Modèles et Architectures d'Agents et de Systèmes Multi-Agents Adaptatifs

Dossier d'habilitation à diriger des recherches

de l'Université Pierre et Marie Curie

Spécialité : Informatique

Zahia GUESSOUM

Laboratoire d'Informatique de Paris 6

Soutenance prévue le 5 décembre devant le jury composé de :

Pascal ESTRAILLIER	Rapporteur
Jacques FERBER	Rapporteur
Les GASSER	Rapporteur
Yves DEMAZEAU	Examinateur
Christian LEMAITRE	Examinateur
Jacques MALENFANT	Examinateur
Jean-Pierre BRIOT	Directeur

Remerciements

à compléter.

Résumé :

La complexité des nouveaux systèmes d'information (par exemple les systèmes accessibles depuis l'Internet) et les applications émergentes récentes (*Grid computing*, services Web, ...) s'est trouvée considérablement accrue du fait de : leur distribution, la grande quantité d'informations qu'ils manipulent, leur aspect coopératif et adaptatif, et leur ouverture. Ces systèmes d'information et ces applications ont toutes les caractéristiques d'un bon domaine d'application des systèmes multi-agents. Ils permettent en effet de valider et de montrer les limites des modèles et architectures multi-agents proposés.

Les connaissances nécessaires à la gestion et au contrôle des systèmes complexes sont très nombreuses, il est long et difficile de toutes les exprimer en les donnant directement au système. La modélisation de ces systèmes complexes nécessite donc des agents adaptatifs. Les agents doivent en effet être capables de réagir aux changements et perturbations de leur environnement qui est complexe et dynamique. L'adaptation du comportement ou de la structure interne des agents permet ainsi d'acquérir dynamiquement les connaissances qui n'ont pas été fournies par le concepteur. Cependant, cette adaptation entraîne l'émergence de comportements globaux qui sont parfois indésirables. Contrairement aux simulations multi-agents où les phénomènes émergents sont détectés par un observateur externe, ces systèmes multi-agents ont plusieurs caractéristiques qui les rendent plus difficiles à observer que les systèmes mono-agents ou les simulations. Ils sont souvent ouverts, distribués, hétérogènes et à large échelle.

Le système doit être en effet capable de s'auto-observer afin de détecter les phénomènes émergents et adapter son comportement et sa structure à ces phénomènes émergents et à l'évolution de son environnement. La modélisation de ces systèmes complexes nécessitent ainsi l'adaptation au niveau local (micro) et au niveau global (macro).

Dans ce dossier, j'ai choisi de détailler trois principaux sujets : architectures d'agents, systèmes multi-agents tolérants aux pannes et simulation de modèles économiques. Ces trois sujets résument mes travaux sur les agents et systèmes multi-agents adaptatifs. Un premier chapitre présente mes travaux sur les architectures d'agents et montre que les architectures modulaires permettent de dépasser la dichotomie classique : agents réactifs et agents cognitifs. Ce chapitre montre également que la modularité facilite le développement d'agents adaptatifs en s'appuyant sur l'exemple d'agents interactifs qui peuvent adopter et exécuter dynamiquement des protocoles d'interaction. Le but des deuxième et troisième chapitres est de montrer l'intérêt des systèmes multi-agents adaptatifs et d'essayer de définir l'architecture de ces systèmes. Le deuxième chapitre décrit le mécanisme de réplication adaptatif que nous avons introduit pour fiabiliser les systèmes multi-agents à large échelle. Le troisième chapitre résume nos travaux sur la simulation de modèles économiques. Il décrit le modèle de firmes ainsi que le modèle de formes organisationnelles que nous avons élaborés. Il définit ensuite les interactions entre les firmes et les formes organisationnelles. Le quatrième chapitre résume nos réflexions sur les systèmes multi-agents auto-adaptatifs et montre l'intérêt de l'adaptation pour la modélisation de systèmes complexes. Il donne ensuite un aperçu de mes projets en cours ainsi que mes perspectives de recherche.

Mots clés : Agents, systèmes multi-agents, adaptation, modèles, architectures, systèmes complexes.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction générale

1.1 Agents et systèmes multi-agents

Les techniques d'intelligence artificielle ont été appliquées, avec succès, pour résoudre des problèmes de diagnostic, de conception et de classification. En revanche, leur application au contrôle ou à la simulation de processus dynamiques complexes tels que les écosystèmes, la robotique, le monitorage de procédés industriels, la surveillance de patients en soins intensifs ou l'évolution économique, soulève de nombreux problèmes et fait encore l'objet de diverses recherches. Elles sont souvent mal adaptées à la modélisation des systèmes comportant plusieurs entités qui interagissent et évoluent dans un environnement dynamique.

Pour aborder des problèmes complexes, un effort particulier a été porté ces dernières années sur l'intelligence artificielle distribuée et les systèmes multi-agents ([42], [36], [12]). Les systèmes multi-agents s'appuient sur la métaphore de l'organisation collective, par opposition à l'intelligence artificielle classique qui s'appuie sur la métaphore du penseur isolé. Ils proposent de modéliser des systèmes complexes à l'aide d'une société d'entités appelées agents. Chaque agent a ses propres compétences, mais il a besoin d'interagir avec les autres pour résoudre les problèmes qui dépendent de son domaine d'expertise et éviter les conflits. L'objectif de ces systèmes est donc de trouver une solution à des problèmes globaux ou de simuler des comportements complexes à l'aide d'un ensemble d'agents ayant les propriétés suivantes :

- autonomie : l'agent agit sans l'intervention des humains ou des autres agents, et il contrôle ses actions en fonction de son état interne et de son environnement.
- proactivité : l'agent a en effet sa propre activité et son propre but. Contrairement aux objets dont l'action est simplement une réponse aux messages reçus d'autres objets, l'agent lui, a une activité dirigée vers son but.
- adaptation : l'agent est capable de réguler ses aptitudes (communicationnelles, comportementales, etc.) en fonction de l'agent avec lequel il interagit et/ou de l'environnement dans lequel il évolue. Autrement dit, un agent adaptatif a la capacité de modifier les propriétés de ses différentes tâches afin de satisfaire les demandes internes et externes.

Les systèmes multi-agents permettent de modéliser des systèmes hétérogènes, complexes, dynamiques, non linéaires et évolutifs, et de faire apparaître une intelligence et des capacités qui sont différentes et globalement supérieures à celles des agents qui les composent. Cette intelligence émerge de la coexistence et de la coopération d'agents plus ou moins autonomes. Dans l'approche informatique classique de résolution de problème, la tâche globale est décomposée en sous-tâches et le programme code les différentes étapes de résolution. Ces différentes étapes consistent à parcourir les chemins prédéfinis pour chercher la solution. Dans cette approche, le programme code les agents et les interactions en se basant sur un modèle organisationnel. Les agents interagissent dans l'environnement et la solution émerge de ces interactions.

Dans la majorité des applications existantes telles que l'application Manta de Drogoul [27] et la gestion de crises (voir les travaux de Cardon [16]) la simulation multi-agents offre un outil puissant pour expliquer les changements au sein d'un système naturel ou artificiel en fonction des changements des individus qui le composent. Les phénomènes émergents peuvent être détectés et expliqués par un observateur externe qui est l'utilisateur de la simulation. La détection de ces phénomènes émergeants repose souvent sur l'interprétation par l'observateur externe des résultats de simulation [89].

1.2 Agents vs. systèmes multi-agents adaptatifs

La complexité des nouveaux systèmes d'information (par exemple les systèmes accessibles depuis l'Internet) et les applications émergentes récentes (*Grid computing*, services Web, ...) s'est trouvée considérablement accrue du fait de : leur distribution, la grande quantité d'informations qu'ils manipulent, leur aspect coopératif et adaptatif, et leur ouverture. Ces systèmes d'information et ces applications ont toutes les caractéristiques d'un bon domaine d'application des systèmes multi-agents. Ils permettent en effet de valider et de montrer les limites des modèles et architectures multi-agents proposés.

Les connaissances nécessaires à la gestion et au contrôle des systèmes complexes sont très nombreuses, il est long et difficile de toutes les exprimer en les donnant directement au système. La modélisation de ces systèmes complexes nécessite donc des agents adaptatifs. Les agents doivent en effet être capables de réagir aux changements et perturbations de leur environnement qui est complexe et dynamique. L'adaptation du comportement ou de la structure interne des agents permet ainsi d'acquérir dynamiquement les connaissances qui n'ont pas été fournies par le concepteur. Cependant, cette adaptation entraîne l'émergence de comportements globaux qui sont parfois indésirables. Contrairement aux simulations multi-agents où les phénomènes émergents sont détectés par un observateur externe, ces systèmes multi-agents ont plusieurs caractéristiques qui les rendent plus difficiles à observer que les systèmes mono-agents ou les simulations. Ils sont souvent ouverts, distribués, hétérogènes et à large échelle.

Le système doit être en effet capable de s'auto-observer afin de détecter les phénomènes émergents et adapter son comportement et sa structure à ces phénomènes émergents et à l'évolution de son environnement. La modélisation de ces systèmes complexes nécessitent ainsi l'adaptation au niveau local (micro) et au niveau global (macro).

1.3 Contributions scientifiques

Mes thèmes de recherche se sont principalement organisés autour du domaine des systèmes multi-agents adaptatifs. L'objectif général de mes projets de recherche est de proposer des modèles et des architectures multi-agents adaptatifs pour modéliser, implémenter et déployer des systèmes complexes à large échelle. Pour définir ces modèles et architectures, je m'appuie sur le rapprochement des techniques issues du génie logiciel (modélisation et programmation par objets [44] [97]), de l'intelligence artificielle (représentation et traitement des connaissances [99]), et de la programmation concurrente (voir par exemple les travaux d'Agha et Hewitt [1] et ceux de Briot [11]) et répartie (voir par exemple les travaux d'Estraillier [32] [33] et ceux de Sens [105]).

Mes principales contributions sont dans les sous-domaines :

- des architectures d'agents [50] [48] [49],
- du passage des objets actifs aux agents autonomes [51],
- des environnements de développement multi-agents [58],
- de la tolérance aux pannes des systèmes multi-agents [52] [53] [106],
- des systèmes multi-agents adaptatifs [54] [55] [15],
- de la simulation multi-agents des modèles économiques [31] [60] [59].

A première vue, ces contributions semblent diversifiées, cependant elles s'organisent autour du thème : adaptation au niveau individuel (agents) et au niveau collectif (organisation).

1.4 Applications

Ma démarche est souvent expérimentale et vise la conception d'architectures logicielles multi-agents génériques (frameworks) validées sur différents terrains d'expérimentation. Plusieurs projets ont ainsi été développés dans le but de valider les modèles et architectures proposés. Outre la simulation de modèles économiques présentée dans le chapitre 4, et le contrôle de congestion adaptatif de réseaux ATM [9], deux exemples d'applications sur lesquelles j'ai travaillé sont :

- la détection d'intrusions (cette application a été développée dans le cadre de la thèse de Karima Boudaoud [7] [8]) : Les systèmes de détection d'intrusions existants ont été conçus pour des environnements connus et bien définis. Ils ne sont donc pas bien adaptés à des environnements dynamiques tels que les réseaux actifs. Dans ce type d'environnements où les besoins en sécurité sont en perpétuelle augmentation, flexibilité et adaptabilité deviennent des critères primordiaux.
  Nous avons utilisé les systèmes multi-agents pour introduire une nouvelle génération de systèmes de détection d'intrusions souple et flexible pour s'adapter aux changements et à l'évolution complexe et non-prédictive des réseaux.
- le contrôle de la ventilation artificielle (cette application a été développée dans le cadre de ma thèse [48], [26]) : Ce système est utilisé pour contrôler la ventilation artificielle de patients hospitalisés dans des unités de soins intensifs à l'hôpital Henri Mondor (Créteil). Il contrôle, en temps réel, différents signaux ventilatoires

(débit inspiratoire, fréquence respiratoire et pression de CO2 expiré), diagnostique l'état courant du patient et adapte en conséquence soit la thérapie, soit l'assistance mécanique fournie par le respirateur. Pour réaliser cela, il utilise des agents autonomes adaptatifs qui coordonnent leurs actions pour éviter les conflits. Chaque agent développe un raisonnement relativement complexe pour construire à différents niveaux d'abstraction une image de l'évolution de l'état du patient.

1.5 Organisation du document

J'ai choisi de détailler trois principaux sujets : architectures d'agents, systèmes multi-agents tolérants aux pannes et simulation de modèles économiques. Ces trois sujets résument mes travaux sur les agents et systèmes multi-agents adaptatifs.

Ce document est organisé de la manière suivante : un premier chapitre présente mes travaux sur les architectures d'agents et montre que les architectures modulaires permettent de dépasser la dichotomie classique : agents réactifs et agents cognitifs. Ce chapitre montre également que la modularité facilite le développement d'agents adaptatifs en s'appuyant sur l'exemple d'agents interactifs qui peuvent adopter et exécuter dynamiquement des protocoles d'interaction. Le but des deuxième et troisième chapitres est de montrer l'intérêt des systèmes multi-agents adaptatifs et d'essayer de définir l'architecture de ces systèmes. Le deuxième chapitre décrit le mécanisme de réplication adaptatif que nous avons introduit pour fiabiliser les systèmes multi-agents à large échelle. Le troisième chapitre résume nos travaux sur la simulation de modèles économiques. Il décrit le modèle de firmes ainsi que le modèle de formes organisationnelles que nous avons définis. Il définit ensuite les interactions entre les firmes et les formes organisationnelles. Le quatrième chapitre résume nos réflexions sur les systèmes multi-agents auto-adaptatifs et montre l'intérêt de l'adaptation pour la modélisation de systèmes complexes. Il donne ensuite un aperçu de mes projets en cours ainsi que mes perspectives de recherche.

Chapitre 2

Architectures d'agents
Vers des agents adaptatifs

2.1 Problématique

Les systèmes multi-agents existants sont classés en général en deux groupes : les systèmes cognitifs et les systèmes réactifs. Les systèmes multi-agents cognitifs sont fondés sur la coopération d'agents capables, à eux seuls, d'effectuer des opérations complexes. Chaque agent dispose d'une capacité de raisonnement, d'une aptitude à traiter des informations diverses liées au domaine d'application, et d'informations relatives à la gestion des interactions avec d'autres agents et l'environnement.

Dans un système multi-agent réactif, le comportement complexe du système émerge de la coexistence et de la coopération d'agents au comportement simple. Les agents réactifs ne disposent que d'un protocole de communication réduit. Ils répondent uniquement à une loi de type stimulus/réponse (voir les travaux de Ferber et Drogoul [2] [27] et ceux de Parunak [94]).

Une synthèse s'est ensuite amorcée entre ces deux orientations et les architectures d'agents hybrides ont été introduites (voir par exemple l'architecture Touring proposée par Ferguson [38], l'architecture Interrap proposée par Müller et M. Pischel [87], et l'architecture proposée par Occello et Demazeau [90]). L'ancienne architecture de DIMA développée dans le cadre de ma thèse [48] était également hybride. Ces agents hybrides combinent des propriétés cognitives et réactives généralement logées dans des modules différents. Les architectures hybrides apportent une solution au problème de l'intégration des aspects réactifs et cognitifs. Ces architectures présentent des qualités indéniables de génie logiciel. Leur organisation modulaire permet entre autre l'intégration de capacités très hétérogènes en terme de programmation. Cette hétérogénéité est nécessaire pour répondre à l'intégration des tâches très diversifiées qui sont remplies par un agent depuis le raisonnement jusqu'à l'interaction. L'intérêt est de pouvoir réutiliser des méthodes existantes, notamment des méthodes d'intelligence artificielle.

La modularité introduit beaucoup de souplesse dans la gestion des capacités en autorisant à moindre coût l'échange de modules dans un but d'évolutivité ou de test.

Une architecture modulaire fait de l'agent un système ouvert [58]. Cependant, les architectures hybrides sont complexes et ne sont pas bien appropriées à la modélisation d'agents dont le comportement est très simple. Par exemple, la conception d'un agent avec Interrap [87] nécessite la conception des trois modules correspondant aux trois couches (comportement, planification, coopération) de l'architecture. Il est donc très lourd d'utiliser cette architecture pour concevoir un agent dont le comportement est très simple (par exemple, basé sur le principe de l'éco-résolution). Ces architectures hybrides ne résolvent pas de ce fait le problème de granularité. Elles ne permettent pas la conception de systèmes multi-agents dont la granularité des agents est variable.

Les nouvelles plates-formes multi-agents opérationnelles et génériques telles Madkit [62] [61] et Jade [5] n'ont été fondées sur aucune architecture d'agents existante. Elles fournissent un noyau générique, souvent représenté par une classe JAVA, qui n'offre que les fonctionnalités de base telles que la communication asynchrone entre agents. L'utilisateur de ces plates-formes doit réutiliser cette classe et implémenter toutes les autres fonctionnalités requises par son application.

Le nouveau modèle d'architecture d'agents DIMA, décrit dans ce chapitre, est fondé sur la conclusion suivante : les travaux nombreux sur les architectures d'agents ont engendré plusieurs résultats intéressants (voir par exemple la synthèse proposée dans ma thèse [48] et celle proposée par Boissier [58], la classification proposée par Müller [88] et celle proposée par Wooldridge et Jennings [116]). Cependant, il est très difficile de comparer ces architectures dans le but de trouver la meilleure architecture. Chacune est bien appropriée à une catégorie d'agents. Une bonne plate-forme devrait intégrer les différentes architectures d'agents existantes ainsi que d'éventuelles nouvelles architectures. Ainsi, le modèle d'architecture d'agents que nous proposons peut être vu comme un modèle 'ouvert', une proposition d'agent minimal est faite permettant, par raffinements successifs, d'ajouter des fonctionnalités fournies par les différentes bibliothèques de la plate-forme. La force de DIMA réside donc à la fois dans sa proposition de modèle d'architecture d'agents modulaire, mais également dans les différentes bibliothèques disponibles.

Ce chapitre présente le modèle d'architecture de DIMA et les différents types d'agents. Il présente ensuite les protocoles d'interaction pour illustrer les bibliothèques de composants réutilisables de DIMA.

2.2 Une architecture modulaire

Le modèle d'architecture d'agents DIMA propose de décomposer chaque agent en différents composants dont le but est d'intégrer des paradigmes existants, notamment des paradigmes d'intelligence artificielle. Un agent peut ainsi avoir un ou plusieurs composants qui peuvent être réactifs ou cognitifs. Ce modèle permet de dépasser la dichotomie classique réactif/cognitif en permettant de définir des agents hétérogènes au sein d'une même application. Chaque agent est composé d'un ou de plusieurs composants. La modularité offre ainsi plusieurs avantages à cette architecture :

– possibilité de définir des agents à granularité variable.

– possibilité de définir des agents à structure adaptative. Chaque agent peut dynamiquement changer ses composants ainsi que les relations entre ces différents composants.
– possibilités d'intégrer différents modèles d'agents.
– possibilité de définir des bibliothèques de composants réutilisables.

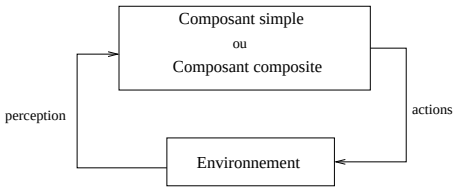

FIG. 2.1 – L'architecture d'agents de DIMA

Chaque agent est un composant simple ou composant composite (voir la figure 2.1) qui gère l'interaction de l'agent avec son environnement et qui représente son comportement interne. L'environnement regroupe tous les autres agents ainsi que les entités qui ne sont pas des agents.

Dans les sections suivantes, nous présentons d'abord les composants proactifs, la brique de base de cette architecture. Ensuite nous décrivons les différents modèles d'agents qui sont fondés sur ces composants proactifs.

2.2.1 Composants proactifs

Contrairement aux objets dont l'activité est restreinte au traitement des messages envoyés par les autres objets, le composant proactif a différentes compétences et son activité n'est pas restreinte à l'envoi et à la réception de messages. Un composant proactif représente une entité autonome et proactive. Le noyau de base de DIMA est un framework de composants proactifs. Il est composé de plusieurs classes et de leurs méthodes. Nous avons choisi un ensemble minimal de classes et de méthodes définissant les fonctionnalités des composants proactifs. Ces fonctionnalités peuvent être étendues dans les sous-classes. Ce noyau minimal est principalement composé de la classe `ProactiveComponent` (voir figure 2.2 [1]).

Une instance de la classe `ProactiveComponent` décrit :
– le but du composant proactif : il est implicitement ou explicitement décrit par la méthode `isAlive()`.
– les comportements de base du composant proactif : un comportement est une suite d'actions qui permettent de changer l'état interne de l'agent ou d'envoyer des messages aux autres agents. Dans notre implémentation, un comportement pourrait être implémenté sous forme d'une méthode Java.
– la méthode `step()` : elle définit la manière dont les comportements sont sélectionnés, séquencés et activés en fonction du but.

[1] Les différents diagrammes de ce chapitre sont des schémas UML générés avec Rational Rose

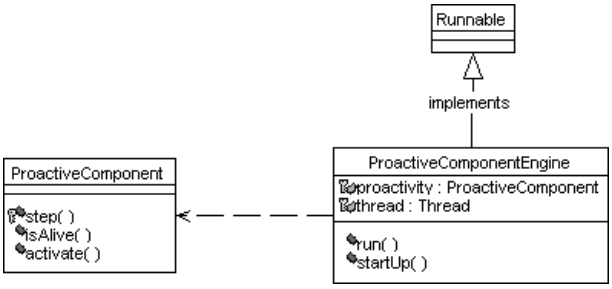

FIG. 2.2 – Le *Structure générale d'un composant proactif*

Les principales méthodes de cette classe sont brièvement décrites dans la table 2.1.

TAB. 2.1 – Principales méthodes de `ProactiveComponent`

Méthodes	Description
public abstract boolean isAlive()	Teste si le composant proactif n'a pas atteint son but.
public abstract void step()	Décrit un cycle de base du méta-comportement du composant proactif.
void proactivityLoop()	Représente la proactivité du composant. public void proactivityLoop() {while (this.isAlive()) {this.preActivity() ; this.step() ; this.postActivity() ; }}
public void startUp()	Initialise et active la proactivité. public void startUp() { this.proactivityInitialize() ; this.proactivityLoop() ; this.proactivityTerminate() ;}

Le composant proactif peut être considéré comme une bonne brique de base pour construire des agents. Par exemple, nous pouvons implémenter avec `ProactiveComponent` des agents réactifs tels qu'ils sont décrits par J. Ferber et A. Drogoul [2]. Les comportements d'un agent réactif, dans ce cas, correspondent aux comportements de base de l'éco-résolution tels que : rechercher satisfaction, fuir, agresser. La méthode step() implémente le principe de l'éco-résolution.

Pour définir des agents interactifs, les composants proactifs (voir section précédente) ont été enrichis d'un module de communication (voir figure 2.3). Ce dernier gère les interactions avec les autres agents. Les comportements de ces agents interactifs intègrent

notamment deux types d'actions :

- des actions liées aux concepts d'agents tels que l'envoi et la réception de messages définis par les méthodes *readMail()* et *sendMessage()*,
- des actions liées au domaine.

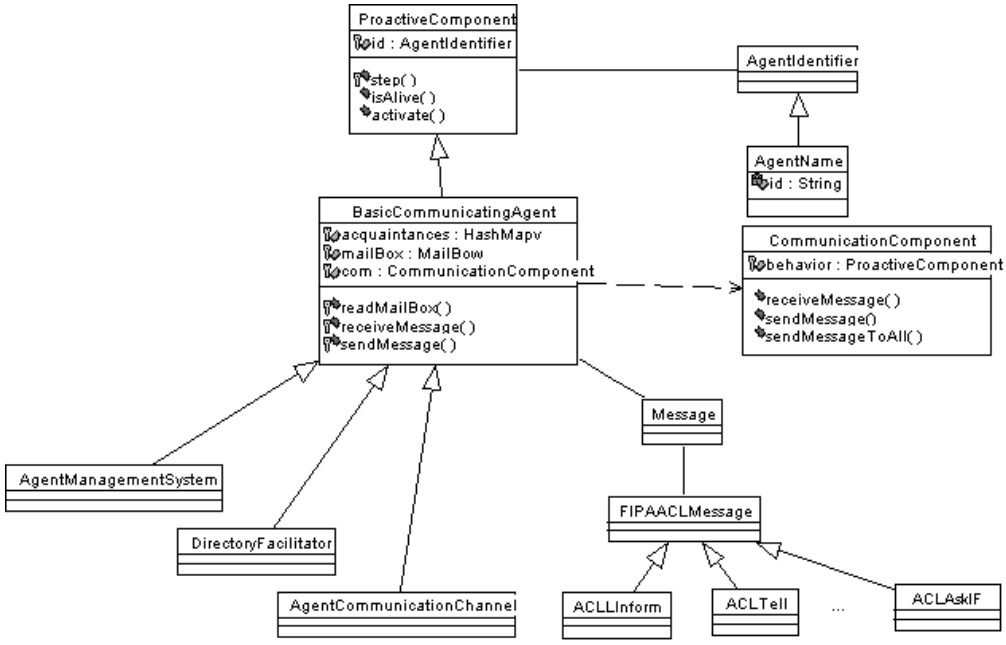

FIG. 2.3 – Composants de base d'un agent interactif et agents de l'architecture FIPA

Ainsi, la définition d'un agent s'opère d'une façon relativement simple. La conception d'un agent réactif minimaliste, puis d'un agent interactif, peut se faire par raffinements successifs de la classe de base proposée. Cette simplicité d'utilisation de la plate-forme DIMA est héritée de la programmation par objets.

2.2.2 Architecture méta-niveau

Pour concevoir un agent, il est nécessaire de développer une structure décisionnelle de base, chargée de choisir une des options auxquelles se trouve confronté l'agent en fonction du contexte (implémentée par la méthode `step()`). Une telle structure est souvent complexe à réaliser pour le programmeur, car il doit prendre en compte tous les cas possibles (états possibles de l'agent et de son environnement) et tous les comportements de base. Ce travail n'est alors réalisable dans de bonnes conditions que pour des agents extrêmement simples où tous les stimuli sont connus *a priori* et ne sont pas très nombreux. Dans le cas des systèmes complexes, les différentes variations de l'environnement ne peuvent être connues *a priori*. L'architecture d'agent doit, dans ce cas, offrir la possibilité d'adapter facilement le processus de décision de l'agent à l'évolution de son contexte.

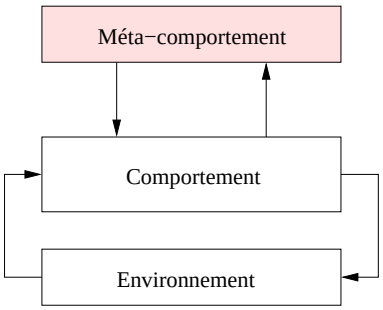

FIG. 2.4 – Architecture d'agents adaptatifs

Plusieurs chercheurs ont souligné l'intérêt de la représentation explicite et séparée du contrôle ou de l'aspect réflexif des architectures de méta-niveaux (voir par exemple les travaux de Pitrat [98], de Rao [101], de McAffer [84] et ceux de Malenfant [81]). En adoptant cette représentation explicite et séparée du contrôle, nous proposons d'introduire un méta-comportement dans notre architecture d'agents (voir la figure 2.4). Ce méta-comportement donne à chaque agent la capacité de prendre des décisions appropriées au sujet du contrôle ou d'adapter son comportement, avec le temps à de nouvelles circonstances. Il fournit à l'agent un mécanisme d'auto-contrôle pour adapter dynamiquement ses comportements selon son état interne et celui de son environnement.

Le méta-comportement se base sur les données de l'agent lui-même, son environnement et le système de décision utilisé. Il est basé sur deux types d'éléments : conditions et actions. Pour implémenter cette structure, nous réutilisons la classe `ProactiveComponent`. La nouvelle classe a les principaux composants suivants :

- un ensemble de conditions destinées à être utilisées par le méta-comportement pour tester le contexte de l'agent,
- un ensemble d'actions qui peuvent être utilisées par le méta-comportement pour modifier l'état de l'agent et/ou l'état de son environnement.

Cette structure est très générale et peut être utilisée avec n'importe quel système décisionnel (ATN, base de règles, etc.), pourvu qu'il fournisse un lien entre les conditions et les actions. La figure 2.5 donne des exemples de classes d'agents adaptatifs. Dans la classe `ATNBasedProactiveComponent`, le méta-comportement est décrit par un ATN et la méthode step() permet, dans ce cas, d'activer une transition dont la condition est vérifiée.

```
public void step() {
 currentState = currentState.crossTransition(this);
}
```

Les états de cet ATN représentent les points de décision. Ils sont utilisés pour choisir la transition suivante parmi toutes les transitions associées à l'état courant.

Les transitions relient les états initiaux aux états finaux. Chaque transition est désignée par : *if* condition *then* action. Les conditions de transition testent l'occurrence

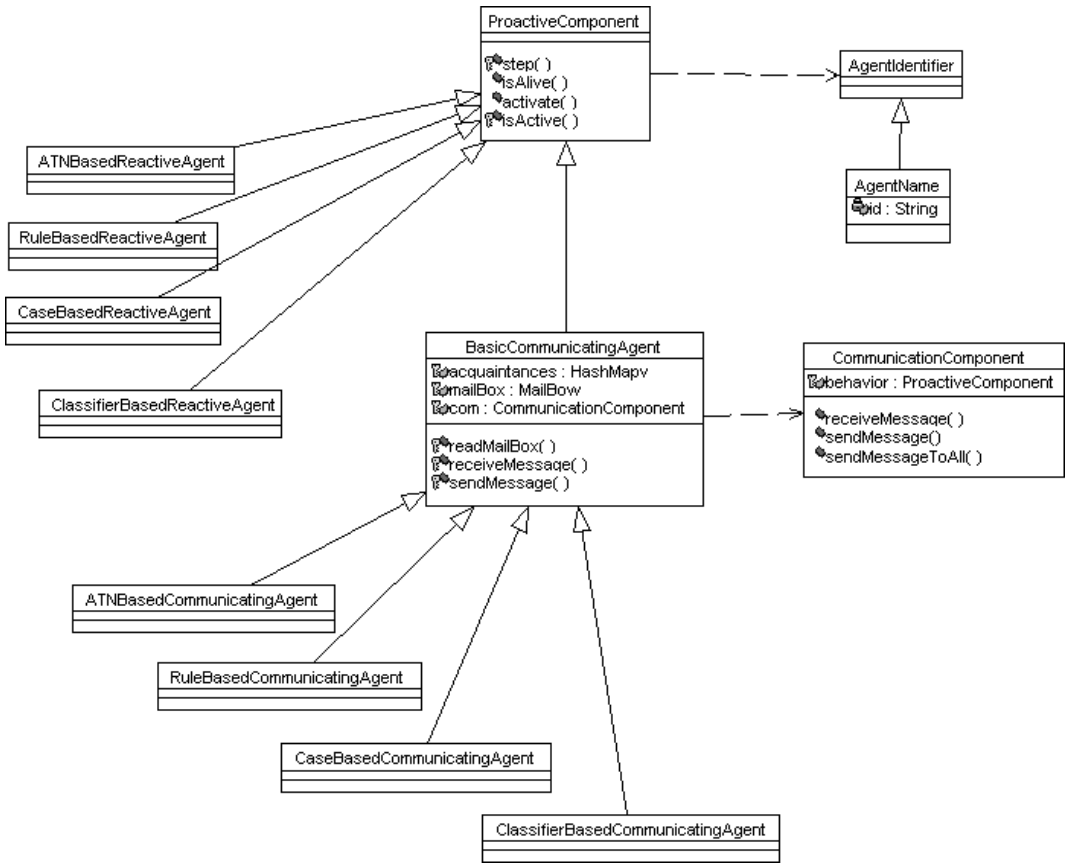

FIG. 2.5 – Examples de classes d'agents

d'un événement (réception d'un message, réception d'un stimulus ...) ou d'un changement de l'état interne de l'agent. Les actions de transition représentent les actions et les décisions nécessaires pour répondre à ces événements. Elles sont implémentées par des méthodes.

Pour construire un composant proactif, on doit décrire son méta-comportement et ses comportements de base en sous-classant la classe `ProactiveComponent`. On doit également décrire la méthode `isAlive()`. Dans le cas des agents adaptatifs où le méta-comportement serait décrit par un ATN, cette méthode teste si l'état final n'est pas atteint. Elle est indépendante du domaine d'application.

```
public boolean isAlive() {
return !(currentState.isFinal());
 }
```

Par exemple, dans le modèle *éco-résolution*, le méta-comportement d'un agent peut être décrit plus facilement par un automate dont les états sont : satisfait, rechercheSatisfaction, rechercheFuit, fuite et les transitions correspondent aux règles (voir figure

2.6). Le concepteur peut ainsi ajouter facilement des états et/ou des transitions pour compléter les méta-comportement et l'adapter aux différentes évolutions de l'environnement.

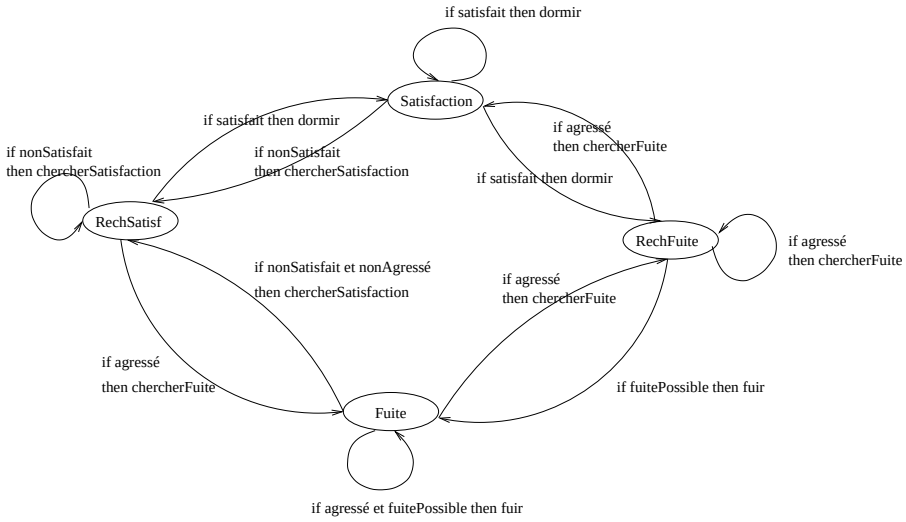

FIG. 2.6 – Méta-Comportement d'un agent dans le modèle de l'éco-résolution

Un méta-comportement a ainsi deux rôles principaux :

- adaptation structurelle : il s'agit d'adapter la structure de l'agent à l'évolution de son environnement. Par exemple, l'arrivée de nouveaux agents dans le système ou tout simplement dans la liste des accointances d'un agent peut nécessiter la réorganisation des accointances.
- adaptation comportementale : il s'agit d'adapter le processus de décision de l'agent à l'évolution de son environnement. Par exemple, dans le cas d'un agent économique, le choix d'une stratégie dépend fortement de l'évolution de la compétition. Il serait donc intéressant de doter cet agent d'un ensemble de méta-règles qui permettent de mettre à jour les données sur l'évolution de la compétition qui favorisent le choix d'une stratégie.

Ces deux rôles peuvent être combinés pour avoir des agents dont la structure et le comportement sont adaptatifs. Ce modèle peut être utilisé hors-ligne pour faciliter la conception d'un agent. Il suffit donc de simuler l'évolution de l'environnement des agents, observer les différentes évolutions de l'environnement et les réactions des agents pour adapter ensuite leur méta-comportement à ces différentes évolutions. Les agents peuvent ensuite être activés avec les nouveaux méta-comportements. L'adaptation, dans ce cas, est statique ; elle est réalisée par le concepteur. La section suivante décrit un exemple d'adaptation dynamique.

2.2.3 Adaptation comportementale

Pour permettre l'adaptation dynamique de l'agent à l'évolution de son contexte, on propose de concevoir un agent comme la somme de deux entités indépendantes : un système décisionnel (par exemple une base de règles) et un système d'adaptation. Ainsi, selon les évolutions du contexte, les agents s'adaptent, et la structure décisionnelle de l'agent est générée et/ou mise à jour automatiquement.

Ces systèmes d'adaptation, bien que pouvant gérer directement le système décisionnel de l'agent, sont surtout là pour mesurer, tester et améliorer l'efficacité de l'agent. Pour ce faire, ils doivent disposer d'informations sur l'agent et son environnement, mais surtout d'informations concernant le fonctionnement du système décisionnel (méta-comportement), des méta-informations. Ils doivent de plus adapter ce système décisionnel par le biais de méta-actions et, ainsi, adapter l'agent à son contexte. Un système d'adaptation peut en effet être vu comme un méta méta comportement de l'agent.

Pour faciliter l'implémentation de ces agents adaptatifs, nous nous sommes appuyés sur les systèmes de classeurs. Ceux-ci sont une extension des systèmes à base de règles. L'innovation réside dans le fait qu'un apprentissage s'effectue au niveau du choix des règles appliquées (appelées classeurs). A chacune de ces règles est associée une évaluation de l'efficacité de la règle. Une sélection ou tirage aléatoire parmi les meilleures règles applicables (celles dont la condition *matche* le contexte) détermine celle qui définit le comportement à chaque pas. Ensuite, lorsqu'une information suggère une amélioration de la situation, les notes des dernières règles utilisées sont augmentées. De même, si la situation empire, les notes sont diminuées. Il est aussi possible d'éliminer les règles les moins bien notées ou d'en insérer de nouvelles.

Les principes de l'algorithme sont décrits dans l'algorithme 1. Pour la réalisation de ces agents, nous avons utilisé le framework à base de classeurs XCS [71].

Le développement d'agents adaptatifs semble ainsi très simple. Cependant, l'utilisation d'un framework tel que XCS soulève de nombreux problèmes. Il présente le défaut d'être particulièrement lent lors de son adaptation parce que les notes n'évoluent que lorsque les règles sont utilisées.

Ces frameworks et les algorithmes sous-jacents sont issus de travaux en apprentissage dans les systèmes mono-agents. Ils étudient l'adaptation d'un agent à un environnement composé d'un ensemble d'entités passives. La perception de ces environnements représente le contexte de l'agent. Il s'avère ainsi plus simple de construire le modèle de cet environnement ; l'adaptation est donc rapide. Dans un système complexe, les environnements sont très complexes et très dynamiques. L'adaptation est donc souvent coûteuse. Une solution à ce problème est l'utilisation de ces systèmes d'adaptation comme outils d'aide à la conception. Les agents adaptatifs sont activés en simulation, les variations de l'environnement sont simulées. Cette simulation permet d'initialiser la base de règles (ou système de classeurs) initiale. Les agents peuvent ensuite être activés dans leur environnement réel avec cette base de règles et le système d'adaptation est utilisé pour faire face aux modifications imprévues.

Un exemple d'application de ces agents est donné dans le chapitre 4.

Algorithme 1 Construction de la base de règles par XCS

Require: CL Ensemble des classeurs (CL peut être vide) et B Ensemble de comportements ;
Ensure: pour un contexte c
1: $M \leftarrow \phi$
2: **for** tout $r \in CL$ **do**
3: **if** r est similaire à c **then**
4: ajouter r à M
5: **end if**
6: **end for**
7: **if** M ==ϕ **then**
8: pour tout $b \in B$ rajouter à M le classeur (c, b)
9: **end if**
10: choisir un classeur de M noté mc,
11: exécuter l'action du classeur mc,
12: calculer son *reward*,
13: mettre à jour le classeur mc.

2.2.4 Principales caractéristiques de l'architecture

Grâce à sa modularité, l'architecture proposée aide à décomposer le comportement arbitrairement complexe d'un agent en un ensemble de petits comportements spécialisés, et éventuellement en dirigeant ces divers comportements par un méta-comportement. Les avantages de cette architecture correspondent aux différentes propriétés recherchées d'un système multi-agents :

- multi-granularité : des agents de diverses granularités (taille, comportements internes, connaissances) peuvent être implémentés avec l'architecture proposée. Cette propriété est très importante pour la conception des systèmes complexes. Elle permet de dépasser la dichotomie classique entre les agents réactifs et les agents cognitifs.
- dynamicité et ouverture : des agents peuvent être dynamiquement créés et/ou détruits. Ils peuvent intégrer de nouveaux comportements et changer leurs connaissances en fonction des informations qu'ils reçoivent et/ou qu'ils perçoivent. De nouveaux comportements peuvent être créés et intégrés par le méta-comportement d'un agent.
- réflexion : notre architecture met en application un modèle d'agents adaptatifs basé sur la réflexion dans lequel chaque agent a son propre méta-comportement qui régit ses divers comportements, par exemple prendre des décisions appropriées au sujet du contrôle ou adapter ses comportements à de nouvelles circonstances.
- hétérogénéité : c'est une propriété très importante dans les systèmes multi-agents. Cependant, la majorité des systèmes existants ne fournissent pas la possibilité de réaliser des agents hétérogènes. Dans DIMA, différents types d'hétérogénéité sont considérés :

- multiplicité des plates-formes d'exécution,
- multiplicité des formalismes de représentation des connaissances,
- multiplicité des modèles d'agents.

2.3 Protocoles d'interaction

Un système multi-agents est un ensemble organisé d'agents interactifs. L'interaction est ainsi un concept clé des systèmes multi-agents. Pour définir les interactions entre agents, différents protocoles d'interaction ont été proposés (voir par exemple les travaux fait par FIPA [40]). Les protocoles d'interaction sont des descriptions de patterns standard d'interactions entre deux ou plusieurs agents. La majorité des travaux existants, notamment les travaux de la FIPA, décrit en effet les séquences de messages échangés et leurs contenus (performative, émetteur, receveur ...).

Les représentations proposées des protocoles d'interaction (voir les travaux de El Fallah-Seghrouchni [34], d'Odell [92] et de Yoo [117]) fournissent plusieurs facilités pour concevoir des systèmes multi-agents. Cependant, les actions locales et les processus de décision d'un agent ne sont pas toujours explicitement représentés. L'implémentation des agents qui exécutent ces protocoles d'interaction nécessite plus d'effort et de savoir-faire du développeur. Par ailleurs, les protocoles d'interaction sont souvent implémentés de façon *ad hoc*. Il est donc impossible de changer dynamiquement de protocole d'interaction.

L'implémentation des protocoles d'interaction a souvent été désignée comme un objectif par la FIPA. Cependant, le *Contract Net* est le seul protocole d'interaction qui est implémenté dans la plate-forme JADE [5]. Celle-ci ne fournit que des méthodes abstraites que l'utilisateur doit implémenter.

L'objectif de cette section est de montrer que les protocoles d'interaction peuvent être définis comme des modules réutilisables. L'idée que nous suggérons est de construire une bibliothèque de rôles, chaque protocole d'interaction est implémenté par les deux rôles correspondants. Les agents interactifs peuvent ainsi :

- être facilement implémentés,
- adopter de nouveaux protocoles d'interaction et les exécuter dynamiquement.

Pour illustrer l'intérêt de cette adaptation structurelle, nous proposons le scénario suivant : un agent $Agent_i$, qui a le rôle initiateur du *Contract Net*, diffuse un appel à propositions pour une tâche t. Il ne reçoit qu'une seule proposition de l'agent $Agent_j$. $Agent_i$ relance un autre appel à propositions et il ne reçoit qu'une seule proposition de $Agent_j$. Si $Agent_i$ est adaptatif et essaie d'optimiser les communications, il remplacerait le protocole d'interaction *Contract Net* par le protocole d'interaction *Request* [40].

Cette section analyse les protocoles d'interaction et les comportements nécessaires à leur exécution. Elle présente un exemple de protocole d'interaction : le *Contract Net*. Elle analyse les différentes comportements que nécessite l'exécution de ce protocole d'interaction et identifie les paramètres nécessaires à cette exécution. Elle résume ensuite les résultats de notre analyse des principaux protocoles d'interaction existants [83]. Le but de cette analyse est de montrer que nous pouvons fournir une implémentation

(sous forme de bibliothèque) des différents rôles et les comportements nécessaires à leur exécution (méthodes JAVA). Enfin, nous décrivons le framework d'agents interactifs et adaptatifs que nous avons élaboré pour implémenter des agents qui utilise cette bibliothèque de rôles. Grâce à cette bibliothèque et ce framework, le développement d'un système multi-agents est rendu plus facile et les agents peuvent changer dynamiquement leurs rôles.

2.3.1 Contract Net

Le *Contract Net* est le protocole d'interaction le plus utilisé dans les systèmes multi-agents. Il repose sur un mécanisme d'allocation de tâches régi par le protocole d'appel d'offres qui est utilisé dans les organisations humaines. Le modèle est basé sur une communication par envoi de messages. Dans les réseaux contractuels, un agent peut avoir deux rôles par rapport à une tâche : initiateur ou participant. Le protocole est composé de trois phases (voir figure 2.7) :

- annonce d'une tâche : l'initiateur fait une annonce de tâches (ou contrat) aux agents du système,
- offre d'un service : les agents évaluent leur intérêt en fonction de leurs ressources. Si la tâche est intéressante, ils soumettent une offre,
- attribution d'une tâche : l'initiateur choisit un ou plusieurs agents candidats pour exécuter la tâche et informe les agents de ce choix.

Le modèle d'interaction de chaque rôle (initiateur ou participant) peut être facilement déduit de la description du protocole d'interaction. Nous analysons ici le modèle d'interaction de l'initiateur afin de proposer une représentation générique de ce rôle et de définir les actions et décisions nécessaires pour son exécution.

Le rôle *initiateur* est basé sur un contrat (ou service). Son exécution nécessite l'initialisation de ce contrat. Les différentes étapes de celle-ci sont :

1. un appel à proposition est envoyé aux participants.
2. les participants répondent en envoyant les propositions. Le type du message est soit *propose* soit *refuse*.
3. l'initiateur évalue les propositions reçues et en sélectionne une. Cette évaluation est basée sur sa stratégie. L'initiateur peut ainsi sélectionner la première proposition reçue ou la meilleure proposition.
4. l'initiateur envoie les messages *accept* aux participants dont la proposition est acceptée et *reject* aux autres participants.
5. les participants qui acceptent la proposition informent l'initiateur. Le participant attend alors le message *confirm*.

Pour la représentation des rôles, nous avons choisi les ATN (voir section 2.2.3). La figure 2.8 décrit l'activité de l'initiateur. Cet ATN représente le (ou une partie du) méta-comportement de l'agent qui contrôle les comportements locaux de l'agent, en fonction des messages reçus. Par exemple, la méthode *sendCallForProposal* définit un message dont le contenu est le contrat et l'envoie à tous les agents. Les conditions et

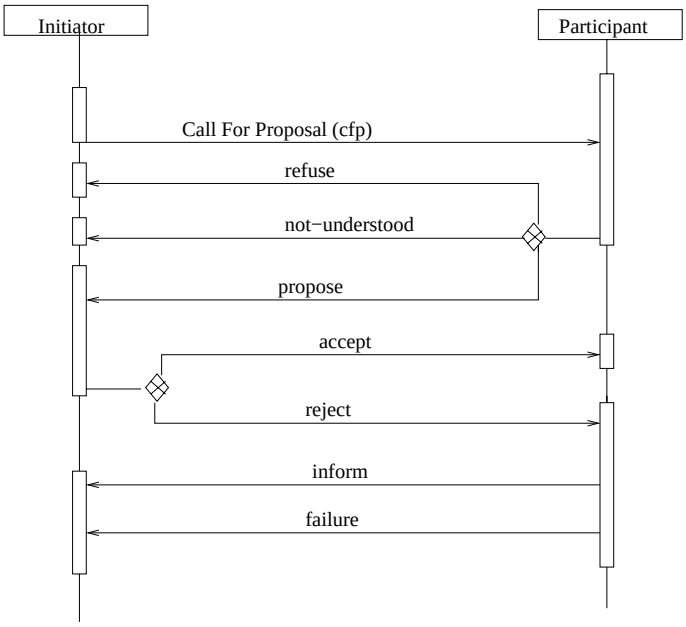

FIG. 2.7 – *Contract Net*

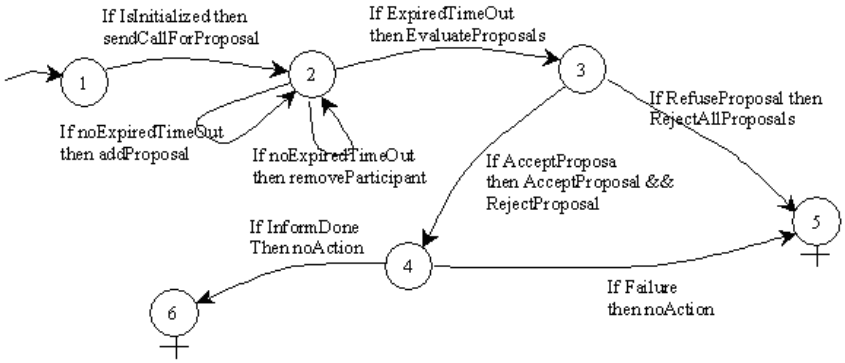

FIG. 2.8 – Meta-comportement de l'initiateur du *Contract Net*

les actions de cet ATN sont génériques. Elles dépendent du contrat et de la stratégie d'évaluation de l'agent. Les deux rôles du *Contract Net* peuvent être en effet définis par des composants (ou boîtes noires) qui nécessitent des paramètres en entrée. L'initiateur nécessite un contrat et une stratégie, et le participant nécessite une stratégie. L'exécution d'un initiateur ou d'un participant nécessite donc l'initialisation de ses paramètres.

2.3.2 Analyse des protocoles d'interaction

Dans la table 2.2, nous résumons les résultats de notre analyse des différents protocoles d'interaction. Cette analyse a permis de définir une représentation générique des rôles correspondant à ces protocoles d'interaction et d'identifier les paramètres d'entrée de ces rôles.

TAB. 2.2 – Exemples de protocoles d'interaction, de rôles et de leurs paramètres

Protocole d'interaction	Rôles	Paramètres
Contract Net	Initiateur	un contrat, un *time out*, une stratégie d'évaluation.
	Participant	une stratégie de décision.
Point à Point	Initiateur	un contrat, une stratégie d'évaluation, une stratégie de réitération, une stratégie de construction.
	Participant	une stratégie d'évaluation, une stratégie de réitération une stratégie de construction.
Enchère Anglaise	Initiateur	un contrat, un *time out*, un incrément, première offre, dernière offre.
	Participant	une stratégie de construction.
Enchère Hollandaise	Initiateur	un contrat, un *time out*, un décrément, la première offre, dernière offre.
	Participant	une stratégie de construction.
Enchère de Vickrey	Initiateur	un contrat, un *time out*, une stratégie d'évaluation.
	Participant	une stratégie de construction.

Les actions et les décisions correspondant aux différents rôles sont ainsi génériques, mais elles nécessitent des paramètres tels que le contrat et la stratégie. Pour faciliter la description de cette bibliothèque, nous avons introduit une ontologie en nous basant sur les résultats de notre analyse. Cette dernière est représentée dans la figure 2.9.

Nous avons ensuite utilisé cette ontologie pour définir une bibliothèque de rôles. Celle-ci contient tous les rôles des protocoles d'interaction décrits dans la littérature multi-agents [40] [83]. La figure 2.10 donne l'implémentation des rôles du *Contract Net*. Chaque rôle est représenté par une classe et les comportements sont implémentés par

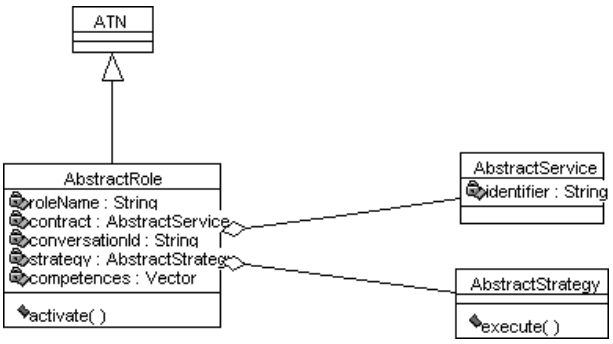

FIG. 2.9 – L'ontologie minimale des protocoles d'interaction

des méthodes de cette classe. Ces méthodes sont génériques.

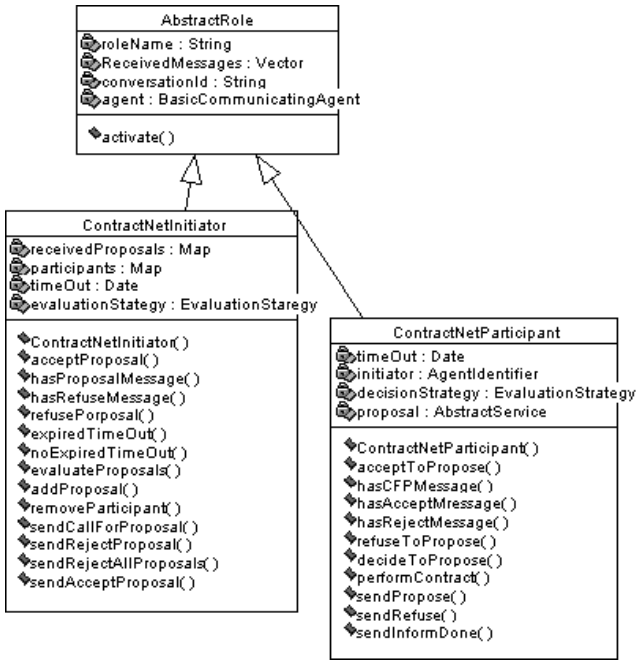

FIG. 2.10 – Implémentation de l'initiateur et du participant du *Contract Net*

La section suivante décrit le framework que nous avons introduit pour définir des agents interactifs et faciliter l'exploitation de cette bibliothèque de rôles.

2.3.3 Un framework d'agents interactifs et adaptatifs

Dans la section 2.2, nous avons présenté une architecture d'agents interactifs. Ces agents disposent d'un ensemble de comportements élémentaires pour envoyer et recevoir des messages. L'utilisation de cette architecture nécessite souvent un grand savoir-faire

pour la synchronisation des interactions entre agents. Les protocoles d'interaction offrent une solution élégante à ce problème de synchronisation. Cependant, leur implémentation est une tâche fastidieuse. L'utilisation d'une bibliothèque de rôles (voir section précédente) facilite l'implémentation de ces agents interactifs.

Nous avons en effet défini une structure d'agents interactifs et adaptatifs. Ces agents peuvent changer leur structure (changer un rôle, adopter un nouveau rôle, ...) pour s'adapter à l'évolution de leur environnement. Par exemple, un agent qui a seulement le rôle initiateur du *Contract Net* peut décider de répondre à un appel à propositions envoyé par un autre agent. Il doit ainsi acquérir dynamiquement le rôle participant. Pour définir ces agents adaptatifs, nous nous sommes appuyés sur les agents interactifs et les avons enrichis :

- d'une liste de rôles,
- de comportements élémentaires pour acquérir un nouveau rôle, activer un rôle, activer plusieurs rôles, ...

La figure 2.11 donne un aperçu de ce framework. Ce framework fournit ainsi les comportements élémentaires qui peuvent être utilisés par le méta-comportement pour adapter la structure de l'agent. Pour construire des agents interactifs adaptatifs, il est nécessaire d'utiliser des méta-règles. Par exemple, dans la vente aux enchères un acheteur $Acheteur_i$ peut utiliser une méta-règle pour acquérir automatiquement les rôles correspondant à la proposition reçue. Si $Acheteur_i$ n'a que le rôle initiateur du *Contract Net* et si le vendeur utilise l'enchère hollandaise au lieu du *Contract Net*, $Acheteur_i$ ajouterait à sa liste le rôle participant de l'enchère hollandaise pour répondre à l'appel à propositions reçu.

Une base de méta-règles est en cours d'élaboration dans le cadre de la thèse de Tarek Jarraya qui a développé la bibliothèque de rôles ainsi que le framework d'agents interactifs. Pour l'élaboration de cette base de méta-règles, nous nous appuyons sur l'exemple du commerce électronique [74].

Dans le projet Méta-DIMA, décrit dans le dernier chapitre, nous nous sommes appuyés sur ce framework et la bibliothèque de protocoles d'interaction pour la transformation des modèles en un système multi-agents opérationnel (voir [114]).

2.4 DIMA : vers un environnement de développement

DIMA est un environnement de développement de systèmes multi-agents dont le développement a débuté en 1993. Il est basé sur le modèle d'architecture proposé dans ce chapitre. La première version de DIMA a été implémentée en Smalltalk-80 et a été ensuite portée en JAVA.

La richesse des bibliothèques proposées par DIMA permet aux utilisateurs de développer des applications à base de systèmes multi-agents de façon intuitive en se basant sur un modèle de conception solide. De nombreux chercheurs et stagiaires de DEA ont contribué à l'élaboration du noyau de base, et également des bibliothèques de la plateforme tout en essayant de conserver la "philosophie" DIMA : permettre la conception incrémentale d'agents en partant d'un modèle relativement simple. En cela, DIMA peut

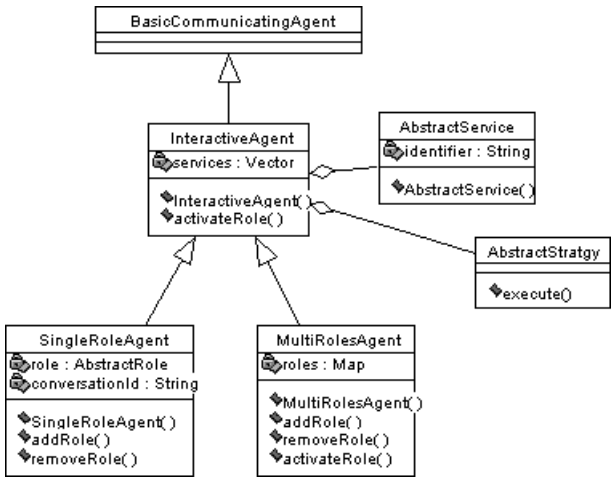

FIG. 2.11 – Implémentation des agents interactifs avec rôles

être vu comme un environnement de conception de systèmes multi-agents simple à prendre en main (le point de départ étant un modèle d'agent minimal) mais pouvant se révéler très puissant via l'exploitation des différents modèles d'agents et/ou des options fournies par la plate-forme (bibliothèques de composants, architectures d'agents, modèles de communications, etc.).

Pour améliorer le processus de développement de systèmes multi-agents avec la plate-forme DIMA, un outil de méta-modélisation basé sur les propositions de l'OMG sur les Model-Driven Architectures [93] est en cours de développement (nos premiers exemples sont décrits dans [75] et [114]). L'intérêt de l'approche par méta-modélisation provient notamment de la diversité des paradigmes en multi-agents (différents modèles d'agents, différents modèles organisationnels, ...) qu'il convient de combiner. Cette approche permettra de définir une méthodologie intégrant les différentes étapes du cycle de base de génie logiciel. Son avantage est qu'elle est basée sur une plate-forme multi-agent et un outil de méta-modélisation opérationnels [102]. Ce projet est le résultat d'une coopération avec Gilles Blain, Jean-François Perrot et Arnaud Thiefaine.

2.5 Conclusion

Aujourd'hui, l'intérêt d'une méthodologie de conception et d'implémentation de systèmes multi-agents n'est plus à démontrer. De nombreux travaux de recherche continuent à explorer les possibilités que fournirait une méthodologie telle que Aalaadin [37], ADELFE [6], ADEPT [69], Cassiopée [21], Gaia [80], INGENIAS [95], MASSIVE [78], MITAIS [113], Nemo [64], PASSI [22] Toutefois, la plupart de ces propositions se base sur une architecture d'agent purement conceptuelle, l'idée n'étant pas de fournir un moyen à partir d'une spécification d'un problème donné, de déduire une solution opérationnelle, mais de se restreindre à l'agentification du problème.

Nous pensons que ces approches, aussi intéressantes soient-elles, limitent leurs potentialités (au sens "utilisation potentielle") par une trop grande généricité (qui a d'ailleurs beaucoup de mal à être atteinte). Proposer une méthodologie de conception de systèmes multi-agents générique (i.e. qui permet de potentiellement agentifier un problème posé, sans se soucier de sa partie opérationnelle) est louable à de nombreux points de vue. Mais cette généricité apporte plus d'inconvénients qu'elle ne propose de solutions. Ce type de méthodologie force à proposer des solutions dans lesquelles les hypothèses d'implémentation restent trop vagues, et où peu de réelles caractéristiques des entités logicielles générées sont clairement spécifiées.

L'approche que nous proposons ici peut être qualifiée de *bottom-up* : plutôt que de partir sur un modèle d'agent générique couvrant l'ensemble des domaines mais restant hautement conceptuel, nous avons commencé par proposer une première solution opérationnelle minimale. Via cette proposition et sa confrontation avec un ensemble d'utilisateurs, le modèle s'est affiné afin de répondre aux besoins spécifiques de différents domaines d'application, tout en conservant une cohérence globale. Les bibliothèques et frameworks composants DIMA en font un environnement de conception de systèmes multi-agents riche. Elles permettent des solutions opérationnelles associées cohérentes.

Chapitre 3

Systèmes multi-agents tolérants aux pannes

Le travail présenté dans ce chapitre est une partie du projet ALSACE (Adaptation dynamique de Logiciels par des Composants Evolutifs) [1]. Il s'inscrit dans le cadre d'une collaboration entre les thèmes OASIS (Objets et Agents pour Systèmes d'Information et de Simulation) et SRC (Systèmes Répartis et Coopératifs) du LIP6.

3.1 Problématique

Les systèmes multi-agents offrent une vision décentralisée et coopérative de la résolution de problème. Ils sont donc particulièrement bien adaptés aux problèmes dynamiques et répartis physiquement. Les domaines d'application sont nombreux : gestion de crise, contrôle de processus, ateliers de production flexibles, robotique collective, etc. Cependant, une notion fondamentale des systèmes répartis à large échelle est leur sensibilité aux pannes de diverses origines : machine, réseau, etc.

La tolérance aux pannes est un domaine très étudié dans les systèmes répartis. Les différents mécanismes de réplication proposés ont été appliqués avec succès à plusieurs applications distribuées (i.e. bases de données [70]). Cependant, dans la majorité des cas, la réplication est décidée par le programmeur et elle est appliquée de façon statique avant l'exécution de l'application. Ces différentes applications sont fiables parce que la criticité de leurs composants peut être déterminée par le concepteur et ne varie pas durant l'exécution de l'application.

Dans le cas des applications multi-agents dynamiques et adaptatives, la criticité des agents, qui sont souvent adaptatifs, peut évoluer durant l'exécution en fonction de l'évolution des comportements des agents et du comportement collectif. Par ailleurs, les ressources disponibles sont souvent limitées. La réplication simultanée de tous les composants (quelle que soit leur criticité) d'un système multi-agents complexe n'est

[1] Participants : S. Aknine, M. Bertier, J.- P. Briot, Z. Guessoum (co-responsable), O. Marin et P. Sens (co-responsable)

donc pas toujours possible.

Notre idée consiste à appliquer dynamiquement les mécanismes de réplication en fonction de l'évolution de la criticité des agents. La solution que nous proposons permet ainsi de répondre de façon dynamique aux questions suivantes :

- quel est l'agent à répliquer ?
- quel est le nombre de réplicats ?
- comment répliquer (quelle stratégie) ?
- où répliquer ?

Ce chapitre est organisé comme suit : la deuxième section présente l'état de l'art. La troisième section définit la réplication et décrit le framework que nous utilisons pour répliquer nos agents. La quatrième section présente les solutions que nous concevons pour déterminer la criticité des agents de façon automatique et dynamique. La cinquième section introduit une méthode pour déterminer le nombre de réplicats en fonction de la criticité des agents. Enfin, est brièvement décrite la plate-forme multi-agents que nous proposons pour développer (concevoir, implémenter et déployer) des systèmes multi-agents tolérants aux pannes. Quelques expérimentations illustreront alors les mécanismes et l'architecture étudiés.

3.2 Différentes approches pour la tolérance aux pannes

Différentes approches ont été proposées pour fiabiliser les systèmes à agents : une pour les agents mobiles et une autre pour les systèmes multi-agents.

Pour les agents mobiles, des solutions sont proposées pour traiter des problèmes de sécurité tels que les intrusions [7] et les pannes [111]. Cet aspect (mobilité) n'est pas considéré dans notre projet.

Pour fiabiliser les systèmes multi-agents, différentes solutions ont été suggérées. Le premier type de solutions a été introduit par les systèmes multi-agents réactifs. Il est fondé sur la redondance. Les systèmes basés sur la métaphore des fourmis offrent un très bon exemple. Les agents réactifs sont souvent similaires, la panne d'un agent n'affecte donc pas le fonctionnement global du système.

Cependant,

- on ne peut pas concevoir toutes les applications avec les systèmes multi-agents réactifs. Ces systèmes sont plus appropriés aux simulations. Ils ne sont pas souvent utilisés dans des applications distribuées.
- cette redondance ne peut pas être appliquée facilement aux systèmes multi-agents cognitifs, les différentes copies des agents cognitifs pouvant provoquer des incohérences. Un contrôle de cohérence de ces copies s'avère en effet nécessaire.

S. Kumar et al. [72] proposent de faire appel à une équipe d'agents *brokers* pour assurer la tolérance aux pannes des systèmes multi-agents. Un *Broker* est un agent intermédiaire qui peut offrir divers services tels que la recherche d'agents capables d'effectuer une certaine tâche, router des requêtes et des réponses, etc. Un tel agent peut être intégré dans un système multi-agents dans le but d'assurer une tâche donnée. Toutefois, si cet agent tombe en panne, c'est tout le système qui risque de l'être aussi. Ainsi,

si un *broker* est déclaré en panne, l'équipe de *brokers* permettra d'assurer les services de l'agent défaillant. Pour ce faire, l'approche consiste en un modèle d'agents *brokers* : *Adaptive Agent Architecture (AAA)* qui modélise les engagements et les croyances des agents *brokers* par une logique modale. L'équipe d'agents *brokers* établit les intentions du groupe et doit les maintenir continuellement. L'architecture AAA permet de détecter et corriger (en choisissant un nouvel agent *broker*) une panne d'un agent *broker*, et d'assurer la continuité du service de tolérance aux pannes, c'est-à-dire l'existence continuelle d'au moins un agent *broker* dans l'équipe.

S. Hagg [63] introduit le concept de sentinelle pour protéger les agents des états indésirables. Les sentinelles représentent une structure de contrôle de leurs systèmes multi-agents. Elles construisent un modèle de chaque agent et contrôlent les communications pour détecter les pannes. Le concepteur associe à chaque fonctionnalité du système multi-agents une sentinelle. Cette sentinelle contrôle les différents agents qui interagissent pour réaliser cette fonctionnalité. Une analyse de ses croyances sur les autres agents lui permet de détecter les fautes quand elles se produisent.

Cette approche semble très intéressante. Cependant, les sentinelles sont également sources de défaillance. Par ailleurs, la conception de ces systèmes est très complexe.

A. Fedoruk et R. Deters [35] proposent une approche de tolérance aux pannes par réplication transparente des agents. Les réplicats d'un agent A_i sont organisés sous forme de groupe de réplicats. Tout agent A_j peut communiquer avec tous les réplicats de l'agent (A_i), ou avec un seul réplicat A_{ik} sélectionné suivant un critère déterminé (l'inverse étant aussi possible c'est-à-dire que A_{ik} peut communiquer avec A_j). Pour ce faire, A. Fedrouk et R. Deters utilisent un proxy pour gérer les interactions entre un groupe de réplicats et le reste des agents du système. L'architecture du proxy n'impose ici aucune modification au niveau applicatif et assure la communication entre les agents du système et un groupe de réplicats. Elle permet aussi de synthétiser les réponses des réplicats aux requêtes des agents du système en adoptant une stratégie de vote, d'agrégation, etc. Pour pallier au risque de panne du proxy (celui-ci étant considéré comme un point de centralisation, qui pourrait défaillir), des copies en réplication passive lui sont créées.

Cette approche est basée sur plusieurs idées intéressantes mais elle manque de flexibilité et en particulier de réutilisabilité. Les expérimentations ont été réalisées avec la plate-forme FIPA-OS [100] qui, à notre connaissance, ne fournit aucun mécanisme de réplication. La réplication est simulée par le concepteur avant l'exécution, le mécanisme de contrôle de réplication avancé est donc *ad hoc*.

Dans les systèmes distribués, différents outils intègrent des facilités et des mécanismes de réplication pour construire des applications fiables. Cependant, la majorité des outils n'est pas assez flexible pour implémenter une réplication adaptative. MetaXa [85] implémente en Java la réplication active et passive de façon flexible. Java est étendu avec une architecture réactive réflexive. Comme dans DarX, la réplication est transparente. Cependant, MetaXa est basé sur un interpréteur JAVA modifié. GARF [46] réalise des machines Smalltalk tolérantes aux pannes en utilisant la réplication active. Comme MetaXa, GARF recourt à une architecture réflexive et élabore différentes stratégies de

réplication. Mais il ne fournit pas de mécanisme adaptatif pour appliquer ces stratégies de façon automatique et dynamique.

La définition d'une bibliothèque de mécanismes de réplication réutilisables offre une meilleure flexibilité aux utilisateurs. Le concepteur du système peut choisir, en fonction de ses besoins, les mécanismes les plus appropriés et le système peut ensuite adapter ces mécanismes en fonction de son évolution et de l'évolution de son environnement. En effet, le système doit avoir la capacité de s'auto-observer afin de déterminer la criticité des agents et d'adapter le nombre de réplicats et la stratégie de réplication à l'évolution de cette criticité et à l'évolution de l'environnement (voir section 3.4).

3.3 Réplication

La réplication de données et/ou de calculs est le seul moyen efficace de réaliser la tolérance aux fautes dans les systèmes distribués. Un composant logiciel répliqué est défini comme un composant logiciel qui possède une représentation sur deux ou plusieurs machines [47].

Il existe deux types fondamentaux de protocoles de réplication :

- le protocole actif où tous les réplicats effectuent les traitements de façon concurrente,
- et le protocole passif où un seul réplicat poursuit son exécution et transmet périodiquement son état courant aux autres réplicats pour maintenir la cohérence de l'ensemble du groupe de réplication.

La réplication active entraîne une surcharge importante. En effet, le coût de traitement pour chaque composant est multiplié par son degré de réplication, c'est-à-dire par le nombre de ses réplicats. De même, les communications additionnelles pour maintenir la cohérence au sein du groupe de réplication sont loin d'être négligeables.

Dans le cas de la réplication passive, les réplicats ne sont sollicités qu'en cas de panne. Le principe est le suivant : si le réplicat actif est perdu, un nouveau réplicat est choisi parmi l'ensemble des passifs et l'exécution est redémarrée à partir d dernier état du composant. Cette technique est moins coûteuse que l'approche active, mais le délai nécessaire au recouvrement des traitements perdus est plus important. De plus, il est beaucoup moins évident de garantir un recouvrement total dans l'approche passive, puisqu'on repart forcément du dernier point de mise à jour.

Plusieurs outils [115] intègrent des facilités de réplication pour construire des applications tolérantes aux fautes. Cependant, la majorité des produits n'est pas assez flexible pour implémenter des mécanismes adaptatifs. Rares sont les systèmes qui permettent de modifier la stratégie de réplication choisie durant l'application. Plus encore, aucun système ne donne la possibilité de laisser à l'application répartie le choix de sa politique de réplication.

Le paragraphe suivant présente le framework DarX. Ce dernier fournit un mécanisme flexible de tolérance aux fautes pour les systèmes multi-agents. DarX offre les deux types de stratégies et surtout, permet de passer dynamiquement d'une stratégie à une autre au cours de l'exécution.

3.3.1 DarX

DarX est un framework pour concevoir des applications distribuées tolérantes aux fautes. Chaque tâche peut être répliquée plusieurs fois. DarX possède plusieurs mécanismes pour contrôler les groupes de réplicats (rajouter et supprimer un réplicat). Il fournit également un multi-cast atomique pour la communication interne d'un groupe de réplicats.

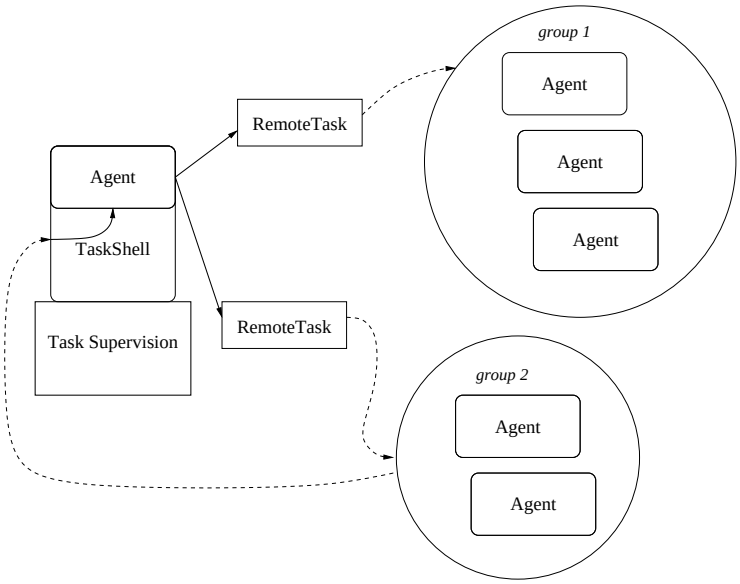

FIG. 3.1 – Architecture DarX

Un groupe de réplication est une entité opaque qui gère chaque tâche de l'application. Le nombre de réplicats et la stratégie interne de réplication d'une tâche spécifique sont totalement cachés aux autres tâches de l'application. Chaque groupe de réplication contient un unique leader qui représente le groupe auprès des autres tâches. Le leader contrôle également la fiabilité de chacun de ses réplicats. En cas de panne du leader, un remplaçant est automatiquement élu parmi l'ensemble des réplicats restants. DarX fournit un système de nommage. Chaque tâche répliquée possède un identifiant global au sein de l'application ; cet identifiant est indépendant des caractéristiques du groupe de réplication ainsi nommé. Afin de tirer profit des capacités de tolérance aux fautes offerte par DarX, chaque agent hérite des fonctionnalités d'un objet DarxTask (voir figure 3.1), permettant au système de manipuler les mécanismes d'exécution et de communication de l'agent. Il devient ainsi possible que DarX agisse en tant qu'intermédiaire pour l'agent et décide :

- quand un agent doit être activé ou suspendu,
- et quand une réception de message doit être effectuée.

Chaque tâche est à son tour encapsulée dans un TaskShell. Cet élément sert à

gérer le groupe de réplication ; cela inclut la prise en charge des communications et la préservation de la cohérence dans le groupe de réplication. Ainsi la tolérance aux fautes est assurée de manière totalement transparente pour les agents. Par exemple en ce qui concerne les communications, le TaskShell intercepte les messages reçus et gère un cache de réception ; il est alors possible de garantir le séquencement des messages au sein du groupe et de supprimer les messages dupliqués par erreur. Les mécanismes de base fournis aux applications multi-agents pour la tolérance aux fautes sont donnés dans la table 3.1).

TAB. 3.1 – Mécanismes de base de DarX

startTask(DarxTask agent) terminateTask(DarxTask agent) TaskShell.suspend() TaskShell.resume()	Ce sont les primitives qui permettent de contrôler l'exécution de chaque tâche au sein de l'application
TaskShell replicateTo(Location hostURL) killReplicantAt(Location hostURL)	Ces deux primitives implémentent les mécanismes de création et de destruction de réplicats à l'intérieur du groupe de réplication.
TaskShell setLeader(TaskShell agent) setReplicationStrategy(ReplicationStrategy r)	Il s'agit des méthodes permettant l'adaptation dynamique de l'application, notamment le choix d'un nouveau leader, ainsi que le changement de la politique de réplication.

La communication entre tâches distinctes se fait au moyen d'un proxy : le RemoteTask. Chaque RemoteTask pointe sur le leader courant du groupe de réplication référencé. Ceci permet de communiquer indifféremment avec des entités locales ou distantes, que ce soit des groupes de réplication ou des agents.

3.3.2 Discussion

DarX fournit des mécanismes pour répliquer les agents et modifier la stratégie de réplication dynamiquement. Cependant, nous ne pouvons pas toujours répliquer tous les agents d'un système à large échelle parce que les ressources sont souvent limitées. De plus, le nombre de ressources varie durant l'exécution du système.

Un bon mécanisme de réplication devrait donc adapter la stratégie de réplication à l'évolution de l'environnement. La solution que nous proposons consiste ainsi à appliquer automatiquement et dynamiquement les mécanismes de réplication aux agents. Cette solution, décrite dans la section suivante, est basée sur un processus d'observation et un processus de monitoring.

3.4 Criticité d'agents

L'analyse de la criticité d'un agent permet de définir son importance et l'influence de sa défaillance sur le comportement normal et la fiabilité du système multi-agents. Aucune définition de la criticité d'agents n'a été proposée. Nous allons donc nous baser sur les concepts de base des systèmes multi-agents pour définir cette criticité. L'objectif est d'élaborer une approche automatique afin de ne pas rendre le développement des systèmes multi-agents plus complexe. Cependant, notre solution peut également utiliser des données fournies *a priori* par le concepteur de l'application. Ce dernier peut, par exemple, choisir de figer la stratégie de réplication d'un agent ou de l'ensemble des agents du système.

Pour nous, la criticité d'un agent dépend de deux types d'informations :

- *Informations du niveau système.* Elles sont basées sur des mesures standard (charge de communication, temps de traitement ....). Elles sont notamment utilisées pour évaluer le degré d'activité d'un agent.
- *Informations du niveau sémantique.* Elles dépendent du domaine d'application et du paradigme choisi (e.g. agent).

La résolution de problèmes dans un système multi-agents émerge des interactions entre un ensemble d'entités organisées appelées agents. Il est ainsi principalement caractérisé par les structures organisationnelles qui peuvent être statiques ou dynamiques. Par exemple, Ferber [37] et Odell [92] utilisent le concept de *rôle* pour définir les structures organisationnelles. Ce concept est bien approprié aux domaines d'applications où les structures organisationnelles sont connues *a priori*.

Par ailleurs, dans beaucoup de domaines complexes, les structures organisationnelles et leurs évolutions ne peuvent pas être définies *a priori*. Elles émergent des interactions entre agents. Castelfranchi [19] cite plusieurs exemples de structures organisationnelles émergentes : les réseaux de communication, les réseaux d'accointances, les réseaux d'interdépendance, etc.

Nous proposons deux solutions complémentaires :

- l'une basée sur le concept de *rôle*, elle est détaillée dans [53],
- l'autre basée sur les *réseaux d'interdépendances*.

Notre approche est générique, elle ne dépend pas d'un langage d'interaction ou d'un domaine d'application. Cependant, les agents, qui peuvent être cognitifs ou réactifs, communiquent avec un langage de communication tels que ACL [40] et KQML [39].

Les sections suivantes présentent l'architecture proposée et montre comment les réseaux d'interdépendances sont définis, mis à jour et utilisés pour évaluer la criticité d'agents.

3.4.1 Une architecture multi-agents

L'évaluation de la criticité d'agents est basée sur l'observation. Le système multi-agents est enrichi d'un mécanisme d'auto-observation qui facilite l'adaptabilité dynamique de la réplication. Il est ainsi doté d'un méta-niveau qui permet :

- d'analyser les informations observées afin de déterminer la criticité d'agents,

– d'adapter dynamiquement la réplication.

Dans la majorité des travaux sur l'observation des systèmes multi-agents, le mécanisme d'observation est centralisé (voir par exemple [83]). L'observation est utilisée pour collecter des informations sur l'exécution du système et les analyser à la fin de l'exécution. Le but de cette observation est d'expliquer le comportement du système. Par ailleurs, les domaines d'application considérés ne sont pas très complexes (petit nombre d'agents, structures organisationnelles connues, ...).

Ces mécanismes d'observation centralisés ne sont pas bien appropriés aux systèmes à large échelle où les informations observées doivent être analysées en temps réel pour adapter le système multi-agents à l'évolution de son environnement.

Nous proposons de distribuer le mécanisme d'auto-observation afin d'améliorer l'efficacité du système d'auto-observation et sa robustesse. Cette distribution est fondée sur une organisation d'agents réactifs appelés Moniteurs dont le but consiste à observer et contrôler les agents du domaine. Dans notre architecture multi-agents, nous associons un agent Moniteur à chaque agent du domaine (voir figure 3.2).

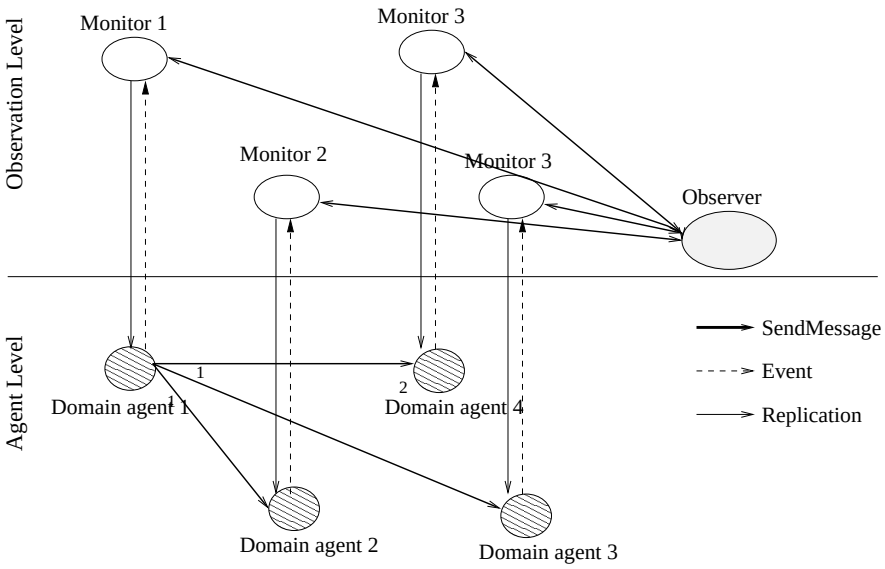

FIG. 3.2 – Architecture multi-agents

Pour observer les agents, un module d'observation est associé au serveur DarxServer (voir figure 3.3). Les informations fournies par ce module sont gérées par un agent appelé Observateur. Chaque agent Moniteur s'abonne à cet agent Observateur pour recevoir les informations et les événements correspondant à l'agent du domaine auquel il est lié. Il analyse ensuite ces informations pour évaluer la criticité de l'agent du domaine et mettre à jour sa réplication.

Les agents de contrôle (Moniteurs et Observateurs) sont organisés de façon hiérarchique. Chaque agent Moniteur ne communique qu'avec l'agent Observateur. Les

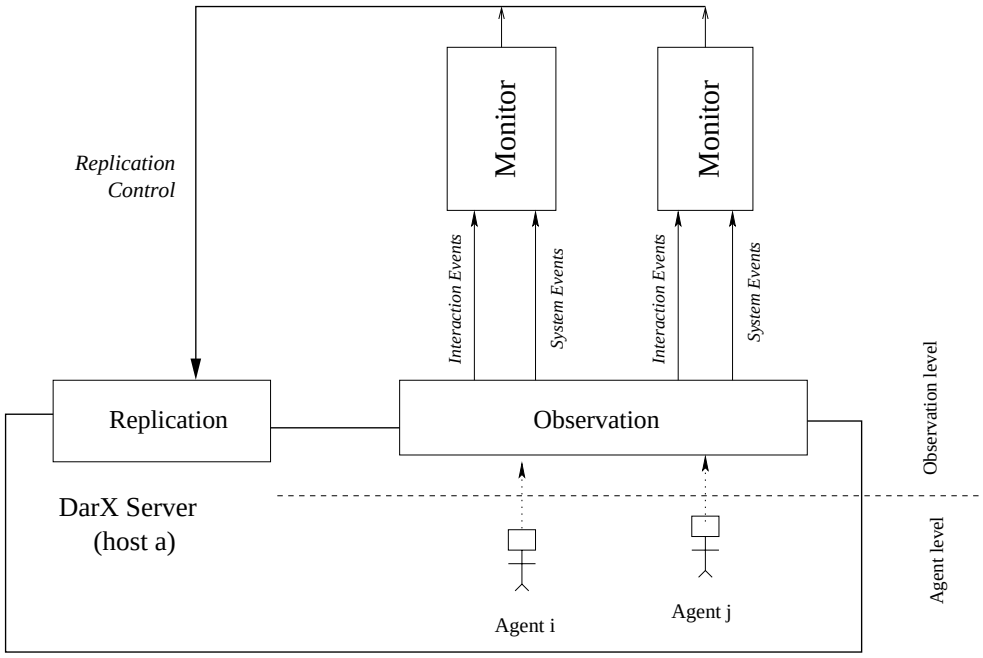

FIG. 3.3 – Architecture générale du module de contrôle de réplication

agents Observateurs s'échangent des informations locales afin d'en déduire des informations globales du système (nombre de réplicats global, quantité d'informations globale échangée entre agents, ...).

3.4.2 Réseaux d'interdépendances

Dans un système multi-agents, chaque agent est défini comme une entité autonome. Cependant, les agents ne sont pas complètement autonomes. Pour réaliser leur(s) but(s), les agents n'ont pas toujours toutes les compétences et/ou les ressources nécessaires. Ils dépendent donc des agents qui ont les compétences et/ou les ressources requises pour réaliser leur(s) but(s). Les réseaux d'interdépendances ont en effet été introduits [109] [110] [18] pour décrire les structures organisationnelles de ces agents. Ils ont été utilisés dans de nombreux travaux qui définissent souvent les réseaux d'interdépendances de façon statique avant l'exécution du système multi-agents. Cependant, les systèmes multi-agents complexes sont caractérisés par des structures émergentes [108].

Dans ce paragraphe, nous allons d'abord montrer comment ces réseaux d'interdépendances émergents sont définis et mis à jour en nous basant sur l'observation des agents. Ensuite, nous soulignerons comment ces réseaux sont exploités pour déterminer la criticité d'agents.

Définition du réseau d'interdépendances

A chaque agent du domaine est associé un nœud. Ce dernier définit le rôle organisationnel de l'agent qui montre que celui-ci n'est pas une entité indépendante, il est contraint par (et contraint) son organisation. Ce nœud représente donc l'interface entre l'agent et son organisation.

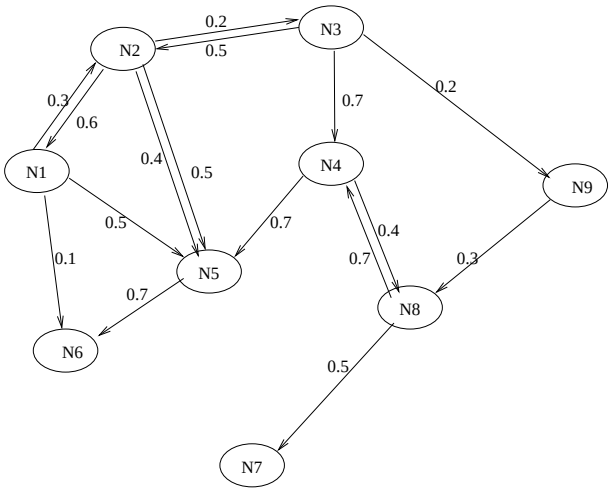

FIG. 3.4 – Exemple de graphe d'interdépendances

L'ensemble des nœuds, appelé réseau d'interdépendances, est représenté par un graphe orienté (N, L). N définit l'ensemble des nœuds du graphe et L définit les arcs.

$$N = \{N_i\}_i \tag{3.1}$$

et

$$L = \{L_{i,j}\}_{i,j} \tag{3.2}$$

où $L_{i,j}$ est le lien entre les nœuds N_i et N_j. Ce lien est caractérisé par un poids $W_{i,j}$ qui quantifie l'interdépendance entre les agents associés : $Agent_i$ et $Agent_j$. La figure 3.4 donne un exemple de graphe où les liens avec un poids nul ne sont pas représentés. Un nœud est ainsi relié à un ensemble de nœuds qui peut être vide (l'agent ne dépend d'aucun autre agent) ou regrouper tous les autres nœuds du réseau. Cet ensemble est dynamique puisque des liens peuvent disparaître et de nouveaux liens peuvent apparaître.

Chaque nœud est géré par l'agent Moniteur associé à l'agent du domaine correspondant. Les nœuds sont automatiquement générés à l'activation du système. L'ensemble initial des liens de chaque nœud est vide.

Pour définir les dépendances $W_{i,j}$ de $Agent_i$ et $Agent_j$ ($Agent_j$ dépend de $Agent_i$), nous proposons de considérer deux types d'informations :

– $QI_{i,j}$: la charge de communication entre ($Agent_i$ et $Agent_j$),

– $NbM_{i,j}$: le nombre de messages envoyés de $Agent_i$ à $Agent_j$.

Nous proposons de définir $W_{i,j}$ en fonction de $QI_{i,j}$, $NbM_{i,j}$ ainsi que d'informations globales telles que la charge de communication moyenne et le nombre de messages moyen.

Adaptation de l'interdépendance

Toute modification de l'agent du domaine entraîne une modification de son nœud. Celle-ci peut ensuite être propagée aux nœuds voisins en modifiant les liens entre le nœud d'origine et les nœuds voisins.

Les modifications du réseau d'interdépendances sont contrôlées par les agents Moniteurs. Ces derniers permettent d'adapter le graphe d'interdépendances aux différents changements des agents du domaine (par exemple, l'arrivée de nouveaux agents, ...). La mise à jour du graphe est distribuée. Chaque agent met à jour le dépendance de son nœud avec les autres nœuds et envoie les informations concernant son nœud à l'agent Observateur. Les différents agents Observateurs construisent ensuite des informations globales (charge de communication moyenne QI, nombre de message moyen NbM, ...) et les envoient aux agents Moniteurs.

Pour un intervalle de temps Δt, la charge de communication globale $QI(\Delta t)$ et le nombre de messages global $NbM(\Delta t)$ sont calculés comme suit :

$$QI(\Delta t) = Mop1(QI_{1,1}(\Delta t), QI_{1,2}(\Delta t)..., QI_{n,n}(\Delta t)) \tag{3.3}$$

$$NbM(\Delta t) = Mop2(NbM_{1,1}(\Delta t), NbM_{1,2}(\Delta t)..., NbM_{n,n}(\Delta t)) \tag{3.4}$$

où Mop1 et Mop2 sont des opérateurs d'agrégation. Pour nos expérimentations nous avons utilisé la moyenne.

Algorithme 2 donne un aperçu de l'opération d'adaptation du réseau d'interdépendances. Cette adaptation est fondée sur des informations locales (charge de communication, ...) et des informations globales (agrégation des différentes charges de communication). Cet algorithme est utilisé par chaque agent Moniteur pour gérer le nœud associé.

Dans cet algorithme, nous n'avons considéré que deux types d'informations : nombre de messages et la quantité d'informations échangés. Ces deux types d'informations sont souvent suffisantes pour évaluer les interdépendances. Cependant, dans beaucoup d'applications, il est nécessaire de tenir compte des :

– types de messages (performatifs par exemple),
– priorité des messages,
– séquences de messages (protocole utilisé).

L'algorithme proposé peut en effet être réutilisé pour intégrer ces différents types d'informations.

Algorithme 2 Algorithme d'adaptation des interdépendances d'un agent

Algorithme d'adaptation des interdépendances d'un agent $Agent_i$.

Require: $QI(\Delta t)$ et $NbM(\Delta t)$ fournis par l'agent Observateur
Ensure:

1: **for** tout i **do**
2: **for** tout j différent de i **do**
3: Calculer :

$$myQI = ((QI_{i,j}(\Delta(t) - QI(\Delta t)/QI(\Delta t) \tag{3.5}$$

$$MyNbM = (NbM_{i,j}(\Delta(t) - NbM(\Delta t))/Nb(\Delta t)) \tag{3.6}$$

4: Mettre à jour les poids en utilisant la règle suivante :

$$W_{i,j}(t + \Delta t) = W_{i,j}(t) + (a * MyQI + b * MyNbM)/(a + b) \tag{3.7}$$

5: **end for**
6: **end for**

3.4.3 Evaluation de la criticité d'agents

Dans un système multi-agents, l'activité interne d'un agent ne peut pas toujours être facilement observée. L'observation est en effet souvent restreinte aux événements et aux informations du niveau système. Pour évaluer le degré d'activité d'un agent du domaine, nous nous basons sur des mesures du niveau système fournies par le module d'observation de DarX. Pour un agent $Agent_i$ et un intervalle de temps Δt, ces mesures donnent, par exemple, le temps CPU utilisé (cp_i). Ce dernier est utilisé pour évaluer le degré d'activité de l'agent aw_i comme suit :

$$aw_i = cp_i/\Delta t \tag{3.8}$$

Les interdépendances et le degré d'activité de chaque agent du domaine sont utilisés pour déterminer sa criticité. La criticité de $Agent_i$ peut être calculée comme suit :

$$w_i = (a_1 * AgOp(W_{j,ij=1,m}) + a_2 * aw_i)/(a1 + a2) \tag{3.9}$$

où `AgOp` est un opérateur d'agrégation, a_1 et a_2 sont les poids donnés aux deux types de paramètres (interdépendances et degré d'activité). Leurs valeurs par défaut sont (a_1=0.5 et a_2=0.5), mais ils peuvent être initialisés par le concepteur.

Par ailleurs, la méthode à base de rôles que nous avons élaborée (voir [53]) peut être utilisée pour déterminer l'importance d'un agent. La cricitité dans ce cas sera calculée en fonction de son degré d'activité, de ses dépendances et de ses rôles.

3.5 Nombre de réplicats

La criticité d'un agent est utilisée pour déterminer le nombre de ses réplicats. Pour chaque agent $Agent_i$, sa criticité w_i est utilisée pour déterminer le nombre de réplicats. $Agent_i$ est répliqué en fonction de :

- w_i : sa criticité,
- W : la somme des criticités de tous les agents, W n'est pas calculée à chaque pas,
- rm : le nombre minimum de réplicats, il est défini par le concepteur,
- Rm : les ressources disponibles qui définissent le nombre de réplicats possibles.

Le nombre de réplicats nb_i de $Agent_i$ peut-être déterminé comme suit :

$$nb_i = rounded(rm + w_i * Rm/W) \tag{3.10}$$

nb_i est ensuite utilisé par DarX pour mettre à jour le nombre de réplicats de $Agent_i$.

3.6 Vers un environnement de développement de systèmes multi-agents résistants aux pannes

L'objectif du projet ALSACE (Adaptation dynamique de Logiciels par des Composants Evolutifs) est d'offrir une plate-forme multi-agents intégrant un service de tolérance aux pannes transparent, adaptatif et indépendant du niveau applicatif des systèmes multi-agents. Cette plate-forme est basée sur le framework DarX (proposé par les membres de SRC) et implémente le mécanisme de réplication adaptatif que nous (membres d'OASIS) proposons. Les agents Moniteurs et Observateurs sont intégrés à DarX et sont automatiquement activés.

Pour faciliter l'intégration des plates-formes multi-agents existantes et ce middleware, nous avons élaboré en collaboration avec des stagiaires de DEA un adaptateur qui permet de réaliser une passerelle entre les plates-formes multi-agents et DarX (voir figure 3.5). Cet adaptateur permet d'encapsuler les agents dans des DarxTask. Les agents peuvent ainsi être distribués et répliqués dynamiquement. Pour la validation de ce travail nous avons utilisé la plate-forme DIMA.

L'architecture modulaire de DIMA a facilité cette intégration. Toute application écrite avec DIMA peut en effet être réactivée avec DIMA+DarX. Dans DIMA, un agent est décrit indépendamment de la plate-forme d'exécution (avec ou sans DarX). Mais au moment de son activation, on choisit la plate-forme d'exécution et la méthode d'activation correspondante. La réception des messages est assurée par la DarxTask, qui lorsqu'elle reçoit un message, le dépose dans la boîte aux lettres de l'agent correspondant. Les agents DIMA héritent ainsi de toutes les caractéristiques des DarxTask. Ils peuvent être répliqués et déployés dynamiquement.

Dans cette partie, nous présentons l'exemple d'agents *agenda électronique* qui nous a servi de fil conducteur pour nos tests. Les expériences sont ensuite explicitées avant d'en voir les résultats et conclusions qui en ressortent.

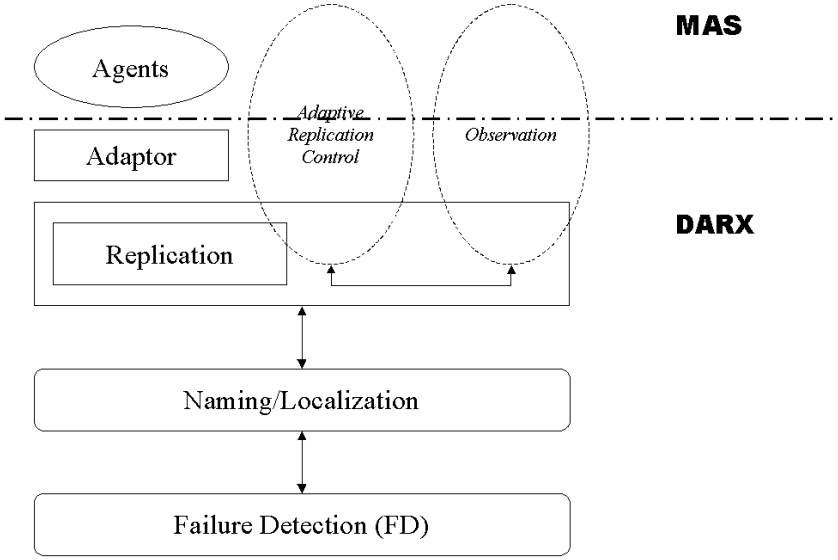

FIG. 3.5 – Vue générale de l'environnement de développement de systèmes multi-agents résistants aux pannes

3.6.1 Exemple de l'agenda électronique

L'exemple test de notre plate-forme est un système multi-agents utilisé en tant qu'assistant pour la prise de rendez-vous automatique. Chaque participant utilisateur possède un agenda électronique qui lui présente son emploi du temps. Cet agenda constitue l'interface qui existe entre l'utilisateur et le monde des agents : derrière chaque agenda se trouve un agent en charge de le gérer. Un utilisateur peut commander à son agent de prendre un rendez-vous à une date précise et avec certaines personnes et l'agent s'activera afin de répondre au mieux à cette requête. Les agents des différents utilisateurs se consultent les uns les autres afin de satisfaire au mieux les intérêts de chacun, et modifient en conséquence l'agenda dont ils ont la charge. Les interactions entre agents tournent autour de la prise de rendez-vous pour des meetings. Les éléments qui décrivent un meeting sont les suivants : un titre, une description, une durée, le dernier jour où ce meeting peut toujours être valide (date maximale de planification), les participants nécessaires et optionnels, si la personne créant le meeting est nécessaire ou optionnelle, le nombre de minutes avant lesquels le meeting ne doit pas être planifié (garantissant ainsi un certain temps avant le meeting).

Un utilisateur peut remplir ces informations et demander à l'agent de trouver la meilleure date pour tous les participants au meeting, puis de l'inscrire dans l'agenda. Pour ce faire, les agents suivent dans notre exemple des protocoles d'interactions tels que le *Contract Net* de FIPA [19].

3.6.2 Evaluation de performances

Pour valider la plate-forme réalisée (DIMA + DarX), nous avons effectué plusieurs expérimentations afin d'évaluer différents paramètres tels que le temps de transmission de messages et le temps de réplication des agents.

Le premier test s'est porté sur l'évaluation du coût de la transmission d'un message (de taille variable) entre deux agents communicants, activés sur des machines distantes. Ce test nous permet d'évaluer l'influence de la distribution des agents sur des machines distantes, sachant que le temps de transmission de messages entre agents locaux est négligeable. Nous avons considéré les paramètres suivants :

- taille du message transmis (variations entre 100 octets et 6400 octets).
- nombre de réplicats en stratégie active, de l'agent récepteur : cas sans réplication, un réplicat, deux réplicats ou trois réplicats.

Note : Les tests que nous avons réalisés, ont été effectués sur six machines de type Intel(R) Pentium(R) 4 CPU 2.00GHz, 526 M Octets de mémoire vive RAM.

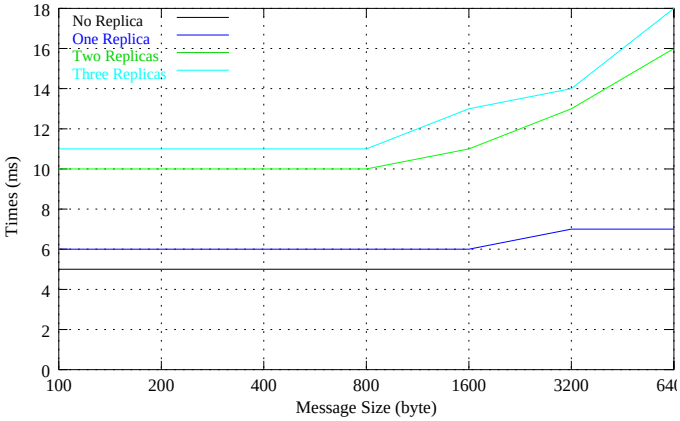

FIG. 3.6 – Temps de transmission de messages

Sur la figure 3.6, nous remarquons que les temps de transmission de messages entre les deux agents, sans réplication du récepteur, est constant (5ms). La variation de la taille du message entre 100 et 6400 octets n'affecte pas le temps de réponse du système. Pour un seul réplicat actif, nous remarquons que le temps commence à changer à partir d'une taille de message supérieure à 1600 octets. En revanche, pour deux et trois réplicats, la différence des temps commence à être relativement significative à partir d'une taille de messages supérieure à 800 octets. Toutefois, dans le cadre d'applications de système multi-agents, de telles tailles de messages se présentent rarement.

3.6.3 Evaluation de la criticité

Ce deuxième test s'est porté sur l'évaluation de la méthode de calcul de la criticité d'agent que nous avons proposée dans ce chapitre. Nous avons considéré 20 agents assistants : un initiateur et plusieurs participants. Pour le calcul des interdépendances, nous n'avons considéré que les paramètres suivants :

- le nombre de messages transmis.
- la taille de ces messages (quantité d'informations transmises).

Note : Ces tests ont également été effectués sur six machines de type Intel(R) Pentium(R) 4 CPU 2.00GHz, 526 M Octets de mémoire vive RAM.

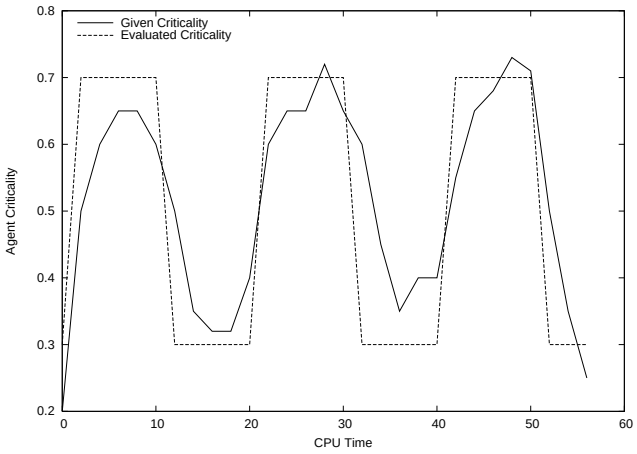

FIG. 3.7 – Criticité d'un agent

La figure 3.7 compare les deux criticités :

- la criticité donnée par le concepteur,
- la criticité calculé par le système.

Nous remarquons que la criticité évaluée par le système est très proche de celle donnée par le concepteur. Cependant, nous avons effectué plusieurs expérimentations pour fixer les paramètres tels que l'intervalle d'observation (Δt) et la fonction d'agrégation.

3.6.4 Tests de robustesse

Pour faire les tests de robustesse, nous avons implémenté un simulateur de pannes. Ce simulateur choisit un agent de façon aléatoire et détruit la thread correspondante. Pour chaque expérimentation, nous définissons :

- DS : la durée de simulation,
- NP : le nombre de pannes.

Le simulateur génère en effet une panne tous les Δt défini comme suit :

$$\Delta t = \frac{DS}{NP} \tag{3.11}$$

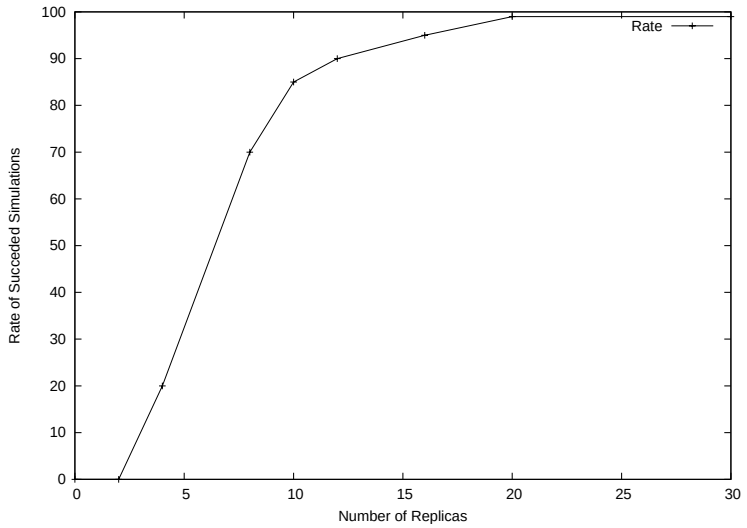

FIG. 3.8 – Taux de simulations réussies pour chaque taux de réplication

Dans cette expérimentation, nous avons considéré 100 agents distribués sur 10 machines. La durée de simulation $DS = 10$ mn et le nombre de pannes NP=100. Nous avons répété l'expérimentation plusieurs fois en variant le nombre de ressources (nombre total de réplicats).

la figure 3.8 montre le taux de succès SR en fonction du nombre de réplicats.

$$SR = \frac{NSS}{TNS} \tag{3.12}$$

où

- NSS est le nombre de simulations qui n'ont pas échoué,
- TNS est le nombre total de simulations.

Une simulation échoue si une panne détruit un agent critique qui n'a aucun réplicat. Ces expérimentations montrent que le nombre de ressources doit être supérieur au nombre d'agents critiques.

Ces résultats sont préliminaires, mais ils sont encourageants et montrent l'intérêt de l'approche que nous avons adoptée.

3.7 Conclusion

L'approche de tolérance aux pannes que nous proposons est fondée sur le framework DARX qui offre plusieurs stratégies de réplication et permet de changer dynamiquement la stratégie de réplication. Cette approche présente certaines similitudes avec celle de Fedoruk et Deters [35], du point de vue des mécanismes de tolérance aux pannes (réplication et groupe de communication) introduits. Cependant, plusieurs avantages sont à signaler au niveau architectural et fonctionnel, notamment la gestion des réplicats,

et les communications agents-agents et agents-réplicat. Notre approche se veut être plus transparente, et complètement distribuée. Nous introduisons le concept d'adaptation dynamique qui, jusqu'à présent n'a jamais été traité dans le cadre d'approches de réplication d'agents.

L'implémentation des mécanismes et des modèles élaborés a permis de développer une plate-forme multi-agents. Cette plate-forme de développement de systèmes multi-agents intègre les mécanismes de tolérance aux pannes, et offre ce service d'une manière transparente, n'imposant aucune modification fonctionnelle ou structurelle au niveau des comportements des agents développés. Une telle solution devrait pouvoir fiabiliser les applications multi-agents sans aucune contrainte de compétences et de connaissances des agents. Cependant, les expérimentations réalisées sont très simples. Une des perspectives de ce travail consiste en effet à valider cette plate-forme sur une application réelle complexe (large échelle, agents adaptatifs, ...).

Chapitre 4

Firmes et formes organisationnelles

Ces travaux s'inscrivent dans le cadre d'une collaboration avec Rodolphe Durand de l'Ecole de Management de Lyon. Cette collaboration a démarré en 1995 dans le cadre de ma thèse où nous avons montré l'intérêt des systèmes multi-agents en modélisant une population simplifiée de firmes. Elle s'est poursuivie après la thèse en s'appuyant sur des modèles théoriques.

4.1 Problématique

De nombreux travaux sur la modélisation et l'étude de l'évolution des firmes et des organisations ont été élaborés ces dernières années. Deux principales approches ont été clairement identifiées : l'écologie organisationnelle et le management stratégique. Elles étudient séparément les deux principaux problèmes de sélection et d'adaptation :

- *l'écologie des organisations* : selon l'argument écologique, les évolutions dans les organisations sont principalement dues aux processus de variation, de sélection et de rétention à différents niveaux tels que les firmes et les formes organisationnelles. Par exemple, Baum et Rao [4] ainsi que Lewin et Volberda [77] se basent sur des données historiques pour étudier l'influence des facteurs institutionnels, écologiques et stratégiques sur les performances et la survie des firmes et des organisations. Ils étudient notamment les taux de création et de disparition des firmes.
- *le management stratégique* : selon l'argument stratégique, les changements dans les organisations sont essentiellement dus au processus d'adaptation au niveau des firmes. Les travaux proposés (voir par exemple les travaux de Mizuta et Yamagata [86], Lomi et Larsen [79], Levinthal [76], et Bruederer et Singh [13]) modélisent la structure interne des firmes, leur processus de décision ainsi que les différentes relations qui peuvent exister entre elles.

Dans ces deux approches, les deux mécanismes (adaptation et sélection) sont étudiés séparément. La sélection n'est ainsi pas utilisée pour expliquer ou pour être expliquée en fonction de l'adaptation des firmes.

De récentes réflexions s'orientent vers l'unification des deux approches (voir [30]). Elles soulignent l'intérêt d'étudier à la fois le processus écologique et le processus stratégique ainsi que leurs interactions. Par exemple, le problème de l'émergence ou de l'évolution de nouvelles formes organisationnelles a fait l'objet de nombreuses recherches. Ces formes organisationnelles sont souvent données comme le résultat des activités des firmes entrantes. Ces travaux se basent sur des techniques d'Intelligence Artificielle telles que les algorithmes génétiques.

La sélection et l'adaptation sont maintenant présentées comme des processus complémentaires. Cependant, aucun modèle, à notre connaissance, n'a encore été proposé pour étudier simultanément les deux problèmes jusque-là abordés indépendamment. Ceci est notamment dû à la complexité des systèmes économiques. L'étude de ces systèmes nécessite la modélisation des deux populations (firmes et formes organisationnelles), la dynamique de chacune de ces populations ainsi que l'étude du rapport dynamique entre elles.

Le but des travaux présentés dans ce chapitre est de pallier cette lacune en suggérant un modèle qui intègre les deux théories afin de permettre une analyse à deux niveaux : firmes et formes organisationnelles. Notre solution est basée sur l'utilisation des systèmes multi-agents adaptatifs. Nous allons donc montrer que les systèmes multi-agents adaptatifs permettent d'étudier les deux problèmes (adaptation et sélection) simultanément. Nous introduisons un modèle multi-agents qui intègre les firmes (niveau micro) et les formes organisationnelles (niveau macro), et qui représente les différentes relations entre ces deux niveaux. Ceci dans le but de vérifier :

- comment l'adaptation au niveau des firmes entraîne t-elle l'émergence de nouvelles formes organisationnelles ?
- comment les formes organisationnelles influent-elles sur les firmes et comment les firmes interprètent-elles ces influences ?

Le chapitre est organisé en cinq sections. Nous présentons respectivement, dans les première et deuxième sections, le modèle des firmes et celui des formes organisationnelles. Dans la troisième section, nous décrivons les différentes relations entre les firmes et les formes organisationnelles. La dernière section présente les principaux apports de notre modèle et les travaux en cours.

4.2 Firmes

Différentes théories ont été conduites pour l'étude de la croissance et l'évolution des firmes. Durand [29] en fait un synthèse. Pour répondre aux attentes des décideurs, Penrose [96] introduit une approche originale basée sur la réalité des managers. Elle propose d'enrichir la représentation des firmes en considérant plus d'attributs que ceux émis dans la théorie classique tels que la quantité et la qualité de production. La fonction d'une entreprise consiste à utiliser des ressources productives en vue de fournir des biens et des services selon des plans conçus et mis en œuvre par son organisation. Une entreprise est ainsi caractérisée par la manière dont l'autorité gestionnaire emploie ses ressources qui sont soit physiques (équipement, terrains, matières premières, ...) soit

humaines (personnel administratif, technique, ...). Le but de la gestion des ressources et de leur allocation est la recherche de l'accroissement des profits. La firme est ainsi considérée comme une entité administrative autonome qui exploite une collection de ressources physiques et humaines.

Le modèle économique que nous utilisons s'inscrit dans cette théorie. Il est fondé sur la théorie des ressources. Il définit un ensemble de firmes en compétition les unes avec les autres qui interagissent indirectement via le marché. Chaque firme, représentée par un agent, observe les autres et essaie de choisir une stratégie appropriée en fonction du contexte et, ce, dans le but de croître, de survivre et d'améliorer ses performances.

4.2.1 Modèle de firmes

Dans notre modèle, une firme est principalement définie par :
- un ensemble de ressources représentées par un vecteur de variables (noté X),
- un ensemble de performances représentées par un vecteur de variables (noté Y). Ces Y dépendent des X,
- un capital K et un budget B,
- un ensemble de stratégies. Chaque stratégie élabore un plan d'utilisation des ressources. Ce plan donne une combinaison adéquate des ressources disponibles créées ou acquises en fonction du budget. Il instaure un ordre de priorité des ressources et une méthode d'affectation du budget aux ressources.

Le comportement de la firme est défini par les étapes suivantes :
- observation de la compétition,
- détermination des performances relatives,
- calcul du budget à investir,
- choix et application d'une stratégie,
- calcul des nouvelles performances.

Après la mise à jour de la représentation de la compétition, basée sur une perception du marché où se trouvent toutes les informations relatives aux autres firmes, la firme calcule l'indicateur de performance V qui correspond à une évaluation de sa performance relative. V est calculé en fonction de l'évolution de ses performances et de celles de l'ensemble des autre firmes dans le marché. Cette valeur V est ensuite utilisée pour déterminer des paramètres internes tels que le capital K et le budget B. Celui-ci correspond au flot d'investissements dont dispose une firme pour améliorer ses stocks de ressources et de compétences (X).

Une stratégie est par la suite choisie en tenant compte, notamment, des performances de la firme et de celles des autres firmes. En fait, toute possibilité de croissance est conditionnée par la combinaison adéquate entre les différents types de ressources et de compétences.

Une fois les ressources X mises à jour par la stratégie, elles sont transformées selon le modèle de Lisrel [28] jusqu'à l'obtention des performances Y. Le modèle de Lisrel correspond à une méthode statistique permettant de tester, d'une part, les rapports de correspondance entre les variables observables et les concepts (tels que la non-transférabilité

des ressources productives, la non-imitabilité, la non-substituabilité) dont ils sont les expressions et, d'autre part, d'analyser les relations statistiques entre les concepts ainsi opérationnalisés.

4.2.2 Processus de décision

Le processus de décision d'une firme consiste à sélectionner une stratégie en fonction du contexte. Différents processus de décision peuvent être utilisés (voir chapitre 2) : moteur d'inférences, *case based-reasoning*, systèmes de classeurs, etc. Nous avons choisi deux types de mécanismes :

- un mécanisme réactif : une base de règles a été définie par l'expert à la conception,
- un mécanisme adaptatif : un système de classeurs a été utilisé pour construire dynamiquement la base de règles et l'adapter à l'évolution du contexte de l'agent.

La base de règles définie par l'expert prend en compte la variation des valeurs des paramètres internes de la firme et de celles de sa perception de la compétition. Ceci est un exemple de règle :

```
if (Vt - Vt-1) > 0 then strategy_t
```

où *strategy_t* est la stratégie appliquée à l'étape précédente.

L'ensemble des variations considérées est ainsi connu *a priori*.

Le mécanisme adaptatif tient compte des variations dynamiques du contexte de la firme (état interne et perception des autres firmes). Chaque firme construit dynamiquement une représentation interne de son contexte (voir Chapitre 2). Cette représentation est ensuite utilisée pour choisir une stratégie.

Pour construire dynamiquement ces règles de décision, nous avons utilisé les systèmes de classeurs. Un classeur est défini comme suit :

```
[context] [strategy] reward
```

Le contexte intègre des données internes (K, X, V,...) et des données issues de la perception de la compétition (Ymin, Yaverage, Ymax).

Exemple de classeur :

```
 Classifier_1 :
[K = 500,
V = 0.4
X ={2,1,5,3,4,8,6,2}
Y_Min= {33,12, 45.11, 70.90}
Y_Max= {36.2, 52.2, 76.1}
Y_average= {34.1, 49.6, 75.7}]
[Strategy= strategy_1]
r= 0.5
```

L'utilisation d'un système de classeurs permet de déterminer la meilleure stratégie et donc les meilleures variations afin de les réutiliser. Il permet ainsi d'introduire le modèle VSR (Variation, Sélection, Rétention) qui assure que l'organisation ayant connu, subi ou engendré les variations soit en meilleure posture après la survenance des variations conservées [30].

4.2.3 Expérimentations

Dans notre modèle, une firme est principalement caractérisée par son processus de décision. Toutes les populations de firmes utilisées sont dynamiques ; des firmes peuvent disparaître ou entrer durant la simulation.

Le but de nos premières expérimentations est la comparaison des performances des deux types de firmes : réactives et adaptatives.

Nous considérons deux populations de 300 firmes. Ces deux populations ont les mêmes caractéristiques initiales (capital, ressources, ...). Elles ne se distinguent que par le processus de décision (base de règles ou système de classeurs). Nous avons observé l'évolution de ces deux populations. Les résultats (voir figure 4.1) montrent que les firmes adaptatives sont plus performantes à long (moyen) terme que les réactives. Ceci est notamment dû à l'adaptation de la base de classeurs aux différents changements au niveau du contexte de la firme. Cependant, ces firmes adaptatives ont beaucoup de difficultés dans la première phase du fait du choix aléatoire de la stratégie. Ceci peut être évité par l'initialisation de la population de classeurs établie, par exemple, sur la base de règles fournie par l'expert.

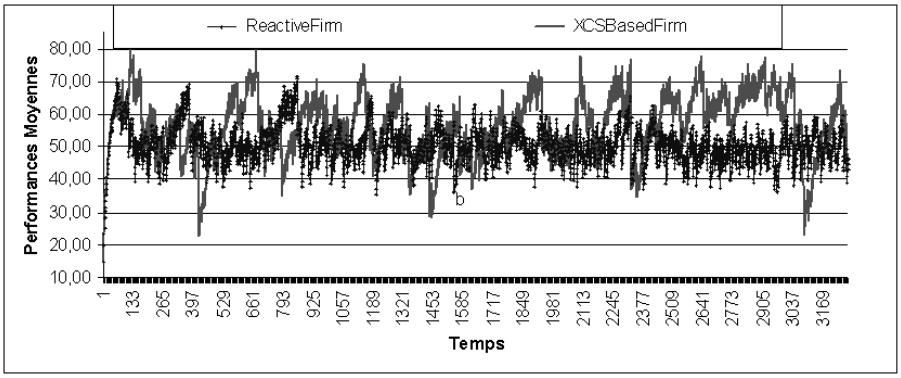

FIG. 4.1 – Performances des firmes réactives et adaptatives

La figure 4.2 montre qu'après une première phase difficile, la population de firmes adaptatives est légèrement plus résistante que la population de firmes réactives. Cependant, le nombre de firmes est très bas pour les deux populations.

4.2.4 Discussion

Comme nous avons pu le remarquer durant nos expérimentations, face à un environnement caractérisé par une forte compétition, le taux de mortalité des firmes (réactives ou adaptatives) est très élevé. Baum [4] explique ce taux de mortalité par les comportements de surexploitation et de surexploration. Les firmes réactives utilisent les mêmes stratégies ; elles ont donc un comportement de surexploitation. En revanche, les firmes adaptatives commencent souvent leur activité par un comportement de surexploration. Ceci est notamment dû au fait que leur base de règles initiale est vide.

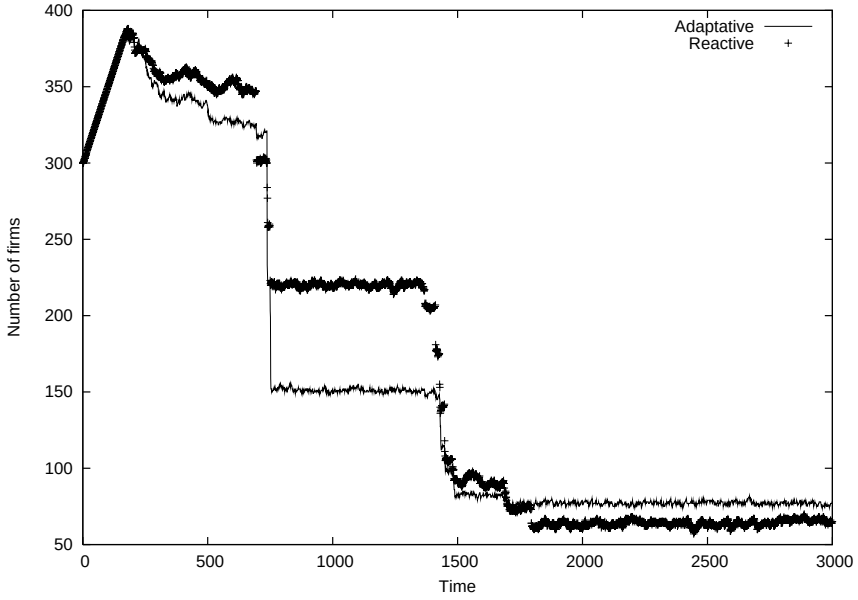

FIG. 4.2 – Nombres de firmes réactives et adaptatives

Les firmes qui survivent sont les firmes adaptatives qui résistent aux risques du comportement de surexploration durant la première phase. Ce comportement permet d'initialiser la base de classeurs en sélectionnant les variations (et donc les stratégies) les plus adéquates. Elles essaient ensuite d'exploiter les stratégies sélectionnées tout en essayant d'en explorer de nouvelles. Ces firmes sont capables :

- de faire face à la compétition dans un environnement turbulent ;
- d'établir un équilibre entre les capacités d'exploration (recherche de nouvelles routines) et d'exploitation.

Cependant, ces firmes se basent seulement sur les variations locales car elles n'ont pas de perception des variations des autres firmes. Une perception de ces variations permettrait de généraliser la sélection et d'imiter les firmes les plus performantes. Celles-ci ont donc besoin d'une perception de leur contexte organisationnel ou formes organisationnelles. Ces dernières sont définies et modélisées dans le paragraphe suivant.

4.3 Formes organisationnelles

Les formes organisationnelles ont été définies et représentées de différentes manières en économie. Nous allons d'abord exposer quelques définitions de la littérature puis nous introduirons un nouveau modèle.

4.3.1 Quelques définitions

Différents travaux en économie et en management ont étudié les formes organisationnelles, leur évolution et les facteurs qui favorisent leur émergence. Ces travaux s'appuient sur des exemples de formes organisationnelles mais la majorité ne propose aucune définition ou formalisation.

Pour Romanelli [104], le concept de forme organisationnelle désigne les caractéristiques d'une organisation qui, d'une part, l'identifient comme une entité distincte des autres et, d'autre part, la classent dans un groupe d'organisations similaires.

Dans la définition de Laszlo et al. [73], une forme organisationnelle est une identité externe. Ils accordent beaucoup d'importance aux perceptions et opinions des observateurs externes.

Pour définir l'évolution organisationnelle, Baum et Rao [4] ont eu recours à deux types de hiérarchies : la hiérarchie écologique et la hiérarchie généalogique. Les formes organisationnelles font partie de la hiérarchie généalogique décrite comme étant une mémoire institutionnelle permettant de préserver et de propager l'information d'organisation et de production. Cette hiérarchie est composée d'entités qui, avec le temps, persistent dans le même état ou avec des altérations résultat de leur reproduction. Par ailleurs, une hiérarchie écologique reflète la structure économique et l'intégration des systèmes organisationnels. Elle est composée d'entités historiques provenant de l'effet cumulatif de la variation et de la sélection à travers le temps, et des expressions structurelles et comportementales d'entités de la classe généalogique.

Une firme est donc une expression structurelle et comportementale d'une entité de la hiérarchie généalogique, à savoir une forme organisationnelle. L'ensemble des firmes qui adoptent la même forme organisationnelle génère une population dans la hiérarchie écologique. On peut donc remarquer la nécessité des deux hiérarchies pour la théorie de l'évolution organisationnelle.

D'après Baum et Rao [4], il y a deux niveaux d'évolution :

- *le niveau micro*, défini par les firmes dans lesquelles certaines interactions entre elles et avec leur environnement peuvent causer une variabilité dans les formes organisationnelles qu'elles adoptent ;
- *le niveau macro*, défini par les différentes formes organisationnelles dont l'interaction entre elles et avec l'environnement peuvent engendrer de nouvelles formes organisationnelles.

Une forme organisationnelle est ainsi une entité qui regroupe des firmes similaires. Ces dernières ont un comportement et une structure similaires.

4.3.2 Exemple

Dans le domaine de la vente de livres, la forme organisationnelle des librairies généralistes est différente de celle représentant les chaînes telles que la FNAC ou Virgin. Par ailleurs, les services de vente et de livraison en ligne sont une autre forme organisationnelle. Ces formes organisationnelles diffèrent par leurs stratégies, traduites par l'utilisation des différentes ressources physiques et humaines telles que :

- l'espace occupé : les librairies de quartier ne requièrent qu'un espace réduit leur permettant d'avoir un stock réduit de livres ; c'est ce qui engendre leur spécialisation dans un certain type de livres. La FNAC, quant à elle, nécessite plus d'espace du fait de la variété des produits qu'elle vend. En revanche, AMAZON ne nécessite qu'un espace moyen pour stocker ses produits.
- le personnel : le nombre d'employés diffère d'une forme organisationnelle à une autre. Pour AMAZON et la FNAC, le nombre très important est de l'ordre de milliers d'employés. Ce qui n'est pas le cas des librairies de quartiers qui nécessitent rarement plus d'un employé plus spécialisé que les premiers. Il connaît en général les livres qu'il vend et conseille les clients dans leurs acquisitions.
- le marketing : contrairement aux librairies de quartier, la FNAC et surtout AMAZON investissent beaucoup en marketing (achat d'annuaires, publicité sur le net, ...).
- les nouvelles technologies : les librairies virtuelles telles qu'AMAZON investissent beaucoup en matériel informatique (surtout lors de la création du site pour la gestion des ouvrages et des clients),

Les structures de ces trois firmes sont en effet très différentes. Ceci explique leur appartenance à trois formes organisationnelles.

4.3.3 Modèle de formes organisationnelles

Notre modèle de firmes (voir section 2) est basé sur les concepts de ressource et de stratégie. Chaque stratégie définit des variations de ressources. Les stratégies entraînent ainsi des différences ou des similarités structurelles des firmes qui les adoptent. Ces dernières peuvent en effet être organisées en fonction de ces variations qui sont le résultat des stratégies sélectionnées.

Une forme organisationnelle est une agrégation de firmes similaires, elle est ainsi définie par :

- des variations de ressources,
- des performances qui sont fonction des performances de ses différentes firmes.

Les variations de ressources sont décrites par une discrétisation de l'intervalle de variations possibles [0, 1].

- 0 correspond à aucune variation de la ressource,
- 1 correspond à l'allocation d'un budget maximum à cette ressource.

Différentes méthodes et techniques peuvent ainsi être utilisées pour déterminer cette décomposition. Nous distinguons deux principales catégories : la granulation exacte et la granulation floue. Zadeh [118] montre que la granulation floue est inspirée des capacités humaines à agir et à raisonner sur des informations issues de la perception. Elle fournit une meilleure représentation des différentes classes de variations et un très bon processus de décision.

Nous avons donc choisi la granulation floue. Chaque variation d'une ressource est ainsi décrite par des valeurs symboliques tels que petit, moyen, grand (voir figure 4.3). Dans notre représentation des formes organisationnelles, nous utilisons les mêmes sym-

boles pour les différentes ressources.

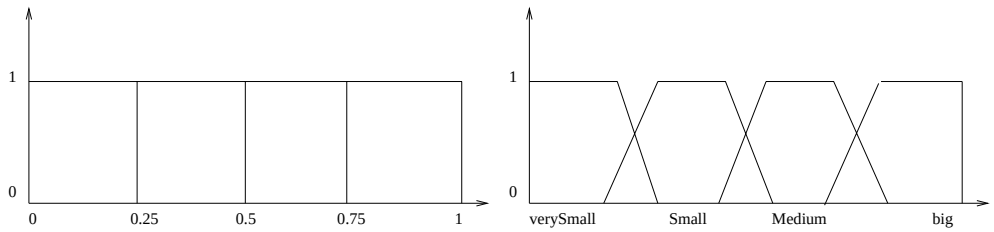

FIG. 4.3 – Example of granulation

Une forme organisationnelle est ensuite associée à un vecteur de valeurs floues. Par exemple, si nous considérons trois ressources, les valeurs floues (small, medium et big) peuvent correspondre a une forme organisationnelle qui donne la priorité à la troisième ressource.

Etant donné n ressources et un ensemble de k valeurs floues FZ,

$$FZ = (fv1, fv2, ..., fvk) \tag{4.1}$$

chaque forme organisationnelle fo_j peut être alors associée à un ensemble de n valeurs floues (fvj1, fvj2, ..., fvjn) où $fvji_{i=1..n} \in FZ$. Cet ensemble de valeurs floues permet de déterminer l'ensemble des firmes qui appartiennent à cette forme organisationnelle.

Les performances Pfo_j d'une forme organisationnelle fo_j sont ensuite définies par une agrégation des performances de ces firmes.

$$Pfo_j = aggreg(Y_1, Y_2, .., Y_m) \tag{4.2}$$

où $Y_{i,i=1,m}$ sont les performances des m firmes que la forme organisationnelle regroupe et *aggreg* est un opérateur d'agrégation. Différents opérateurs d'agrégation peuvent être utilisés (moyenne arithmétique, médiane, minimum et maximum, ...).

4.4 Des agents aux systèmes multi-agents adaptatifs

Différents modèles et expérimentations ont été introduits pour étudier les deux processus d'adaptation et de sélection. Cependant, la majorité de ces travaux n'a pas abordé la relation entre l'adaptation au niveau firme et la sélection au niveau population de firmes ou formes organisationnelles. Levinthal [76] est un des premiers à avoir montré que l'adaptation et la sélection ne sont pas des forces indépendantes, qu'elles sont fortement liées. Il suggère un modèle fondé sur les algorithmes génétiques. Dans ce modèle, chaque firme est représentée par N attributs dont la valeur est 0 ou 1. L'ensemble des formes organisationnelles est défini en considérant les 2^N combinaisons possibles des valeurs des N attributs. La *fitness* de chaque firme est déterminée en identifiant sa forme organisationnelle dans le paysage de la fonction fitness. Il montre que les firmes qui survivent sont les firmes qui réussissent à s'intégrer dans une forme organisationnelle performante.

Ce travail a ainsi montré que l'adaptation et la sélection sont des forces liées et non pas indépendantes. Cependant, il ne permet pas d'étudier l'évolution d'une population de firmes adaptatives et l'émergence de nouvelles formes organisationnelles.

La section suivante présente une solution pour répondre à ce problème.

4.4.1 Modèles multi-agents adaptatifs

Pour étudier l'évolution des firmes en tenant compte des deux problèmes (adaptation et sélection), nous proposons un modèle qui intègre des firmes et des formes organisationnelles et qui gère les liens entre elles. Ces deux types d'entités représentent les deux niveaux :

- niveau micro (firmes) : une organisation de firmes qui ont la capacité de percevoir leur environnement (les autres firmes, la population de formes organisationnelles) et agissent continuellement en fonction de leur état interne et de l'évolution de leur environnement.
- niveau macro (formes organisationnelles) : une population d'entités qui représentent les formes organisationnelles. Ce niveau donne au système (notamment la population de firmes) la capacité d'adaptation et la capacité d'auto-organisation.

Nous considérons des modèles économiques plus proches des organisations réelles où des firmes et des formes organisationnelles peuvent apparaître et disparaître dynamiquement. Le but de ce modèle est d'offrir des réponses aux questions suivantes :

- comment les firmes influent-elles sur les formes organisationnelles ?
- comment les firmes et les formes sont-elles sélectionnées ?
- comment les formes organisationnelles influent-elles sur les firmes ?

Les différents travaux réalisés en économie et en management ont montré que les modèles économiques sont des systèmes complexes dont l'évolution au niveau micro (firmes) affecte le niveau macro (formes organisationnelles). Par ailleurs, toute modification des formes organisationnelles affecte les firmes. Par exemple, la disparition d'une forme organisationnelle peut entraîner la disparition de toutes les firmes associées.

Pour gérer ces relations micro-macro, nous avons introduit une composante de couplage qui joue le rôle d'intermédiaire entre les firmes et les formes organisationnelles. Elle interprète les informations dans les deux sens. Elle observe les ressources et leurs variations et les traduit pour les formes organisationnelles. A l'inverse, elle détecte certaines variations dans les formes organisationnelles telles que leurs performances ou l'émergence d'une nouvelle forme organisationnelle et les communique aux firmes. Chaque firme génère ainsi, d'une part, ses propres variations et, d'autre part, elle a sa propre perception de la population de formes organisationnelles.

Cette composante de couplage permet ainsi d'observer les firmes et de gérer les relations entre les firmes et les formes organisationnelles. Nous proposons d'associer à chaque firme un agent moniteur pour gérer ses interactions avec la population de formes organisationnelles (voir figure 4.4).

Les interactions entre les deux niveaux (micro-macro) sont décrites dans les sections suivantes.

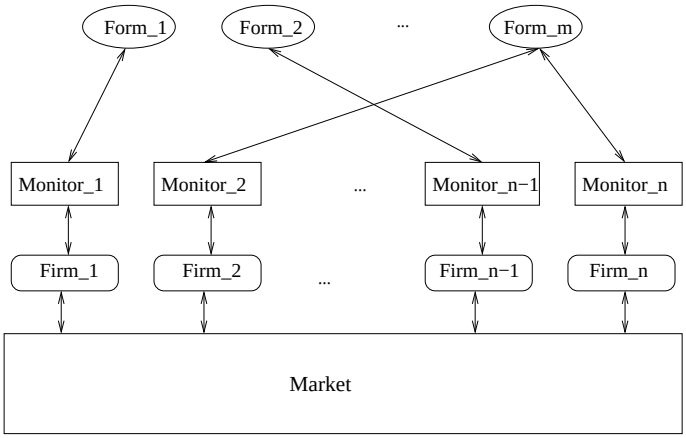

FIG. 4.4 – Firmes et formes organisationnelles

4.4.2 Interactions entre firmes et formes organisationnelles

Les interactions entre les firmes et les formes organisationnelles sont réalisées par une organisation de moniteurs. Ceux-ci maintiennent deux types d'informations : la tendance des variations des ressources au niveau des firmes et la tendance des variations de la population des formes organisationnelles. Toute modification des firmes est alors perçue par les agents moniteurs et propagée aux formes organisationnelles qui s'adaptent en conséquence. De plus, toute modification d'une forme organisationnelle est signalée aux firmes.

Un agent moniteur gère les interactions entre une firme et les formes organisationnelles. Il observe les variations de ressources de la firme et détermine la forme organisationnelle correspondante en utilisant les valeurs floues. Il calcule les degrés d'appartenance de la firme f_i aux différentes formes organisationnelles fo_j :

$$\pi(f_i, fo_j) = \sum_{k=1}^{n} \pi(fv_{jk}/\triangle_{i,k})/n \tag{4.3}$$

où :

- fv_{jk} est la valeur floue de la ressource k pour la forme organisationnelle fo_j,
- $\triangle_{i,k}$ est la variation de la ressource k pour la firme f_i.

$$\pi(fv_{jk}/\triangle_{i,k}) = \begin{cases} 0 & if \triangle_{i,k} < a_j ou \triangle_{i,k} >= d_j, \\ \frac{d_j - \triangle_{i,k}}{b_j - a_j} & if a_j <= \triangle_{i,k} < b_j, \\ 1 & if b_j <=_{i,k} < c_j, \\ \frac{\triangle_{i,k} - c_j}{d_j - c_j} & if c_j <= \triangle_{i,k} < d_j. \end{cases} \tag{4.4}$$

a_j, b_j, c_j et d_j sont les caractéristiques de la valeur floue fv_j (voir figure 4.5).

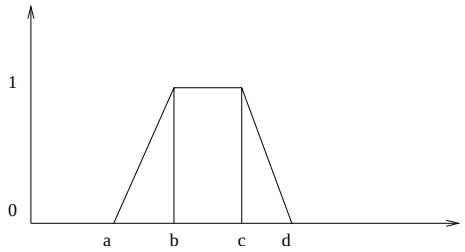

FIG. 4.5 – Caractéristiques d'une valeur floue

La firme f_i appartient ainsi à la forme organisationnelle fo_l telle que :

$$\pi(f_i, fo_l) = max_{j=1,m}\pi(f_i, fo_j) \tag{4.5}$$

et

$$\pi(f_i, fo_l) >= thGm \tag{4.6}$$

$thGm$ est un seuil défini par l'expert et m est le nombre de formes organisationnelles. Certaines firmes ne sont en effet attachées à aucune forme organisationnelle. Elles sont dites orphelines.

4.4.3 Emergence de formes organisationnelles

De nouvelles formes organisationnelles peuvent émerger et celles existant peuvent disparaître. Les formes organisationnelles sont tenues d'avoir de bonnes performances. Elles peuvent disparaître lorsque leurs performances se détériorent.

Pour contrôler les formes organisationnelles, nous avons introduit un agent appelé Contrôleur.

La création d'une nouvelle forme organisationnelle est conditionnée par l'existence d'une firme orpheline très performante. La meilleure firme orpheline f_i est utilisée pour créer une nouvelle forme organisationnelle. La définition de cette dernière est basée sur les variations de ressources de cette firme. Pour chaque ressource k de f_i et sa variation $\triangle_{i,k}$, et pour chaque valeur floue $fv_j \in FZ$, le Contrôleur calcule le degré d'appartenance :

$$\pi(fv_j/\triangle_{i,k}) = \begin{cases} 0 & if \triangle_{i,k} < a_j or \triangle_{i,k} >= d_j, \\ \frac{d_j - \triangle_{i,k}}{b_j - a_j} & if a_j <= \triangle_{i,k} < b_j, \\ 1 & if b_j <=_{i,k} < c_j, \\ \frac{\triangle_{i,k} - c_j}{d_j - c_j} & if c_j <= \triangle_{i,k} < d_j. \end{cases} \tag{4.7}$$

où a_j, b_j, c_j et d_j sont les caractéristiques de la valeur floue fv_j.

La valeur floue correspondant à la ressource k est fv_l telle que :

$$\pi(fv_l/\triangle_{i,k}) = max_{j=1,m}(\pi(fv_j/\triangle_{i,k})) \quad (4.8)$$

L'ensemble des valeurs floues $fvl_{l=1,n}$ est ensuite utilisé par le Contrôleur pour définir la nouvelle forme organisationnelle. Il calcule les degrés d'appartenance des autres firmes orphelines à cette nouvelle forme organisationnelle et met à jour l'ensemble de ces firmes.

4.4.4 Des firmes adaptatives

Pour tenir compte des formes organisationnelles, nous avons enrichi le modèle de firmes adaptatives proposé dans la section 2. Une firme adaptative est une firme qui prend en compte l'évolution du modèle de compétition et celle de la population des formes organisationnelles. Dans la version précédente, le contexte était défini par le modèle de la compétition. Ce modèle peut être défini par des indicateurs de performances tels que le minimum, le maximum et la moyenne. Ce contexte a été étendu par des indicateurs d'évolution de la population des formes organisationnelles tels que la meilleure forme organisationnelle ou la moyenne.

Exemple de classeur :

```
Classifier_f :
 [K = 500, V= 0.4
  X ={2,1,5,3,4,8,6,2}
  Y={35.25, 52.33, 75.12}
  Y_Min (33,12, 45.11, 70.90)
  Y_Max (36.2, 52.2, 76.1)
  Y_average(34.1, 49.6, 75.7)
  Form = Form_2,
  BestForm = Form_4
  WorstForm = Form_6]
[Strategy= strategy_1]
 r= 0.7
```

4.4.5 Expérimentations

L'implémentation du modèle intégrant les firmes et les formes organisationnelles a permis d'obtenir un simulateur ("Marché Virtuel") pour la validation des différentes théories et hypothèses des chercheurs en économie et en management. Nos premières expérimentations ont pour but la validation de cet outil et de nos hypothèses sur l'adaptation et la sélection.

Nous considérons deux populations de 500 firmes chacune. Elles ont les même caractéristiques mais utilisent respectivement :

- des systèmes de classeurs en ne considérant que les firmes,
- des systèmes de classeurs en considérant les firmes et les formes organisationnelles.

Nous avons réalisé plusieurs expérimentations. La figure 4.6 compare les performances des deux types de firmes adaptatives (avec et sans formes organisationnelles). Elle montre que les firmes adaptatives qui tiennent compte des formes organisationnelles sont généralement plus performantes que les autres.

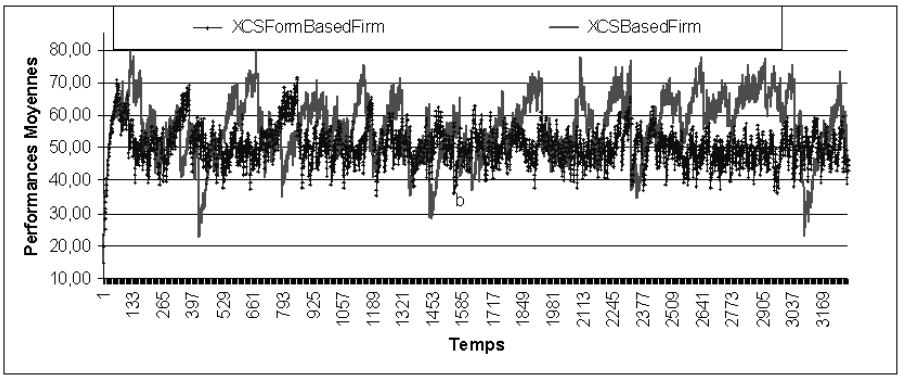

FIG. 4.6 – Comparaison des types de firmes adaptatives (sans et avec formes organisationnelles)

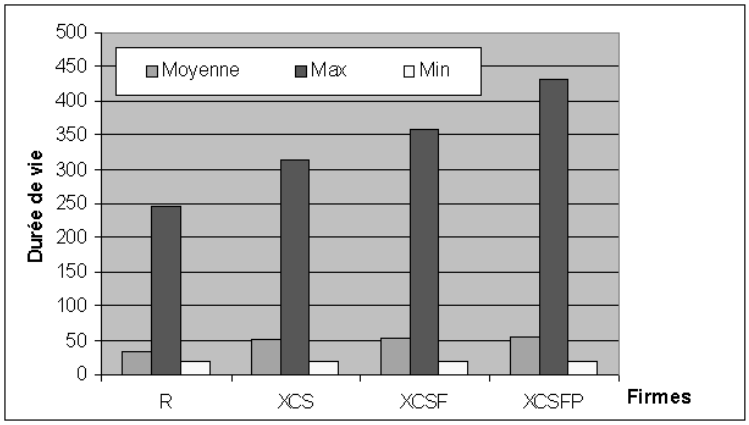

FIG. 4.7 – Durée de vie des différents types de firmes

La figure 4.7 montre les durées de vie des trois types de firmes : réactives (R), adaptatives (XCS), adaptatives avec formes organisationnelles sans ou avec populations initiales de classeurs (XCSF et XCSFP). Les durées de vie moyennes de ces firmes sont respectivement : 35, 51, 54, 56. Les firmes adaptatives qui tiennent compte des formes organisationnelles sont plus résistantes ; elles ont ainsi une durée de vie plus grande. Par ailleurs, l'initialisation de la population de classeurs permet d'éviter les risques des choix aléatoires qui permettent d'initialiser cette population. Cependant, le taux de disparition de ces firmes reste encore trop élevé. Ceci est notamment dû au problème de convergence de la base de classeurs (voir figure 4.9).

La figure 4.8 résume les expérimentations que nous avons réalisées pour comparer le nombre de formes organisationnelles en considérant deux catégories de populations de firmes dont le nombre de firmes et le nombre de formes initiaux sont identiques (1500 firmes) :

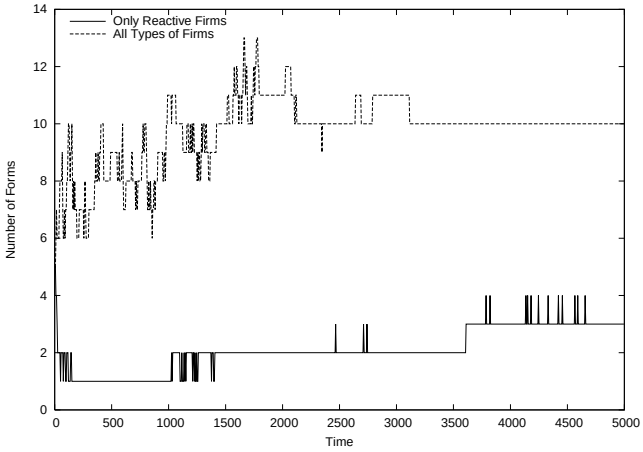

FIG. 4.8 – Nombre de formes organisationnelles

- une population de firmes réactives,
- une population de trois types firmes : réactives, adaptatives ne tenant pas compte des formes organisationnelles et adaptatives tenant compte des formes organisationnelles.

Les résultats montrent que l'adaptation au niveau des firmes facilite l'émergence de nouvelles formes organisationnelles.

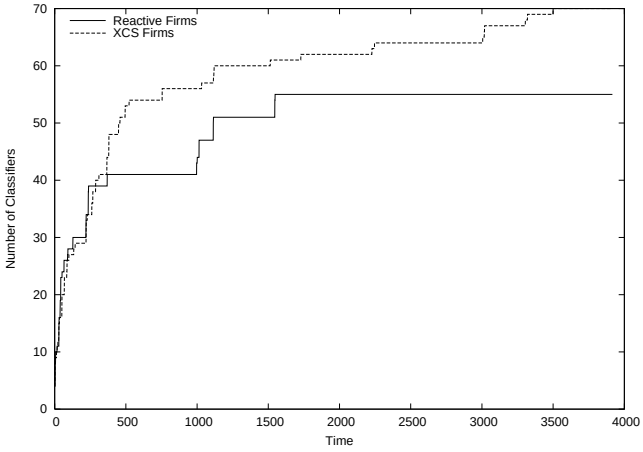

FIG. 4.9 – Nombre de classeurs

Dans les dernières expérimentations, nous avons considéré deux populations de 500 firmes : des réactives, et des adaptatives ne tenant pas compte des formes organisationnelles. Nous avons rajouté à chacune des populations une firme adaptative et nous avons observé l'évolution du nombre de classeurs de cette firme. Ces expérimentations

(voir figure 4.9) montrent que le nombre de classeurs se stabilise assez rapidement ($\sim$ 1500 périodes) dans le cas où la firme est dans une population réactive. Cependant, cette période est beaucoup plus importante ($\sim$ 3500 périodes) quand la firme est dans une population de firmes adaptatives.

4.5 Conclusion

Dans ce chapitre, un exemple d'application des systèmes multi-agents adaptatifs à la simulation de modèles économiques a été présenté. Cette application est réalisée dans le cadre d'une coopération avec Rodolphe Durand (Ecole de Management de Lyon) qui a démarré en 1995.

Nous avons introduit un modèle de firmes basé sur la théorie des ressources. Ce modèle est très simple, mais reflète des caractéristiques importantes des firmes (Il est détaillé dans [31]). Une première phase de ce projet (réalisée par des stagiaires de DEA) a permis d'illustrer l'intérêt des agents adaptatifs en comparant les deux populations de firmes : réactives et adaptatives. Elle a également montré les limites des modèles économiques qui représentent un ensemble de firmes en interaction indirecte via le marché sans tenir compte des formes organisationnelles. Dans le cadre de la thèse de Lilia Rejeb, nous avons proposé un modèle de formes organisationnelles [60] [59]. Nous avons ensuite élaboré un modèle multi-agents adaptatifs pour l'intégration des deux modèles (firmes et formes organisationnelles) et les différentes relations qui peuvent exister entre elles. Les premiers résultats sont cohérents avec les différentes théories de la littérature.

La première perspective de ce projet consiste à finaliser le simulateur réalisé pour que les chercheurs en économie et en management puissent l'utiliser pour valider leurs différentes théories.

Une deuxième perspective est l'étude de l'algorithme d'apprentissage par renforcement utilisé. Dans notre modèle, chaque firme a une perception des autres firmes et des formes organisationnelles. La taille du contexte (ou environnement) de chaque agent est en effet plus importante que celles considérées dans la littérature et dans les expérimentations des chercheurs en apprentissage par renforcement (voir les travaux de Gérard et Sigaud [45]). Par ailleurs, l'environnement de la firme est dynamique car les firmes évoluent continuellement. La population des firmes est également dynamique. La convergence de l'algorithme d'apprentissage (stabilisation de la base de règles pour les systèmes de classeurs) est en effet plus difficile que dans la majorité des exemples déjà considérés. Par exemple, pour faciliter la convergence, il serait utile de proposer des méta-règles pour le choix d'une stratégie (exploration ou exploitation). Ce choix est actuellement fait de façon aléatoire.

Chapitre 5

Vers des systèmes multi-agents auto-adaptatifs

5.1 Introduction

Les premiers travaux en intelligence artificielle distribuée se sont fondés sur la métaphore humaine. Les organisations étaient vues comme des organisations humaines. Le principal problème était la coordination des différents agents qui sont souvent conçus comme des systèmes intelligents. Par la suite, beaucoup de travaux se sont attachés à définir des structures organisationnelles issues de plusieurs domaines : la biologie, la physique, la chimie, les mathématiques, l'informatique, etc. Un panorama de ces travaux est présenté par Gasser [43]. Les structures organisationnelles modélisées peuvent être statiques, donc conçues *a priori* par le programmeur, ou dynamiques. Elles dépendent de l'environnement dans lequel elles évoluent, des ressources disponibles, et donc du problème à résoudre ou du système à simuler.

Le comportement global du système émerge d'un ensemble d'interactions locales entre agents ou entre les agents et un ensemble de structures organisationnelles. L'émergence repose ainsi sur le comportement dynamique du système et sur son évolution. Cette évolution peut concerner (voir figure 5.1) :

- le niveau micro : la structure interne ou les connaissances internes de l'agent pendant son exécution,
- ou le niveau macro : les structures organisationnelles ou sociales, telles que le réseau d'interdépendances et d'interférences, et le réseau d'accointances (réseau relationnel) définis par Castelfranchi [19].

L'émergence nécessite l'étude du rapport dynamique entre le micro et le macro, c'est-à-dire entre les agents et leur organisation (voir par exemple les travaux de Marcenac [82]). Le système doit posséder des mécanismes qui tendent à faire apparaître des éléments de cognition d'un niveau supérieur à celui des agents. Ces éléments caractérisent le phénomène global émergent. Müller [89] souligne que ce phénomène est observé soit par un observateur externe qui est l'utilisateur de la simulation (sens faible), soit par le système lui-même (sens fort) en des termes distincts de la dynamique sous-jacente.

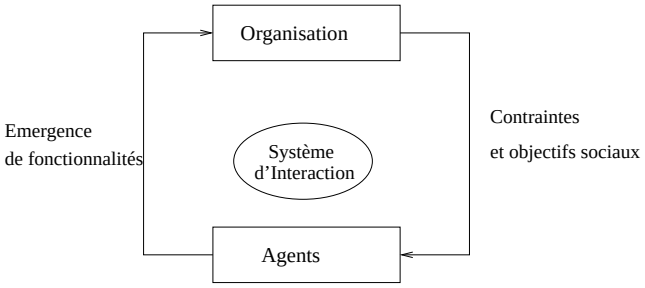

FIG. 5.1 – Les relations micro-macro par Ferber [36]

Ces deux types d'observations correspondent aux deux types d'émergence : émergence comme explication et émergence comme processus.

Dans les systèmes complexes [23], les phénomènes émergents ne peuvent pas être détectés par un observateur externe. Le système doit en effet être doté d'un mécanisme pour détecter automatiquement ces phénomènes. Ce chapitre a pour but de montrer l'apport des systèmes multi-agents adaptatifs pour rendre intelligibles et compréhensibles, ou contrôler des systèmes complexes. Dans la deuxième section, nous présentons les principaux modèles multi-agents adaptatifs existants. Dans la troisième section, nous introduisons un nouveau modèle multi-agents auto-adaptatifs. La dernière section résume nos travaux en cours et futurs sur les systèmes auto-adaptatifs.

5.2 Adaptation dans les systèmes multi-agents

Contrairement aux systèmes classiques, qui sont définis pour résoudre un problème défini avant l'exécution, les systèmes complexes sont en interaction continuelle avec leur environnement. Les actions des agents modifient leur environnement qui perturbe en retour leurs comportements. Pour faire face à ces perturbations, le mécanisme de rétroaction introduit en cybernétique (voir les travaux de Von Foerster [41]) est souvent utilisé pour définir des mécanismes adaptatifs. Cette solution est également adoptée par la majorité des systèmes multi-agents. Par exemple, dans Manta [27], chaque agent a un ensemble figé de tâches. Chacune de ces tâches est qualifiée par une note. La sélection d'une tâche est fonction de l'environnement courant et des notes. Dans ce cas, l'apprentissage se fait au niveau de la note attribuée à chaque tâche. Cette note évolue en fonction des résultats obtenus par la tâche lors de son exécution.

Dans la théorie AAMAS [14], les agents cherchent l'adéquation fonctionnelle en modifiant les liens existants entre eux. L'organisation est donc implicitement représentée par des liens entre agents.

Ishida et Gasser [67] [66] démontrent l'intérêt des organisations adaptatives en étudiant les systèmes à base de règles de production. L'idée principale de ce travail est le contrôle d'exécution d'un ensemble de bases de règles qui ont une mémoire de travail (base de faits) commune et qui doivent réagir à des événements asynchrones. Chaque

base de règles est dotée d'un mécanisme de contrôle de méta-niveau pour contrôler l'activation des règles de production en fonction de ses dépendances avec les règles de production des autres bases de règles.

Une autre catégorie de travaux consiste à étudier l'adaptation au niveau organisationnel. Par exemple, Carley utilise le modèle ORGAHEAD [17] pour étudier l'apprentissage multi-agents à différents niveaux. Les populations d'agents sont divisées en plusieurs catégories, avec beaucoup d'agents évoluant de manière très simple et quelques agents capables de modifier l'organisation des relations entre agents. Ainsi, la tâche organisationnelle est gérée par un ensemble d'agents. Cette approche a l'avantage de réduire les communications qui sont souvent prohibitives dans les systèmes multi-agents large échelle.

Cependant, le cas d'organisations dynamiques et adaptatives a rarement été abordé. Dans la majorité des systèmes multi-agents existants, l'adaptation des structures organisationnelles est fondée sur l'adaptation des comportements des agents et il n'est pas bien montré comment le comportement complexe du système émerge à partir de l'adaptation des simples règles locales des agents. Par ailleurs, l'adaptation au niveau local n'entraîne pas toujours une adaptation au niveau global. Par exemple, dans la gestion des réseaux [9], l'utilisation des agents adaptatifs (à chaque nœud est associé un agent) qui essaient de maximiser leurs performances locales peut entraîner un problème de congestion. Ce problème a notamment été souligné par Carley [17] et Odell [91]. Ce dernier cite l'exemple du *Crash* boursier qui a été causé par un ensemble d'agents logiciels adaptatifs dont le but de chacun est de maximiser son profit.

Les systèmes multi-agents auto-adaptatifs permettent d'éviter ce problème en introduisant une représentation de l'organisation et une perception de cette représentation par les agents.

5.3 Systèmes multi-agents auto-adaptatifs

Plusieurs domaines d'application émergents sont très complexes et dynamiques. Par exemple, le *grid computing* [65], les services Web [20] et l'intelligence ambiante [107] sont les exemples d'applications émergentes les plus connus. Ces applications sont basées sur un ensemble d'entités coopératives et distribuées qui gèrent un ensemble de ressources hétérogènes et fournissent des services aux utilisateurs. La gestion de ces ensembles ouverts de ressources est un problème complexe, des ressources peuvent être dynamiquement rajoutées et les ressources existantes peuvent être changées ou disparaître.

Les concepts de base des systèmes multi-agents (émergence, auto-organisation, adaptation) sont très utiles pour comprendre, expliquer et contrôler ces systèmes complexes. L'objectif principal des nombreux travaux sur les systèmes multi-agents est de proposer des modèles organisationnels pour modéliser des systèmes hétérogènes, complexes, dynamiques, non linéaires et évolutifs. Ces modèles montrent une intelligence et des capacités qui sont supérieures à celles des agents qui les composent. Elles émergent de la coexistence et de la coopération des agents plus ou moins autonomes.

Néanmoins les modèles organisationnels et les architectures multi-agents existants ne

satisfont que de manière partielle les besoins des applications émergentes. Ces dernières sont souvent caractérisées par :

- un très grand nombre d'agents,
- des comportements d'agents adaptatifs,
- des structures organisationnelles dynamiques.

La complexité croissante de ces applications impose ainsi l'élaboration des nouvelles architectures de systèmes ouverts et adaptatifs. Les systèmes à large échelle ne peuvent pas être contrôlés par un observateur externe ; les phénomènes émergents doivent être détectés par le système lui-même. Le système doit en effet avoir la capacité de s'auto-observer.

L'auto-observation est un mécanisme très important dont les avantages ont été bien définis par Pitrat qui souligne " an intelligent system must have the ability to observe its own behavior " [98]. Ce mécanisme consiste à faire observer à un système son propre comportement en le dotant d'un méta niveau. Les informations récoltées à ce méta niveau sont utilisées pour améliorer ou expliquer son comportement. Plusieurs systèmes sont conçus selon ce principe, par exemple pour détecter les boucles ou améliorer leurs performances. Nous avons utilisé ce principe dans les deux projets décrits dans les deuxième et troisième chapitres précédents pour détecter la variation de la criticité d'agents et adapter la stratégie de réplication, et pour détecter de nouvelles formes organisationnelles.

La section suivante décrit l'architecture multi-agents que nous avons introduite pour doter les systèmes multi-agents d'un mécanisme d'auto-observation.

5.3.1 Architecture d'un système multi-agents auto-adaptatif

Les systèmes complexes sont conduits par des tendances fondamentales telles que : une bonne fiabilité, une meilleure qualité de service, un équilibrage de charge ou de meilleures performances. L'adaptativité est en effet une propriété importante pour représenter la dynamique et faire face à la complexité de ces systèmes. Dans [15], nous avons adopté une approche nouvelle quant à l'adaptativité. Nous considérons que celle-ci n'est pas une propriété comme les autres mais est le caractère fondamental du système le conduisant à modifier toutes ses autres propriétés en les situant dans un cadre adaptatif. L'adaptativité n'est donc ni une fonction ni un rôle représenté par une structure ou un mécanisme à part ajouté aux agents, mais le fondement de l'architecture du système.

Nous nous sommes ainsi basés sur l'architecture multi-agents proposée pour fiabiliser les systèmes multi-agents (voir chapitre 3) et celle proposée pour modéliser l'évolution d'un système économique (voir chapitre 4) pour élaborer une architecture d'un système multi-agents auto-adaptatif. Les sections suivantes présentent les composants et l'activité d'un tel système.

Composants d'un système multi-agents auto-adaptatif

Un système multi-agents auto-adaptatif est un système ouvert qui modifie continuellement sa structure en fonctionnant. Il a une représentation de l'état de son envi-

ronnement et une représentation de sa propre organisation. Il adapte continuellement son organisation pour réagir aux variations de son environnement. L'organisation est par conséquent adaptative.

Nous considérons l'organisation comme une composante importante d'un système multi-agent. Nous adoptons la définition de Les Gasser : " ... an organization is a particular set of settled and unsettled questions about beliefs and actions through which agents view other agents. So, the organization relies on : mutual commitments, global commitments, and mutual beliefs ... " [42].

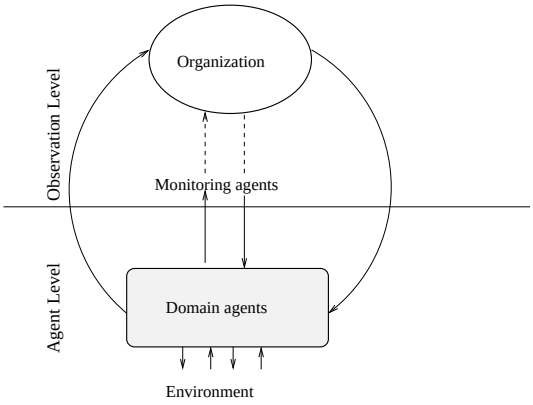

FIG. 5.2 – Différents composants d'un système auto-adaptatif

Nous proposons ainsi de réifier la partie adaptative de l'organisation [54] [15]. Pour cela, le système est architecturé autour (voir figure 5.2) :

- d'une organisation d'agents (représentant le niveau micro) qui ont la capacité de bien traiter une certaine classe de problèmes et qui agissent continuellement en fonction de leurs buts et en fonction de l'évolution de leur environnement. Cette organisation regroupe l'ensemble des agents du domaine dans un réseau classique d'accointances. Ce réseau définit la communication explicite avec des messages spécialisés explicites entre agents (par exemple, des messages de KQML ou de ACL), et les connaissances de domaine.
- d'une composante adaptative (représentant le niveau macro) qui représente l'organisation. Elle peut être définie par un ensemble de nœuds reliés et qui forment le réseau d'interdépendances. Elle réifie l'ensemble des tendances du système. Comme cela a été souligné par C. Castelfranchi, " this part represents the glue of groups, it links the agent with joint goal and the common solution, it links members with each other " [19]. Chaque nœud de ce réseau correspond à un agent du domaine et les connexions entre ces nœuds représentent les interdépendances entre ces agents. Ce réseau peut être donc représenté par une matrice W où $W_{i,j}$ décrit la dépendance entre les agents du domaine $Agent_i$ et $Agent_j$.
- d'une composante de couplage représentée par une organisation d'agents moniteurs qui observent continuellement la première composante (niveau micro), éta-

blissent un état de cette composante et fournissent cette information à la composante adaptative (niveau macro). Ils peuvent également contrôler l'organisation d'agents du domaine. Le but de ces agents consiste ainsi à réaliser le couplage entre les deux niveaux : micro et macro.

Les structures organisationnelles ne sont ainsi pas invariantes et indépendantes de la dynamique du système.

Activité d'un système multi-agents auto-adaptatif

Un système multi-agents auto-adaptatif a trois types d'activités :
- l'activité de l'organisation d'agents du domaine,
- le monitoring qui est l'activité des agents moniteurs,
- l'adaptation de la représentation de l'organisation.

Algorithme 3 Activité d'un système multi-agents auto-adaptatif

Require: AD : ensemble des agents du domaine, AM : ensemble des agents moniteurs et CO : représentation de l'organisation.

Ensure:

1: **while** Les agents AD sont actifs **do**
2: Activité de l'ensemble des agents du domaine,
3: Activité des agents moniteurs,
4: Mise à jour de CO
5: **end while**

Les trois processus décrits dans la boucle *tant que* de l'algorithme 3 sont activés en parallèle :
- Les agents du domaine opèrent de façon autonome selon les règles de leur proactivité et l'évolution de leur environnement. La fin de l'activité de ces agents entraîne la fin de l'activité du système. Ils ont souvent la capacité de se modifier pour adapter leur structure à l'évolution de leur environnement.
- Les agents moniteurs observent et analysent les informations sur le comportement des agents du domaine pour en extraire des indications pertinentes et les transmettre à l'organisation. L'activité de ces agents moniteurs est déclenchée par les modifications au niveau des agents du domaine. Par ailleurs, toute modification de la représentation de l'organisation est également perçue par les agents moniteurs. Cette perception est ensuite transmise aux agents du domaine qui l'utilisent pour mettre à jour leurs croyances.
- L'adaptation de la représentation de l'organisation vise à offrir aux agents du domaine une représentation de leur contexte organisationnel. Un agent adaptatif est ainsi un agent qui prend en compte l'évolution de ce contexte organisationnel.

5.3.2 Discussion

L'architecture des systèmes auto-adaptatifs que nous avons présentée fait apparaître ceux-ci comme des systèmes ouverts et organisationnellement complexes. Ces systèmes fonctionnent en se stabilisant sur des états de stabilité éphémère, en adaptant systématiquement leur comportement à la fois à l'évolution de l'environnement et à leur propre organisation.

Cette architecture est actuellement expérimentée par des réalisations effectives à partir de la plate-forme DIMA. Les systèmes multi-agents tolérants aux pannes et la simulation des modèles économiques sont deux exemples de réalisation. Le but de ces réalisations est de montrer la robustesse et la fiabilité des systèmes multi-agents auto-adaptatifs par rapport à des systèmes plus classiques.

Cette nouvelle approche trouve des applications dans différents domaines notamment les applications émergentes telles le *grid computing* [65], les services web [20] et l'intelligence ambiante [107]. Elle ouvre sur une problématique de recherche importante : comment créer automatiquement, à partir d'un système qui est initialement composé d'agents adaptatifs, un système auto-adaptatif, qui génère ses propres tendances et les réifie dans une structure organisationnelle ? Autrement dit, quelles sont les connaissances des agents moniteurs et quelle est la représentation de l'organisation ?

5.4 Travaux en cours et futurs

Pour mieux étudier la problématique des systèmes auto-adaptatifs, plusieurs travaux sont en cours dans le cadre des projets de thèse. Ces travaux sont brièvement présentés dans les sections suivantes.

5.4.1 Emergence de protocoles d'interaction par observation

Le paradigme agent propose une bonne solution pour la modélisation des systèmes complexes. La résolution de problème est fondée sur les interactions entre agents. La spécification des interactions se base souvent sur les notions de rôle, les relations entre rôles et les groupes d'agents. Ces approches mènent souvent à une spécification statique du réseau d'interaction et des protocoles d'interaction.

L'objectif de ce projet est l'apprentissage de nouveaux protocoles d'interaction en se basant sur l'observation des échanges inter-agents. La première étape de cette approche consiste à observer et ensuite structurer les communications inter-agents. La deuxième étape consiste à extraire de la structure ainsi obtenue les séquences de messages représentant les protocoles d'interaction déjà définis et les séquences qui sont intéressantes et qui ne sont pas définies, puis utiliser la trace de la résolution de problème pour expliquer le comportement du système. La troisième étape consiste à généraliser ces nouvelles séquences pour construire des nouveaux protocoles qui peuvent ensuite être transmis aux agents pour améliorer leurs interactions futures.

Dans le cadre de ce projet, le langage de communication inter-agents est ACL (proposé et spécifié par FIPA). Par ailleurs, les agents sont adaptatifs, leurs règles d'interaction dépendent du contexte.

5.4.2 Un modèle économique pour la gestion de ressources

Ce projet s'inscrit dans le projets systèmes multi-agents tolérants aux pannes et s'appuie sur l'architecture multi-agents proposée dans le chapitre 3.

Le problème de détermination du nombre de réplicats des agents en fonction de leur criticité et de la répartition de ces réplicats est un problème de gestion de ressources. L'ensemble de ces ressources est dynamique car de nouvelles machines peuvent être dynamiquement rajoutées et les machines existantes peuvent tomber en panne d'une part. Par ailleurs, les systèmes multi-agents sont souvent ouverts.

Jamali et al. [68] soulignent l'intérêt de tenir compte de certaines notions de l'économie pour la gestion des ressources. L'utilisation de ces notions nous permet une gestion intelligente des ressources et améliore la fiabilité et l'efficacité du système. Il est ainsi préférable d'avoir un seul réplicat sur une machine très fiable que quatre réplicats sur des machines qui sont souvent en panne. Par ailleurs, il est judicieux de répartir les réplicats de façon à optimiser le temps de réponse.

Dans notre modèle, les ressources de chaque machine sont gérées par l'agent *Observateur*. Les ressources considérées représentent le nombre de réplicats que peut héberger cette machine. Nous associons également un coût CM_i à chaque ressource de la machine M_i. Nombre et coût sont initialisés par le concepteur. Cependant, le coût est ensuite automatiquement mis à jour par l'agent *Observateur* en fonction des statistiques fournies par le détecteur de pannes et le module d'observation. Il peut être calculé comme suit :

$$CM_i(t+1) = CM_i(t) * (1 - pp_i(t+1) + pp_i(t)) \tag{5.1}$$

où pp_i est la probabilité de panne de la machine M_i.

Par ailleurs la criticité de l'agent du domaine est utilisée pour déterminer le nouveau budget $B_i(t)$ que l'agent Moniteur peut utiliser pour allouer ou désallouer des ressources (acheter ou céder des ressources).

L'allocation de ressources se fait ensuite par un processus de négociation entre les agents Moniteurs et les agents Observateurs.

5.4.3 Méta-DIMA : vers un environnement de développement

Dans les travaux de recherches actuels sur le multi-agents, on trouve des contributions sur les architectures d'agents, sur les outils de développement liés à l'implémentation des agents, sur l'organisation des systèmes d'agents et sur les méthodologies d'analyse et de conception orientées-agent. Les architectures d'agents traitent des composants fonctionnels des agents pris individuellement et des interactions entre ces composants et sont basées sur des travaux tels que ceux de présentés par Castelfranchi [19], Decker,

Durfee et Lesser[24], Ferber [36], Gasser [42], Avouris et Gasser [3]. Les travaux sur l'organisation multi-agents sont plus récents et apportent des concepts tels que les groupes, les rôles, etc. Des méthodologies (Aalaadin [37], Cassiopée [21], Gaia [80], TROPOS [10], INGENIAS juan, ADELFE [6], ADEPT [69], MASSIVE [78], MITAIS [113], Nemo [64], PASSI ([22], Voyelles [25] [103]) permettant d'analyser des problèmes complexes et de concevoir des systèmes multi-agents sont apparues, mais elles ne prennent pas en charge le cycle de développement complet (analyse, conception, codage, déploiement, ...). Qui plus est, elles ne prennent pas en compte l'existant en terme d'architectures d'agents et d'outils de développement orientés-agent.

L'objectif de la démarche élaborée dans ce projet est de combler le fossé existant entre les architectures d'agents et les méthodologies et outils de développement orientés-agent. Nous nous basons sur des architectures et outils existants afin d'élaborer des méta-modèles supportant l'expression des connaissances réellement pertinentes impliquées dans le cycle de développement. Notre approche est donc résolument ascendante. Cette nouvelle approche s'inspire des propositions de l'OMG sur le Model-Driven Architecture (MDA) qui prône la séparation de la logique applicative des considérations techniques liées aux plates-formes d'implémentation, dans le but d'améliorer la réutilisabilité et de faciliter le développement [93]. Le fait que les connaissances liées aux besoins devraient être permanentes, alors que les aspects techniques sont souvent limités à une technologie donnée (dont le succès peut être éphémère) constitue une des idées clés du MDA [112].

Le MDA définit plusieurs niveaux de modèles [93], principalement les *platform-independant models* (PIM), et les *platfom-specific models* (PSM). Les PIM expriment la logique applicative du système, les PSM représentent le code à générer et on cherche à passer des PIM aux PSM par des transformations qu'il s'agit de définir et d'automatiser. Ainsi, au lieu d'être perturbé par les aspects techniques, le concepteur peut se concentrer sur les besoins fonctionnels de l'application.

La définition et l'automatisation des transformations en question fait intervenir un niveau supérieur de modélisation, celui des méta-modèles. Dans une large mesure, la mise en œuvre du MDA repose sur l'élaboration de méta-modèles adéquats, préliminaire nécessaire à l'écriture des règles de transformations entre PIM et PSM.

5.4.4 Goliath : une méthode à base d'agents pour la construction d'interfaces graphiques

Il s'agit de contribuer à résoudre le problème de l'assemblage de composants logiciels à l'aide d'outils à base de connaissances. L'hypothèse centrale est que les techniques à base d'agents logiciels répondent bien à ce problème.

Une difficulté majeure des outils à base de connaissances consiste à contrôler l'explosion combinatoire induite par les connaissances élémentaires. Comme il s'agit d'aider l'humain et non de le remplacer, ce dernier joue un rôle essentiel dans ce contrôle mais le défi consiste à éviter que sa tâche devienne fastidieuse. L'écriture de tactiques à base de scripts utilisée par exemple dans les systèmes interactifs de développement transfor-

mationnel ou de preuve semble insuffisante. L'expression de ce contrôle dans un système multi-agents offre des perspectives nouvelles qui pourraient permettre une avancée significative. Par ailleurs, la notion d'agent semble bien adaptée h à la médiation entre un système et un utilisateur. Enfin, on peut voir l'intégration d'un composant dans une architecture logicielle comme une adaptation car les composants peuvent rarement être utilisés par simple juxtaposition. Une voie à explorer consistera à ce que des agents prennent en charge les composants pour les accompagner, en quelque sorte les encapsuler, pendant tout ou partie du cycle de vie. En effet, typiquement l'adaptation d'un composant revient à le placer dans un réceptacle qui communique avec le reste de l'architecture logicielle ce qui peut être fait de façon très souple par un agent logiciel au moins dans une phase de prototypage. Lors du déploiement de l'application les agents d'adaptation seront si besoin dégradés (en quelque sorte spécialisés) dans des formes moins souples mais plus efficaces.

5.4.5 Discussion

Les travaux présentés dans cette section s'inscrivent dans le thème "Agent-Oriented Software Engineering" dont nous décrivons les principales perspectives dans [56] et le thème "systèmes multi-agents adaptatifs". Ils visent :

- *à court/moyen termes* : l'élaboration de frameworks pour l'implémentation et le déploiement des systèmes multi-agents adaptatifs,
- *à long terme* : l'élaboration d'une méthodologie pour l'analyse, la conception, l'implémentation et le déploiement des systèmes multi-agents adaptatifs.

Pour mieux comprendre le monitoring et les problèmes de conception des systèmes complexes, nous avons également des projets sur la conception et le monitoring des systèmes à base de composants : le projet COgents (voir section 6.1.3) et le projet Goliath (voir section 5.4.4).

Bibliographie

[1] G. Agha and C. Hewitt. Concurrent programming using actors : Exploiting large scale parallelism. In S. N. Maheshwari, editor, *5th Conference On Foundations of Software Technology & Theoretical Computer Science*, number 206 in LNCS, pages 19–41. Springer Verlag, 1985.

[2] N. A. Avouris and L. Gasser, editors. *Distributed Artificial Intelligence : Theory and Praxis*, chapter Using Reactive Multi-Agent Systems in Simulation and Problem Solving. Kluwer Academic, London, 1992.

[3] N. A. Avouris and L. Gasser, editors. *Distributed Artificial Intelligence : Theory and Praxis*. Kluwer Academic Publisher, 1992.

[4] A. Joel A.C. Baum and Hayagreeva Rao. *Handbook of Organizational Change and Development*, chapter Evolutionary Dynamics of Organizational Populations and Communities. Oxford University Press, 1999.

[5] F. Bellifemine, A. Poggi, and G. Rimassa. Jade - a fipa-compliant agent framework. In *PAAM*, pages 97–108, London, 1999.

[6] C. Bernon, M.-P. Gleizes, G. Picard, and P. Glize :. The adelfe methodology for an intranet system design. In *Agent-Oriented Information Systems*, Toronto, 2002.

[7] K. Boudaoud. *Intrusion Detection : a New Approach using a Multi-Agent System.* PhD thesis, Institut Eurecom and EPFL, Sophia Antipolis, 2001.

[8] K. Boudaoud, H. Labiod, Z. Guessoum, and R. Boutaba. Network security management with intelligent agents. In *International Network Operation and Management Symposium (NOMS'00).* IFIP/IEEE, 2000.

[9] N. Boukhatem. *L'approche multi-agents pour un contrôle de congestion adaptatif de réseaux ATM.* PhD thesis, Université de Versailles, 1997.

[10] P. Bresciani and F. Sannicol. Requirement analysis in tropos : a self referencing example. In H. Tianfield J. Mller and R. Unland, editors, *Agent Technologies, Infrastructures, Tools, and Applications for e-Services*, volume 2592. Springer-Verlag, 2003.

[11] J.-P. Briot. Actalk : a testbed for classifying and designing actor languages in the smalltalk-80 environment. In *Proceedings of ECOOP'89*, volume LNAI 1069, pages 109–129, Nottingham, 1989.

[12] J.-P. Briot and Y. Demazeau, editors. *Principes et architecture des systèmes multi-agents*. Collection IC2. Hermes Science Publications, 2001.

[13] E. Bruederer and J. Singh. Organizational evolution, learning, and selection : A genetic algorithm-based model. *Academy of management journal*, 39(5) :1322–1349, 1996.

[14] D. Capera, J.-. Georgé, M. P. Gleizes, and P. Glize. The amas theory for complex problem solving based on self-organizing cooperative agents. In *WETICE*, pages 383–388, 2003.

[15] A. Cardon and Z. Guessoum. Systèmes multi-agents adaptatifs. In *JFIAD-SMA'2000*, 2000.

[16] A. Cardon and F. Lesage. Toward adaptive information systems : Considering concern and intentionality. In *KAW'98*, pages 108–124, April 1998.

[17] K. M. Carley. Adaptive organizations and emergent forms. *Organization Science*, 769(2-3), 1998.

[18] C. Castelfranchi. *Decentralized AI*, chapter Dependence relations in multi-agent systems. Elsevier, 1992.

[19] C. Castelfranchi. Modelling social action for AI agents. *Artificial Intelligence*, 103 :157–182, 1998.

[20] Lawrence Cavedo and Zakaria Maamar, editors. *Workshop on Web Services and Agent-Based Engineering*, Melbourne, July 2003. *ACM*.

[21] A. Collinot, A. Drogoul, and P. Benhamou. Agent oriented design of a soccer robot team. In *KAW'98*, pages 41–47, April 1998.

[22] M. Cossentino and C. Potts. A case tool supported methodology for the design of multi-agent systems. In *The 2002 International Conference on Software Engineering Research and Practice*, Las Vegas (NV), USA, June 24-27 2002. SERP'02.

[23] Agnès Guillot Emmanuel Daucé, editor. *Modèles Multi-Agents et Systèmes Dynamiques. In Systèmes Dynamiques.* Hermès, Décembre 2002.

[24] K. S. Decker, E. H. Durfee, and V. R. Lesser. *Distributed Artificial Intelligence*, chapter Evaluating research in cooperative distributed problem solving, pages 485– 517. Morgan Kaufmann, 1989.

[25] Y. Demazeau. Multi-agents systems methodology. In *14th Brazilian Symposium on Artificial Intelligence, SBIA'98*, Porto Alegre, 1998.

[26] M. Dojat, F. Pachet, Z. Guessoum, D. Touchard, A. Harf, and L. Brochard. Néo-Ganesh : a working system for the automated control of assisted ventilation in icus. *Artificial Intelligence in Medicine*, 11 :97–117, September/October 1997.

[27] A. Drogoul. *De la simulation multi-agents à la résolution collective de problèmes. Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents.* PhD thesis, LAFORIA, Unievsrité de Paris 6, 1993.

[28] R. Durand. *Management Stratégique des ressources et analyse de la performance.* PhD thesis, HEC, 1997.

[29] R. Durand. *Entreprise et évolution économique*. Belin, 2000.

[30] R. Durand, editor. *Développement de l'Organisation, nouveaux regards*. ECONOMICA, 2002.

[31] R. Durand and Z. Guessoum. Competence systemics and survival, simulation and empirical analysis. In *Competence 2000 Conference*, pages 10–24, 2000.

[32] P. Estraillier and C. Girault. Applying petri net theory to the modelling, analysis and prototyping of distributed systems. In *International Workshop on Emerging Technologies for Factory Automation : State of the Art and Future Directions*. IEEE/SICE, August 1992.

[33] P. Estraillier and F. Kordon. Structuration of large scale petri nets : an association with higher level formalisms for the design of multi-agent systems. In *International Conference on Systems, Man and Cybernetics Information, Intelligence ans Systems*, Beijing, China, October 1996.

[34] A. El Fallah-Seghrouchni, S. Haddad, and H. Mazouzi. Protocol engineering for multiagent interactions. In *MAAMAW'99*, number 1647 in LNAI, pages 128–135. Springer Verlag, 1999.

[35] A. Fedoruk and R. Deters. Improving fault-tolerance by replicating agents. In *AAMAS2002*, Boulogna, Italy, 2002.

[36] J. Ferber. *Les systèmes multi-agents, vers une intelligence collective*. InterEditions, Paris, 1995.

[37] J. Ferber and O. Gutknecht. Alaadin : a meta-model for the analysis and design of organizations in mutli-agent systems. In Y. Demazeau, editor, *ICMAS'98*, pages 128–135, Paris, 1998.

[38] I. A. Ferguson. *TouringMachines : An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, University of Cambridge, Clare Hall, 1992.

[39] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an agent communication language. In *Third international conference on information and knowledge management*. ACM Press, November 1994.

[40] FIPA. Specification. part 2, agent communication language, foundation for intelligent physical agents, geneva, switzerland. http ://www.cselt.stet.it/ufv/leonardo/fipa/index.htm, 1997.

[41] H. Von Foerster. *Understanding Understanding : Essays on Cybernetics and Cognition*. Springer-Verlag, 2003.

[42] L. Gasser. Conceptual modeling in distributed artificial intelligence. *Journal of the Japanese Society of Artificial Intelligence*, 5(4), July 1990.

[43] L. Gasser. Perspectives on organizations in multi-agent systems. In *9th ECCAI Advanced Course ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS*, number 2086 in LNAI, pages 1–32. Springer Verlag, 2001.

[44] N. Giambiasi and C. Oussalah. Langages à objets. *Génie Logiciel et Systèmes Experts*, 1(22) :52–68, Mars 1991.

[45] P. Gérard, W. Stolzmann, and O. Sigaud. YACS : a new learning classifier system using anticipation. *Journal of Soft Computing : Special Issue on Learning Classifier Systems*, 3-4(6) :216–228, 2002.

[46] R. Guerraoui, B. Garbinato, and K. Mazouni. Lessons from designing and implementing garf. In *Proceedings Objects Oriented Parallel and Distributed Computatio*, volume LNCS 791, pages 238–256, Nottingham, 1989.

[47] R. Guerraoui and A. Schiper. Software-based replication for fault tolerance. *IEEE Computer*, 30(4) :68–74, April 1997.

[48] Z. Guessoum. *Un environnement opérationnel de conception et de réalisation de systèmes multi-agents*. PhD thesis, Laforia, Paris 6, 1996.

[49] Z. Guessoum. A hybrid agent model : a reactive and cognitive behavior. In *ISAD97*, pages 25–32, Berlin, Germany, 1997. IEEE.

[50] Z. Guessoum. Dima : Une plate-forme multi-agents en smalltalk. *Objet*, 3(4) :393–410, Décembre 1998.

[51] Z. Guessoum and J.-P. Briot. From active objects to autonomous agents. *IEEE Concurrency*, 7(3) :68–76, 1999.

[52] Z. Guessoum, J.-P. Briot, S. Charpentier, O. Marin, and P. Sens. A fault-tolerant multiagent framework. In ACM, editor, *First International Workshop on Challenges in Open Agent Systems, AAMASŠ02*, pages 672–673, Bologna, July 2002.

[53] Z. Guessoum, J.-P. Briot, O. Marin, A. Hamel, and P. Sens. *Software Engineering for Large-Scale Multi-Agent Systems*, chapter Dynamic and Adaptative Replication for Large-Scale Reliable Multi-Agent Systems, pages 182–198. Number 2603 in LNCS. Springer Verlag, April 2003.

[54] Z. Guessoum, A. Cardon, and A. Ramdani. Self adjustable autonomy in multi-agent systems. In *AAAI 1999 Spring Symposium Series, Agents with adjustable autonomy*, march 1999.

[55] Z. Guessoum, A. Cardon, and A. Ramdani. Toward self-adaptive multi-agent systems. In *MAAMAW'99*, Valencia, 1999.

[56] Z. Guessoum, M. Cossentino, and J. Pavon Mestras. *In Methodologies and Softawre Engineering for Agents Systems*, chapter A Roadmap of Agent-Oriented Software Engineering : The European Agentlink Perspective. Kluwer, January 2004.

[57] Z. Guessoum, O. Nadjemi, B. Braunschweig, P. Roux, A. Yang, E. S. Fraga, I. D. Stalker, D. Pinol, M. Serra, and D. Paen. Agent-based computer-aided process engineering. In *2nd. Workshop on Intelligent Computing in the Petroleum Industry ICPI 2003*, Acapulco, Guerrero, México, 2003. AAAI.

[58] Z. Guessoum and M. Occello. *Principes et architecture des systèmes multi-agents*, chapter Environnements de dévéloppement. Hermès, 2001.

[59] Z. Guessoum, L. Rejeb, and R. Durand. Emergence of organizational forms. In *AAMAS'03*, Aberystwyth, UK, April 2003. AISB.

[60] Z. Guessoum, L. Rejeb, and R. Durand. Multi-agent simulation of firms and organizational forms. In *AAMAS*, Melbourne, 2003. ACM.

[61] O. Gutknecht. *Proposition d'un modèle générique de systèmes multi-agents - Examen de ses conséquences formelles, implémentatoires et méthodologiques.* PhD thesis, Université Montpellier 2, September 2001.

[62] O. Gutknecht, J.Ferber, and F. Michel. Madkit : Une expérience d'architecture de plate-forme multi-agent générique. In *JFIADSMA'2000*. Hermès, 2000.

[63] S. Hagg. A sentinel approach to fault handling in multi-agent systems. In C. Zhang and D. Lukose, editors, *Multi-Agent Systems, Methodologies and Applications*, number 1286 in LNCS, pages 190–195. Springer Verlag, 1997.

[64] M.-P. Huget. Nemo : an agent-oriented software engineering methodology. In *Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies*, Seattle, USA, Nov 2002.

[65] INRIA, editor. *Ecole thèmatique GRID'2002*, Aussois, France, Décembre 2002.

[66] T. Ishida. *Real-Time Search for Learning Autonomous Agents.* Kluwer Academic Publishers, 1997.

[67] T. Ishida, L. Gasser, and M. Yokoo. Organization self-design of distributed production systems. *IEEE TKDE,*, 4(2) :123–134, 1992.

[68] N. Jamali, P. Thati, and G. Agha. An actor-based architecture for customizing and controlling agent ensembles. *IEEE Intelligent Systems, Special Issue on Agents*, 1999.

[69] N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, and B. Odgers. Autonomous agents for business process management. *Int. Journal of Applied Artificial Intelligence*, 14(2), 2000.

[70] B. Kemme and G. Alonso. Don't be lazy be consistent : Postgres-r, a new way to implement database replication. In *Conf on Very Large Databases (VLDB*, 2000.

[71] T. Kovacs. Evolving optimal populations with XCS classifier systems. Technical Report CSRP-96-17, October 1996.

[72] S. Kumar, P. R. Cohen, and H. J. Levesque. The adaptive agent architecture : Achieving fault-tolerance using persistent broker teams. In *The Fourth International Conference on Multi-Agent Systems ICMAS*, Boston, *USA*, 2000.

[73] P. Laszlo, M. T. Hannan, G. Peli, and G. R. Caroll. Form and identity : On te structure of organizational forms. In *14th Colloquim of Euro. group of Organizations*, 2002.

[74] C. Lemaitre, C. Excelente, and A. El Fallah-Seghrouchni. Multi-agent organization approach to electronic business automation. In *5th International Conference on Decision Science Institute*, Athènes, June 1999.

[75] B. Lesueur, Z. Guessoum, G. Sunyé, G. Blain, and J.-F. Perrot. La métaphore du dossier. In *INFORSID'99*, Tour, 1999.

[76] D. A. Levinthal. Adaptation on rugged landscapes. *Management Science*, 43(7) :934–950, 1997.

[77] A. Y. Lewin and H. W. Volberda. Prolegomena on coevolution : A framework for research on strategy and new organizational forms. *Organization Science*, 10(5) :519–534, 1999.

[78] J. Lind. *Iterative software engineering for Multi-Agent Systems, The MASSIVE Method*. Springer Verlag, 2001.

[79] A. Lomi and E. R. Larsen. Interacting locally and evolving globally : A computational approach to the dynamics of organizational populations. *Academy of Management Journal*, 39(4) :1287–1321, 1996.

[80] N. Jennings M. Wooldridge and D. Kinny. The methodology gaia for agent-oriented analysis and design. *AI*, 10(2) :1–27, 1999.

[81] J. Malenfant. ARM : un modèle réflexif asynchrone pour les objets répartis et réactifs. In *LMO'03*, volume 9 of *L'objet*, 2003.

[82] P. Marcenac. Modélisation de systèmes complexes par agents. *TSI*, 16(8) :1013–1038, Octobre 1997.

[83] H. Mazouzi, A. El Fallah-Seghrouchni, and S. Haddad :. Open protocol design for complex interactions in multi-agent systems. In *AAMAS*, Bologna, Italy, July 2002. ACM.

[84] J. McAffer. *A Meta-level Architecture for Prototyping Object Systems*. PhD thesis, University of Tokyo, Tokyo, 1995.

[85] M.Golm. Metaxa and the future of reflection. In *OOPSLA -Workshop on Reflective Programming in C++ and Java*, pages 238–256. Springer Verlag, 1998.

[86] H. Mizuta and Y. Yamagata. Agent-based simulation for economic and environmental studies. In *JSAI 2001 Workshops*, number 2243 in LNAI, pages 142–152. Springer Verlag, 2001.

[87] J. Mller and M. Pischel. Modelling reactive behaviour in vertically layered agent architectures. In A. G. Cohen, editor, *Eleventh European Conference on Artificial Intelligence (ECAI'94)*, pages 709–713, Amsterdam, (NL), 1994.

[88] J. Muller. Agent architectures. In *ATAL'98*, page 1 :8, 1998.

[89] J.-P. Muller. Modélisation organisationnelle en systèmes multi-agents. In *Septième école d'été de l'ARCo*, Bonas, 2000.

[90] M. Occello and Y Demazeau. Building real time agents using parallel blackboards and its use for mobile robotics. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, San Antonio, Texas, USA, october 1994.

[91] J. Odell. Agents and complex systems. *Journal of Object Technology*, 1(2) :35–45, 2002.

[92] J. J. Odell, H. V. Dyke Parunak, and B. Bauer. Representing agent interaction protocols in UML. In *Fourth International Conference on Autonomous Agents*, 2000.

[93] OMG TC Document ormsc/2001 07-01. Model driven architecture (mda). Technical report, OMG, 2001.

[94] H. V. D. ParunaK, S. Brueckner, and J. Sauter. Digital pheromone mechanisms for coordination of unmanned vehicles. In *AAMAS*, Bologna, Italy, 2002. ACM.

[95] J. Pavn and J. Gmez-Sanz. Agent oriented software engineering with ingenias. In *CEEMAS*, Springer Verlag, page 2691, 2003.

[96] E. Penrose. *The theory of the growth of the firm.* John Wiley, 1959.

[97] J.-F. Perrot. Langage à objets et programmation par objets. Technical report, LAFORIA, Université de Paris 6, 1992.

[98] J. Pitrat. An intelligent system must and can observe it own behaviour. In *Cognitiva 90*, 1990.

[99] J. Pitrat. *Méta-connaissances.* Hermès, Paris, 1994.

[100] S. J Poslad., S.J. Buckle, and R. Hadingham. The fipa-os agent platform : Open source for open standards. In *PAAM 2000*, Manchester, UK, 2000.

[101] R. Rao. From the board notion of reflection to the engineering practice of object-oriented metalevel architectures. In *OOPSLA-91 Workshop on Reflection and Metalevel Architectures*, pages 27–31, 1991.

[102] N. Revault, H. Sahraoui, G. Blain, and J-F Perrot. A metamodeling technique : The métagen system. In *TOOLS Europe'95*, pages 127–139, 1995.

[103] P.-M. Ricordel and Y. Demazeau. La plate-forme volcano, modularité et réutilisation pour les systèmes multi-agents. *TSI, Numéro spécial : Environnements de développement de systèmes multi-agents*, 21(4), 2002.

[104] E. Romanelli. The evolution of new organizational forms. *Annuel Rev.*, 1(17) :79–103, 1992.

[105] P. Sens. Contribution à l'intégration de la tolérance aux fautes dans les environnements répartis thèse d'habilitation. Technical report, Université de Paris 6, 2002.

[106] P. Sens and Z. Guessoum. Systèmes multi-agents tolérants aux pannes. In *Ecole thèmatique GRID'2002*, Aussois, France, 2002. INRIA.

[107] D. Servat and A. Drogoul. Combining amorphous computing and reactive agent-based systems : a paradigm for pervasive intelligence ? In ACM, editor, *AAMAS'02*, 2002.

[108] J. S. Sichman and R. Conte. Multi-agent dependence by dependence graphs. In *AAMAS2002*, pages 483–490, Boulogna, Italy, 2002. ACM.

[109] J. S. Sichman, R. Conte, and Y. Demazeau. Reasoning about others using dependence networks. In *Actes de Incontro del gruppo AI*IA di interesse speciale sul inteligenza artificiale distribuita*, Roma, Italia, 1993.

[110] J. S. Sichman, R. Conte, and Y. Demazeau. A social reasoning mechanism based on dependence networks. In *Proceedings of ECAI'94 - European Conference on Artificial Intelligence*, Amsterdam, The Netherlands, August 1994.

[111] F. De Assis Silva and R. Popescu-Zeletin. An approach for providing mobile agent fault tolerance. In S. N. Maheshwari, editor, *Second International Workshop on Mobile Agents*, number 1477 in LNCS, pages 14–25. Springer Verlag, 1998.

[112] R. Soley. Model driven architecture, omg white paper draft 3.2. Technical report, Object Management Grou, 2000.

[113] L. Sun and K. Liu. A method for interactive articulation of information requirements for strategic decision support. *Information and Software Technology*, 43, 2001.

[114] A. Thiefaine, Z. Guessoum, J.-F. Perrot, and G. Blain. Génération de systèmes multi-agents à partir de modèles. In Jean-Pierre Briot et Khaled Ghédira, editor, *JFSMA03*, Tunisie, 2003.

[115] R. van Renesse, K. Birman, and S. Maffeis. Horus : A flexible group communication syste. *CACM*, 39(4) :76–83, 1996.

[116] M. Wooldridge and N. Jennings. Intelligent agents : Theory and practice. *The Knowledge Engineering Review*, 10(2) :115–152, June 1995.

[117] M.-J. Yoo. *Une approche componentielle pour la modélisation d'agents coopératifs et leur validation.* PhD thesis, Université de Paris 6, 1999.

[118] L. A. Zadeh. A new direction in ai : Toward a computational theory of perceptions. *AI Magazine*, 22(1) :73–84, 2001.

Chapitre 6

Activités de recherche

6.1 Projets

Plusieurs projets auxquels j'ai participé ou dont je suis responsable mettent en œuvre mes travaux de recherche : Fibof, EMA (responsable), COgents (responsable) et CellPop. Cette section donne un aperçu de chacun de ces projets.

Les deux projets ALASACE (voir Chapitre 2) et MOVECO (voir chapitre 3) ne sont pas décrits ici.

6.1.1 Fibof : Financial Business Framework (1994-1996)

Le projet européen Esprit IV Fibof a été fait en collaboration avec El Corte Ingles (une banque sévillane) et Soleri Cigel (une entreprise française). Gilles Blain était le responsable LIP6.

L'objectif de Fibof était de définir un framework applicatif pour les secteurs bancaire et financier, adaptable à chaque institution financière. Ce framework doit permettre de spécifier simplement dans un langage compréhensible par l'utilisateur de nouveaux produits financiers. Il doit ensuite exploiter cette description pour générer automatiquement une application de gestion et de suivi de ces produits.

En collaboration avec Gilles Blain, Bruno Lesueur et Jean-François Perrot, nous avons proposé une démarche de développement de systèmes informatiques, appelée "la métaphore du dossier" [75]. Son objectif est de permettre une prise en compte équilibrée des points de vue des utilisateurs et des informaticiens. Cette démarche est outillée par MétaGen [102], qui allie la méta-modélisation et l'acquisition de connaissances pour offrir un environnement de développement de haut niveau. Dans ce cas précis, MétaGen a été utilisé pour engendrer des applications multi-agents distribuées, implémentées en Java ou Smalltalk. Le point de vue des utilisateurs est modélisé selon deux aspects : fonctionnel et organisationnel. Le premier est fondé sur la notion de dossier (vu comme un dépôt de données) et sur les événements et règles de gestion qui agissent sur celui-ci. L'aspect organisationnel permet de décrire les différents utilisateurs du système et de leur attribuer les responsabilités dégagées lors de l'analyse de la gestion des dossiers.

Le point de vue des informaticiens prend en compte l'ensemble des moyens techniques (frameworks, systèmes transactionnels, etc.) nécessaires à la mise en œuvre du système.

Notre démarche permet ainsi de préserver l'indépendance des "managers", des organisateurs et des implémenteurs, tout en assurant une mise en œuvre opérationnelle.

6.1.2 EMA : Etude du MArché électrique (1996-2000)

Le projet EMA s'inscrit dans la cadre d'une collaboration avec EDF. Il a eu pour objectif la proposition d'une modélisation multi-agents du marché concurrentiel électrique en se basant sur les textes de loi sur l'ouverture du marché électrique européen.

Une première phase a permis d'analyser ce marché et les textes de loi afin d'identifier les agents, leurs structures et leur connaissances. La modélisation retenue se décompose en trois parties : le pôle production, le pôle demande et le pôle régulation. Pour chaque pôle, nous avons identifié l'ensemble des acteurs qui le constituent. Par exemple, le pôle production est constitué des différents producteurs qui interviennent sur le marché.

Dans une deuxième phase, nous avons développé un simulateur multi-agents des différents acteurs tels que l'autorité de régulation, l'EDF, les producteurs indépendants et les consommateurs.

Ce simulateur multi-agents est utilisé par EDF pour comparer les résultats de la simulation multi-agents avec d'autres études statiques, comme l'étude basée sur l'ouverture du marché des télécommunications. EDF visait également à généraliser cette simulation à d'autres domaines d'application en proposant un modèle générique d'une population d'agents concurrentiels.

6.1.3 COgents : Agent-Based Architecture For Numerical Simulation (2002 -2004)

Le projet européen IST COgents est fait en collaboration avec l'Institut Français du Pétrole (coordinateur), le Department of Chemical Engineering (University College, London), RWTH Lehrstuhl für ProzessTechnik (The Aachen University of Technology) et Aspen Technology (Barcelona). Ce texte est repris du résumé présenté à la journée LMCS (Logiciels dédiés à la Modélisation et au Calcul Scientifique) organisée le 25/4 à l'IFP.

Dans le cadre de COGents, nous avons proposé une approche nouvelle de la simulation numérique à l'aide de la technologie multi-agents, des composants logiciels et de l'Internet. Nous démontrons l'utilisation des agents cognitifs pour faciluter la modélisation et pour l'interopérabilité dynamique et opportuniste de composants CAPE-OPEN (Computer-Aided Software Engineering) distribués sur l'Internet, facilitant ainsi la fourniture de services d'application en modélisation de procédés.

Pour permettre cette interopérabilité dynamique et opportuniste, nous avons défini des représentations des connaissances de modélisation et de composition de services. Ces représentations sont sous la forme d'une ontologie de la modélisation de procédés. Elles fournissent un complément sémantique au standard CAPE-OPEN, qui ne définit que la syntaxe des entrées-sorties des composants. Cette ontologie est à la base d'une

organisation d'agents qui résolvent le problème d'appariement entre les besoins des utilisateurs et les services disponibles.

Nous allons ensuite développer des agents cognitifs personnalisés (voir [57]), en encapsulant des composants par des agents, leur déléguant ainsi la représentation des composants dans le processus d'assemblage. Ces agents cognitifs contiennent des connaissances explicites de simulation, leur permettant ainsi de négocier pour établir la configuration de simulation la plus apte à traiter un problème posé par un utilisateur.

Dans le cadre de ce projet, je co-encadre avec Bertrand Braunschweig la thèse cifre d'Othmane Nadjemi.

6.1.4 CellPop : Analyse, par modélisation informatique et vidéomicroscopie, du comportement dynamique de cellules individuelles et de populations cellulaires, en pathologie cancéreuse (ACI inter-EPST Bio-Informatique 2002 - 2004)

Ce projet (Programme inter-EPST "Bio-informatique") s'inscrit dans le cadre d'une collaboration entre le LERI (Laboratoire des Etudes et Recherches en Informatique de l'Université de Reims) et l'INSERM (U514). Il a démarré en 2003 et vise à mettre en œuvre une plate-forme de modélisation des comportements cellulaires dynamiques, permettant de modéliser les phénomènes intervenant dans le cadre de l'invasion tumorale.

La modélisation multi-agents interviendra comme une étape indispensable à la compréhension du mécanisme global. Elle sera un complément à l'observation de ces phénomènes grâce à la vidéomicroscopie "fonctionnelle" (avec marqueurs fluorescents) et à l'analyse quantitative de séquences d'images, qui sont déjà bien maîtrisés par les biologistes associés au projet. Plus précisément, la démarche de travail sera la suivante : à partir d'un modèle pré-supposé, l'évolution dans le temps d'une population cellulaire sera calculée (sous forme de séquences d'images simulées) et la quantification permettra de déduire des paramètres quantitatifs pertinents. En parallèle, des expériences de vidéomicroscopie (avec les marqueurs adéquats) permettront d'acquérir des séquences d'images réelles desquelles seront déduits les mêmes paramètres quantitatifs. La confrontation des deux jeux de données quantitatives permettra de valider ou d'infirmer le modèle, et d'affiner ses paramètres.

Les phénomènes biologiques étudiés seront : la migration cellulaire, la sociologie cellulaire (distribution spatiale de populations), l'adhésion, la forme et la déformation en relation avec la fonction.

6.2 Encadrement de travaux de recherche

6.2.1 Thèses

- Co-encadrement (50%) de Karima Boudaoud Eurécom, Nice
  Directeur de thèse : Jacques Labetoule
  Sujet : Une approche multi-agents pour la détection d'intrusions

thèse soutenue en Décembre 2000.
Eurécom (Nice), École Polytechnique Fédérale de Lausanne
Situation actuelle : maître de conférences à l'Université de Nice, membre de l'équipe Rainbow de Michel Riveill.

- Co-encadrement (40%) avec Mikal Ziane de David Julien
  Directeur de thèse : Jean-Pierre Briot
  Sujet : Utilisation des systèmes multi-agents pour des construire des interfaces adaptatives
  Université de Paris 6 Soutenance prévue pour 2004

- Encadrement (90%) de Lilia Rejeb
  Directeur de thèse : Herman Akdag
  Sujet : Simulations économiques, étude de l'émergence des formes organisationnelles
  Université de Reims
  Soutenance prévue pour 2004

- Encadrement (80%) de Tarek Jarraya
  Directeur de thèse : Herman Akdag
  Sujet : Agents adaptatifs appliqués au e-commerce
  Université de Reims
  Soutenance prévue pour 2004

- Encadrement (90%) de Nora Faci
  Directeur de thèse : Mohamed T. Laskri
  Sujet : Emergence de protocoles d'interaction par observation
  Université d'Annaba (Algérie)
  Soutenance prévue pour 2005

- Co-encadrement (50%) d'Ana B. Gil Gonzalez
  Thèse en cotutelle avec l'Université de Salamanque
  Directeur de thèse : Jean-Pierre Briot
  Responsable à l'Université de Salamanque : Francisco García
  Sujet : Agents adaptatifs pour le commerce électronique
  Soutenance prévue pour 2005

- Co-encadrement (50%) d'Othmane Nadjemi
  Directeur de thèse : Jean-Pierre Briot
  Responsable à l'IFP : Bertrand Braunschweig
  Sujet : Agents logiciels intelligents pour la simulation numérique : l'exemple des Cogents
  Soutenance prévue pour 2005

6.2.2 DEA

- 2002-2003 : Encadrement de Hager Karoui du DEA Lamsade
  Sujet : Système multi-agents tolérants aux pannes

- 2001-2002 : Encadrement de Marc Montagnier du DEA IARFA de Paris 6, et d'Athmane Hamel du DEA Lamsade
  Sujet 1 : Représentation et implémentation des formes organisationnelles
  Sujet 2 : Système multi-agents tolérants aux pannes

- 2000-2001 : Encadrement de trois DEA IARFA de Paris 6 : Sylvain Baron, Sébastien Charpentier, David Julien
  Sujet1 : Système multi-agents tolérants aux pannes
  Sujet2 : Systèmes d'aide à la conception des systèmes à base de composants
  Sujet 3 : Méthodologie de conception d'agents adaptatifs

- 2000-2001 : Co-encadrement avec Frédéric Peschanski d'un étudiant en maîtrise (Université de Clermont Ferrand) : Olivier Vernin
  Sujet : Déploiement des agents DIMA en utilisant Comet

- 1999-2000 : Encadrement de trois DEA (2 Paris 6 et 1 ENS Lyon) : Michel Quenault, Aldric Mahé et Guillaume Lacôte
  Sujet1 : Agents adaptatifs
  Sujet2 : Organisations adaptatives
  Sujets : SMA tolérants aux pannes

- 1998-1999 : Encadrement d'un DEA (Paris 6), Khadija Aoun et Redouane Djelouah (Dauphine)
  Sujet 1 : Intégration de Bast et DIMA
  Sujet 2 : Un système multi-agents pour l'étude du marché électrique européen

- 1997-1998 : Encadrement de deux binômes DESS GLA (Paris VI), Projet annuel
  Sujet 1 : Méta-modélisation de DIMA
  Sujet 2 : Utilisation de DIMA pour développer un simulateur de réseaux ATM

- 1996-1997 : Encadrement d'un binôme DESS GLA (Paris VI), Projet annuel
  Sujet : Méta-modélisation d'un système multi-agents qui simule l'évolution économique

- 1994-1995 : Encadrement d'un binôme DESS IA (Paris VI), Projet annuel
  Sujet : EDDAKS : un simulateur de procédés industriels de fabrication

6.3 Conférences invitées et séjours à l'étranger

- Invitée à UIUC (University of Urbana-Champaign of Illinois), équipe du Professeur Gul Agha, Juil.-Sept. 1999.
- Z. Guessoum. Adaptive Agents and Multi-Agent Systems. Dagstuhl Seminar Number 03081, Objects, Agents and Features, H.-D. Ehrich (Univ. Braunschweig, D), J.-J. Meyer (Utrecht, NL), M. Ryan (Univ. Of Birmingham, GB)
- P. Sens et Z. Guessoum. Systèmes multi-agents tolérants aux pannes. GRID2002 (Ecole d'hivers), Aussois, décembre 2002.
- Z. Guessoum. Système multi-agents. Tutorial à Seventh Maghrebian Conference on Software Engineeringand Artificial Intelligence (MCSEAI02), Annaba, mai 2002.
- Z. Guessoum. Adaptive agent and multi-agent systems. Workshop Franco-japonais sur les objets distribués et les agents, Tokyo, Dec. 2000
- Invitée à UIUC (University of Urbana-Champaign of Illinois), équipe du Professeur Gul Agha, Mai-Juin, 2003.

6.4 Publications

6.4.1 Editeurs

- X. Briffault, Z. Guessoum et M. Occello. Numéro thématique Technique et Science Informatiques (TSI), Environnements de développement multi-agents. Vol. 21, No. 4, 2002. Rédacteur responsable des 5 papiers publiés.
- Z. Guessoum et M. Occello. Workshop Méthodologies et Environnements pour les Systèmes Multi-agents, Plate-forme AFIA 2001. publié pas l'AFIA.
- O. Boisier, Z. Guessoum et M. Occello. Plates-formes multi-agents. Bulletin de l'AFIA No. 37, Octobre 1999.

6.4.2 Chapitres dans des ouvrages collectifs

- Z. Guessoum, M. Cossentino and J. Pavon Mestras. A Roadmap of Agent-Oriented Software Engineering : The European Agentlink Perspective. In "Methodologies and Softawre Engineering for Agents Systems", Federico Bergenti, Marie-Pierre Gleizes and Franco Zambonelli 'eds.), Kluwer, à paraître en janvier 2004.

- R. Durand and Z. Guessoum. Competence systemics and survival, simulation and emperical analysis. Ron Sanchez (ed.), Elsevier Pergamon Press, To appear 2004.

- Z. Guessoum, J.-P. Briot, O. Marin, A. Hamel and P. Sens. Dynamic and Adaptative Replication for Large-Scale Reliable Multi-Agent Systems. In "Software Engineering for Large-Scale Multi-Agent Systems", Alessandro Fabricio Garcia, Carlos Lucena, Franco Zambonelli, Andrea Omicini and Jaelson Castro (eds.), LNCS 2603, pp. 182-198, April 2003.
- A. B. Gil, F. J. García, Z. Guessoum. Adaptive Agents for E-commerce Applications. In "New Methods and Tools Supporting E-Commerce", R. Corchuelo, A. Ruiz, J. A. Pérez (Eds.), pp. 27-36. Editorial Catedral, Salamanca. ISBN 84-96086-02-X. 2002.
- Z. Guessoum. Modèles multi-agents et systèmes dynamiques. In "Systèmes Dynamiques", Agnès Guillot Emmanuel Daucé (eds.), Hermès, Décembre 2002. 0.2cm
- ASA (www-poleia.lip6.fr/∼guessoum/asa.html). Une comparaison des plates-formes multi-agents. In "Organisation et applications des SMA", René Mandiau, Emmanuelle Grislin- Le Strugeon, André Péninou (eds.), Hermès, pp. 207-240, mai 2002.

- Z. Guessoum et M. Occello. Environnements de développement. In "Les systèmes multi-agents", Jean-Pierre Briot et Yves Demazeau (eds.), Collection IC2. Hermès Science Publications, pp. 177-206, Paris, France, décembre 2001.

6.4.3 Revues avec comité de lecture

- M. Occello, Z. Guessoum et O. Boissier. Un essai de définition de critères pour l'étude comparative de plates-formes multi-agents. Technique et Science Informatiques (TSI), Numéro thématique : Environnement de développement de systèmes multi-agents. pp. 549-553, Vol. 21, No. 4, avril 2002.
- Z. Guessoum. A Multi-Agent Simulation Framework. Transactions of Computer Simulation, Vol. 17, No. 1, pp. 2-11. April 2000.
- Z. Guessoum and J.-P. Briot. From Active Object to Autonomous Agents. IEEE Concurrency, Vol., 7 No. 3, pp. 68-78, July/September, 1999.
- Z. Guessoum. DIMA : Une plate-forme multi-agents en Smalltalk. Revue Objet, Vol. 3 No 4, pp. 393-410, décembre 1998.
- M. Dojat, F. Pachet, Z. Guessoum. Touchard, A. Harf, L. Brochard. NéoGanesh : a Working System for the Automated Control of Assisted Ventilation in ICUs. Artificial Intelligence in Medicine, Vol. 11 No. 2, pp. 97-117, September/October 1997.

6.4.4 Revues sans comité de lecture

- Bertrand Braunschweig, Eric S. Fraga, Zahia Guessoum, Didier Paen, Daniel Pinol, Aidong Yang. COgents : a new IST-funded project on CAPE-OPEN Software Agents, CAPE-OPEN Update Vol. 3.
- Z. Guessoum. La plate-forme DIMA. AFIA No. 39, octobre 1999.

6.4.5 Conférences internationales avec comité de lecture

- D. Julien, M. Ziane and Z. Guessoum. Goliath : an extensible model-based environment to develop user interfaces. To appear in Proc. CADUI'04 (Triennal International Conference on Intelligent User Interfaces), Kluwer, Portugal January 13-16, 2004.
- L. Rejeb, Z. Guessoum, R. Durand and T. Jarraya. Adaptive Multi-Agent Simulation of Firms and Organizational. In proc. International Symposium on Programming and Systems (ISPS) may, 2003 Alger.
- Z. Guessoum, L. Rejeb and R. Durand. Multi-Agent Simulation of Firms and Organizational Forms. In proc. the Second International Joint Conference on Autonomous Agents Multiagent Systems, AAMAS 2003, pp. 1000-1001, July, 2003, ACM 2003.
- B. Braunschweig , E. S. Fraga, Z. Guessoum, D. Paen and A. Yang. Coagents : Cognitive Middleware Agents to Support e-CAPE. In proc. eBusiness/eWork Conference, Prague, October 2002.
- A. B. Gil, F. García, Z. Guessoum : An Adaptive Agent Model for e-Commerce Architecture. Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. In proc. AH 2002, Malaga, Spain, May 29-31, 2002. Lecture Notes in Computer Science 2347 Springer 2002, pp. 592-597.
- Z. Guessoum, J.-P. Briot, S. Charpentier, O. Marin and P. Sens. A Fault-Tolerant MultiAgent Framework. In Proc. The First International Joint Conference on Autonomous Agents Multiagent Systems, AAMAS 2002, pp.672-673, July 15-19, ACM 2002.
- R. Durand and Z. Guessoum. Competence systemics and survival, simulation and emperical analysis. In Proc. Competence 2000 Conference, Helsinki, Finland, to appear, June 10-14, 2000.
- K. Boudaoud, Z. Guessoum : A Multi-agents System for Network Security Management. In proc ; of Sixth International Conference on Intelligence in Networks (SMARTNET 2000), September 18-22, 2000, Vienna, Austria. IFIP Conference Proceedings 178 Kluwer 2000, pp. 407-418.
- K. Boudaoud, H. Labiod, Z. Guessoum et R. Boutaba. Network Security Manage-

ment with Intelligent Agents. in Proc. IFIP/IEEE International Network Operation and Management Symposium (NOMS'00), IEEE Press, pp. 579-792, Hawaii, April 2000.

- R. Durand, K. Cool and Z. Guessoum. Resource accumulation and sustainability of competitive advantage Simulation and empirical analysis. In Proc. Academy of Management Conference, Aug. 1999, Chicago.
- Z. Guessoum and J.-P. Briot. An Autonomous Agent Structure. In Proc. the 7th International Conference on Intelligent Systems (ICIS'98), Fontaine-Blue, Jul. 1998.
- Z. Guessoum. A Hybrid Agent Model : a Reactive and Cognitive Behavior. In Proc. ISAD97, IEEE, Berlin, Germany, pp. 25-32, April 1997.
- Z. Guessoum et R. Durand. A multi-agent system to study the economics agents evolution. In Proc. the third international conference on applications of computer systems, ACS'96, Szczecn, Poland, pp. 137-142, November 1996.
- R. Durand et Z. Guessoum. Stratégies-types et modélisation du processus concurrentiel : exemple d'application des systèmes multi-agents. In Proc. Conférence Internationale sur le Management, Paris, France, pp. 201-214, juillet 1996.
- Z. Guessoum. A framework integrating an object-oriented multi-agent system and discrete event simulation. In Proc. First LAAS IEEE-ICCS'95, Beirut, Lebanon, pp. 165-173, 1995.
- Z. Guessoum et C. Oussalah. An object oriented environment for representing production rules. In Proc. IASTED conference, Alexandria, Egypt, pp. 401-405, May 1992.

6.4.6 Workshops internationaux avec comité de lecture

- Z. Guessoum, O. Nadjemi, B. Braunschweig , A. Yang, D. Pionel and I. Stallker. Agent-Based Computer-Aided Process Engineering. In proc. 2nd. Workshop on Intelligent Computing in the Petroleum Industry ICPI 2003, IJCAI03 conference, Acapulco, August 2003.
- Z. Guessoum, L. Rejeb and R. Durand. Emergence of Organizational Forms. In proc. The third symposium on Adaptive Agent Multiagent Systems (AAMAS 2003), AISB'03, april, 2003.
- Z. Guessoum, J.-P. Briot, Z. Charpentier, S. Aknine, O. Marin and P. Sens. Dynamic and Adaptative Replication for Large-Scale Reliable Multi-Agent Systems. In Proc. ICSE'02 First International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'02), ACM, Orlando FL, U.S.A., may, 2002.
- J.-P. Briot, Z. Guessoum, S. Charpentier, S. Aknine, O. Marin, P. Sens. Dynamic Adaptation of Replication Strategies for Reliable Agents. In proc. AISB'02, pp.

13-21, London, UK, April 2002.

- A. Gil, F. García, Z. Guessoum. Dynamic and Adaptive E-Commerce architecture Based on Agent Technologies. In Proc. the 9th International Workshop Design, Specification, and Verification of Interactive Systems, DSV-IS'2002, Peter Forbrig, Quentin Limbourg, Bodo Urban, Jean Vanderdonckt (Eds.), pp. 282-295. Rostock (Germany), June 2002.
- K. Boudaoud, Z. Guessoum and Ph. Dubois. Polcicy-based Security Management Using Multi-Agent Systems. In Proc. HP-OVUA Workshop, Berlin , June 2001.
- O. Marin, P. Sens, J.-P. Briot and Z. Guessoum. Towards Adaptive Fault-Tolerance for Distributed Multi-Agent Systems. In Proc. ERSADS'2001, Bertinoro, Italy, May 2001.
- Z. Guessoum, J.-P. Briot, P. Sens and O. Marin. Fault Tolerant Multi-Agent systems. Poster in MAAMAW'01, Annecy (France), may 2001.
- Z. Guessoum. Adaptive Multiagent Systems. In Proc. AIB'S, pp. 107-112, York, 20-24 mar, 2001, ISBN 1 902956 17 0.
- Z. Guessoum, M. Quenault and R. Durand. An Adaptive Agent Model. In Proc. AIB'S, pp. 100-116, York, 20-24 mar, 2001, ISBN 1 902956 17 0.
- G. Lacotte, J.-P. Briot, Z. Guessoum and P. Sens. Towards Fault-Tolerant Agents. ECOOP'2000 Workshop on Distributed Objects Programming Paradigms, Cannes, juin 2000.
- B. Huet, Z. Guessoum and G. Blain. A Multi-Agent Simulation of Intensive Care Unit Second International. ICSC Symposium on Engineering of Intelligent Systems, Scotland, U.K.n, June 27-30, 2000.
- Z. Guessoum, A. cardon and A. Ramdani. An Adaptive Multi-Agent Systems. Poster in MAAMAW'99, Valencia, Spain, June 1999.
- Z. Guessoum and A. cardon. Self-Adjustable Autonomy in Multi-Agent Systems. AAAI'99-Self Adjustable Autonomy. Stanford, March 1999.
- Z. Guessoum and J-P Briot. In quest of the missing link between active objects and autonomous agents. Poster in MAAMAW'97, May, 1997.
- Z. Guessoum et P. Deguenon. A Multi-Agent Approach for Distributed Discrete Event Simulation. Proc. the First International Workshop DIMAS'95, Poland, pp. 183-190, November, 1995.

6.4.7 Conférences nationales avec comité de lecture

- A. Thiefaine, Z. Guessoum, J.-F. Perrot et G. Blain, Génération de systèmes multi-agents à partir de modèles, à paraître dans JFSMA03, Tunisie, Novenbre 2003.

- A. Gil, Z. Guessoum and F. García. Recomendadores en un Sistema Multiagente Adaptativo para el Comercio Electrónico.Taller en Sistemas Hipermedia Colaborativos y Adaptativos dentro de las JISBD'2002. El Escorial, 18 al 22 de Noviembre del 2002.
- L. Rejeb, Z. Guessoum et R. Durand. Modélisation multi-agents des formes organisationnelles. JFIADSMA, J-P Müller (ed.), Hermès, pp. 106-109, octobre 2002.
- A. Cardon, Z. Guessoum. Systèmes multi-agents adaptatifs. JFIADSMA'2000, Sylvie Pesty et Olivier Boissier (eds.), Hermès, pp. 100-116, octobre 2000.
- Z. Guessoum, A. cardon et A. Ramdani. Vers des SMA adaptatifs. JFIADSMA'99, Ile de la Réunion, France, pp. 337-339, novembre 1999.
- Lesueur, Z. Guessoum, G. Sunyé, G. Blain et J.-F. Perrot La métaphore du dossier. INFORSID'99, pp. 279-299, Tour, juin 1999.
- Z. Guessoum, J-P. Briot et M. Dojat. Des objets concurrents aux agents autonomes, JFIADSMA'97, Joël Quinqueton, Marie-Claude Thomas et Brigitte Trousse (eds.), Hermès, pp. 93-106, Avril, 1997.
- Z. Guessoum. Des objets concurrents aux agents autonomes, JFLA'97, Lyon, Février 1997.
- Z. Guessoum et R. Durand. Des agents intelligents pour modéliser l'évolution des entreprises. in Intelligence artificielle distribuée, Jean-Pierre Müller et Joël Quinqueton (eds.), Hermès, Paris, pp. 37-58, Avril 1996.
- Z. Guessoum et M. Dojat. Le contrôle dans les systèmes multi-agents. CRAC'96, 30-31 mai, Paris, 1996.
- Z. Guessoum Simulation à événements discrets et modélisation multi-agents. Troisièmes Journées Francophones IADSMA'95, Chambéry- St Baldoph, pp. 211-222, 1995.
- Z. Guessoum. Une plate-forme multi-agents. Deuxièmes Rencontres de Jeunes Chercheurs en IA, Marseille, pp. 299-310, 19-21 septembre, 1994.
- Z. Guessoum. Systèmes asynchrones de production. Deuxièmes Journées Francophones IADSMA'94, Voiron, pp. 169-180, 9-11 mai, 1994.

6.4.8 Thèses

- Z. Guessoum. Un environnement opérationnel de conception et de réalisation de systèmes multi-agents. Thèse de l'Université Paris 6, LAFORIA, mai 1996.
- Z. Guessoum. Un environnement orienté objets pour la représentation des règles de production. Thèse de Magister (équivalent Master), Haut commissariat à la recherche, Centre de développement des technologies avancées, Laboratoire de Robotique et d'Intelligence Artificielle, Alger, mai 1991.

6.4.9 Rapports internes

- A. Cardon et Z. Guessoum. Système multi-agents adaptatifs. Rapport interne lip6, mars 2000.
- Houda Labiod, Zahia Guessoum. An Adaptive BCH Forward Error Correction Scheme for ATM Transmission Over Wireless channels. Rapport interne UVSQ 1998/20, Université de Versaille, 1998.
- M. Dojat, F. Pachet, Z. Guessoum, D. Touchard, A. Harf, L. Brochard. NéoGanesh : a Working System for the Automated Control of Assisted Ventilation in ICUs. Artificial Intelligence in Medecine,rapport LAFORIA, 1997.
- Z. Guessoum. DIMA : un environnement de conception et de réalisation de systèmes multi-agents, GDR programmation, Orléans, Novembre 1996.
- Z. Guessoum et M. Dojat. A Real-Time Agent Model in an Asynchronous Object-Oriented Environement. Raport LAFORIA 19/96, février 1996.

6.5 Responsabilités administratives et scientifiques

6.5.1 Responsabilités administratives

- Membre de la coordination du poleia IA du LIP6 de 1998 à 2000.
- Membre suppléant de la commission de spécialistes (section 27) de l'Université de Reims (depuis 2001).
- Membre titulaire de la commission de spécialistes (section 27) de l'Université d'Evry (depuis 2001).
- Membre du conseil scientifique du LERI (Laboratoire des Etudes et recherches en Informatique de l'Université de Reims) depuis 1999.
- Responsable de stages d'étudiants en deuxième années à l'IUT de Reims de 1998 à 2002.

6.5.2 Responsabilités scientifiques

Comité de programmes

- Membre du comité de programme du Third International Joint Conference on Autonomous Agents and Multi Agents Systems (AAMAS04).
- Membre du comité de lecture de du livre Software Engineering for Large-Scale Multi-Agent Systems, (SELMAS), LNCS, à paraître en 2004.
- Membre du comité de programme du fourth symposium on Adaptive Agent Multiagent Systems (AAMAS 2004), AISB'04 ,Leeds, March 2004.
- Membre du comité de programme de Eighth Maghrebian Conference on Software Engineeringand Artificial Intelligence (MCSEAI04), Tunisie, 2004.

- Membre du comité de programme des JFSMA'03 (Journées Francophones Systèmes Multi-Agents), Tunisie 2003.
- Membre du comité de programme de la conférence ISPS'02 (International Symposium on Programming and Systems), Alger, 2003.
- Membre du comité de programme de RJCIA'03 (Rencontres des Jeunes Chercheurs en Intelligence Artificielle), Laval, 2003.
- Membre du comité de programme des JFIADSMA (Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents) de 1998 à 2002.
- Membre du comité de programme de Maghrebian Conference on Software Engineering and Artificial Intelligence (MCSEAI02), Annaba, 2002.
- Membre du comité de la direction du GdR I3.
- Membre du comité de programme de la plate-forme AFIA 2001.

Membre de comités de lecture

- Conférences AAMAS-2003 et AAMAS-2002.
- Conférence ICMAS-2000 (International Conference on Multi-Agent Systems).
- Workshops MAAMAW-1999, MAAMAW-2001.
- Conférence ECOOP.
- Revue IEEE concurrency.
- Revue TSI.
- Revue Integrated Computer-Aided Engineering Journal, special issue on Agent Technology.
- Revue Knowledge and InformationSystems (KAIS).

Jury de thèse

- Membre du jury (examinateur) de la thèse de Stéphane Durand (Université du Havre).
- Membre du jury (examinateur) de la thèse de Raouf Benamara (Université d'Evry).
- Membre du jury (examinateur) de la thèse de Karima Boudaoud (EPFL et Eurécom).

Organisation de conférences et animation de groupes de recherche

- responsable du groupe AFIA/PRC-I3 (Groupe 2.2 du GDR-I3 depuis juin 2001) ASA (Approches par Sociétés d'Agents)
  http ://www-poleia.lip6.fr/∼guessoum/asa.html)
  organisation des réunions de ce groupe (3 fois par an), rédaction des comptes rendus pour le groupe et pour le bulletin de l'AFIA, etc.

  Le groupe ASA vise à étudier les différentes approches proposées pour l'ingénierie des SMA (architectures, modèles, démarches et méthodes). L'objectif de cette étude est double. D'une part, le groupe ASA souhaite recenser les principales

réalisations et en proposer une synthèse cohérente. D'autre part, il se propose de définir des benchmarks qui permettent l'analyse et la comparaison des différentes étapes de ces approches.
- Organisation d'une journée AFIA-PRC "Recherche/Industrie" en collaboration avec Eunika Mercier-Laurent.
  Compte rendu http ://www-poleia.lip6.fr/∼guessoum/asa/CRSMA1.html.
- Organisation de l'atelier "Méthodologies et Environnements pour les Systèmes Multi-agents" en collaboration avec Michel Occello, Plate-forme AFIA 2001.

Divers

- Membre de Specif (correspondante du poleia du LIP6).
- Membre de l'AFIA.
- Membre de AgentLink.
- Membre (Responsable du noeud LIP6) du réseau d'excellence AAMAS (Adaptive Agents and Multi-Agent Systems).
- Membre de AgentLink II ALAD SIG.
- Membre de AgentLink II ALAD AOSE.