas significatifs de l'exécution. Dans un deuxième temps, nous avons utilisé ces informations pour comparer différents SMA résolvant le même problème de collecte du minerai.

Annexe A

AM 24 - MULTI-AGENT SYSTEMS EXAMINATION

UNIVERSITY OF SOUTHERN DENMARK
AM 24 - MULTI-AGENT SYSTEMS EXAMINATION
ODENSE, AUGUST 2008

On a 2-dimensional grid G representing the surface of a planet, N bases land randomly to collect ore samples. The planet is a torus. The ore is distributed on the planet with a density D. Coordination mode M of the bases is either cooperative when the bases are belonging to the same company, or competitive when they involve different companies. Each base carries societies of robots consisting in X explorers and Y transporters. Robots can move on the grid in the eight directions. Two robots cannot be located at the same time at the same place. Each robot has a limited perception scope P ($1 < PxP < G/100$) and a limited communication scope. This also applies to the bases. Both scopes can be tuned by the robots themselves. When an explorer is at the same position of a sample of ore, it can acquire its coordinates. Explorers can send coordinates of an ore sample it has detected. Robots have a limited size memory to record a maximum of S ($S < X+Y$) coordinates. When a transporter is at the same position of a sample of ore, it can pick the ore sample. Transporter can transport samples back to the base and deposit them here. C of the base is limited. Robots have a limited capacity W of ore sample they can grab and/or carry at a given time. The capacity At each cycle, each robot executes only one action: a message sending, a perception requesting, a body moving, an ore sample picking, ... Robots have limited energy that they consume when acting: a perception request is costing (P) units when a message sending is costing 1 unit, whilst robot motion is the far most expensive action among all possible actions. A robot simply dies when its energy is consumed in its batteries. A robot can acquire additional energy at the base. When the base is full of ore or after a given time T, robots alive return to the base.

Delivery of the solution provided by your system should contain the quantity of ore collected, the time spent for it, and the percentage of robots alive.

11-13 August

- Install Madkit (see www.madkit.org) and become familiar with it through online documents and available examples. In case you find it appropriate, you are allowed to purposively reuse parts of the code you may find there.

14-15 August

- Analyse the domain and the problem of this foraging robots application, and extend specifications if needed. Justify your extensions.
- Perform the analysis and design for the requested MAS. Pure reactive MAS are forbidden, which means that some agent reasoning is expected in your system. Explicit this part in your final report.
- Discuss the A, the E, the I, the O, and the dynamics models you are selecting (cf. lectures until 15 August). In particular:
  - For every type of agent you will define, you should report the state graph describing its life cycle.
  - You should explain every additional variable you consider as being particularly interesting for each type of agents, what it does represent, and how it should evolve during the excution of the your program.
- Implement your first MAS system with MadKit, and experiment it until it runs adequately. To help in the debug and testing,
- Each measurable quantity (time cycle, ore quantity ...) should be associated with an external variable.
- The input user interface of your software should permit to initialize - and store in an external file - the initial values of: C (capacity of each base in number of ore samples), D (density of ore, as a percentage), M (coordination mode between bases, cooperative = 1 or competitive = 0), N (number of bases), P (perception scope), S (memory size of each robot), T (maximal number of cycles), W (maximal number of ore a robot can grab and/or carry) X (number of explorers per base), Y (number of transporters per base), and the ones of the additional important variables you have defined.
- The code should contain the becessary flags and comments indicating when agents enter and leave their states, the localization and the activation of the communicating primitives.

- The output user interface should generate a file containing the results, including the values of the quantity of ore collected, the time spent for it, and the percentage of robots alive, as well as the values of every variable you have identified as being important for your design.

18-19 August

- Experiment the ratio X explorers / Y transporters and how it affects convergence.
- Experiment several energy functions of the robots and how they affect convergence.
- Experiment the evolution of communication and perception scopes and how this affects convergence.
- Revise your first MAS system with MadKit, experiment it until it runs adequately for one base.

21-22 August

- Experiment the extension of the system when several lunar ships are cooperating to collect the ore.
- Experiment the extension of the system when several lunar ships are competing to collect the ore.
- Revise the analysis and design for the revised MAS. Pure reactive MAS are forbidden, which means that some agent reasoning is expected in your system. Explicit this part in your final report.
- Implement your final MAS system with MadKit, experiment it until it runs adequately for several bases. Be careful that variables and entities are precisely identified in the code.
- Write a 15-20 pages (maximum length, including figures) PDF report including analysis and design, final models, experiments, and extensions. Take care that the agent architectures as well as interaction and organisation structures are clearly reported and argued. Send your PDF report to Y. Demazeau (lecturer), H. Joumaa (assistant), and C. Risager (examiner), before 21 August 4:00 pm.
- Send your MAS software, meaning the executable code, your observations and the results of the experimentation, to Y. Demazeau, H. Joumaa, and C. Risager, before 22 August 4:00 pm. Within the same message, propose and argue a set of recommended values for W, X and Y with (X+Y)*W = 20 for the given setup of C = 200, D = 5

26 August

– Between 1 pm and 4 pm, be prepared to deliver a 30 mn presentation of your work with support of slides. Your presentation will be public. The presentation will be followed by a set of individual questions asked to students. Send your PDF presentation to Y. Demazeau, H. Joumaa, and Claus Risager, before 25 August 4:00 pm. You are not allowed to modify your slides after this deadline.

Bibliographie

Y. Arrouye : *Environnements de visualisation pour l'évaluation des performancesdes systèmes parallèles : étude, conception et réalisation*. Thèse de doctorat, Institut National Polytechnique de Grenoble, 1995.

R.-A. Aydt : The pablo self-defining data format. *In Rapport technique, University of Illinois at Urbana Champaign*, 1992.

P. Bayer et M. Svantesson : Comparison of agent-oriented methodologies analysis and design, mas-commonkads versus gaia. *Blekinge Institute of Technology, Student Workshop on Agent Programming*, p. 0–1, 2001.

F. Bellifemine, A. Poggi et G. Rimassa : Developing multi-agent systems with jade. *In Intelligent Agents*, 2001.

P. Berreur : Evaluer les systèmes multi-agents. Rap. tech. Pellucid 5FP IST-200134519, Université Joseph Fourier, juin 2005.

H. Bincheng, L. Jiming et J. Xiaolong : From local behaviors to global performance in a multi-agent system. *In Intelligent Agent Technology*, p. 0–1, 2004.

G. Bonnet et A. Tessier : Evaluation d'un système multiagent physique : retour sur expérience. *In Journées Francophones sur les Systèmes MultiAgents (JFSMA 2008)*, p. 599–604, 2008.

P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos et A. Perini : Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, p. 0–1, 2003.

P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos et A. Perini : Tropos: An agent-oriented software development methodology. *In Journal of Autonomous agents and Multiagent Systems*, 2004.

S. Browne, J. Dongarra et K. London : Review of performance analysistools for mpi parallel programs. *In Proceedings of the 8th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, p. 0–1, 2001.

L. Cernuzzi et G. Rossi : On the evaluation of agent oriented modeling methods. *In Agent Oriented Methodology Workshop*, p. 0–1, 2002.

J. ChassinDeKergommeaux, . Maillet et J. m. Vincent : Monitoring parallel programs for performance tuning in cluster environments. *In Pa-*

rallel Program Development for Cluster Computing: Methodology, Tools and Integrated, p. 131–150, 2001.

A.-L. Couch : Categories and context in scalable execution visualization. *Journal of Parallel and Distributed Computing*, p. 195–204, 1993.

M. T. Cox et S. DeLoach : Multiagent systems and mixed initiative planning course. *Wright State University, http://www.cs.wright.edu/people/faculty/mcox/Teaching/Cs790/*, p. 0–1, 2000.

P. Davidsson et S. Johansson : Evaluating multi-agent system architectures: A case study concerning dynamic resource allocation. *In Third International Workshop on Engineering Societies in the Agents*, p. 0–1, 2002.

S. A. DeLoach, M. F. Wood et C. H. Sparkman : Multiagent systems engineering. *The Intl. Jour. of SE and KE*, p. 0–1, 2001.

Y. Demazeau : From interactions to collective behaviour in agent-based systems. *In First European conference on cognitive science*, p. 0–1, 1995.

Y. Demazeau : Mas project subject 2004. 2004a.

Y. Demazeau : Systmes multi-agents. *Volume ARAGO n29, OFTA, Lavoisier*, p. 0–1, 2004b.

Y. Demazeau et A. R. Costa : Populations and organisations in open multi-agent systems. *In First Symposium on parallel and Distributed Artificial Intelligence*, p. 0–1, 1996.

M.-P. G. E. Kaddoum, P. G. J.-P. Georgé et G. Picard : Analyse des critères d'évaluation de systèmes multi-agents adaptatifs. *In JFSMA*, p. 0–1, 2009.

A. E. Fallah-Seghrouchni, S. Haddad et H. Mazouzi : Etude des interactions basée sur l'observation répartie dans un système multi-agents. *In Actes des JFIADSMA'98*, p. 89–101, 1998.

A. E. Fallah-Seghrouchni, S. Haddad et H. Mazouzi : Protocol engineering for multi-agent interaction. *In MAAMAW*, p. 89–101, 1999.

A. E. Fallah-Seghrouchni, S. Haddad et H. Mazouzi : A formal study of interaction in multi-agent systems. *In International Journal of Computers and their Applications*, p. 89–101, 2000.

J. Ferber : Les systèmes multi-agents: vers une intelligence collective. *In InterEdi- tions*, p. 0–1, 1995.

T. Finin, R. Fritzson, D. Mckay et R. Mcentire : Kqml as en agent communication language. *In 3rd international conference on iformation and knowledge managment*, p. 0–1, 1994.

H. Fiorino : Core-dms plate-forme pour la construction des sma. *In MAGMA - LEIBNIZ*, p. 0–1, 2003.

FIPA : Fipa 97 specification. part 2, agent communication language. *In HTTP://WWW.FIPA.ORG*, p. 0–1, 1997.

P. A. Fishwick : Computer simulation : growth through extension. *Transactions of the Society for Computer Simulation International*, p. 0–1, 1997.

G. Gaspar : Communication and belief changes in a society of agents: Towards a formal model of autonomous agent. *In D.A.I. 2*, p. 0–1, 1991.

C. Guilloud : *Traçage flexible d'exécutions de programmes parallèles.* Thèse de doctorat, Institut Nationale Polytechnique de Grenoble, 2004.

A.-H. Hayes, M.-L. Simmons, J.-S. Brown et D.-A. Reed : Debugging and performance tuning for parallel computing systems. *IEEE Computer Society*, p. 0–1, 1996.

M.-T. Heath et J.-A. Etheridge : Visualizing the performances of parallel programs. *IEEE Trans. Softw. Eng.*, p. 29–39, 1991.

R. Hofmann : Monitoring and evaluation of parallel and distributed systems. *In European School on Parallel Programming Environments, ESPPE'96*, p. 135–153, 1996.

M. Hurfin, N. Plouzeau et M. Raynal : Erebus: a debugger for asynchronous distributed computing systems. *In Third Workshop on Future Trends of Distributed Computing Systems*, p. 93–98, 1992.

C. A. Iglesias, M. Garijo et J. C. Gonzalez : A survey of agent-oriented methodologies. *Intelligent Agent, Springer*, p. 0–1, 1999a.

C. A. Iglesias, M. Garijo et J. C. Gonzalez : A survey of agent-oriented methodologies, 1999b.

C. A. Iglesias, M. Garrijo, J. Gonzalez et J. R. Velasco : Analysis and design of multiagent systems using mas-commonkads. *Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages (ATAL)*, p. 0–1, 1998.

R. Jacobson, X.-J. Zhang, R. DuBose et B.-W. Matthews : Monitoring and evaluation of parallel and distributed systems. *In E.coli. Nature, vol. 369*, p. 761–766, 1986.

R. Jain : The art of computer systems performance analysis. 1991.

N. R. Jennings, K. Sycara et M. J. Wooldridge : A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, p. 0–1, 1998.

N. R. Jennings et M. Wooldridge : Agent-oriented software engineering. *Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press*, p. 0–1, 2000.

H. Joumaa : Evaluer les systemes multi-agents. Rap. tech. Pellucid 5FP IST-200134519, Université Joseph Fourier, juin 2006.

K. Jurasovic, G. Jezic et M. Kusek : A performance analysis of multi-agent systems. *In International Transactions on Systems Science and Applications*, p. 0–1, 2006a.

K. Jurasovic, G. Jezic et M. Kusek : A performance analysis of multi-agent systems, 2006b.

E. KRAEMER et J.-T. STASKO : The visualization of parallel systems : An overview. *Journal of Parallel and Distributed Computing*, p. 105–117, 1993.

Y. LABROU et T. FININ : A proposal for a new kqml specification. *In technical report*, p. 0–1, 1997.

J.-R. LARUS : Efficient program tracing. *IEEE Computer*, p. 0–1, 1993.

R. LESZCZYNA : Evaluation of agent platforms. Rap. tech. Pellucid 5FP IST-200134519, Joint Research Centre, Institute for the Protection and Security of the Citizen, juin 2004.

E. LEU : *La réexecution, pierre angulaire de la mise au point des programmes parallèles.* Thèse de doctorat, Ecole Polytechnique Fédérale de Lausanne, 1992.

E. LEU et A. SHIPER : Execution replay: A mechanism for integrating a visualization tool with a symbolic debugger. *In Second Joint International Conference on Vector and Parallel Processing: Parallel Processing*, p. 55–66, 1992.

L. LEVROUW, K. AUDENAERT et J. V. CAMPENHOUT : A new trace and replay system for shared memory programs based on lamport clocks. *In Second Euromicro Workshop on Parallel and Distributed Processing*, p. 471–478, 1994.

E. MAILLET : *Performance Observability.* Thèse de doctorat, University of Illinois, 1990.

E. MAILLET : Issues in performance tracing with tape/pvm. *In Proceedings of EuroPVM'95*, p. 143–148, 1995.

E. MAILLET : *Traçage de logiciel d'applications parallèles : conception et ajustement de qualité.* Thèse de doctorat, Institut National Polytechnique de Grenoble, 1996.

A.-D. MALONY et H. W. A. REED : Performance measurement intrusion and perturbation analysis. *IEEE Transactions on parallel and distributed systems*, p. 0–1, 1992.

H. MAZOUZI, A. E. FALLAH-SEGHROUCHNI et S. HADDAD : Open protocol design for complex interactions in multi-agent systems. *In AAMAS*, p. 517–526, 2002.

J. M. MELLOR-CRUMMEY : *Debugging and analysis of large-scale parallel programs.* Thèse de doctorat, University of Rochester, 1989.

T. MEURISSE : *Simulation multi-agent : du modèle à lÕopérationnalisation.* Thèse de doctorat, Université Pierre et Marie CURIE, 2004.

B. P. MILLER : What to draw? when to draw? an essay on pa- rallel program visualization. *Journal of Parallel and Distributed Computing*, p. 265–269, 1993.

B. P. MILLER, M. D. CALLAGHAN, J. M. CARGILLE, J. K. HOLLINGSWORTH, R. B. IRVIN, K. L. KARAVANIC, K. KUNCHITHAPADAM et T. NEWHALL : The paradyn parallel performance measurement tool. 1995.

L. Mulet, J. M. Such et J. M. Alberola : Performance evaluation of open-source multiagent platforms. *In AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, p. 1107–1109, New York, NY, USA, 2006. ACM. ISBN 1-59593-303-4.

J. Mylopoulos, M. Kolp et P. Giorgini : Agent-oriented software development. *In SETN*, p. 0–1, 2002a.

J. Mylopoulos, M. Kolp et P. Giorgini : Agent-oriented software development. *In SETN*, p. 599–604, 2002b.

R. Netzer et B. Miller : Optimal tracing and replay for debugging message-passing parallel programs. *In Supercomputing '92*, p. 502–511, 1992.

T. G. Nguyen et T. T. Dang : Agent platform evaluation and comparison. Rap. tech. Pellucid 5FP IST-200134519, Institute of Informatics, Slovak Academy of Sciences, juin 2002.

N. H. S. E. NHSE : Parallel tools library software catalog. *In http://www.cs.vu.nl/albatross/*, p. 0–1, 1999.

Q. nhu Numi Tran, G. Low et M. anne Williams : A feature analysis framework for evaluating multi-agent system development methodologies. *In In*, p. 613–617, 2003.

J. F. O. Gutknecht, F. Michel : The madkit agent platform architecture. Rap. tech. Pellucid 5FP IST-200134519, Université Montpellier 2, juin 2000.

B. Plateau, J. Briat, I. Ginzburg et M. Pasin : Athapascan runtime :efficiency for irregular problems. *In EURO-PAR97 Parallel Processing*, p. 0–1, 1997.

F. Post et A. Hin : Focus on computer graphics. *In* F. Post et A. Hin, éds : *Advances in Scientific Visualization*. Springer verlag, 1991.

D.-A. Reed : Performance instrumentation techniques for parallel systems. *In Performance Evaluation of Computer and Communication Systems*, p. 0–1, 1993a.

D.-A. Reed : Scalable performance analysis : The pablo performance analysis environment. *In Proceedings of the Scalable Parallel Libraries Conference*, p. 104–113, 1993b.

D.-A. Reed : Experimental performance analysis of parallel systems : Techniques and open problems. *In Proceedings of the 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, p. 25–51, 1994.

D.-A. Reed et R.-L. Ribler : Performance analysis and visualization. *In The Grid : Blueprint for a New Computing Infrastructure, éd. par Foster (I.) et Kesselman (C.)*, p. 0–1, 1998.

R. Ribler, A. Mathur et M. Abrams : Visualizing and modeling categorical time series data. *In Rapport technique, Virginia Polytechnic Inst. and State University*, p. 0–1, 1995.

A. Sabas, S. Delisle et M. Badri : A comparative analysis of multiagent system development methodologies: Towards a unified approach. *In In Third International Symposium From Agent Theory to Agent Implementation (AT2AI-3*, p. 599–604, 2002.

R. E. Shannon : Introduction to the art and science of simulation. *Proceedings of the 30th conference on Winter simulation, IEEE Computer Society Press*, p. 0–1, 1998.

B. Stein et J. C. Kergommeaux : Interactive visualization environment of multi-threaded parallel programs. *In Parallel Computing : Fundamentals, Applications and New Directions*, p. 131–150, 1998.

A. Sturm et O. Shehory : A framework for evaluating agent-oriented methodologies. *In Proc. of the Int. Bi-Conference Workshop on Agent-Oriented Information Systems, AOIS 2003, volume 3030 of LNCS*, p. 94–109. Springer, 2003.

J. Sudeikat, L. Braubach, A. Pokahr et W. Lamersdorf : Evaluation of agent - oriented software methodologies - examination of the gap between modeling and platform. *In* P. Giorgini, J. P. Müller et J. Odell, éds : *Agent-Oriented Software Engineering V, Fifth International Workshop AOSE 2004*, p. 126–141. Springer Verlag, 7 2004.

R. Trillo, S. Ilarri et E. Mena : Comparison and performance evaluation of mobile agent platforms, 2007.

M. Wooldridge et P. Ciancarini : Agent oriented software engineering: the state of the art. *In First international Workshop on Agent-Oriented Software Engineering*, p. 0–1, 2000.

M. Wooldridge et N. Jennings : Intelligent agents: Theory and practice. *In The Knowledge Engineering Review*, p. 0–1, 1995.

M. Wooldridge, N. R. Jennings et D. Kinny : The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi Agent Systems*, p. 0–1, 2000.

B. J. N. Wylie et A. Endo : Annai/pma multi-level hierarchical parallel program performance engineering. 1996.

J.-C. Yan : Performance tuning with aims an automated instrumentation and monitoring system for multicomputers. *In Proc. of the Twenty-Seventh Annual Hawai Conference on System Sciences*, p. 625–633, 1994.

ZEUS : Isr agent research - zeus. http://www.labs.bt.com/projects/agents/zeus/, November 2000.

L. D. R. Y. Zhang et D.-A. Reed : Svpablo : A multi-language performance analysis system. *In Lecture Notes in Computer Science*, p. 131–150, 1998.

Q. A. Zhao et J. T. Stasko : Visualizing the execution of threads-based parallel programs. *Rapport technique nNo GIT-GVU-95-01, Georgia Institute of Technology*, p. 0–1, 1995.