Organisation de Sociétés d'Agents pour la Visualisation d'Informations Dynamiques

THÈSE

présentée et soutenue publiquement le 18 décembre 2001

pour obtenir le titre de

Docteur de l'Université Pierre et Marie Curie - Paris VI
(spécialité Informatique)

par

Valérie RENAULT

Composition du jury

Président :	Jean-Charles Pomerol : Professeur, Université de Paris 6
Rapporteurs :	Jean-Pierre Müller : CIRAD, Montpellier Bernard Pavard : Directeur de Recherche CNRS, IRIT GRIC, Toulouse
Examinateurs :	Guillaume Hutzler : Maître de Conférence, Université d'Evry Claude Kintzig : France Telecom R&D, Issy Moulineaux
Directeur de thèse :	Alexis Drogoul : Maître de Conférence, Université de Paris 6

Remerciements

En premier lieu, je tiens à remercier Alexis Drogoul d'avoir accepté d'encadrer ces trois années de recherche. La confiance et la liberté qu'il m'a accordées ont été aussi enrichissantes que son soutien scientifique et moral, toujours prodigué avec sympathie, même dans les moments critiques...

Je tiens aussi à remercier Jean-Pierre Briot de m'avoir accueillie au sein de l'équipe OASIS du Laboratoire d'Informatique de Paris 6.

Je remercie vivement Jean-Charles Pomerol de m'avoir fait l'honneur de présider mon jury ainsi que Jean-Pierre Müller et Bernard Pavard d'avoir accepté d'être rapporteurs de ma thèse et d'avoir porté leur regard de spécialiste sur une thèse qui s'est voulu pluridisciplinaire. Merci aussi à Claude Kintzig d'avoir participé à ce jury.

Un grand merci à Guillaume Hutzler et à Bernard Gortais d'avoir initialisé le projet des *Jardins de Données* et de m'avoir ainsi permis de partager avec eux leur projet de recherche d'où ont découlé les problématiques de mes travaux de thèse. Merci à tous les membres des équipes OASIS et MIRIAD, ainsi qu'à tous ceux du LIP6 qu'il m'a été donné de croiser pendant ces trois années, sans oublier la machine à café du $7^{\text{ème}}$ sans laquelle de nombreuses discussions intra et inter-équipes n'auraient pas eu lieu.

Merci à l'équipe du Studio créatif de France Telecom R&D. Je remercie tout particulièrement Jean-Marc Raibaud, Annie-Claire Papadopoulo et Laurent Ponthou de m'avoir accueillie au sein du Studio créatif. Merci aussi à Joëlle Fezay sans qui j'aurais souvent dû voyager en vélo et loger à la belle étoile lors de mes différentes missions. And thanks to Bénédicte Athimon-Pillard et Christelle Tenière pour tous les jeux de l'oie, les chercheurs d'or et les cowboys. Je remercie Denis Chene et Naima Lankri pour leur collaboration ergonomique sur LEA.

Tous les petits agents de LEA et d'OSCAR se joignent à moi pour dire mille mercis à Laurent Breton et à Mélanie Courtine pour leur soutien au quotidien, c'est-à-dire pour les nombreux petits déj', croissants, chocolats, teurgoule et fous rires compris. Et merci encore plus pour l'aide qu'ils m'ont apportée grâce à leurs compétences en Java et en XML et pour les nombreuses discussions que nous avons eues autour de l'architecture d'OSCAR et de LEA. Et aussi merci pour les nombreuses lectures, relectures et re-relectures de ce mémoire.

Des milliers de mercis à toute ma famille de ne pas m'en avoir trop voulu de l'avoir un peu délaissée ces dernières années, en particulier des milliers de mercis à Lucien, Nanou et Germaine. Et j'aimerais enfin et surtout remercier ceux que l'on ne voit jamais mais sans le soutien desquels tout aurait été tellement plus compliqué voire impossible : Mistic, Laetitia, Sandrine, Laurent, Jean, Nicole et Hervé.

à E.T. et au Petit Prince

Table des matières

Conclusion

Bibliographie

Annexes

Table des figures

Table des algorithmes

Introduction

«- Le point commun à toutes ces machines, c'est qu'elles n'opèrent que sur des données fournies à leurs opérateurs internes par les usagers. Une machine à qui l'on ne pose pas un problème défini reste incapable d'initiative.

- Et pourquoi n'a-t-on pas essayé de les doter d'une conscience et d'un raisonnement ?

- Parce qu'on s'est aperçu qu'il suffisait de les munir de quelques fonctions réflexes élémentaires pour qu'elles prennent des manies pires que celles des vieux savants. Achetez dans un bazar une petite tortue électronique de gosse, et vous verrez à quoi ressemblaient les premières machines électro-réflexes : irritables, fantasques... douées en somme d'un caractère.»

— Boris Vian, *Le danger des Classiques*, 1950

---

1 Problématique et objectifs

Les systèmes multi-agents sont de plus en plus «en vogue» en particulier grâce au développement des *agents intelligents* et des *robots sociaux* [dossier Sciences et Avenir 2001]. Les agents restent cependant peu exploités dans des domaines de recherche tels que les interfaces homme-machine et la visualisation d'informations. Pourtant, ces derniers présentent des caractéristiques très proches des systèmes multi-agents et spécialement des sociétés d'agents.

De plus en plus, les interfaces de visualisation doivent traiter des données dynamiques et hétérogènes issues de diverses sources. La visualisation doit donc intégrer des processus d'organisation de l'information et des mécanismes d'interactions avec l'utilisateur tout en conservant une certaine autonomie afin de s'adapter à un environnement changeant.

De façon similaire, les sociétés d'agents sont composées de nombreuses entités autonomes et hétérogènes. Les nombreuses interactions entre ces entités conduisent à des organisations capables de s'adapter aux modifications de leur environnement.

Malgré ces similitudes, il existe peu d'interfaces *multi-agents* de *visualisation* de données [Proctor et Winter 1998, Ishizaki 1996]. Les interfaces *agents* qui existent actuellement sont généralement conçues à partir de quelques entités, peu nombreuses et relativement complexes [Lieberman *et al.* 1999]. Ces entités disposent donc de nombreuses connaissances et sont ainsi souvent fortement dépendantes du domaine d'application.

À partir des similitudes entre les objectifs des interfaces de visualisation et les possibilités des sociétés d'agents, nous avons défini une double problématique. D'une part, montrer comment la conception de systèmes multi-agents pour la visualisation d'informations peut conduire à la construction de nouveaux modèles d'organisation de sociétés d'agents. D'autre part, décrire comment une interface multi-agent de visualisation peut répondre à des contraintes propres à la visualisation telles que la présentation organisée d'informations hétérogènes ou l'adaptation dynamique aux actions de l'utilisateur.

Nous proposons ainsi une alternative aux interfaces existantes en concevant une interface de visualisation multi-agent basée sur une société d'agents relativement peu évolués, mais dont les nombreuses interactions conduisent à une certaine «intelligence» du système dans sa globalité.

2 Démarche suivie

Notre démarche consiste à prendre en compte les contraintes et les caractéristiques issues de la visualisation d'informations [Card *et al.* 1999, Leung et Apperley 1994] afin de concevoir des agents dont le rôle est de recueillir, d'analyser et de représenter ces informations.

Afin de permettre à ces agents d'interagir et de s'organiser, nous les avons dotés de comportements de regroupement issus de modèles éthologiques [Reynolds 2000], [Resnick 1994a, Bonabeau et Theraulaz 1994]. La visualisation d'informations à l'écran repose ainsi sur la visualisation de ces groupes d'agents et de leurs interactions. Enfin, ces agents sont dotés de capacités d'interactions avec l'utilisateur [Sarkar et Brown 1992]. Ce dernier peut alors modifier dynamiquement leur organisation à l'écran et ainsi personnaliser son interface.

Notre travail se situe ainsi à la confluence de plusieurs disciplines dont les principales sont la visualisation d'informations, les systèmes multi-agents et l'éthologie.

3 Ce que l'on ne fera pas

Nos travaux reposent sur l'étude des systèmes multi-agents utilisés en tant que noyau organisationnel dans des interfaces de visualisation. Nous axerons donc nos recherches sur les modèles éthologiques permettant de mettre en œuvre des organisations - ou sociétés - d'agents et sur l'influence que peuvent avoir ces organisations sur la visualisation de données numériques ou textuelles. C'est pourquoi, dans ce mémoire, nous ne traitons pas de problèmes spécifiques aux données textuelles tels que la **détection automatique de mots-clés** ou le **traitement sémantique** de texte.

L'un des intérêts des organisations multi-agents est la possibilité de s'adapter dynamiquement et de façon autonome aux modifications de l'environnement. Ainsi nous avons préféré positionner nos travaux autour des systèmes de visualisation de données n'entrant pas dans le cadre des **systèmes d'alarme**. En effet, ces derniers sont principalement «intéressants» au moment précis où un incident particulier survient alors que nous nous sommes plutôt attachés à étudier comment les modifications progressives de flux de données peuvent être traduites en termes d'évolution d'une société d'agents.

Nous n'aborderons pas non plus les mécanismes de **contrôle** de systèmes industriels complexes. Ces mécanismes permettent d'agir directement sur les processus conduisant à l'émission des données et requièrent donc de la part des utilisateurs des activités de *perception*, de *traitement de l'information* et de *résolution de problèmes*. Les utilisateurs sont aussi placés dans un contexte de *communication* avec une équipe et d'actions physiques sur des systèmes de commande [Berliner *et al.* 1964]. Les aspects communication et commande ne sont pas pris en compte dans nos travaux, les interactions entre l'utilisateur et les agents n'ayant pas pour rôle de modifier les données mais d'en modifier leur visualisation. Cependant les études menées autour des activités de perception et de traitement de l'information, telles que celles mises en œuvre dans le contrôle aérien [Boudes et Amaldi 1996], par exemple, sont à prendre en compte lors de la conception d'une interface de visualisation d'informations dynamiques.

LEA, l'application que nous présentons dans ce mémoire, a pour objectif d'organiser des données textuelles, en l'occurrence des courriers électroniques. Le choix de recourir à ces données textuelles est venu de la nécessité de disposer d'un grand nombre de données, évoluant dynamiquement et pouvant posséder des relations entre elles. Afin de tester et de valider notre approche, il était aussi nécessaire de manipuler des données facilement intelligibles par le plus grand nombre d'utilisateurs. L'application à la visualisation de boîtes aux lettres électroniques a été choisie car elle possède toutes ces propriétés, tout en s'intégrant parfaitement aux recherches menées au sein de France Telecom R&D.

Pour finir, LEA n'est hélas pas conçue pour **répondre automatiquement au courrier** à la place de l'utilisateur.

4 Organisation du manuscrit

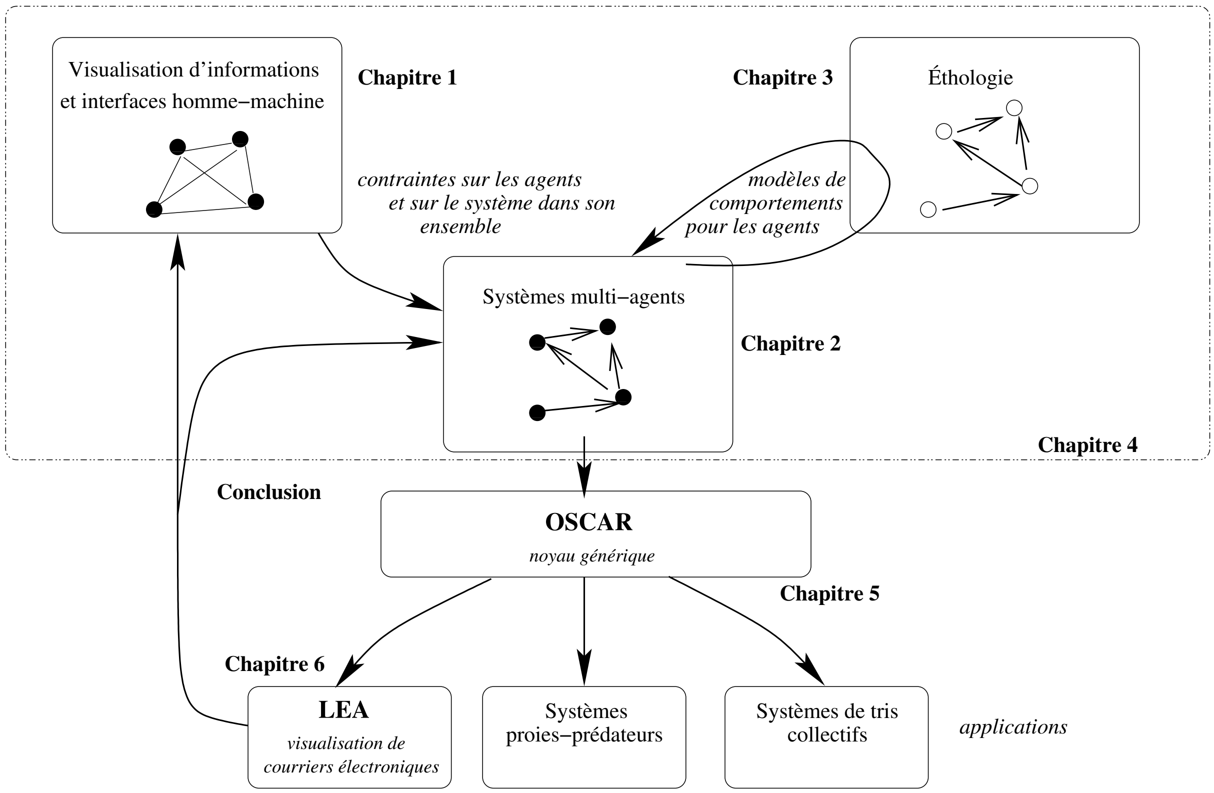


FIG. 1 – Organisation du manuscrit

La figure 1 résume l'organisation de ce mémoire.

Le chapitre 1 introduit le domaine des interfaces homme-machine et plus particulièrement celui des **systèmes de visualisation d'informations**. Ces systèmes ne peuvent être conçus sans d'abord prendre en considération les études psychologiques menées sur le **système perceptif** humain. Les études menées sur la construction perceptive, la mémoire et l'attention permettent de prendre en compte les caractéristiques et les limites du système perceptif lors de la conception d'une interface visuelle. Nous présentons ensuite un état de l'art non exhaustif des systèmes de visualisation et des moyens d'interactions homme-machine. Nous nous focalisons sur certains points-clés que nous retrouverons lors la conception d'interfaces multi-agents, tels que la cartographie [Honkela *et al.* 1997] ou les techniques de «Fisheye views» [Furnas 1986]. Nous concluons ce chapitre en synthétisant les propriétés «idéales» que devraient posséder une interface de visualisation de données dynamiques.

Le chapitre 2 récapitule les travaux menés dans le domaine des **agents** et des **systèmes multi-agents**. Dans un premier temps, nous présentons les principales caractéristiques de ces systèmes, en particulier, nous mettons en évidence l'importance du rôle de leur environnement, des interactions des agents et de leurs capacités d'adaptation. Nous présentons ensuite leurs capacités d'auto-organisation afin d'aborder les problèmes que peut poser la conception d'un système à plusieurs niveaux de description et d'interprétation. Nous finissons ce chapitre en présentant quelques agents autonomes d'interface existants.

Dans le domaine des systèmes multi-agents réactifs, les sociétés d'agents et leurs organisations sont souvent issues de métaphores éthologiques. Comme nous attendons de nos agents qu'ils apportent leurs compétences organisationnelles aux systèmes de visualisation, le chapitre 3 détaille des modèles d'organisations multi-agents. Ce chapitre montre notamment comment les **modèles éthologiques**, et en particulier les modèles issus des sociétés d'insectes sociaux, peuvent fournir des modèles comportementaux pour nos agents. Nous discutons aussi, dans ce chapitre, de l'utilisation de métaphores dans les systèmes multi-agents.

Par l'intermédiaire de la présentation des quelques systèmes pouvant être rapprochés de notre problématique, le chapitre 4 explique l'**agencement des différents domaines** décrits aux trois chapitres précédents. Il nous permet de **positionner nos recherches** par rapport à ces domaines. La conception multi-agent de systèmes de visualisation ne peut être mise en œuvre sans tenir compte des travaux déjà menés dans les domaines de la psychologie et des interfaces homme-machine. Ces travaux contraignent donc le choix des relations entre les données et les agents, mais aussi les comportements des agents eux-mêmes ainsi que ceux du système multi-agent dans sa globalité. Nous postulons ici que des agents, dont les comportements et les interactions sont inspirés de modèles éthologiques, peuvent générer des sociétés d'agents organisées dont la visualisation traduira l'organisation des données initiales.

Le chapitre 5 présente **OSCAR, Outil de Simulation Comportementale par Attraction-Répulsion**. OSCAR est un noyau générique permettant de construire des systèmes multi-agents à base d'agents réactifs. Ces agents communiquent via leur environnement par des émissions de phéromones virtuelles. Cette plate-forme permet à un utilisateur de créer ses propres agents et de définir leurs interactions dans des fichiers XML de configuration. Les différents modules de ce noyau sont expliqués au travers d'un exemple de tri d'objets par des termites virtuels.

Le chapitre 6 présente **LEA, Learning E-mail Agents**, un système de visualisation de boîtes aux lettres électroniques utilisant le noyau multi-agent OSCAR. Le but de cette application est d'aider l'utilisateur lors de la consultation de son courrier électronique. Chaque agent-message prend en charge un courrier et émet des stimuli locaux dans son environnement en fonction des mots-clés qu'il contient. Les agents peuvent alors se regrouper en agents-groupes ou se repousser en fonction de ces stimuli. En interagissant avec les agents graphiques de LEA, l'utilisateur a la possibilité de filtrer, d'organiser et de synthétiser les informations contenues dans ses messages.

Enfin, nous concluons ce mémoire en synthétisant les mécanismes que nous avons utilisés pour atteindre les propriétés «idéales» d'un système de visualisation définies au chapitre 1. Cependant, même si ces propriétés peuvent être considérées comme trop complexes à la vue de l'application LEA, c'était une étape nécessaire. En effet, la réalisation concrète et opérationnelle de cette application constitue un premier pas vers la conception d'interface multi-agent de visualisation d'informations complexes.

Chapitre 1

Visualisation d'informations

Sommaire

L'objectif de ce chapitre est, tout d'abord, d'appréhender les connaissances neuropsychologiques sur les «mécanismes» de la perception visuelle qui peuvent être pertinentes lors de la conception de systèmes (multi-agents) de visualisation d'informations dynamiques. En effet, il est nécessaire de connaître l'opérateur humain pour mieux l'assister [Kolski 1997] car ses activités mentales vont influer sur son comportement devant une machine. Nous définissons ensuite les systèmes de visualisation d'informations et les axes de recherches qu'ils recouvrent. Ces axes font l'objet des deux parties suivantes : l'exploration de données et les interactions homme-machine. Nous présentons ainsi certaines techniques existantes d'exploration de données en nous focalisant sur les caractéristiques que nous retrouverons lors de la conception d'interfaces multi-agents. Nous nous concentrons, en particulier, sur la cartographie de données et la représentation de données structurées. Puis, comme de tels systèmes d'informations ne prennent sens que lorsqu'ils sont en interaction avec des utilisateurs, la section suivante présente quelques techniques d'interaction homme-machine. Nous concluons enfin en mettant en évidence les propriétés idéales d'un système de visualisation d'informations dynamiques.

1.1 Éléments de psychologie de la perception visuelle

De nombreuses métaphores visuelles sont employées dans le vocabulaire courant pour énoncer des processus de pensée : «je *vois* ce que vous voulez dire», «j'ai *vu* arriver le moment où il se mettait en colère», etc. La présence du verbe *voir* n'est pas innocente et indique la relation très forte qui existe entre la pensée et la perception visuelle. Un individu augmente ses capacités à *expliquer*, mais aussi à *comprendre* des objets plus ou moins complexes à l'aide d'un schéma ou d'un dessin. Ce n'est pas un hasard si la plupart des cartons d'invitation à un mariage comportent un plan indiquant comment accéder au lieu de la cérémonie [FIG. 1.1], ni si des explications sur l'ensemble des réactions chimiques impliquées dans les processus de mémorisation au niveau neuronal doivent être complétées par un schéma [FIG. 1.2]. Ces artefacts visuels permettent d'*organiser* les différentes informations mises en jeu : où se situe la salle de réception par rapport à la mairie ? quel rôle jouent les ions calcium dans la mémorisation ? Cependant, un étudiant ayant des connaissances approfondies sur les protéines et un lecteur n'ayant aucune connaissance particulière dans ce domaine *verront-ils* vraiment le schéma de la même façon ? Rien n'est moins sûr.

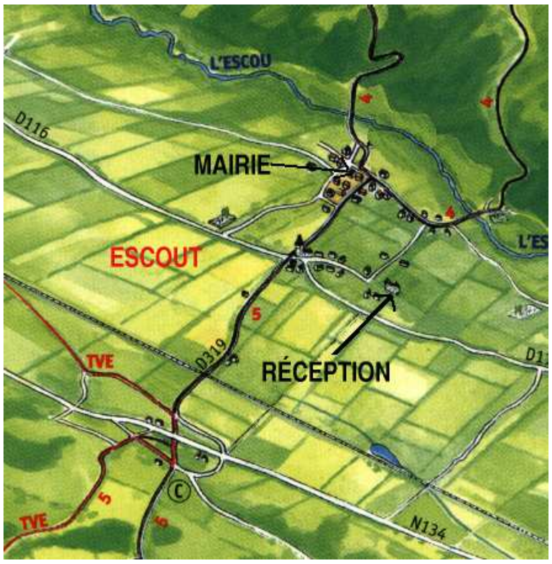


FIG. 1.1 – Plan routier

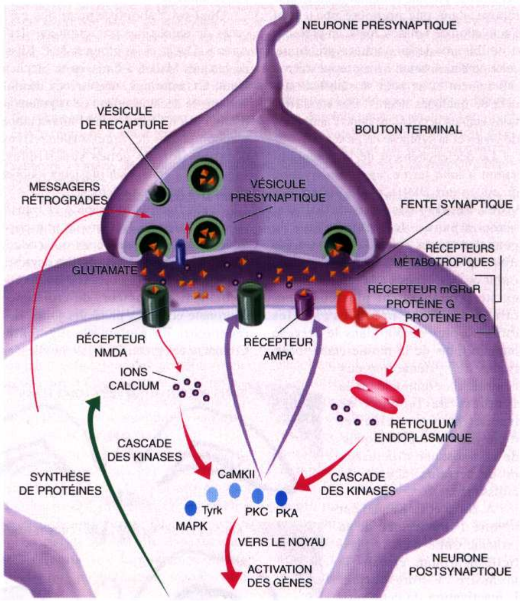


FIG. 1.2 – Schéma des processus chimiques se déroulant dans les neurones lors de la mémorisation [dossier Pour la Science 2001]

1.1.1 Quelques repères

La façon dont un être vivant perçoit son environnement dépend des spécificités inhérentes à son propre système perceptif. L'étude de la perception commence avec les premières réflexions philosophiques sur les représentations et la connaissance avec Platon, Descartes ou Kant. Elle s'est développée, par la suite, essentiellement en psychologie dans le but de comprendre comment les perceptions et les représentations s'établissent chez un individu [Bagot 1999, Francès 1992]. Plusieurs approches ont été développées. L'*introspection*, par exemple, consiste à soumettre un individu à une tâche et, immédiatement après, à lui demander de décrire ses états mentaux et les représentations subjectives par lesquelles il est passé pour réaliser cette tâche. À l'opposé, l'approche *béhavioriste*, popularisée par Watson, réfute l'introspection en établissant une méthode scientifique dans laquelle tous les processus mentaux (conscience, pensée, représentation, attention, etc.) n'ont pas de signification fonctionnelle. Seules les conduites observables sont étudiées [Gregory 1993].

Actuellement, le courant cognitiviste tend à unifier les nombreuses théories existantes. Il semble ainsi aujourd'hui très difficile de parler de la perception sans adopter la vision cognitiviste du *traitement de l'information* [Fortin et Rousseau 1997, Vignaux 1991, Varela 1996].

Le processus de perception est ainsi décomposé en plusieurs étapes. À chaque instant, l'organisme recueille une multitude d'informations. Ces dernières circulent dans le système nerveux sous forme de signaux électriques. Le système nerveux n'a pas seulement pour rôle de transmettre ces signaux, mais il doit aussi les traiter [Reuchlin 1993]. La perception peut ainsi être analysée en deux étapes interdépendantes. D'une part, les mécanismes de *recueil de l'information* permettent de prendre conscience de l'environnement ; d'autre part, les mécanismes de *traitement de l'information* conduisent à la *construction* des représentations mentales. La façon dont l'individu prend conscience de ces informations ou en prend inconsciemment connaissance fait cependant l'objet de nombreux débats [Searle 1996].

1.1.2 Constructions perceptives

> *«Mon problème, avec les classements, c'est qu'ils ne durent pas ; à peine ai-je fini de mettre de l'ordre que cet ordre est déjà caduc. Comme tout le monde, je suppose, je suis pris parfois de frénésie de rangement ; l'abondance des choses à ranger, la quasi-impossibilité de les distribuer selon des critères vraiment satisfaisants font que je n'en viens jamais à bout, que je m'arrête à des rangements provisoires et flous, à peine plus efficaces que l'anarchie initiale. Le résultat de tout cela aboutit à des catégories vraiment étranges ; par exemple, une chemise pleine de papiers divers et sur laquelle est écrit «à classer» ; ou bien un tiroir étiqueté «urgent 1» et ne contenant rien (dans le tiroir «urgent 2» il y a quelques vieilles photographies, dans le tiroir «urgent 3» des cahiers neufs). Bref, je me débrouille.»*
>
> — Georges Perec, *Penser/classer*, 1985

Exploration visuelle

Lorsqu'un objet (image, scène, etc.) apparaît dans le champ visuel, les yeux réagissent par une *réponse de fixation*. Chaque point de fixation fournit une information restreinte sur l'environnement, mais les mouvements rapides de l'œil permettent de couvrir une zone visuelle plus large. Les mécanismes de perception «raccordent» ensuite ces différents champs de vision afin de permettre à l'individu de percevoir un espace continu autour de lui. L'analyse de ces points de fixation a montré l'existence de mécanismes sélectifs fixant le regard sur les points de l'espace qui apportent le plus d'informations perceptives. La réponse de fixation de l'œil consiste à amener l'image de l'objet sur la fovéa, c'est-à-dire sur la région de la rétine où les cellules sensibles à la lumière sont les plus nombreuses

[Reuchlin 1993]. Pour simplifier, nous pourrions dire que dans une situation donnée, il y a un maximum d'informations pertinentes «regardées» par un maximum de récepteurs. Cela suggère que le système perceptif possède des stratégies d'exploration visuelle obéissant à certaines règles de priorité. Cependant la priorité des signaux peut aussi être influencée par l'objectif de l'individu à un moment donné.

Nous allons donc voir maintenant quels sont ces signaux afin de comprendre ce qui «attire l'œil» d'un observateur qui pourrait être, par exemple, devant une interface de visualisation.

Organisation des formes et des catégories

À tout moment, le système nerveux est soumis à bien plus de stimulations qu'il ne peut en traiter. Il s'agit aussi bien des stimulations externes reçues par les organes des sens que des stimulations internes telles que la survenue d'images ou d'idées [Gregory 1993]. Ainsi, à chaque instant, un individu n'est conscient que d'une infime partie de l'information qui arrive à son système nerveux. En effet, toute activité deviendrait vite impossible si chaque individu ne disposait pas d'une représentation relativement cohérente et stable de son environnement. Lorsqu'un sujet est face à des éléments désordonnés d'informations, il devient beaucoup plus difficile pour lui d'orienter et de contrôler son activité : c'est, par exemple, le cas d'un élève au fond d'une classe qui perçoit des informations issues de son instituteur, mais aussi toutes les actions et bavardages de ses camarades de devant, ainsi que des bruits divers provenant du couloir et de l'extérieur. Il est donc nécessaire qu'il existe une *activité organisatrice* qui s'exerce sur l'activité sensorielle.

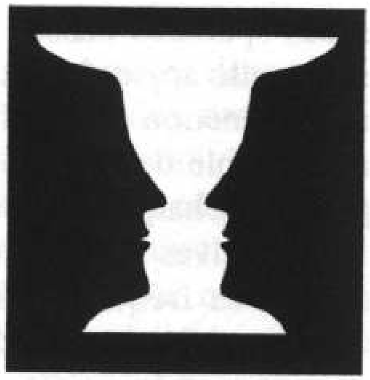


FIG. 1.3 – Figure ambigüe [Bagot 1999]

Les figures ambigües, par exemple, montrent qu'une information peut avoir plusieurs interprétations possibles (FIG. 1.3). Le système perceptif doit donc comporter un *système* *de décision* qui permette de faire un *choix* entre toutes ces interprétations. Cependant l'observateur peut avoir du mal à choisir de façon consciente entre deux interprétations, car, à un instant donné, une seule est accessible [Reuchlin 1993].

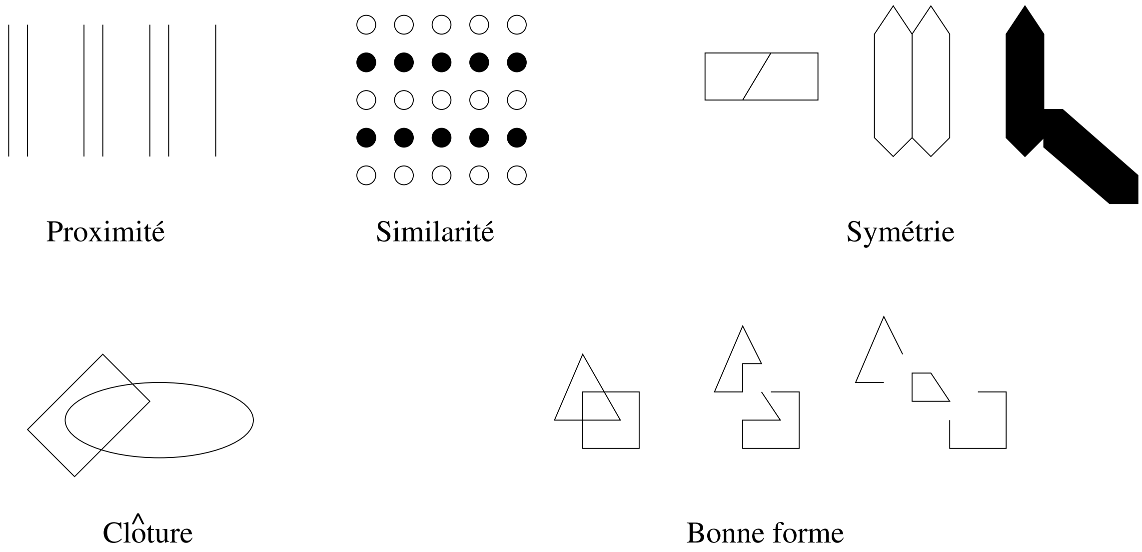


FIG. 1.4 – Organisation perceptive : lois de la Gestalt

La théorie de la *Gestalt*, ou théorie de la «forme» [Dortier 1999], naît au début du 20ème siècle, en Allemagne. D'après cette théorie, *«la perception que nous avons du monde n'est pas une somme d'éléments séparés»*, mais elle *«se constitue en ensembles organisés de* formes *globales qui donnent un sens à ce que nous voyons»* [Denis 1994]. Par exemple, lorsqu'un individu perçoit une mélodie, il ne perçoit pas chaque note ou chaque instrument, mais il perçoit un tout. Penser, c'est effectuer des tris et focaliser son attention sur certaines données, puis les mettre en forme et les assembler sous différentes modalités. Ainsi, pour cette théorie, une forme est une organisation qui ne peut se ramener à une juxtaposition d'éléments. Elle possède une qualité propre qui ne se trouve dans aucun des éléments constituants. La modification d'un seul de ces éléments peut modifier la forme entière. Les lois de la perception (FIG. 1.4) orientent l'activité de perception dans sa recherche de «bonnes formes», c'est-à-dire de formes possédant des propriétés de régularités, de simplicités, etc. [Doron et Parot 1991]. Les zones informatives d'une figure sont alors les frontières, les intersections de lignes, d'angles, etc. Ces zones ont été définies en analysant les points de fixation du regard lors de la présentation d'images [Reuchlin 1993]. Ainsi, le système perceptif visuel utilise ces zones pour organiser les informations qu'il reçoit.

Toute perception est donc une construction qui, en se reposant sur des mécanismes structurant les signaux extérieurs et en tenant compte des connaissances antérieures, permet d'établir des représentations mentales qui donnent alors une signification à l'environnement. Ainsi Vignaux [Vignaux 1999] définit la perception comme *«un processus de* sémantisation*, personnel et collectif, c'est-à-dire d'attributions en permanence de significations aux choses, aux situations, aux autres, et donc de catégorisation»*.

1.1.3 Processus cognitifs intervenant lors de la visualisation d'une interface

Placé devant une interface de visualisation, un utilisateur fait intervenir de nombreux processus cognitifs. La connaissance de ces processus et de leurs limites doit être prise en compte lors de la conception d'une interface. Il est important de considérer les points suivants :

- le rôle des représentations et des images mentales dans la compréhension et la mémorisation des informations ;
- les capacités limitées de perception et de mémorisation de l'utilisateur face à un trop grand nombre d'informations ;
- l'influence du niveau de vigilance et de motivation de l'utilisateur sur sa capacité à mémoriser et à comprendre l'information.

De la perception à la connaissance : rôle des représentations et des images mentales

Le terme de *représentation* désigne aussi bien le *processus* par lequel une information est assimilée que le *résultat de ce processus* [Denis 1994]. Ce processus correspond ainsi à la mise en correspondance de deux entités : le *représentant* répète, remplace ou «représente autrement» le *représenté* [Doron et Parot 1991]. L'existence et le support des représentations ont fait l'objet de nombreuses discussions philosophiques et épistémologiques [Désesquelles 2001], certains courants mettant l'accent sur le représentant et d'autres sur la dynamique du processus. De nombreux modèles ont ainsi été mis en œuvre afin d'expliquer les mécanismes structurels et fonctionnels des représentations. Il est possible de regrouper ces modèles en quatre catégories : les *modèles analogiques*, les *modèles propositionnels*, les *modèles procéduraux* et les *modèles à connaissances distribuées* [Doron et Parot 1991].

Dans le cadre des modèles analogiques, la théorie des modèles mentaux postule un isomorphisme entre les représentations et le monde représenté. Ce modèle permet de montrer l'importance du *support imagé* dans la perception, la compréhension et la mémorisation. Ainsi, la construction d'*images mentales* permet de répondre immédiatement à la question «quelle différence y a-t-il entre un croissant et une chocolatine ?». Le point pertinent qui nous intéresse ici est la relation étroite qui semble exister entre l'image mentale et la perception : en effet, l'image mentale semble jouer le même rôle d'identification, de structuration et de caractérisation de l'objet que la perception directe de l'objet. D'un côté, les expériences de Standing [Standing 1973] montrent une meilleure rétention de l'information avec des images qu'avec des mots. De l'autre, d'après Shepard [Shepard 1967], le risque d'erreur lors de la reconnaissance de matériel visuel est moins élevé que lors de la reconnaissance de matériel verbal. Ces modèles tendent donc à prouver l'importance de l'utilisation d'images (mentales ou non) lors de la perception, de la structuration et de la mémorisation de l'information.

Limites de la perception

De nombreuses théories ont été énoncées sur la mémoire [dossier Pour La Science 2001, Baddeley 1994, Reuchlin 1993]. Un des principaux modèles consiste à envisager la mémoire comme un système de stockage à trois niveaux. Dans un premier temps, le système de stockage de l'information sensorielle (ou mémoire immédiate) conserve une image assez précise de l'environnement tel qu'il est capté par les systèmes sensoriels (visuel, auditif, tactile, olfactif et gustatif). Ce système conserve ainsi une image moins d'une seconde. Puis, la mémoire à moyen terme (d'une capacité de stockage de quelques secondes) permet de choisir les informations pertinentes à un moment donné, d'en faire une interprétation, puis de synthétiser et de structurer les données issues de la mémoire immédiate. Ces deux premiers systèmes, souvent réunis sous le terme de mémoire de travail [Baddeley 1992] ou de mémoire à court terme [Atkinson et Shiffrin 1968], ont une capacité très limitée. Généralement, lorsqu'un sujet entend une liste de mots et doit les retrouver, il en mémorise en moyenne sept quelque soit la longueur initiale de la liste. La mémoire à long terme, quant à elle, permet de maintenir une organisation complexe des connaissances. C'est à ce niveau de mémorisation que le stockage et la récupération d'informations sont facilités par la compréhension et par la possibilité de rattacher l'information à d'autres connaissances préexistantes.

Afin de faciliter la compréhension et la mémorisation, plusieurs aides visuelles peuvent être fournies au sujet apprenant. Dans un premier temps, la structuration, la formation de catégories, la différentiation visuelle des données [Vignaux 1999] et, plus généralement, l'organisation des données présentées peuvent aider l'utilisateur dans sa tâche de perception et de mémorisation. Face à la limite de sa capacité perceptive, une solution est de présenter une information déjà groupée et synthétique des données. Ce regroupement est utilisé quotidiennement, par exemple, lorsqu'il s'agit de mémoriser un numéro de téléphone, l'apprentissage se faisant alors sur des nombres (des groupes de chiffres) et non sur chacun des chiffres. L'utilisateur mémorise alors les groupes de plus haut niveau ainsi que les nouvelles informations relatives à chacun de ces groupes. La perception visuelle peut aussi être rapidement guidée par des indices graphiques. C'est ce qui existe, par exemple avec l'utilisation de codes de couleur : rouge pour l'alerte et vert pour une situation normale.

Influence du niveau d'attention

Plusieurs facteurs propres au sujet peuvent améliorer sa perception et sa mémorisation [Reuchlin 1993]. Dans un premier temps, l'existence de liens entre les éléments facilite leur mémorisation. Ces liens peuvent être des liens phonétiques, des liens sémantiques ou des liens que le sujet confère lui-même aux éléments, c'est-à-dire, résultant d'une *activité de structuration* du sujet lui-même. Ehrlich [Reuchlin 1993] a demandé à des sujets d'apprendre des listes de mots ne présentant pas d'organisation préétablie. Au moment du rappel, il a constaté la formation de petits groupes de mots selon un principe qui peut être soit sémantique (cheval, charrue, paysan), soit phonétique (château, chapeau, chariot), soit grammatical (regarder, porter), etc. Puis, il a demandé aux sujets d'apprendre à nouveau la même liste de mots, mais donnée dans un ordre différent. Il a ainsi constaté que ces groupements s'enrichissent, puis se stabilisent : le principe de catégorisation adopté par un sujet (et non induit par les énoncés) a donc favorisé la mémorisation. Ainsi, l'engagement actif du sujet aide à la mémorisation, qu'il s'agisse d'une activité de structuration ou de toute autre activité telle que la récitation, la manipulation, l'explication à d'autres, etc.

Ce niveau d'activité est lié à l'*attention* du sujet. L'attention permet de sélectionner certaines informations : parmi les nombreux stimuli qu'un individu peut recevoir à un instant donné, un certain nombre seulement vont être prioritaires. Cela se traduit par une facilitation de leur perception, du choix et de l'exécution des réponses adéquates, par un traitement plus achevé ou encore par un accès à la conscience [Doron et Parot 1991]. D'autres stimuli, au contraire, vont être partiellement ou totalement ignorés. L'individu peut «jouer» sur son attention afin de se concentrer sur une information particulière. C'est ce qui se passe lors d'une réception où les invités, malgré le brouhaha, vont se concentrer sur telle ou telle conversation («effet cocktail»).

Pour finir, la *motivation* et la *familiarité* avec les données peuvent être considérées comme deux autres facteurs importants pouvant influencer la capacité de reconnaissance et de mémorisation. En effet pour qu'un sujet «passe à l'action», il faut que sa conduite ait un sens pour lui, *«être motivé, c'est d'abord pouvoir trouver un sens à son action»* [Mucchielli 2000]. Enfin, un niveau élevé de familiarité du sujet avec les données va améliorer la mémorisation. Il est généralement plus facile de mémoriser des listes de mots usuels que des mots inconnus.

La neuro-psychologie nous permet de connaître les aptitudes perceptives visuelles telles que la détection rapide de regroupements et d'organisation de données ou la reconnaissance rapide des objets connus. Ces aptitudes sont de plus en plus prises en compte lors de la conception d'interfaces graphiques et, en particulier, lorsqu'il s'agit d'alléger les charges cognitives d'un utilisateur devant un système de visualisation d'informations [Andrews 2000].

1.2 Introduction aux systèmes de visualisation d'informations

> *«It is the job of information visualization systems to set up visual representations of data so as to bring the properties of human perception to bear.»*
> — Stuart Card, [Card *et al.* 1999]

Un document textuel nécessite un effort important de compréhension et de mémorisation de la part d'un lecteur qui souhaite en retirer un certain contenu informatif. Assimiler de nouvelles informations, les comprendre et en extraire de nouvelles connaissances nécessitent souvent de pouvoir se les représenter visuellement. Du dessin d'enfant représentant sa famille aux villes virtuelles [Cremer *et al.* 2000], de nombreux artefacts existent pour nous permettre de visualiser ou de construire nos propres représentations mentales à partir de stimulations externes. La *visualisation d'informations* est l'un de ces artefacts.

1.2.1 Contexte

Terminologie

Card *et al.* [Card *et al.* 1999] définissent la *visualisation* comme l'utilisation de l'informatique pour représenter visuellement des données. Les représentations obtenues permettent à l'utilisateur d'interagir avec ces données et ainsi de mieux les comprendre. Ces auteurs distinguent la *visualisation d'informations* de la *visualisation scientifique*. La visualisation scientifique permet de représenter des données scientifiques, c'est-à-dire issues de données réelles ou de données de simulation. C'est dans cette catégorie que se placent les systèmes de représentation de molécules chimiques ou d'interactions physiques. La visualisation d'informations, quant à elle, est définie comme la visualisation de données «abstraites» et «non physiquement situées». Il peut, par exemple, s'agir de visualiser un ensemble de sites Web ou de données boursières. Les termes de données «abstraites» et «non physiquement situées» semblent malgré tout relativement discutables. En effet, il est possible de s'interroger sur la catégorie à laquelle appartient un système de visualisation permettant de représenter les stocks de livres d'une bibliothèque ou le contenu d'une boîte aux lettres électronique, même s'il est vrai que ces données n'ont pas une situation géographique précise qui permette de les situer les unes par rapport aux autres.

Nous utiliserons le terme de visualisation d'informations dans le sens de représentation de données permettant d'approfondir les connaissances sur un système complexe, en donnant à l'utilisateur des moyens d'interaction. En effet, la perception et la compréhension des données proviennent d'une part de leur représentation *visuelle* et *structurée* et d'autre part de la possibilité d'*interagir* avec elles et de les *manipuler* dynamiquement.

Comme le montrent les études psychologiques, plus la représentation de ces données est organisée et structurée à l'écran, plus l'utilisateur peut les interpréter facilement. De plus, les mécanismes d'interaction permettent à l'utilisateur de *s'impliquer* dans le système en demandant, par exemple, des changements d'angles de vue. Dès que les données sont visualisées sur un écran, elles acquièrent une *position topologique* les unes par rapport aux autres. Ces positions relatives peuvent intervenir lors de la construction de la représentation mentale, la proximité sur l'écran étant alors traduite en proximité sémantique ou relationnelle. Enfin, l'informatique offre la possibilité d'exploiter un très grand nombre de données tout en réduisant la charge de travail de l'utilisateur puisqu'il peut confier les opérations de prétraitement des données à la machine.

Cependant, rendre une information *lisible*, *compréhensible* et *accessible* sur un écran d'ordinateur n'est pas si évident que cela, pour preuve la profusion d'informations qu'il peut y avoir sur une seule page web [Shneiderman 1997]. Cette profusion ne tient souvent compte ni de la taille de la fenêtre de visualisation limitée des terminaux, ni des capacités limitées du système visuel humain [Furnas 1981]. Ainsi, le domaine de la visualisation d'informations est un domaine de recherche à part entière qu'il est nécessaire de décomposer en plusieurs axes de recherche.

Axes de recherche dans le domaine de la visualisation d'informations

La classification des systèmes de visualisation peut se faire selon plusieurs points de vue : selon le type de données traitées (numériques, symboliques, dynamiques, multi-critères, etc.), selon la structure des données (information linéaire, hiérarchique, en réseaux, par attributs, etc.) [Andrews 2000] ou selon le type de représentation (tableau, arbre, graphe, cartographie, info-sphère, etc.) [Card *et al.* 1999]. Sans vouloir faire un état de l'art exhaustif des systèmes de visualisation suivant une de ces classifications, les sections suivantes illustrent les techniques ayant servi de sources d'inspiration et de points de comparaison avec la conception d'interfaces multi-agents.

Les techniques de visualisation d'informations se décomposent en quatre axes de recherche : les techniques de *prétraitement des données* (techniques d'analyse factorielle [Harman 1967], d'analyse en composantes principales [Dunn et Everitt 1982], de segmentation, etc.), les techniques d'*exploration visuelle des données*, les techniques de *distorsion* et les techniques d'*interaction* [Keim 2000]. Après une brève section sur les moyens graphiques de représenter une donnée, les deux dernières parties de ce chapitre seront consacrées l'une à l'exploration visuelle des données et l'autre aux techniques de distorsion et d'interaction.

1.2.2 Icônes, symboles, représentation des attributs

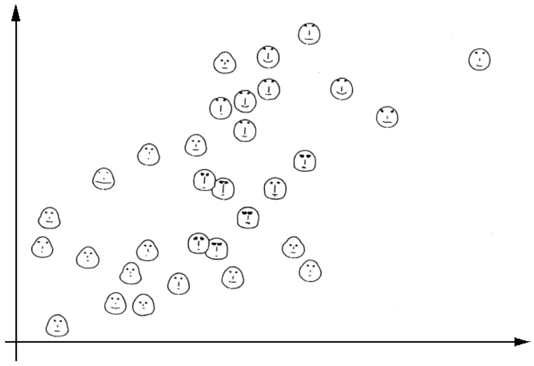


FIG. 1.5 – Faces de Chernoff [Chernoff 1973]

La plupart des systèmes représentent les données individuelles sous forme de rectangle, de triangle ou d'un texte indiquant un ou plusieurs mots-clés. Chernoff [Chernoff 1973, Tufte 1983] s'est intéressé à l'utilisation d'icônes pour représenter des informations. Les figures de Chernoff (FIG. 1.5) visualisent les valeurs des données comme des caractéristiques d'icônes. Chaque donnée correspond à un visage. La forme du nez, de la bouche, des yeux et du visage lui-même correspond à un attribut de la donnée. Le principal avantage de cette méthode est que l'utilisation d'icônes variés (mais pas trop nombreux) peut rendre une interface plus conviviale. En revanche, le principal inconvénient repose sur le nombre limité d'attributs que peut avoir une valeur. Ainsi, la proximité graphique des figures risque de ne plus permettre à l'utilisateur de différencier les attributs. L'utilisation d'icônes et de symboles variés s'est particulièrement développée avec l'introduction des bureaux virtuels comme environnement de travail sur les ordinateurs. L'utilisation de la métaphore du bureau pour représenter l'environnement de travail de l'utilisateur montre la pertinence de l'utilisation d'icônes particuliers pour représenter certaines informations, comme c'est le cas des icônes de la corbeille, des dossiers ou des icônes chargés de représenter un logiciel particulier. Une légère modification de cet icône peut alors représenter une variation de l'état de l'objet représenté, comme par exemple, l'utilisation d'icônes différents lorsque la «poubelle» est «vide» ou «pleine».

1.3 Exploration des données dans les systèmes de visualisation

L'*exploration des données* a été définie par Keim [Keim 2000] comme un processus de recherche et d'analyse de bases de données afin d'en extraire une information implicite et potentiellement «utile». L'extraction d'informations implique la diminution du nombre initial de données, ce qui est avantageux pour la machine en terme de temps de calcul et pour l'homme en terme de charge visuelle. La notion d'information *utile*, quant à elle, sous-entend la prise en compte de l'intérêt de l'utilisateur à un moment donné et dans un contexte particulier. Elle implique donc la nécessité pour l'utilisateur de pouvoir exprimer des changements d'intérêts ou des requêtes, en résumé de pouvoir *interagir* avec le système. Les techniques de distorsion permettant par exemple de zoomer sur une partie d'un environnement et les techniques d'interaction donnant par exemple la possibilité à l'utilisateur de poser des requêtes, ne peuvent être mise en place que lorsque la première étape d'exploration a été définie.

1.3.1 Cartographie de données

Plusieurs systèmes de visualisation ont été développés en utilisant la métaphore du paysage en deux ou trois dimensions. Cette métaphore offre un double avantage pour l'utilisateur. D'une part, la proximité entre les données dans l'environnement virtuel est facilement interprétée comme la proximité des caractéristiques des données réelles. D'autre part, ces systèmes proposent souvent la possibilité de se «promener» dans le paysage, ce qui revient à explorer l'ensemble des données.

Landscape

Wise et *al.* [Wise *et al.* 1995] présentent un paysage en perspective permettant de visualiser des ensembles d'articles ou de news (Fig. 1.6). Toute la difficulté de ce type

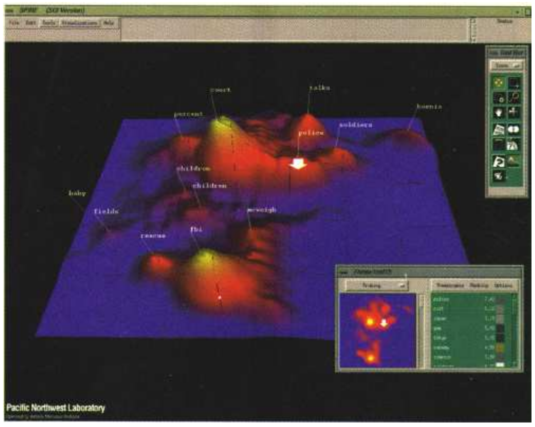


FIG. 1.6 – Landscapes [Wise *et al.* 1995]

d'application réside dans la transposition des données dans une représentation spatiale en trois dimensions, «possible mais artificielle » [Wise *et al.* 1995], qui préserve les caractéristiques des données.

Chalmer [Chalmers 1993] a étendu cette approche à un environnement de dimension n. Après avoir extrait les mots du corpus de documents, chaque mot est associé à une dimension. Un document est ensuite positionné dans cet environnement selon des coordonnées correspondant à la fréquence d'occurrences de chaque mot.

WEBSOM

WEBSOM [Honkela *et al.* 1997] est très proche par son aspect visuel des systèmes précédents (FIG. 1.7). Il offre lui aussi des cartes représentant un ensemble de documents ou de sites web. Les mécanismes sous-jacents de prétraitement sont cependant différents puisqu'ils reposent sur les cartes auto-organisatrices de Kohonen [Kohonen 1995].

La construction de la carte se fait en quatre étapes :

- codage du document par la construction d'un histogramme de mots ;
- prétraitement des entrées ;
- formation de cartes par catégories de mots ;
- formation de cartes des documents.

L'intérêt de ces cartes de visualisation, qu'il s'agisse de Landscapes ou de WEBSOM, est de définir des *zones d'attraction* pour les données dans certaines parties de l'écran. De la même façon qu'un utilisateur d'ordinateur a l'habitude d'organiser son bureau virtuel en zones consacrées à telle ou telle tâche, ces interfaces permettent d'organiser l'écran en «zones sémantiques». L'utilisateur peut se déplacer dans cet environnement et focaliser son

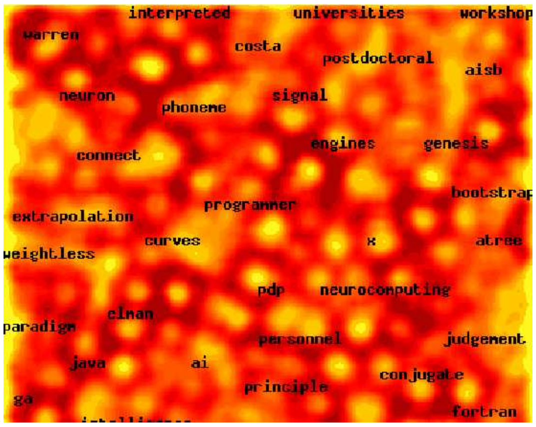


FIG. 1.7 – WEBSOM [Honkela *et al.* 1997]

attention sur une zone. Cependant, ces systèmes n'offrent généralement pas la possibilité à l'utilisateur de réorganiser lui-même ces zones les unes par rapport aux autres selon ses propres critères.

1.3.2 Représentation de l'organisation hiérarchique des données

Les *treemaps* et les *arbres coniques («cone trees»)* sont deux catégories de systèmes de visualisation permettant de représenter et de parcourir des données organisées selon une structure hiérarchique.

Treemaps

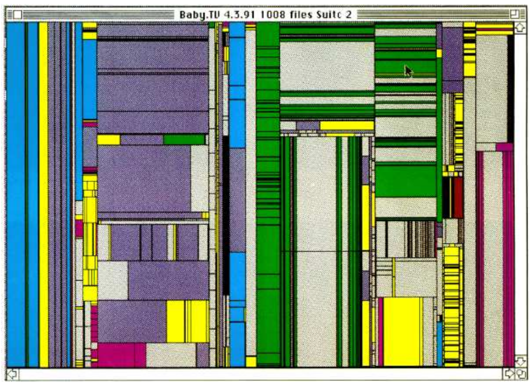


FIG. 1.8 – Treemaps [Shneiderman 1992]

Le concept de treemaps a été développé essentiellement par B. Shneiderman et son équipe [Johnson et Shneiderman 1991, Johnson 1993]. Shneiderman définit la notion de treemap comme *«the notion of turning a tree into a planar space-filling map»*. Les treemaps ressemblent au premier abord à une cartographie. La différence majeure avec la cartographie est la possibilité de visualiser des données qui se trouvent à des niveaux hiérarchiques différents (FIG. 1.8). L'écran est divisé en plusieurs régions selon différents attributs, la taille de chaque région étant alors proportionnelle à son *poids*. Le calcul du poids est variable selon le type d'applications : il peut correspondre, par exemple, aux nombres d'attributs ou à la somme des valeurs de certains d'entre eux. La couleur de chaque région peut, elle aussi, correspondre à un élément supplémentaire.

Plusieurs applications ont été développées à partir de ce modèle telles que TreeViz [Shneiderman 1992] et SmartMoney [Wattenberg 1999]. TreeViz permet de visualiser les fichiers contenus sur le disque dur d'un ordinateur. Il offre la possibilité de supprimer des fichiers choisis ce qui confère une certaine dynamique au niveau de la réorganisation de la représentation visuelle. SmartMoney, quant à lui, est un logiciel de visualisation de données boursières. Il comporte des fonctionnalités supplémentaires par rapport aux systèmes précédents comme par exemple la diminution du nombre de données représentées ou le regroupement de certaines informations.

Arbres coniques

Afin de visualiser des données hiérarchiquement organisées, des travaux sur leur représentation en trois dimensions ont été développés dans les années 1990. Les arbres coniques, par exemple, représentent la structure hiérarchique des données sous forme d'un arbre où chaque ensemble de fils est disposé dans un cône (FIG. 1.9). Comme pour les treemaps, plusieurs applications ont été conçues autour de la visualisation de l'arborescence des fichiers sur un disque dur [Robertson *et al.* 1991, Carrière et Kazman 1995, Jeong et Pang 1998]. Cependant, lorsque les données deviennent trop nombreuses et que la profondeur de la hiérarchie devient très importante, ce type de représentation devient difficilement interprétable et il est alors nécessaire d'introduire des mécanismes d'élagage d'arbres ou de zoom. Avec Cheops (FIG. 1.10), les auteurs [Beaudoin *et al.* 1996] proposent une représentation alternative de l'arbre, plus compacte et en deux dimensions seulement. Le principe de base est de regrouper l'information grâce à une technique d'empilement de triangles afin de synthétiser l'information représentée à l'écran.

L'un des principaux inconvénients de ces techniques repose sur leur limite lorsque le nombre de données ou de niveaux augmente. Il est donc nécessaire de recourir à des «subterfuges» afin d'optimiser ces interfaces. Ces méthodes consistent à ne visualiser qu'une partie de l'information jugée «pertinente» pour un utilisateur à un moment donné. Des techniques dynamiques d'interaction doivent donc être mises en place pour permettre

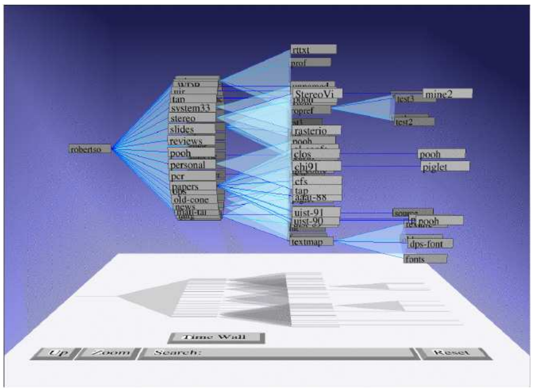


FIG. 1.9 – Cone Tree [Robertson *et al.* 1991]

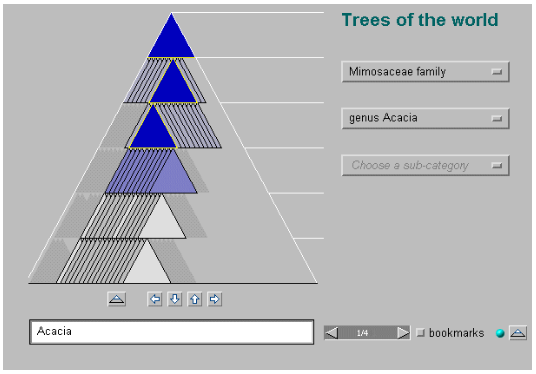


FIG. 1.10 – Cheops [Beaudoin *et al.* 1996]

à l'utilisateur de signifier au système ses centres d'intérêt.

1.4 Interaction homme-machine

L'intérêt d'une représentation à l'écran par rapport à une représentation fixe sur un support papier repose sur la possibilité qu'a l'utilisateur de manipuler dynamiquement les données, de changer les paramètres visibles, de changer l'angle de vue de la représentation ou de demander des informations supplémentaires au système.

Nous regroupons ici sous le terme «interaction homme-machine» les techniques de distorsion et les techniques d'interaction proprement dites. Cependant, ces techniques sont généralement distinguées l'une de l'autre. Les techniques de distorsion sont utilisées pour concentrer l'attention de l'utilisateur sur une zone particulière de l'écran. Les techniques d'interaction, quant à elles, sont mises en œuvre pour répondre à une requête de l'utilisateur sur un ensemble de données particulières à un instant précis. Dans les deux cas, ces techniques permettent de ne visualiser qu'une partie de l'information tout en conservant le contexte global. Elles ont ainsi pour but de faciliter la recherche d'informations, leur interprétation et leur mise en relation avec d'autres données [Leung et Apperley 1994]. La technique de *fisheye* a été développée à l'origine en tant que technique de distorsion. Comme nous allons le voir, elle peut aussi être utilisée dans le cadre de l'interaction. C'est ce que nous ferons en l'utilisant comme moyen d'interaction entre nos agents et l'utilisateur. C'est pour cette raison que nous avons choisi de regrouper dans une même partie l'ensemble de ces techniques.

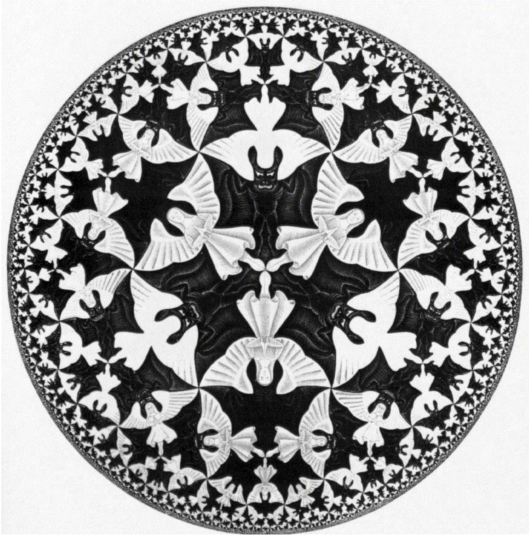


FIG. 1.11 – Circle Limit IV (Heaven and Hell) de Escher [Locher 2000]

1.4.1 Techniques de distorsion

Nous avons rappelé, dans les principes d'organisation perceptive, l'importance des points de fixation du regard et le nombre important de récepteurs visuels qui sont attribués au traitement des zones de fixation. Mais ce n'est pas pour autant que notre perception périphérique ne nous renvoie pas d'informations, bien au contraire. La vision périphérique nous permet de garder en mémoire le contexte global dans lequel nous nous trouvons. C'est ce contexte global qui va orienter notre perception et notre compréhension de l'information locale. Lorsque l'utilisateur concentre son attention sur un point particulier de l'écran, l'interface devrait lui permettre de zoomer visuellement sur ce point sans pour autant supprimer les informations environnantes [Beaudoin *et al.* 1996]. En effet, si le contexte n'est plus affiché à l'écran, il est nécessaire que l'utilisateur le mémorise afin de décoder correctement l'information locale. Cette mémorisation représente une surcharge de travail pour l'utilisateur qui peut nuire à sa (vitesse de) compréhension.

Les techniques de distorsion [Leung et Apperley 1994] reposent donc sur des transformations visuelles de l'information qui permettent de modifier la structure visuelle afin de créer un focus tout en maintenant une vue contextuelle.

Techniques «focus+contexte»

La visualisation d'un large ensemble de données pose de nombreux problèmes dès lors que la signification d'un élément local dérive de l'ensemble de la structure des données. Ceci est le cas par exemple, lorsqu'il s'agit de visualiser un texte relativement long. Afin que l'utilisateur puisse se positionner dans l'ensemble du texte, le curseur dans l'ascenseur indique, par sa position, la position de la zone de texte et, par sa taille, la taille de la zone relativement à l'ensemble du texte. Mais cette information ne suffit pas toujours à l'utilisateur car des informations-clés, telles que le titre du paragraphe, ne sont pas toujours connues. Face à ce problème, d'autres techniques ont été expérimentées comme l'utilisation de plusieurs fenêtres pour représenter séparément la vue locale et la vue globale, mais cela nécessite beaucoup de place sur un écran et demande à l'utilisateur d'intégrer lui-même les deux vues [Sarkar et Brown 1992].

Plusieurs techniques de distorsion ont donc été mises en œuvre afin de permettre à l'utilisateur d'examiner une zone locale en détail, sur une section de l'écran, et au même moment, de présenter une vue globale de l'espace en lui fournissant un contexte général permettant de faciliter la navigation. Les principes les plus couramment employés sont les suivants [Schaffer *et al.* 1996] :

- une distorsion de l'environnement : les éléments sont de moins en moins importants lorsqu'ils sont éloignés du point de focus ;
- un filtrage de l'information : une fonction de «distance» permet de déterminer si les éléments d'information doivent être présents ou non dans la visualisation et induit ainsi des vues partielles filtrées des données ;
- l'utilisation de représentation ou d'abstraction permet de représenter un groupe de données (par exemple un circuit détaillé dans un graphe) par un icône.

Ces principes sont implémentés différemment selon les auteurs. Les techniques les plus répandues sont les suivantes :

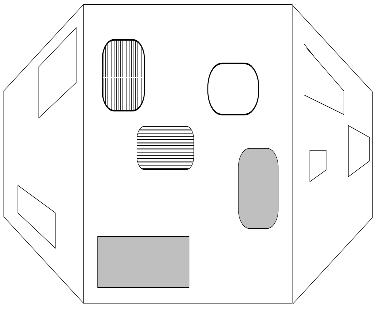


FIG. 1.12 – Mur perspectif [Mackinlay *et al.* 1991]

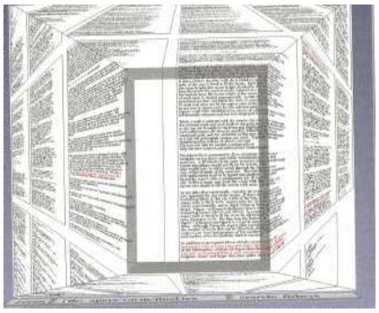


FIG. 1.13 – Document Lens [Robertson *et al.* 1993]

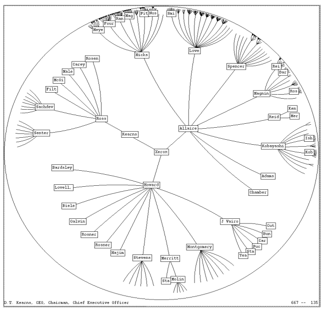


FIG. 1.14 – Browsers et arbres hyperboliques [Lamping *et al.* 1995]

- le *Mur Perspectif* [Mackinlay *et al.* 1991] (FIG. 1.12) présente une information, pouvant être structurée linéairement, en trois parties : une partie centrale visualise la zone pertinente pour l'utilisateur et deux parties périphériques, généralement à gauche et à droite, illustrent un mur «fuyant» en trois dimensions ;

- les systèmes à base de *loupe* ont fait l'objet de nombreuses recherches [Rao et Card 1994, Robertson et Mackinlay 1993]. «Document Lens» (FIG. 1.13), par exemple, permet à l'utilisateur de focaliser son attention sur une partie d'un texte grâce à une loupe rectangulaire. En dehors du rectangle de focus, l'ensemble du document est visualisé de façon «élastique» afin de garder le contexte dans lequel se trouve la partie de texte zoomée [Robertson et Mackinlay 1993] ;
- les *arbres et les browsers hyperboliques* (FIG. 1.14) proposent une approche alternative à la visualisation conventionnelle d'arbres et de hiérarchies. Un nœud de l'arbre (le centre d'intérêt) est initialement représenté au centre d'un cercle dans lequel les différentes branches sont réparties. Ces techniques peuvent être utilisées afin de chercher des informations dans des pages Web, des mots-clés permettent alors de structurer ces pages et servent d'indice pour se déplacer dans l'arborescence [Lamping *et al.* 1995, Munzner 1997] ;
- le concept de *«Fisheye views»* permet de déformer la représentation à partir d'un élément d'information servant de point de focus. Nous développons ce concept plus en détail dans la section suivante car nous l'utiliserons pour la conception de notre interface multi-agent.

Principes des «Fisheye Views» graphiques

Le concept générique de *Fisheye Views* (FIG. 1.15) a été initialement décrit par Furnas [Furnas 1981, Furnas 1986]. Il permet de représenter la structure globale des données (textes structurés, réseaux, etc.) tout en privilégiant la visualisation d'une information particulière. Le *point de focus* correspond à l'élément d'information de l'écran sur lequel se focalise l'utilisateur. Le principe du *Fisheye Views* consiste à déformer la représentation des autres éléments d'information environnants afin de mettre en évidence les informations ayant un intérêt par rapport au point de focus tout en visualisant en périphérie les points ayant moins d'intérêt. Pour chaque élément d'information, il est possible de mesurer son *degré d'intérêt* : cette valeur prend en compte d'une part la pertinence *a priori* des éléments d'information dans l'ensemble de la structure et d'autre part la distance entre ces éléments et le point de focus en cours.

De nombreuses implémentations de ces principes ont été proposées [Schaffer *et al.* 1996, Leung et Apperley 1994], elles diffèrent aussi bien par leurs applications que par les résultats obtenus. Sarkar et Brown [Sarkar et Brown 1992] ont étendu le concept de Furnas avec un formalisme mathématique pour les applications graphiques. Les détails de cette implémentation sont donnés en Annexe A.

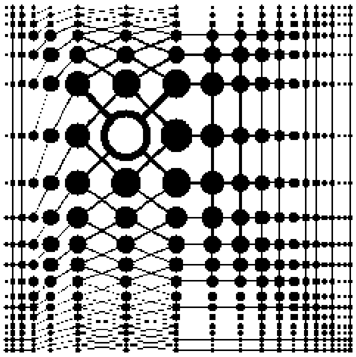


FIG. 1.15 – Fisheye Views [Sarkar et Brown 1992]

1.4.2 Techniques d'interaction

La génération dynamique de représentations visuelles et la possibilité d'interagir avec ces représentations permettent d'exploiter plus en profondeur les données [Keim 2000]. Les études psychologiques sur la mémorisation et l'attention montrent que l'engagement actif de l'individu impliqué dans une tâche augmente ses capacités cognitives et sa mémorisation. Ainsi, la possibilité d'interagir avec les données permet, d'une part, de modifier la façon dont nous comprenons ces données en les manipulant et, d'autre part, d'explorer plusieurs ensembles de données rapidement (présentation sous différents angles, etc.). Enfin cette interaction peut aider à comprendre l'influence de certains paramètres sur le système global.

Il existe de nombreuses techniques d'interaction [Card *et al.* 1999, Keim 2000] parmi lesquelles nous pouvons citer les plus connues :

- le «data-to-visualization mapping» : il s'agit d'établir une carte dynamique entre les attributs des données (valeur, seuil, etc.) et des paramètres de visualisation (axes, couleur, taille, icône, etc.) [Ahlberg et Wistrand 1995a, Ahlberg et Wistrand 1995b] ;
- les techniques de *filtrage* [Ahlberg *et al.* 1992, Young et Shneiderman 1993] consistent à déterminer dynamiquement des sous-ensembles de données soit par *sélection* directe d'un sous-ensemble désiré, soit grâce à des *requêtes* qui permettent de spécifier les propriétés de l'ensemble désiré ;
- les techniques de *zoom* permettent de visualiser un large ensemble de données en les réduisant (resp. en les augmentant) afin de fournir une vision plus large de ces données et de leur contexte (resp. fournir plus de détails sur ces données) [Keim 2000]. Les principes mis en œuvre avec les «Fisheye views» peuvent aussi être utilisés comme techniques de zoom [Schaffer *et al.* 1993] ;

- le *détail à la demande* consiste à obtenir plus d'informations sur une donnée particulière, par exemple en cliquant dessus.

1.5 Synthèse des propriétés idéales d'un système de visualisation

L'état de l'art développé jusqu'à présent permet d'aboutir à une synthèse des propriétés idéales que devrait posséder une interface de visualisation d'informations.

- PROPRIÉTÉ 1 : Les premières contraintes imposées à l'interface proviennent des données à visualiser. En effet, le système doit avoir la possibilité de prendre en charge et de représenter des données nombreuses ayant des caractéristiques parfois **hétérogènes**. De plus, les sources d'informations dont proviennent ces données peuvent être **distribuées**.

- PROPRIÉTÉ 2 : Il est nécessaire que le système puisse aider à la construction perceptive, d'une part en diminuant la charge de mémorisation de l'utilisateur et d'autre part en utilisant des représentations en accord avec les mécanismes sélectifs et les stratégies d'exploration visuelle. C'est le cas des systèmes de cartographies d'informations dans lesquels la représentation spatiale des données induit une interprétation de proximité des caractéristiques de ces données. Cette interprétation correspond ainsi à la loi d'organisation perceptive de proximité de la Gestalt. Des mécanismes de **structuration et d'organisation visuelle dynamique de l'information** peuvent donc permettre d'orienter la perception de l'utilisateur et de faciliter ainsi sa compréhension des informations visualisées.

- PROPRIÉTÉ 3 : De même, la mémorisation est facilitée lorsque les informations sont structurées et organisées. Cependant, l'utilisateur a une capacité perceptive limitée et il ne pourra pas appréhender un trop grand nombre d'informations présentées de façon simultanée. Il est donc nécessaire que l'interface de visualisation puisse aussi lui fournir une **synthèse visuelle dynamique de l'information**. L'organisation et la synthèse peuvent ainsi conduire à des mécanismes de création de **hiérarchies** visuelles de l'information, chaque niveau de cette hiérarchie représentant alors un degré de détail.

- PROPRIÉTÉ 4 : De plus, quelles que soient les données, leurs caractéristiques peuvent varier de façon plus ou moins importante dans le temps. Face à cette dynamique, l'interface doit être suffisamment **autonome** pour s'adapter d'elle-même sans que l'utilisateur ait l'obligation de la remettre constamment à jour. La dynamique peut aussi provenir de changements dans les centres d'intérêts de l'utilisateur, ce qui nécessite que l'interface puisse **s'adapter** à l'utilisateur et **apprendre** à partir des actions de ce dernier.

– Propriété 5 : Enfin, l'interactivité et la manipulation de l'interface éveillent l'attention et l'intérêt de l'utilisateur. Différentes possibilités d'**interaction** doivent donc être mises en œuvre. L'utilisateur doit ainsi avoir la possibilité de parcourir l'information en changeant d'angle de vue ou en émettant des requêtes. Par exemple, lorsque l'interface lui présente une information synthétique, l'utilisateur doit avoir la possibilité d'obtenir les informations ayant permis d'aboutir à cette synthèse, tout en conservant le contexte global de ces données. Comme le montre l'organisation des «bureaux virtuels» de chaque personne, la personnalisation de l'interface est un élément important pour l'utilisateur. Cependant, actuellement, peu de systèmes de visualisation peuvent **s'adapter à une réorganisation dynamique** de l'information par l'utilisateur, tel qu'il peut le faire avec son bureau. Les systèmes de cartographie d'informations, bien que très proches de la notion de bureau, ne possèdent généralement pas cette caractéristique.

Comme nous le montrons dans le chapitre suivant, les systèmes multi-agents semblent posséder les propriétés nécessaires à un système idéal de visualisation d'informations. Si nous partons de l'hypothèse que chaque agent représente une donnée ou un ensemble de données, il devient alors envisageable de disposer des caractéristiques des systèmes multi-agents dans une application de visualisation de données. Cette application pourrait ainsi bénéficier des propriétés suivantes :

– possibilité d'interactions entre des agents-données hétérogènes : différents types d'agents peuvent interagir entre eux et avec l'utilisateur ;
– distribution du traitement des données dans un environnement ne possédant pas de contrôle centralisé ;
– mise en œuvre autonome de processus dynamiques d'organisation et de hiérarchisation afin de structurer, en temps réel, la représentation visuelle de l'information, les regroupements d'agents correspondant alors à des regroupements d'éléments d'information ;
– possibilité pour les agents-données de posséder des mécanismes adaptatifs et ainsi de s'adapter aux variations de leur environnement, que ces changements soient issues de variations dans les flux de données ou de changement d'intérêt de l'utilisateur.

Il est donc nécessaire que nous nous intéressions maintenant plus en détail aux caractéristiques des systèmes multi-agents.

Chapitre 2

Systèmes multi-agents

Sommaire

L'objectif de ce chapitre est de présenter la diversité des entités informatiques qui se cachent derrière le terme «agent» afin de positionner nos travaux de recherche. Nous introduisons les différences qui existent entre l'«intelligence» d'un agent et l'«intelligence» de l'ensemble d'un système multi-agent. Pour cela, nous présentons les caractéristiques principales de ces systèmes et en particulier nous montrons l'importance du rôle de l'environnement et des interactions entre les agents. Nous terminons en présentant des exemples de travaux développés dans le cadre d'agents autonomes d'interface.

2.1 Contexte

2.1.1 Intelligence Artificielle et Intelligence Artificielle Distribuée

> *«Est intelligente une machine qui fait illusion et passe pour intelligente aux yeux des hommes.»*
>
> — Alan M. Turing, *Machine à calculer et intelligence*, 1950

Au début des années 1970, les recherches en Intelligence Artificielle ont essentiellement porté sur les systèmes experts et les systèmes à base de connaissances. Par exemple, le système MYCIN [Shortliffe 1976] a été un succès dans le domaine des diagnostics médicaux des maladies infectieuses. L'expression des connaissances du domaine sous forme de règle de production en a fait un des premiers systèmes experts. Ces systèmes reposent sur la manipulation de représentations symboliques du monde. Cependant, la difficulté à énoncer toutes les connaissances sous forme symbolique impose des limites sur l'adaptabilité de ces systèmes face à de nouveaux faits ou à de nouveaux domaines [Brooks 1991].

Une alternative à ces systèmes, considérés comme experts dès leur apparition, repose sur des «systèmes-enfants» doués de capacités d'apprentissage et d'adaptation face à leur environnement. Ces systèmes sont issus d'une démarche «bottom-up» dans laquelle le système, à l'origine «naïf», va se construire peu à peu au travers des interactions avec son environnement, de même que l'intelligence individuelle humaine se développe grâce aux interactions avec les autres. De plus, les systèmes informatiques devenant de plus en plus complexes, il est devenu nécessaire de les décomposer en unités relativement indépendantes qui peuvent être facilement manipulées et contrôlées. Les notions de «système distribué», d'«objet» puis d'«agent» apparaissent alors avec le développement de l'Intelligence Artificielle Distribuée. Parallèlement au développement de ces notions, le domaine de la Vie Artificielle est apparu.

2.1.2 Vie Artificielle

Langton [Langton 1994] définit la Vie Artificielle comme *«l'étude de la vie telle qu'elle pourrait être, et non de la vie telle qu'elle est»*. Ferber [Ferber 1995] décompose ce domaine en quatre grands thèmes de recherches :

- l'analyse de la dynamique des phénomènes complexes à l'aide d'automates cellulaires ou d'équations différentielles non linéaires (Wolfram [Wolfram 1986], etc.) ;
- l'évolution de populations par utilisation d'algorithmes génétiques (Koza [Koza 1991], etc.) ;
- la réalisation d'animats : créatures autonomes capables d'agir et donc de survivre dans un environnement non entièrement spécifié (Brooks [Brooks 1991], etc.) ;
- l'étude de phénomènes naturels collectifs à partir de l'interaction d'un ensemble d'agents réactifs (Steels [Steels 1991], Deneubourg [Deneubourg *et al.* 1992], Ferber [Ferber 1995], Drogoul [Drogoul 1993], etc.).

Bien que le terme d'«agent» y soit peu employé, une grande partie des recherches autour des systèmes multi-agents est effectuée dans le domaine de la Vie Artificielle.

2.2 Des agents aux systèmes multi-agents

2.2.1 Définition

Peu de compromis existent lorsqu'il s'agit de définir le terme «agent». Ce terme, pris hors d'un contexte, peut s'appliquer à presque n'importe quelle entité. C'est pourquoi les définitions sont le plus souvent données en fonction de principes qui sous-tendent le comportement d'un agent : agent *rationnel*, agent *autonome*, etc. Citons quelques exemples de définitions :

> *«La société de l'esprit est un système selon lequel chaque esprit est composé d'un grand nombre de petits processus appelés* ***agents****. Chaque agent ne peut, à lui seul, effectuer que quelques tâches simples ne demandant ni esprit ni réflexion. Le regroupement de ces agents en sociétés - selon des modalités bien particulières - peut aboutir à la véritable intelligence.»*
>
> — Marvin Minsky, *The Society of Mind*, 1988

> *«Un* ***agent*** *est tout ce qui peut être vu comme percevant son environnement au travers de capteurs et agissant sur son environnement au travers d'effecteurs.»*
>
> — Russell, [Russell et Norving 1995]

Que l'agent soit virtuel, réel, cognitif, réactif, intelligent, etc., ces définitions se rejoignent sur le rôle *actif* de l'agent dans *son environnement*. Ceci implique que l'agent possède certaines caractéristiques. D'une part, il doit avoir des *récepteurs adaptés* afin de *percevoir* cet environnement même de manière limitée. D'autre part, il doit pouvoir y *agir* grâce à ses *effecteurs*. Enfin, il doit posséder une certaine *autonomie* dans le choix de ses actions (déplacer des objets, communiquer avec les autres, etc.). L'agent ne peut ainsi pas se définir en dehors de son environnement.

À noter que nous adoptons ici un point de vue *individu centré* selon lequel tout ce qui n'est pas l'agent lui-même fait partie intégrante de l'environnement, c'est-à-dire que, pour un agent donné, les autres agents font partie de son environnement.

2.2.2 Agents cognitifs versus agents réactifs

Nous pouvons dégager deux positions extrêmes de conception de systèmes multi-agents [Ferber 1995, Mephu-Nguifo 1993].

La première repose sur les notions d'*agents intelligents*, d'*agents rationnels* [Russell et Norving 1995] ou d'*agents cognitifs*. Ces agents disposent d'un certain nombre de croyances, de connaissances et de savoir-faire sur leur environnement qu'ils peuvent se communiquer. L'intelligence du système provient de l'interaction des «intelligences» des agents, chaque agent étant souvent à lui seul l'équivalent d'un système expert indépendant. Ces systèmes possèdent généralement un nombre peu important d'agents (et parfois même un seul) capables de résoudre seuls certains problèmes. Cette école *cognitive* repose sur des analogies sociales, notamment avec les travaux de Gasser [Gasser et Huhns 1989]. Elle est l'héritière directe de l'Intelligence Artificielle «classique».

La seconde position, influencée par la Vie Artificielle, ne présuppose pas d'intelligence *a priori* de la part des agents. Les agents sont capables d'interagir entre eux par le biais de messages élémentaires et leur comportement repose souvent sur l'activation d'actions réflexes de type stimulus-réponse. Généralement, chaque agent est incapable de résoudre seul un problème et ne dispose pas de représentations symboliques de ses connaissances. Ces systèmes impliquent généralement l'intervention d'un nombre important d'agents. C'est de leurs nombreuses interactions qu'*émerge* l'intelligence du système. En effet, malgré la simplicité de leurs agents, ces systèmes peuvent exhiber une variété de comportements intéressants au niveau collectif. Ils permettent ainsi de se concentrer sur l'étude de structures d'interactions telles que la coopération ou la coordination sans nécessairement développer des systèmes de communication complexes. Cette école *réactive* est proche du quatrième axe de recherche décrit par Ferber. Elle s'inspire fréquemment de mécanismes existants dans les sociétés d'insectes et en particulier des processus pouvant conduire à l'émergence d'organisations. C'est par exemple le cas des travaux de [Deneubourg *et al.* 1992], de Steels [Steels 1991] ou de Minsky [Minsky 1988]. La distribution des connaissances parmi les agents, sans que chaque agent ait individuellement connaissance du problème global, offre plusieurs avantages, notamment de tolérance aux pannes, de robustesse, de simplicité de conception au niveau individuel, etc.

Ces deux positions extrêmes ne sont cependant pas à opposer strictement et il existe des architectures hybrides recouvrant les deux domaines. Une dernière approche, à différencier de ces deux courants, est celle de la *formalisation logique* des agents rationnels autonomes avec Castelfranchi, Wooldridge, Jennings, Cohen et Levesque [Ferber 1997].

2.2.3 L'environnement comme support de communication

> *«Pour un agent, interagir avec un autre constitue à la fois la source de sa puissance et l'origine de ses problèmes.»*
>
> — Jacques Ferber, [Ferber 1997]

Brooks [Brooks 1991] montre que l'intelligence d'un agent (virtuel ou humain) est déterminée par la dynamique de son interaction avec son environnement direct. Un agent est *situé* s'il évolue dans un environnement avec d'autres entités. Il perçoit alors son environnement en fonction de ses capacités perceptives («vision», détection de «trace», sonar, etc.) et il est capable d'agir sur son environnement en modifiant les positions et les relations entre les objets [Ferber 1995].

Dans [Russell et Norving 1995], les auteurs distinguent plusieurs propriétés intrinsèques à l'environnement dans lequel évolue un agent :

- accessible vs. inaccessible : l'agent a-t-il accès à tous les paramètres de son environnement ?
- déterminé vs. indéterminé : l'état futur de l'environnement est-il totalement déterminé par l'état actuel et les actions sélectionnées par les agents ?
- épisodique vs. non épisodique : l'expérience de l'agent est-elle divisée en épisodes perception-action où chaque action ne dépendrait que de l'épisode en cours ?
- statique ou dynamique : l'environnement change-t-il sans action de la part de l'agent ?
- discret ou continu : les actions ou les perceptions sont-elles clairement délimitées en nombre et dans le temps ?

La perception de son environnement permet à l'agent d'*interagir* avec les autres agents du système. La communication est une forme particulière d'interaction entre individus. Les agents peuvent communiquer directement entre eux en partageant un langage de représentation externe, sans préjuger des représentations internes des autres. Ce langage repose sur la mise en place de protocoles définissant le dialogue entre les agents [Mephu-Nguifo 1993]. Cette communication peut, par exemple, utiliser des *actes de langages* [Nilsson 1998] partagés par les individus.

Les agents n'ont pas toujours un langage de communication direct, mais peuvent aussi communiquer par le biais d'un système de partage d'informations. L'architecture de *tableau noir* (FIG. 2.1) a été très utilisée dans les systèmes multi-agents cognitifs symboliques. Cette architecture offre un moyen de partager l'information entre plusieurs spécialistes, nommés *Knowledge Sources (KS)*. Les KS peuvent poser, modifier ou retirer des informations (hypothèses, solutions) sur le tableau noir. La résolution d'un problème repose sur l'activation opportuniste des KS. Quand deux ou plusieurs KS sont en conflit pour accéder à une information, un module de *résolution de conflits* décide quel KS va agir. Il organise ainsi de façon centralisée le fonctionnement du système. Le tableau peut aussi être modifié par un sous-système perceptuel qui calcule des données issues d'un système sensoriel. Chaque KS est donc un «expert» de la partie du tableau noir qu'il «regarde» [Nilsson 1998, Ferber 1995].

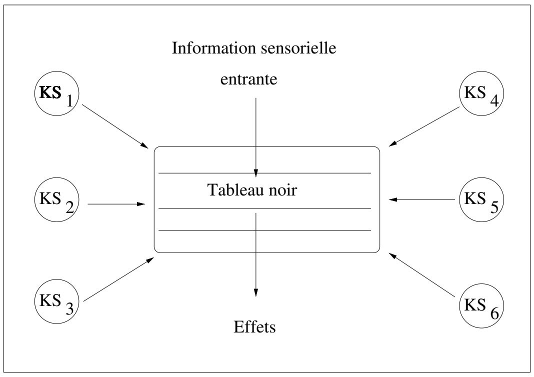


FIG. 2.1 – Communication avec Tableau Noir d'après [Nilsson 1998]

Dans le cadre des agents réactifs, le système de partage de l'information est généralement l'environnement lui-même. Les agents peuvent alors modifier leur environnement que se soit en y déplaçant des objets ou en y laissant des traces. Ce sont ces modifications qui, en agissant sur le comportement des autres agents, servent de communication «non-intentionnelle». La métaphore de la diffusion de phéromones, stimuli chimiques des insectes, est souvent employée pour décrire les informations laissées par les agents [Drogoul 1993]. Un agent peut ainsi émettre un signal qui se diffuse dans son environnement (FIG. 2.2). Généralement, l'intensité de la diffusion décroît lorsque la distance à

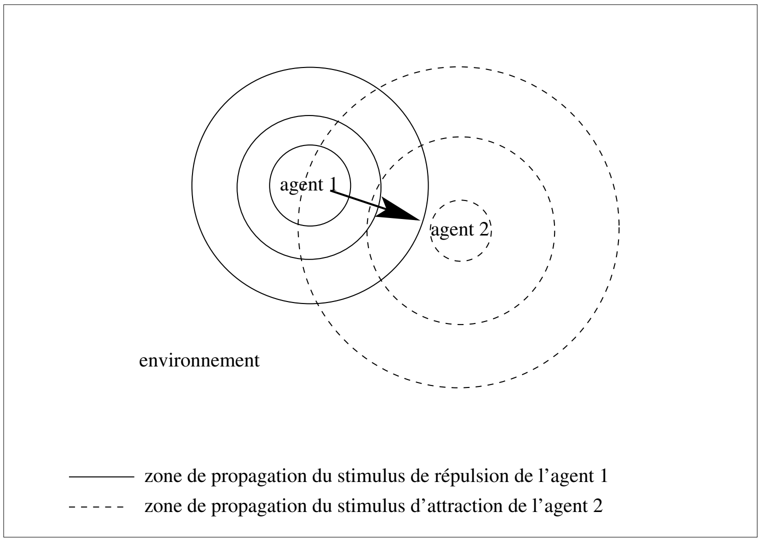


FIG. 2.2 – Communication indirecte avec propagation de stimuli dans l'environnement
L'agent 1 perçoit le stimulus d'attraction de l'agent 2, il se déplace donc vers lui en suivant le gradient. L'agent 2 ne perçoit pas encore le stimulus de répulsion de l'agent 1.

l'agent augmente. Le signal peut soit être rattaché à l'agent, soit être laissé dans l'environnement. Dans le premier cas, cela correspond, par exemple, à un signal de faim qui se déplace avec le déplacement de l'agent. Dans le deuxième cas, cela correspond à une trace laissée dans l'environnement qui permet à un autre individu de suivre le même parcours. Le comportement de diffusion peut se complexifier avec la présence d'obstacles. Les agents réactifs n'ont le plus souvent qu'une représentation partielle de leur environnement. Ils réagissent à leur environnement selon un principe de stimulus/réponse en fonction des stimulations qu'ils reçoivent de leurs congénères. Certaines stimulations peuvent ainsi correspondre à un message répulsif (obstacle) et d'autres à un message attractif (but). L'environnement est donc structuré par des zones favorables ou défavorables à la présence de certains agents. Ces champs de potentiel sont adaptés lorsque les actions principales des agents consistent à se déplacer ou à déplacer des objets.

2.2.4 Autonomie et adaptation

La notion d'autonomie correspond à l'idée qu'un agent dispose d'une certaine marge de liberté dans le choix de ses actes. L'autonomie résulte donc de l'indépendance avec laquelle un agent peut choisir ses actions afin de satisfaire ses buts individuels (tels que sa survie ou sa satisfaction) [Ferber 1997].

L'autonomie repose aussi sur la capacité des agents à apprendre de leurs expériences. Si un fabricant de montres sait que son futur propriétaire part dans six jours en Australie et qu'il inclut dans la montre un mécanisme permettant à celle-ci de changer d'heure au bon moment, ce mécanisme pourra être considéré comme intelligent par l'utilisateur. Cependant l'intelligence viendra plus du concepteur que de la montre elle-même [Russell et Norving 1995]. Le comportement d'un agent est spécifié par le comportement que lui donne son concepteur, mais il doit aussi pouvoir être basé sur les connaissances que l'agent construit au fur et à mesure de son interaction avec l'environnement particulier dans lequel il est immergé. L'agent devient donc autonome, c'est-à-dire que son comportement n'est plus sous contrôle humain mais qu'il est déterminé par sa propre expérience. L'adaptation peut être envisagée aussi bien du point de vue de l'*apprentissage* (au niveau de l'individu) que du point de vue de l'*évolution* (au niveau de la société).

L'autonomie repose aussi sur les ressources et les compétences dont dispose un agent : par exemple, un robot peut être autonome par rapport au chargement de ses batteries en gérant lui-même le moment et la durée de la recharge, ce qui implique qu'il sache où aller se recharger, qu'il soit assuré que la borne de recharge soit disponible et qu'il sache se recharger lui-même [Muñoz *et al.* 2001, Birk 1997, McFarland 1994].

2.3 Organisation et structuration des sociétés d'agents

2.3.1 Définition

> *«Une organisation peut être définie comme un agencement de relations entre composants ou individus qui produit une unité, ou système, dotée de qualités inconnues au niveau des composants ou individus. L'organisation lie de façon interrelationnelle des éléments ou évènements ou individus divers qui dès lors deviennent les composants d'un tout. Elle assure solidarité et solidité relative, donc assure au système une certaine possibilité de durée en dépit de perturbations aléatoires.»*
>
> — Edgar Morin, [Morin 1977]

La notion d'organisation permet d'envisager un ensemble (d'agents, d'individus, etc.) comme une entité décomposable en plusieurs niveaux (ou points de vue) où chaque niveau correspond à l'agrégation d'individus à un niveau inférieur. Par exemple, l'organisation d'une serre peut d'un point de vue global être considérée comme un regroupement spatial de différents genres où chaque genre forme une entité et suit un traitement qui lui est propre. Chaque genre peut lui-même être divisé en espèces, qui sont à leur tour composées d'individus. Chaque individu, selon son état de santé, peut à son tour avoir des besoins spécifiques à un moment donné. Un individu peut appartenir à plusieurs groupes, mais rien ne garantit qu'à chaque instant tous les groupes possibles existent : le groupe des *Phalaenopsis* fleuris ou des individus malades peut n'exister qu'à un moment donné. Ainsi, rien ne garantit la pérennité d'un groupe. De tels groupes correspondent à une *classification* extérieure faite par un observateur. Chaque individu appartient à un groupe d'objets parce que l'observateur a établi des similarités entre ces objets. Aucune interaction n'intervient entre ces objets et ces groupes.

Dans les sociétés d'agents, les *interactions locales* entre les individus sont à l'origine de la formation de groupes et de leur structuration [Resnick 1994a]. Ce mécanisme est souvent désigné par le terme d'*auto-organisation* [Camps 1998]. Un agent *élémentaire* sera considéré comme l'unité de plus bas niveau, dans le cadre de la décomposition choisie, et le système multi-agent sera considéré comme l'entité de niveau supérieur, c'est-à-dire comme ne faisant pas lui-même partie d'une organisation [Ferber 1995].

2.3.2 Des individus à une structure organisée

La modélisation d'entités autonomes et de leurs interactions [Müller 1996] permet de reproduire les dynamiques microscopiques d'un système complexe. Ces dynamiques conduisent, au niveau macroscopique, à l'émergence d'une structure organisée, sans qu'il soit nécessaire de modéliser les caractéristiques globales du système ou de recourir à un contrôle centralisé. De nombreuses micro-simulations ont été développées afin de comprendre comment les comportements et les interactions des individus peuvent influer sur le comportement de l'organisation globale. Par exemple, un modèle a été conçu afin de permettre à des architectes de visualiser comment leurs décisions vont influencer le mouvement d'une foule dans un centre commercial, tout en ne modélisant que le comportement local de chaque piéton [Dijkstra et Timmermans 2000]. Le comportement global de la foule n'est alors pas décrit dans le système mais *visible* à l'écran lors de la simulation par un observateur.

Les métaphores et les modèles biologiques ont inspiré beaucoup de travaux dans l'étude des processus de formation de groupes dans les systèmes multi-agents (réactifs). Une description détaillée de modèles issus de l'éthologie sera donnée au chapitre suivant.

2.3.3 Coexistence des niveaux micro et macro

> *«Toute organisation est le résultat d'une interaction entre agents, et le comportement des agents est contraint par l'ensemble des structures organisatrices.»*
>
> — Jacques Ferber, [Ferber 1995]

La conception, l'observation et/ou l'analyse des systèmes multi-agents peuvent se faire à différents niveaux d'organisation [Ferber 1995]. Le niveau *micro-social* est le niveau le plus élémentaire du système. Il définit l'activité individuelle de chaque agent ainsi que ses interactions avec les autres agents. Des structures de *groupes* d'agents peuvent émerger, au niveau macro, lors des interactions des individus et se maintenir à plus ou moins longue échéance. En effet, les groupes n'ont pas une réalité statique dans l'évolution de la société. *«Un attroupement d'oiseaux n'est pas un gros oiseau»* [Resnick 1996]. Les interactions d'entités à un niveau micro permettent ainsi de créer de nouveaux types d'objets à un niveau macro [Servat *et al.* 1998]. Ces groupes peuvent aussi avoir leurs propres activités et leurs propres interactions. Enfin, l'observateur peut considérer la *société globale* en étudiant, par exemple, la dynamique et l'évolution de la structure du système dans son ensemble.

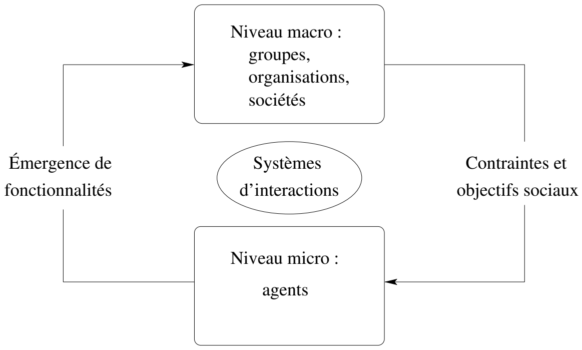


FIG. 2.3 – Relation micro-macro dans les systèmes multi-agents d'après [Ferber 1995]

Gasser [Gasser 2001] souligne les «barrières actuelles» lors de la conception de systèmes d'organisation. En effet, actuellement, il y a peu de recherches sur la façon d'organiser des systèmes interdépendants, hétérogènes et semi-autonomes et sur les moyens d'aboutir à des systèmes globaux ayant des comportements stables et prédictibles. Il n'existe donc pas de solution «prête à l'emploi» pour modéliser et concevoir un système multi-agent organisé à partir d'un problème donné. L'agrégation pose le problème de la causalité circulaire (FIG. 2.3) qui existe entre les individus et les processus organisationnels : *«individuals are the product of organizing processes»*. Il est aussi parfois difficile d'obtenir une organisation qui soit à la fois stable dans ses structures et flexible dans son adaptabilité à un environnement dynamique. S'il est souvent admis que le but du concepteur est de faire en sorte que l'efficacité de l'organisation soit supérieure à la somme de l'efficacité de chaque individu, l'optimisation de l'ensemble du système par rapport à un système centralisé est parfois plus difficile à atteindre, notamment en raison de la distribution de l'information. L'avantage des systèmes multi-agents par rapport à des systèmes centralisés peut aussi être difficile à prouver comme nous le verrons dans le dernier chapitre de ce mémoire. Une réponse rapide à ce dernier point est qu'une colonie de fourmis n'a pas toujours *la* meilleure solution pour un problème donné (qu'elle est théoriquement capable de résoudre) mais qu'elle atteint toujours *une* solution.

2.4 Agents d'interface

Notre mémoire portant sur une interface multi-agent de visualisation, il est maintenant nécessaire de présenter les agents existants sous le nom d'*agents d'interface*. Plusieurs types d'agents peuvent être distingués.

Les agents *intelligents* sont plutôt conçus comme des agents *assistants* ou *collaborateurs*, destinés à aider l'utilisateur dans certaines tâches telles que confirmer ses rendez-vous par mail, réserver des billets d'avion [Morris et Maes 2000] ou faire des achats sur le Web [Chavez et Maes 1996]. L'utilisateur peut déléguer des actions à ces agents. Les agents vont alors dresser son profil utilisateur afin de répondre au mieux à ses attentes. Leur *intelligence* se traduit par des caractéristiques d'autonomie importante (capacité de négocier le prix d'un billet d'avion), d'interactivité avec l'utilisateur (le prévenir d'un message important), etc.

Les agents d'*interface* rentrent généralement dans la catégorie des agents intelligents et rarement dans la catégorie des sociétés d'agents. Par exemple, ces agents peuvent aider l'utilisateur dans la découverte d'un logiciel lors de ses premières utilisations ou lui faire des suggestions lors de certaines manipulations [Microsoft Corporation 2001]. Face au développement d'interfaces de plus en plus complexes, ces agents d'aide sont de plus en plus couramment utilisés.

Les agents *autonomes* d'interface sont définis comme des agents capables de faire des actions sans intervention de l'utilisateur ou en parallèle avec lui, que l'utilisateur soit oisif ou en train de faire d'autres actions [Lieberman 1997]. Les agents n'attendent donc plus de sollicitation de l'utilisateur pour agir. De plus, ces agents opèrent *dans* l'interface et non en arrière-plan du système, à la différence d'un agent de réservation de billets. Ainsi l'utilisateur et les agents interagissent avec l'interface. La plupart de ces interfaces ne se composent que de quelques agents et parfois même d'un seul.

Par exemple, l'agent *Letizia* [Lieberman 1995a] construit un profil de son utilisateur à partir des pages Web récemment visitées. Il peut ensuite filtrer et proposer de nouvelles pages en fonction de celles qui conviennent le mieux à ce profil. *Let's Browse* [Lieberman *et al.* 1999] présente une extension de ce système à plusieurs utilisateurs. Cet agent aide un groupe de personnes à parcourir le Web ensemble en leur proposant de partager des informations sur des centres d'intérêt qu'il suppose être communs. Les travaux de Lieberman reposent aussi sur des agents assistants capables d'aider les «designers» dans leurs travaux d'édition. Ces agents possèdent alors des compétences importantes sur le graphisme mais surtout ils peuvent «capturer» les connaissances graphiques des utilisateurs à partir de leurs actions sur les éléments graphiques [Lieberman 1995b]. Ce type d'apprentissage est appelé *apprentissage par l'exemple*.

Dans le cadre des agents d'interface, le côté «multi» des systèmes multi-agents est souvent négligé et les systèmes agents remplacent les systèmes multi-agents. Il est alors parfois difficile de distinguer la limite entre un agent *intelligent* et un système expert.

2.5 Vers des systèmes multi-agents réactifs d'interface

> *«Il faudrait faire en sorte que tout soit aussi simple que possible,*
> *mais pas plus simple»*
> — Albert Einstein

Les agents d'interface, tels que nous venons de les décrire, requièrent des connaissances importantes sur leurs domaines d'application et sont ainsi fortement couplés à ces domaines. L'implémentation de ces connaissances induit ainsi une complexité de l'architecture qui tend à rapprocher ces agents des systèmes experts et de leurs problématiques.

Les systèmes multi-agents réactifs offrent une alternative à ces agents d'interface. Chaque agent de la société peut alors être chargé de «porter» et de représenter une partie de l'information. Cette distribution de la connaissance permettrait d'implémenter des systèmes évolutifs dans lesquels il serait simple de rajouter ou de supprimer des agents-données. De plus, la possibilité de faire coexister des agents hétérogènes dans un même système assurerait une plus grande modularité et flexibilité du système lors de l'introduction de nouveaux types de connaissances. Enfin, ces agents pourraient hiérarchiser et synthétiser des informations distribuées, sans contrôle centralisé, grâce à des mécanismes d'auto-organisation des agents-données.

Cependant, les problèmes soulevés par Gasser [Gasser 2001] d'équilibre entre la stabilité et la flexibilité d'une organisation se posent de façon tout aussi cruciale dans une interface de visualisation. Il faut donc que les agents, qui n'ont pas de connaissance globale du système, puissent quand même se représenter collectivement de façon stable, structurée et pertinente aux yeux de l'utilisateur. L'objet du chapitre suivant est précisément de montrer comment permettre l'émergence d'organisation stable mais flexible à partir de comportements relativement simples, basés sur des modèles éthologiques, d'agents réactifs.

Chapitre 3

Éthologie et modèles d'organisations

Sommaire

Comme nous l'avons signalé au chapitre précédent, de nombreuses recherches menées sur les systèmes multi-agents réactifs touchent aussi le domaine de la Vie Artificielle. C'est pourquoi ces recherches s'inspirent souvent de la vie *réelle* et, dans le cadre des agents réactifs, des études menées autour des comportements des animaux et en particulier des comportements sociaux des primates ou des insectes. Après avoir défini rapidement le contexte des études menées en éthologie, ce chapitre permet de montrer comment les modèles éthologiques basés sur des comportements individus centrés ont aussi permis de développer de nouveaux modèles organisationnels «pertinents» aussi bien pour la modélisation éthologique que pour la conception de sociétés organisées d'agents.

3.1 Études comportementales des animaux

3.1.1 Contexte

Tinbergen [Tinbergen 1967] définit quatre problèmes centraux abordés par les recherches en éthologie :

- *l'évolution* : Quelle est l'histoire des ancêtres des différentes espèces ? Qu'est-il possible d'en déduire sur l'évolution d'un comportement et sur les pressions qui l'ont façonné ?
- *le développement* : Comment se forme un comportement ? Quels sont les facteurs internes et externes qui influencent la manière dont il se développe au cours de la vie de l'individu ? Comment le processus de développement fonctionne-t-il ?
- *le contrôle* : Comment un comportement est-il contrôlé ? Quels facteurs internes et externes régissent sa survenance et comment ce contrôle fonctionne-t-il ? Qu'est-ce qui régule ce comportement ?
- *la fonction* : À quoi un comportement est-il ordinairement utilisé ? De quelle manière le comportement aide-t-il l'animal à rester en vie et à transmettre ses gènes à la génération suivante ?

L'éthologie couvre ainsi un champ d'études relativement large et des thématiques aussi diverses que :

- les sociétés de primates, par exemple celle des bonobos [de Waal et Lanting 1999] ;
- la pensée et la conscience animale [Vauclair 1992] ;
- la fabrication et l'utilisation d'outils : utilisation d'un piquant de cactus par le pinson-pic afin d'extraire des larves d'un tronc, la pêche au miel des chimpanzés [Boesch et Boesch-Achermann 1991] ;
- la danse des abeilles décrite par von Frisch [Ruwet 1975] ;

- la façon dont les écureuils ou les mésanges stockent et retrouvent leurs provisions ;
- etc.

Pendant longtemps les animaux ont été considérés comme des machines. *«Cette attitude est poussée jusqu'à l'absurde dans le cas de Malebranche qui rudoie sa chienne ; et comme l'animal gémit, un de ses amis la plaint. Pourquoi la plaindre, dit le philosophe : ce ne sont que grincements de poulies à l'intérieur du mécanisme»* [Chauvin 1989]. Lorsque les études éthologiques se sont développées, les différents postulats sur lesquelles elles reposaient ont imposé les techniques d'approche à mettre en œuvre pour analyser les animaux.

Pour Fabre [Fabre 1913], l'instinct est responsable de tous les comportements qui dirigent la vie de l'individu, le but final étant la conservation de l'individu et de l'espèce. Il observe ainsi les insectes dans la nature et tente de les transporter dans son laboratoire pour affiner et préciser ses observations.

Par la suite, la vision béhavioriste, notamment avec Watson, Skinner [Skinner 1969] et Thorndike, limite l'étude du comportement aux relations observables qui existent entre les stimulations et les réponses de l'organisme. Toutes les expérimentations doivent alors se faire dans le laboratoire afin de garantir une méthodologie rigoureuse. Apparaissent ainsi les études sur le conditionnement opérant et les mécanismes de renforcement faites en particulier sur des rats conditionnés à effectuer une tâche particulière (boîte de Skinner, labyrinthe, etc.) [Naville 1963].

À l'inverse le courant naturaliste, avec Lorenz (oies) [Lorenz 1969, Lorenz 1984], Tinbergen (épinoches), Whitman (pigeons) ou Heinroth (oies et canards), revient à l'étude de l'animal dans son milieu naturel, en interaction avec d'autres individus, dans leurs comportements *normaux*, de reproduction, de soin aux jeunes et plus tard de chasse [Chauvin 1989]. Les expérimentations ne sont alors nécessaires que dans un deuxième temps afin de valider les hypothèses.

Actuellement, pour l'éthologie cognitive, penser revient à calculer c'est-à-dire manipuler des représentations [Vauclair 1992]. Cette approche, en se situant dans un postulat de traitement de l'information, autorise des travaux aussi bien en laboratoire que dans le milieu naturel des animaux. Mais elle permet aussi d'introduire des modélisations mathématiques et informatiques afin d'expérimenter les observations directes faites sur les animaux.

Worden [Worden 1996], par exemple, propose un modèle informatique de l'intelligence sociale des primates à partir de scripts. Selon lui, l'intelligence sociale des primates est basée sur des représentations discrètes et symboliques de situations sociales. Les scripts correspondent alors à ces représentations et sont à l'origine de l'apprentissage de compétence sociale. Ces types de modèles nécessitent souvent beaucoup de connaissances et de compétences de la part de l'individu simulé car ils reposent sur des société de primates, de loups, etc., c'est-à-dire sur des espèces ayant des comportements et des connaissances complexes (connaissance des autres individus, intentionnalité, alliance, etc.) [Picault et Collinot 1998].

3.1.2 Sociétés animales : introduction de modélisations multi-agents

Les modèles utilisés en éthologie reposent initialement sur une vision mathématique du monde définissant des équations différentielles entre des variables quantitativement mesurables. C'est le cas du modèle écologique de Lotka-Volterra exprimant la dynamique de la population sur un territoire en fonction du nombre de proies et de prédateurs [Hogeweg et Hesper 1979]. Ces modèles numériques posent cependant des problèmes lorsqu'il s'agit de prendre en compte certains aspects des modèles éthologiques [Ferber 1995]. Par exemple, ces modèles ne prennent pas directement en compte le comportement des individus. Un troupeau de gnous ne va fuir que lorsqu'il se trouve à une certaine distance d'un prédateur. La valeur de cette distance va être calculée en fonction du nombre de calories que va perdre l'ensemble du troupeau lorsqu'il se déplace et de la déperdition énergétique que coûte la perte d'un individu. Cette distance de fuite est donc traduite en terme de quantité d'énergie mais ne tient pas compte des actions individuelles des membres du groupe. Ces modèles ne permettent pas non plus de prendre en compte et de mettre en relation plusieurs niveaux d'analyse (individu, groupe, société).

Les systèmes multi-agents permettent de définir des modèles informatiques *individus centrés* basés sur les *comportements* et les *interactions* entre individus ou entre groupes, mais ils peuvent aussi introduire des paramètres quantitatifs dans la simulation. Ainsi, plusieurs modélisations ont été développées sur la base de systèmes multi-agents. Les caractéristiques individus centrés et l'importance des interactions dans ces systèmes ont conduit à des modélisations de groupes ou de sociétés animales telles que des simulations de sociétés de primates [Picault et Collinot 1998] ou de sociétés d'insectes [Drogoul *et al.* 1995, Drogoul *et al.* 1992] ou de groupes de poissons [Terzopoulos *et al.* 1994].

Dans la suite de ce mémoire, nous aborderons des modèles issus de groupes ou de sociétés d'animaux beaucoup moins complexes individuellement mais où les très nombreuses interactions conduisent à des organisations complexes.

Un des premiers modèles individus centrés a été développé par Hogeweg et Hesper [Hogeweg et Hesper 1979, Hogeweg et Hesper 1983] afin de simuler l'évolution de l'organisation d'une société de bourdons [van Honk et Hogeweg 1981]. La simulation *MIRROR* met l'accent sur les spécifications locales, les interrelations entre les individus et un contrôle distribué grâce à l'utilisation de *démons*. Les entités ne sont plus simulées comme des «parties de l'ensemble» mais comme des «unités autonomes». *«Les modèles de simulation permettent d'étudier comment des entités en interrelation (dynamique) peuvent générer en apparence (i.e. dans les yeux de l'observateur) des propriétés émergentes»* [Hogeweg et Hesper 1979]. Les «macro» comportements de la société sont ainsi issus des «micro» comportements des individus ; il n'est plus nécessaire de définir implicitement ou explicitement des hypothèses sur les spécifications des comportements individuels permettant d'aboutir à des macro-relations. Les auteurs soulignent les avantages de l'utilisation de ce type de modèle en biologie. En particulier, la spécification locale des modules augmente la flexibilité, ainsi que les possibilités d'élargir le modèle mais surtout cela permet une meilleure lisibilité et compréhension du modèle lors de l'observation des comportements des entités.

Ce domaine de modélisation individus centrés a depuis fait l'objet de nombreuses simulations qu'ils s'agissent de modèles éthologiques ou de modèles physiques. Drogoul *et al.* [Drogoul *et al.* 1992, Drogoul 1993] ont montré comment il était possible de reproduire les processus de sociogénèse existant dans les fourmilières avec des sociétés d'agents réactifs. D'autres applications utilisant des modèles individus centrés ont été développées en dehors du domaine éthologique. Les travaux de Servat [Servat 2000] ont montré comment ces modèles permettaient de modéliser des processus de ruissellement, d'infiltration et d'érosion à partir d'agents «goutte d'eau». Dans le domaine de la physique des milieux granulaires, les agents «grain de sable», conçus par Breton *et al.* [Breton *et al.* 1999, Breton 2002], utilisent des mécanismes d'*éco-résolution* [Ferber 1995] afin de résoudre des équations physiques.

À la limite de la modélisation éthologique proprement dite, de nombreux mondes virtuels complexes ont été construits à partir d'agents directement issus du monde animal. Terzopoulos [Terzopoulos *et al.* 1994] a réalisé un monde marin virtuel dans lequel évoluent des poissons artificiels variés (Fig. 3.1). Ces poissons sont des agents autonomes issus de modèles réalistes au niveau individuel (apparence, déplacement et mouvement corporel) mais aussi au niveau des comportements de groupes de poissons. Les poissons sont, par exemple, capables d'apprendre à contrôler leurs muscles dans le but de se déplacer de façon hydrodynamique. Ils peuvent aussi déterminer leur distance avec des objets et identifier ces derniers à travers leur perception visuelle limitée du monde.

Fig. 3.1 – Artificial Fishes [Terzopoulos 1994]

3.2 Modèles éthologiques d'organisations et systèmes multi-agents

Comme l'ont montré les modélisations de Hogeweg [Hogeweg et Hesper 1979] ou de Servat [Servat 2000], les modèles individus centrés permettent de mettre en place des organisations structurées sans recourir à des mécanismes centralisés complexes. Les modèles que nous présentons ci-dessous, bien que constituant parfois plus des analogies avec les modèles éthologiques que de véritables modèles éthologiques, montrent comment il est possible de concevoir des comportements organisationnels relativement complexes à partir de comportements individuels simples. Comme nous le montrons dans le chapitre 4, ces mécanismes organisationnels sont particulièrement bien adaptés lorsqu'il s'agit de mettre en œuvre des processus de structuration dans nos sociétés d'agents d'interface.

3.2.1 Termites et morceaux de bois

Les différentes espèces de termites sont toutes organisées dans une société autour d'une reine et de son mâle. Les autres individus sont, quant à eux, répartis dans des castes d'ouvriers et des castes de soldats. Les nids construits par les ouvriers peuvent être très complexes [Bonabeau *et al.* 1997] et, dans certains cas, les termites peuvent même contrôler l'humidité et la température de l'intérieur de leur termitière autour d'un optimum de 30°C et 98-99% d'humidité [Eibl-Eibesfeldt 1972].

Afin d'expliquer les processus conduisant à la construction d'un nid sans aucun recours à des mécanismes de communication complexe entre les individus, Grassé [Grassé 1959] a introduit le *concept de stigmergie* : le travail de l'insecte est guidé par le produit de son activité antérieure, suivant un schéma stimulus-réponse. Les individus communiquent rarement entre eux de façon directe mais par des modifications que leurs actions entraînent dans leur environnement. C'est la construction du nid en cours qui stimule les ouvriers et oriente leur travail afin de déclencher un acte spécifique. Lorsque des termites, en nombre suffisant, sont en présence de boulettes de terre, il est alors possible d'observer plusieurs types de mécanismes sociaux successifs [Ruwet 1975, Bonabeau et Theraulaz 1994] :

- un déplacement désordonné : quelques individus se mettent à préparer des boulettes de terre qu'ils disposent et abandonnent au hasard, un individu pouvant détruire ce qu'un autre a fait ;
- un travail coordonné : à un moment donné, plusieurs boulettes se trouvent par hasard accolées et constituent alors une masse critique de stimuli qui polarise l'attention de tous les ouvriers ; ceux-ci y ajoutent leurs propres boulettes et, rapidement, la structure grandit. Selon le stade atteint par la construction, les termites construisent des murs, des piliers ou des arches.

Ce serait alors la construction elle-même qui par sa taille, sa forme et son stade d'édification assurerait sa propre régulation. Plusieurs simulations plus ou moins complexes de ce mécanisme ont été conçues [Bonabeau et Theraulaz 1994], certaines s'attardant juste sur le principe de formation de tas, d'autres modélisant l'ensemble de la construction de la termitière.

La modélisation de formation de tas par les termites développée avec StarLogo par l'équipe de Resnick [Resnick 1994a, Resnick 1994b] présente une très grande simplicité dans le comportement de ces agents. Ce modèle présente un algorithme permettant aux termites de déplacer des copeaux de bois afin de former des tas. Le comportement d'un termite est donné dans l'ALGORITHME 3.1.

ALGORITHME **3.1** Comportement d'un termite

Si le termite ne porte rien et trouve un morceau de bois **Alors**
  il le prend
Si le termite porte un morceau de bois et qu'il en trouve un deuxième **Alors**
  il pose le morceau de bois porté

Dans cet algorithme, chaque termite déplace ainsi aléatoirement des copeaux de bois les uns à côté des autres. Il peut, cependant, aussi lui arriver de défaire un tas commencé par d'autres ouvriers. Au bout d'un certain nombre d'itérations, le nombre de copeaux isolés diminue et la probabilité de placer un copeau près d'un tas déjà formé augmente (FIG. 3.2). Peu à peu, le nombre de groupes diminue et la taille de chaque groupe augmente.

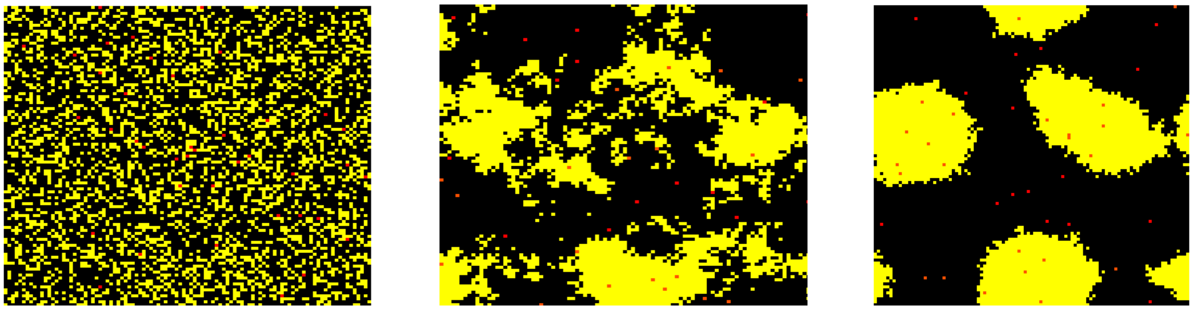


FIG. 3.2 – Formation de tas par les termites [Resnick 1994]

Ainsi, les termites ne possèdent ni de comportements explicites de construction de tas, ni de plan de l'environnement indiquant où se trouvent les groupes, ni enfin de contrôle centralisé sur le comportement collectif mais ils aboutissent malgré tout au regroupement des morceaux de bois.

Les algorithmes de *tri collectif* se rapprochent de ce modèle, les morceaux de bois étant alors des entités hétérogènes et les agents ayant pour rôle de regrouper des entités semblables. Ces algorithmes de tri collectif sont inspirés des comportements de regroupement du couvain observés chez les fourmis ou les abeilles.

3.2.2 Phénomènes d'agrégation

Un autre modèle d'organisation collective d'individus simples est fourni par les phénomènes d'agrégation. Tous les individus d'une espèce sont, par nature, concurrents pour la nourriture, la reproduction, le territoire, les meilleurs emplacements pour faire leur nid, etc. En dehors des espèces sociales, ils ne se regroupent ou ne coopèrent que dans des circonstances particulières et parce qu'ils peuvent y trouver un intérêt immédiat. Le premier avantage des groupes est celui de la sécurité. Dans [Eibl-Eibesfeldt 1972], l'auteur a montré qu'un poisson est mieux protégé dans un banc que tout seul : il devient en effet difficile pour le prédateur de se concentrer sur une cible. De plus, l'alerte est donnée beaucoup plus rapidement par un des membres du groupe en cas de danger.

Les comportements de regroupement des oiseaux migrateurs, des bancs de poissons ou des criquets ont fait l'objet de nombreuses simulations. Reynolds [Reynolds 1987, Reynolds 2000] a défini un modèle distribué d'agrégation de «boids» (créatures artificielles élémentaires) en permettant aux individus de coordonner leurs comportements avec ceux des autres membres du groupe. Chaque individu est partagé entre le fait de rester avec les autres membres du groupe et le fait de se tenir suffisamment éloigné du groupe afin d'éviter les collisions. Quelques règles comportementales simples au niveau individuel suffisent à faire émerger des comportements de groupe. Un boid doit ainsi maintenir une distance minimale par rapport aux autres objets de l'environnement et en particulier par rapport aux autres boids. Il doit aussi adapter sa vitesse à la vitesse moyenne de celle de ses voisins et se déplacer vers le centre de gravité des boids voisins.

ALGORITHME **3.2** Comportement d'un «boid» dans un groupe

ÉVITEMENT DE COLLISION («COLLISION AVOIDANCE») :
Pour tous les voisins proches **Faire**
  calculer une force de répulsion en fonction de leur position et d'un poids de répulsion
s'éloigner de ces voisins

COHÉSION («FLOCK CENTERING») :
Pour tous les voisins proches **Faire**
  calculer le centre de gravité de leur position
se rapprocher de ce centre

ALIGNEMENT («VELOCITY MATCHING») :
Pour tous les voisins proches **Faire**
  calculer leur direction moyenne
  calculer leur vitesse moyenne
se rapprocher de la direction et de la vitesse moyenne

Les règles présentées dans ALGORITHME 3.2 [Reynolds 1999] sont suffisantes pour qu'un observateur voie dans le comportement des boids des mécanismes semblables à ceux d'un vol d'oiseaux migrateurs. Ces comportements de déplacement très fluides n'ont besoin ni d'un contrôle centralisé, ni d'une définition au niveau des groupes. Ceux-ci évoluent dynamiquement au gré des fusions et des divisions qui peuvent se produire. Ces comportements peuvent être à la base de mécanismes d'attroupement, de fuite ou de processus plus complexes utilisés par des personnages d'animation ou de jeux tels que les «pigeons» fuyant devant une voiture télécommandée (FIG. 3.3).

Selon les espèces, l'observateur peut s'interroger sur le degré d'interaction de l'individu avec son groupe [Gregory 1993] : les individus existent-ils encore ou le comportement de groupe efface-t-il le niveau individuel ? le groupe est-il une moyenne des caractéristiques des individus, une référence ou une entité à part ? Dans certaines espèces de fourmis, la répartition du travail en équipes distinctes (polyéthisme) peut se traduire par un véritable polymorphisme. L'individu n'est alors qu'une partie de l'entité fourmilière et n'est pas viable isolé de sa société [Ruwet 1975].

FIG. 3.3 – «Pigeons dans le Parc» [Reynolds 2000]

Toujours selon Reynolds, les comportements du groupe doivent être vus comme le résultat de l'interaction de l'ensemble des comportements des individus, chacun agissant seulement à partir de la perception locale de son environnement. Dans [Reynolds 1987], la perception locale de chaque individu correspond en fait à un accès direct à l'ensemble de la description géométrique de la simulation, sorte de tableau noir stockant la position, la vitesse et la direction des boids. C'est à partir de cette description qu'un individu peut retrouver ses voisins. L'introduction de messages via l'environnement, basée sur la métaphore des phéromones chez les fourmis, offre une solution alternative à ce modèle de perception. Cette métaphore a été mise en œuvre pour définir des comportements d'attraction et de répulsion, par exemple dans les modèles proies-prédateurs.

3.2.3 Comportements d'attraction et de répulsion

Les deux modèles précédents présentent des mécanismes d'organisation sans qu'il soit nécessaire que les agents communiquent entre eux, directement ou indirectement. Seule la modification de la configuration de l'environnement (et donc des autres agents) influe sur le comportement de l'individu. Les individus n'ont alors pas besoin d'émettre des signaux en dehors du signal que peut représenter leur seule présence pour les autres. Les modèles basés sur des comportements d'attraction et de répulsion font, quant à eux, intervenir des comportements de communications rudimentaires, souvent inspirés des phéromones chimiques.

Les mécanismes d'attraction et de répulsion sont à la base de nombreux comportements d'interactions chez les animaux, que ce soit entre congénères ou entre individus entretenant des relations. Dans beaucoup d'espèces, le congénère peut aussi bien avoir un rôle de partenaire que de rival. *«Le résultat de ce double rôle fait que le congénère est très souvent porteur simultanément de signaux de rejet et d'attirance»* [Eibl-Eibesfeldt 1972].

Ces comportements permettent ainsi de maintenir une distance d'équilibre entre les congénères. Les comportements d'attraction et de répulsion se retrouvent dans les compétitions proies-prédateurs. La distance de fuite, par exemple, peut varier suivant l'espèce et suivant les expériences individuelles. *«Les petites espèces ont en général des distances de fuite plus petites que les grandes espèces. Moins une espèce a des moyens de protection plus sa distance de fuite est grande»* [Eibl-Eibesfeldt 1972]. La réaction de fuite peut aussi être différente selon le prédateur : une poule domestique va voler vers un arbre devant un prédateur terrestre tel qu'un renard mais va voler vers un abri face à un rapace. De façon générale, une réponse ne répond pas à un stimulus mais plutôt à une combinaison de stimuli-signaux, appelée *combinaison déclenchante*. Les expérimentations ont aussi montré l'existence d'une loi de sommation hétérogène de stimuli selon laquelle des stimuli qualitativement différents peuvent se remplacer quantitativement [Ruwet 1975].

Les stimuli peuvent s'exprimer au travers de différents types d'interactions acoustiques, visuelles ou olfactives [Vauclair 1992]. La communication peut être directe et s'exprimer par un «langage» comme chez les vervets [Seyfarth et Cheney 1993], par des signaux visuels tels que les couleurs chez les poissons et les oiseaux qui peuvent représenter des signes de menaces intra et interindividuelles, etc. La communication peut aussi s'exprimer indirectement via des signaux chimiques, les *phéromones*, émis dans l'environnement, comme chez les fourmis [Errard *et al.* 1990].

Plusieurs simulations ont expérimenté les principes de distance de fuite, de combinaison de stimuli ou de mécanismes d'interaction directe ou indirecte afin de reproduire des comportements des proies et des prédateurs [Nishimura et Ikegami 1997]. Cliff, par exemple, présente un modèle utilisant un réseau de neurones et des algorithmes génétiques permettant de faire co-évoluer des proies et des prédateurs [Cliff et Miller 1996]. Ce modèle a ainsi permis d'étudier les stratégies issues de cette co-évolution, stratégies évoluant aussi bien d'un point de vue comportemental (rapidité d'action) que d'un point de vue «physiologique» (déplacement de la position des yeux des espèces virtuelles).

Des mécanismes très simples d'attraction-répulsion utilisant la métaphore de diffusion de stimuli locaux via l'environnement peuvent suffire à reproduire l'approche d'un prédateur vers une proie et la fuite de cette dernière (ALGORITHME 3.3). Les proies et les prédateurs émettent des signaux de présence dont l'intensité décroît proportionnellement à la distance. Les proies fuient alors les signaux émis par les prédateurs. Les prédateurs sont d'une part attirés par le signal de présence des proies et d'autre part repoussés plus faiblement par les signaux émis par les prédateurs [Ferber 1995].

À partir de ces simples comportements, des comportements de «chasse» collective peuvent alors être *observés* par un observateur alors qu'ils ne sont pas explicitement

Algorithme 3.3 Comportement d'une proie et d'un prédateur

comportement d'une proie :

émettre un stimulus de présence dans son environnement local

Si un stimulus de présence de prédateur est détecté **Alors**

descendre le gradient de stimulus (fuir le prédateur)

comportement d'un prédateur :

émettre un stimulus de présence dans son environnement local

Si un stimulus de présence de prédateur est détecté **Alors**

descendre le gradient de stimulus (fuir le congénère)

Si un stimulus de présence d'une proie est détecté **Alors**

remonter le gradient de stimulus (attraction vers la proie)

codés dans le système. En effet, l'attraction des prédateurs par les proies et la répulsion entre prédateurs vont conduire à de véritables techniques d'encerclement des proies.

À partir de ces mécanismes élémentaires d'attraction et de répulsion, plusieurs stratégies cognitives complexes ont été définies afin d'optimiser la recherche de proies par les prédateurs : élaboration d'équipes, répartition des prédateurs dans l'environnement, etc. [Ferber 1995].

La façon dont ces modèles seront interprétés dans notre interface multi-agent de visualisation fera l'objet du chapitre 4. Quelques points peuvent déjà être mis en évidence. Tous ces modèles permettent-ils d'organiser visuellement une société d'agents en situant topologiquement les agents dans leur environnement ? Dans le cas des termites, les groupes d'agents bois sont passifs et leur organisation nécessite l'intervention d'un autre type d'agents. Dans le cas des boids, les agents, en réagissant à certaines règles simples, s'organisent d'eux-mêmes afin de former des comportements collectifs cohérents. L'introduction de moyen de communication via l'environnement permet de complexifier les comportements collectifs sans cependant complexifier les comportements individuels. Avant d'analyser, dans le chapitre suivant, comment mettre en œuvre ces principes dans une interface de visualisation, nous allons approfondir les relations existant actuellement entre les systèmes multi-agents et l'éthologie.

3.3 Apports réciproques de l'éthologie et des systèmes multi-agents

> *«Les SMA proposent une nouvelle technologie de construction de logiciels à partir des concepts d'agents et d'interaction, en considérant que chaque unité de programme peut prendre la forme d'un agent qui dispose de sa propre autonomie, de ses propres objectifs et «vie» sur le réseau comme un animal dans un écosystème naturel, coopère ou négocie avec d'autres unités de même nature.»*
> — Jacques Ferber, [Ferber 1997]

Gérard Tisseau [Tisseau 1996] montre les échanges possibles entre la psychologie et l'Intelligence Artificielle. L'Intelligence Artificielle peut fournir des points de comparaisons, des sources d'analogies et des métaphores pour formaliser les activités cognitives, mais peut aussi fournir une aide à la validation ou à la réfutation d'hypothèses pour les psychologues. Inversement, les théories sur les mécanismes cognitifs humains ou sur la compréhension de la langue naturelle en psychologie peuvent offrir de nombreuses ressources de modélisation pour les systèmes d'Intelligence Artificielle. Un enrichissement «circulaire» peut donc être fait entre ces deux domaines, les expériences de l'un pouvant pouvant entraîner une remise en cause et une amélioration des modèles de l'autre. Comme nous venons de le voir avec les modèles éthologiques et leurs simulations avec des systèmes multi-agents des échanges sont tout aussi pertinents à envisager entre l'éthologie et les systèmes multi-agents réactifs : l'éthologie se servant des systèmes multi-agents afin d'expérimenter ses modèles, modèles qui peuvent à leur tour enrichir les modèles d'organisation, de communication, etc., entre les agents [Nehaniv 1999]. Nous allons maintenant voir quels peuvent être ces échanges et ce que peut en retirer chaque domaine.

3.3.1 Similarité terminologique et méthodologique

> *«Un organisme, quel qu'il soit, vit au milieu d'un environnement physique et biologique qui lui pose continuellement un certain nombre de problèmes. L'animal prend connaissance de ceux-ci par l'intermédiaire de son équipement sensoriel. Il perçoit son environnement ; il en reçoit des informations triées par ses organes des sens. Et son comportement est l'expression de cette vie de relation avec le monde extérieur : c'est l'ensemble des conduites innées et acquises par lesquelles il rencontre et résout les difficultés du milieu.»*
> — Jean-Claude Ruwet, [Ruwet 1975]

Cette définition d'un organisme dans son environnement pourrait tout aussi bien s'appliquer à un agent sous réserve de remplacer «l'environnement physique et biologique» par un «environnement physique et/ou logiciel».

Des questions similaires se posent ainsi dans le domaine éthologique et dans le domaine des systèmes multi-agents :

- Comment les individus (agents/animaux) interagissent-ils ?
- Quel est le rôle de cet environnement ?
- Comment les actions localisées d'un individu vont-elles influer sur le reste du groupe ?
- Comment un groupe peut-il être cohérent, alors que les individus qui le composent sont autonomes ?
- Comment coordonner les activités d'individus autonomes sans supervision ?
- Comment valider un ensemble d'hypothèses comportementales ?
- Quels sont les moyens de communications des individus ?
- Qu'est-ce qu'un comportement et comment s'organise un comportement donné ? Où commence le comportement et où finit-il ? Par exemple, *se nourrir*, divisible en *recherche*, *stockage* et *consommation* de nourriture, correspond-il à un seul ou à trois comportements différents ?
- Face à un groupe d'individus, à quel niveau d'analyse faut-il se référer ?

3.3.2 Transposition de modèles éthologiques à des agents informatiques

«I would like to thank flocks, herds, and schools for existing ; nature is the ultimate source of inspiration for computer graphics and animation.»
— Craig Reynolds, [Reynolds 1987]

Les travaux dans le domaine éthologique peuvent utiliser l'informatique afin de modéliser, d'expérimenter et de valider certaines hypothèses comportementales. Les travaux autour des «animats» dans le domaine de la Vie Artificielle montrent comment en retour les modèles éthologiques peuvent être transférés à des «animaux artificiels», qu'il s'agisse de robots ou d'entités logicielles.

Dans le domaine des systèmes multi-agents, Di Caro *et al.* [Di Caro et Dorigo 1998] s'inspirent de métaphores éthologiques afin de construire de nouveaux modèles d'optimisation. Un modèle métaphorique issu du processus de stigmergie est utilisé en résolution de problème afin que la solution émerge des actions locales des individus. Plus précisément :

- l'utilisation de simulations répétées et concurrentes conduit une population de fourmis artificielles à générer de nouvelles solutions ;

– les agents utilisent des mécanismes de recherche locale et stochastique afin de construire des solutions de façon incrémentale ;
– l'utilisation des informations collectées dans les simulations passées orientent la recherche future vers de meilleures solutions [Dorigo *et al.* 1998].

L'algorithme est inspiré des capacités des fourmis réelles à trouver le plus court chemin entre deux lieux (nid et source de nourriture, par exemple). Les fourmis déposent des phéromones volatiles sur leur passage. Au départ, elles choisissent au hasard leur chemin. À partir de la source de nourriture, les premières qui retournent au nid sont celles qui ont emprunté le chemin le plus court. Elles font plus ainsi d'allers-retours par le chemin court que par le chemin le plus long et renforcent d'autant l'attractivité du chemin le plus court. Peu à peu, le deuxième chemin est abandonné par les fourmis. Ce mécanisme a été utilisé par Dorigo pour résoudre les problèmes d'optimisation du voyageur de commerce [Dorigo et Gambardella 1997]. Ce même modèle est utilisé dans AntNet [Di Caro et Dorigo 1998] afin d'introduire des mécanismes d'apprentissage adaptatif dans les tables de routages des réseaux de communications.

3.3.3 Anthropomorphisme et métaphore : danger ou avantage ?

Une tendance naturelle à l'interprétation...

> *«J'estimai finalement que les bonobos se distrayaient en faisant des grimaces ne remplissant aucune fonction de communication. Ce qui est assez intrigant en soi, car cela indique qu'ils possèdent un contrôle volontaire de leurs muscles faciaux. Un animal qui se comporte ainsi ne pourrait-il pas également manipuler les autres ? Quelle que soit la signification du phénomène, ces jeunes singes me firent en tout cas comprendre l'absurdité de l'obsession classificatrice de la science. Se moquaient-ils de moi ? Une fois que j'eus compris à quoi servaient ces acrobaties faciales, je ne pus m'empêcher de penser que parfois ils clignaient de l'œil dans ma direction !»*
>
> — Frans de Waal, [de Waal et Lanting 1999]

> *«Il semble que nous ne puissions faire autrement qu'accepter une certaine incompréhensibilité de l'existence. À vous de choisir. Nous oscillons tous délicieusement entre une vue subjective et une vue objective du monde, et cette perplexité est au cœur de la nature humaine.»*
>
> — Douglas Hofstadter, *Vues de l'Esprit*, 1987

Le regard que les hommes portent sur les animaux repose souvent plus sur des convictions personnelles que sur des expérimentations scientifiques. Les uns considèrent les animaux comme des «presque humains», le langage en moins, les autres les considèrent comme des animaux-machine. L'évolution de l'analyse du comportement animal, du béhaviorisme au cognitivisme, montre la difficulté des humains à analyser le comportement des animaux sans faire intervenir leurs préjugés. La description d'une fourmilière, par exemple, sous les termes de *reine*, de *soldat* ou d'*ouvrière* conduit à la perception d'une société hiérarchique, divisée en castes, où chaque individu a un rôle particulier à jouer.

Deux points de vue s'opposent quant à l'interprétation des capacités animales. L'analyse du comportement est souvent fondée sur le *canon de Morgan* : il s'agit de considérer les animaux de la manière la plus simple possible, jusqu'à ce qu'il y ait de bonnes raisons d'y renoncer. Cependant, pour [Gregory 1993], l'obéissance stricte à cette doctrine conduit à stériliser l'imagination. *«Celui qui étudie les comportements des animaux et qui ne les envisage jamais comme des humains risque de laisser échapper une partie de la richesse et de la complexité de ce qu'ils font. Nombre d'éthologistes expérimentés pensent que cela les aide beaucoup de se mettre dans la situation de l'animal et de réfléchir à la manière dont ils réagiraient s'ils étaient à sa place. Ils prennent alors conscience d'influences importantes qu'ils auraient négligées autrement et réalisent des expériences qu'ils n'auraient pas réalisées sinon.»* Les observations de terrain ont ainsi montré que les chimpanzés ont un concept du moi et sont aussi capables de tromper les autres et de dissimuler de la nourriture.

Au contraire, Vauclair [Vauclair 2000] pose la question de savoir comment interpréter les performances cognitives et perceptives des animaux sans risquer de les traduire en termes humains de fonction et de finalité des comportements observés. Pour lui, plus l'espèce est éloignée des humains, plus le risque est grand. Il est alors nécessaire de considérer que chaque espèce développe des modalités d'adaptation à l'environnement qui sont fonction des particularités de son équipement sensoriel et moteur, des contraintes imposées par le milieu et des besoins spécifiques pour évoluer dans ce milieu.

... pouvant amener à une certaine compréhension

> *«Un mot sur le vocabulaire.(...) J'aurais aimé que, comme les gadgets de Pif, l'éditeur distribue avec le livre un sac de guillemets. Le lecteur aurait été invité à les disposer, à son gré, là où il en aurait senti le besoin. S'il convient de ne pas trop prendre au sérieux le langage anthropomorphique, on doit reconnaître qu'il est éminemment utile sur le plan de la pédagogie.»*
>
> — H. Reeves, *Malicorne, Réflexions d'un observateur de la nature*, 1990

Il suffit de citer des termes tels que les *réseaux neuronaux*, les algorithmes *génétiques* ou la *mémoire* d'un ordinateur pour se rendre compte que l'utilisation de métaphores dans le domaine informatique n'est pas nouvelle [Renault 2001].

Les métaphores permettent de souligner des caractéristiques d'un problème, d'en cacher d'autres, d'envisager le problème sous un autre angle et donc de proposer de nouvelles solutions [Travers 1996]. Elles permettent de poser de nouvelles questions et ainsi de suggérer de nouvelles relations entre certaines caractéristiques. Plus les caractéristiques de la métaphore sont proches de l'objet original, plus il est possible de confondre les deux dans un même raisonnement. Le choix de la métaphore risque bien sûr de biaiser la résolution d'un problème en orientant vers certaines pistes et en en éliminant d'autres. Mais, la mise de côté *temporaire* de certaines caractéristiques du problème n'est pas toujours un inconvénient.

C'est par exemple le cas avec les réseaux de neurones. À l'origine, les réseaux de neurones informatiques étaient très proches de ce qui était connu en biologie. Ils servaient à résoudre des problèmes simples en mettant de côté certaines caractéristiques du problème. Au fur et à mesure que le modèle s'est complexifié, comme cela a été le cas avec le Perceptron, puis avec les réseaux néo-connexionnistes, il est devenu possible de traiter de plus en plus de caractéristiques des problèmes à résoudre et de s'éloigner aussi de la métaphore d'origine.

Comme la plupart des disciplines, l'éthologie est divisée en différents courants de pensée. Roitblat montre comment les paradigmes des modèles éthologiques choisis peuvent influer sur la construction de mécanismes de contrôle des comportements dans les animats [Roitblat 1994]. En effet, un modèle issu de la théorie béhavioriste de l'apprentissage spatial d'un rat dans un labyrinthe de MacFarlan ou un modèle issu de l'apprentissage de cartes cognitives de Tolman n'aboutiront pas aux mêmes mécanismes comportementaux.

De plus, parler d'un agent en terme de fourmi ou de prédateur permet de faire comprendre à un observateur la relation de cet agent avec les autres individus de l'environnement, mais cela l'incite surtout à se servir de cette tendance naturelle à se mettre à la place de l'agent (de la même façon qu'il interprète les comportements d'un animal par rapport à ses propres comportements) pour analyser ce qu'il ferait s'il était à sa place.

3.4 Synthèse des comportements possibles pour des agents d'interface

Les modélisations individus centrés permettent à l'éthologie d'expérimenter de nouveaux types de comportements et de mettre en évidence le rôle des interactions dans des groupes d'individus. Les modèles des termites, de tris collectifs, de formation de groupes et de comportements proies-prédateurs montrent qu'il est possible d'obtenir des modèles relativement évolués à partir d'actions individuelles simples. Ces modèles éthologiques peuvent en retour servir de nouveaux modèles d'interactions pour définir des comportements pour des agents (réactifs ou non).

Il est alors possible, par exemple, d'utiliser la métaphore du regroupement de bois ou du tri de couvain pour imaginer des agents triant des données (numériques ou textuelles) selon les mêmes principes. La visualisation de la dynamique d'un tel système permet de suivre progressivement la construction des groupes par les agents. En effet, l'avantage de ces modèles éthologiques est de s'appliquer à des systèmes ayant des possibilités d'adaptation locale tout en conservant une certaine cohérence et une certaine constance au niveau de l'ensemble du système. Les possibilités d'adaptation peuvent alors être mises en corrélation avec la dynamique nécessaire pour traiter de nouveaux types de données, tandis que la constance au niveau de l'ensemble du système devrait permettre de garder la cohérence nécessaire à une interface de visualisation.

Les différents modèles de groupe posent le problème de savoir ce que représente un groupe pour un individu (qu'il s'agisse d'une fourmi, d'un agent ou d'une donnée) et à quel niveau se place l'observateur. Analyse-t-il le groupe en tant qu'une entité réelle ou observe-t-il les entités individuelles qui le composent ?

L'apport de l'éthologie dans ce mémoire est double. D'une part, l'éthologie nous a servi de source d'inspiration pour le comportement individuel et collectif d'agents constitutifs d'une interface de visualisation. D'autre part, les modèles et le vocabulaire éthologiques utilisés lors de la conception du système de visualisation et par conséquent dans ce mémoire, l'ont aussi été à des fins explicatives et pédagogiques. En effet, il est plus facile d'expliquer les comportements des agents à un utilisateur ne connaissant pas le domaine des systèmes multi-agents en utilisant la métaphore d'émission de stimuli dans l'environnement, d'attraction ou de répulsion d'agents similaires qu'en lui donnant les équations mathématiques utilisées.

Ce chapitre nous a donc permis de mettre en évidence les modèles réactifs d'organisation qui existent avec des systèmes multi-agents. Il est maintenant nécessaire de revenir à notre problématique initiale de visualisation de données et de montrer comment ces modèles peuvent être mis en œuvre dans une interface multi-agent de visualisation.

Chapitre 4

Positionnement et propositions

Sommaire

Le chapitre 1 a permis de mettre en évidence les propriétés idéales d'une interface de visualisation d'informations dynamiques et hétérogènes. Cette interface doit posséder, en particulier, des mécanismes d'organisation et de synthèse afin de fournir une visualisation structurée des données. Elle doit aussi pouvoir s'adapter aux modifications de son environnement, qu'elles proviennent des données elles-mêmes ou de changements dans les centres d'intérêt de l'utilisateur. Le chapitre 2 a présenté les propriétés générales d'un système multi-agent et a ainsi permis de soulever les similitudes entre les caractéristiques de ces systèmes et celles qu'il est souhaitable de trouver dans une interface de visualisation. Une des propriétés les plus importantes pour un système de visualisation repose sur sa capacité à organiser l'information. Le chapitre 3 a ainsi montré comment des comportements «simples» au niveau des agents peuvent conduire à une structuration dynamique de l'ensemble d'un système multi-agent.

Deux interrogations se posent alors :

- comment concevoir une interface multi-agent de visualisation et quels rôles vont jouer les agents dans la prise en charge et la représentation des informations ?
- comment le développement de nouveaux modèles d'organisation dans les systèmes multi-agents peut conduire à l'organisation de l'information à l'écran ?

4.1 Évolution dynamique de sociétés d'agents pour une interface de visualisation

> *«Si nous pouvons parler de «systèmes» organisant à chaque fois des phénomènes entre eux (...), c'est toujours parce qu'on a su distinguer des similitudes ou des différences entre les formes qu'empruntent ces phénomènes pour exister ou telles que nous les percevons. Les similitudes permettent de regrouper, les différences permettent d'opposer.»*
>
> — Georges Vignaux, [Vignaux 1999]

La conception d'une interface multi-agent de visualisation, telle que nous l'envisageons, repose sur quelques principes initiaux. Tout d'abord, chaque agent est porteur d'une donnée ou d'une partie de l'information à représenter. Comme dans une simulation, l'interface visualise alors la société d'agents. Chaque agent a donc un aspect graphique (forme, couleur, taille, etc.) et des comportements dépendants de l'information qu'il contient. À chaque instant, les données peuvent subir des fluctuations qui entraînent alors des modifications dans les agents au niveau de leur aspect et/ou de leurs comportements. Des agents peuvent aussi naître ou mourir selon l'arrivée en temps réel de nouvelles données ou la suppression de certaines informations. Les agents interagissent dans leur société et peuvent, par exemple, se regrouper ou se repousser. Comme dans les simulations éthologiques, les agents peuvent, par leurs interactions locales, permettre l'émergence d'organisations. Ce sont ces organisations des sociétés d'agents qui vont conduire à l'organisation visuelle de l'information à l'écran.

4.1.1 Systèmes existants

Quelques applications sont comparables à notre approche. Un système de visualisation très simple a été développé dans [Minar et Donath 1999]. Il permet de suivre les mouvements de personnes parcourant un site Web de la même façon qu'une caméra permet de suivre une foule dans un espace réel. Ce système est basé sur une cartographie déjà établie des pages du site, sur une représentation des individus par des icônes et enfin sur une animation permettant de suivre les déplacements des icônes sur cette carte. Cependant l'article ne précise pas si les individus qui se connectent au site ont accès à cette représentation. En effet, si les utilisateurs n'ont pas accès à cette représentation (comme le laisse supposer cet article), il paraît difficile de parler d'une dynamique de foule ou de groupe. Contrairement à un mouvement de foule dans un espace réel où chaque individu se déplace en fonction de ses propres buts tout en ayant comme contrainte la présence des autres, dans ce système chaque utilisateur se déplace indépendamment des autres sans qu'aucune interaction n'ait lieu. L'intérêt de ce système est cependant de montrer comment l'observateur regroupe visuellement l'information qu'il perçoit : en effet, comme dans une cartographie, les regroupements sont effectués en fonction de la proximité des informations, les mouvements des agents conduisant alors à déplacer une information dans un autre groupe.

Yoshida et *al.* présentent dans [Yoshida *et al.* 1998] un système multi-agent permettant de visualiser un groupe d'utilisateurs en fonction de la proximité de leurs centres d'intérêt. Chaque utilisateur est représenté par un agent. Le système extrait des mots-clés à partir d'un profil utilisateur. Grâce au poids associé à ces mots-clés, il calcule la pertinence relative d'un agent par rapport à l'ensemble des agents. Les agents sont alors disposés sur l'écran afin que leur distance reflète la proximité de leurs mots-clés. L'utilisateur, dont l'agent personnel est placé au milieu de l'écran, peut interagir avec le système afin de lui signifier qu'il ne partage pas beaucoup de points communs avec un autre utilisateur positionné près de son agent. Des mécanismes d'apprentissage permettent alors à l'agent de modifier ses poids et de se repositionner.

Dans [Ishizaki 1996], Ishizaki propose un modèle générique de visualisation de données basé sur des comportements émergents d'agents réactifs de «design». Il propose deux applications proches, l'une sur la visualisation dynamique de forums de discussion et l'autre sur la visualisation de messages électroniques. Dans cette dernière application, chaque agent graphique est responsable d'un «segment» d'information : l'agent *Sender*, l'agent *Subject*, l'agent *Message*, l'agent *NumMsg*, etc. Les agents communiquent entre eux afin de visualiser à tour de rôle les segments d'information qu'ils représentent dans un espace à trois dimensions. Les travaux portent essentiellement sur les stratégies graphiques utilisées pour représenter les agents. L'approche choisie repose sur un nouveau type de visualisation qui se caractérise par un changement dynamique de la représentation au cours du temps en fonction de l'information déjà présentée à l'utilisateur. Ce type de visualisation fournit une représentation très éloignée et relativement complexe de celle que nous souhaitons atteindre dans notre démarche. Cependant, l'utilisation de systèmes multi-agents réactifs pour représenter une interface de visualisation très dynamique et interactive est particulièrement intéressante.

Les interfaces de discussion en ligne tentent de plus en plus de s'éloigner des lignes de textes et de représenter des dialogues dans des environnements plus conviviaux composés, par exemple, d'avatars. Dans [Viegas et Donath 1999, Donath *et al.* 1999], les auteurs présentent une alternative à ces environnements en proposant une interface graphique abstraite nommée *Chat Circles* (FIG. 4.1). La présence et la participation d'un individu sont représentées par des changements de couleur, de forme et de taille d'un cercle. L'interface présente aussi un système de filtre visuel par proximité permettant à l'utilisateur de suivre une conversation particulière, tout en gardant un suivi des discussions de l'ensemble du système. Les cercles représentant une discussion précise se situent ainsi dans une zone particulière de l'écran. Chaque participant est représenté par un cercle coloré dans lequel apparaissent les phrases qu'il énonce. L'intensité de la couleur du cercle diminue au fur et à mesure que l'utilisateur n'intervient plus. Au moment où un message est posté, le cercle le représentant grossit et met ainsi le message en évidence. Après quelques instants, le cercle revient à sa taille d'origine. Les phrases, issues des autres discussions, apparaissent comme des cercles vides, sans texte à l'intérieur. L'intérêt de ce système repose sur l'utilisation de mécanismes graphiques élémentaires qui permettent de visualiser simplement des informations complexes telles que la dynamique d'une conversation, les tours de parole ou l'activité d'un individu ou d'un groupe.

Proctor et Winter [Proctor et Winter 1998] ont proposé une méthode de visualisation de données basée sur des comportements de regroupements de poissons en banc (FIG. 4.2). La corrélation entre les données est traduite par le temps que passent les poissons les uns à côté des autres. L'application proposée repose sur la visualisation des centres d'intérêt

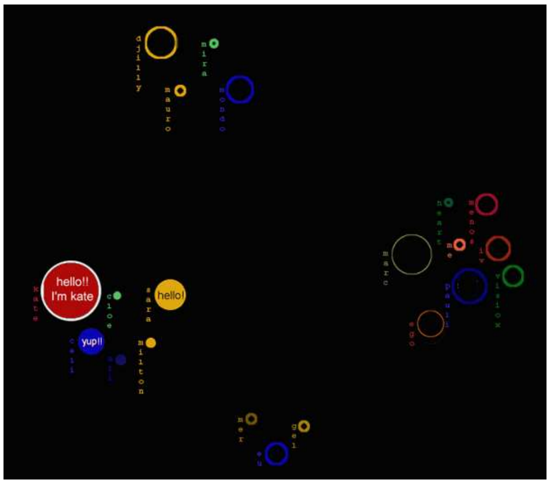


FIG. 4.1 – Chat Circle [Viegas et Donath 1999]

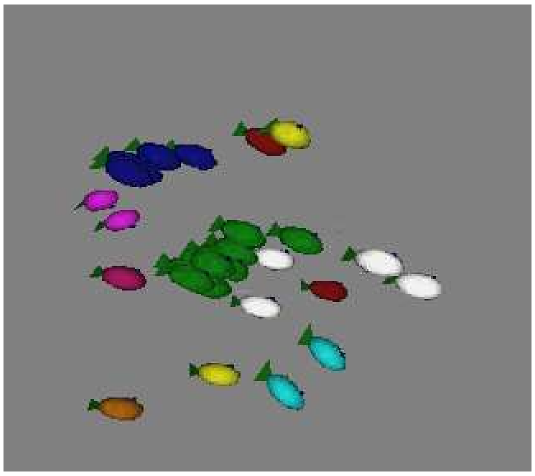


FIG. 4.2 – «Information Flocking» [Proctor et Winter 1998]

d'un groupe d'utilisateurs. Chacun d'eux est représenté par un poisson. Comme dans les modèles éthologiques, ce poisson possède une vision locale de son environnement. Dans le cadre de *l'attroupement d'information* («Information Flocking»), les auteurs proposent de rajouter une quatrième loi aux principes de Reynolds (ALGORITHME 3.2). Cette loi mesure la similarité entre les individus et permet ainsi de modifier leurs comportements pour qu'ils se rapprochent ou s'éloignent les uns des autres en fonction de cette valeur. Le comportement d'attroupement devient ainsi un moyen de visualiser les similarités entre les données représentées par les agents.

4.1.2 Positionnement

Malgré la diversité des interfaces obtenues, ces systèmes proposent tous des modèles de représentation basés sur une répartition de l'information à l'aide de différents éléments graphiques. Ces éléments graphiques permettent une description individu centré des comportements qui facilite la conception de l'interface (Chapitre 2.5) ainsi que la compréhension qu'en a un observateur/utilisateur (Chapitre 3.3.3). Ces systèmes montrent ainsi comment des entités relativement simples peuvent permettre de visualiser des informations hétérogènes, plus ou moins complexes. En effet, la présentation graphique colorée ou iconographique ainsi que la proximité des icônes permet à l'utilisateur d'identifier rapidement des groupes de données ayant des similarités (Chapitre 1.1.2). De plus, ces interfaces permettent, à l'inverse de beaucoup de systèmes plus classiques de visualisation, de prendre en compte la dynamique issue de l'évolution des données ou de l'arrivée de nouvelles informations. C'est aussi grâce à la représentation de cette dynamique qu'il est possible de mettre en évidence visuellement les similarités entre les données.

Cependant, ces éléments graphiques ne comportent généralement pas de moyens d'interactions importants entre les agents et l'utilisateur (sauf pour [Yoshida *et al.* 1998]) ni de processus de synthèse visuelle de l'information. De plus, même s'ils sont généralement capables de s'adapter dynamiquement aux données, ils ne prennent généralement pas en compte la modification des centres d'intérêt de l'observateur. Enfin, à part dans [Proctor et Winter 1998], aucun de ces systèmes n'utilise réellement les potentialités d'interactions et d'organisation des agents. En effet, ces systèmes utilisent les agents comme des entités distribuées positionnées dans un environnement ayant à leur charge un élément d'information. Les agents évoluent alors en tant qu'*agent-donnée* (i.e. porteur d'une information) mais rarement en tant qu'*individu centré* (i.e. en interaction avec leur environnement).

Même s'il est nécessaire d'adapter les comportements des agents en fonction des données qu'ils représentent, ces systèmes n'ont, malgré tout, besoin que de peu de connaissances sur le domaine et sur les données qu'ils représentent. Ces principes semblent assez généraux pour s'appliquer à divers types de visualisation dynamique, qu'il s'agisse de données numériques (données météorologiques, etc.) ou de données textuelles (courriers électroniques, sites Web, etc.).

Comme le signale Gérard Tisseau, dans [Tisseau 1996], un système d'intelligence artificielle ne suffit pas, il faut aussi donner à l'utilisateur des moyens d'interagir pour lui poser des problèmes et y introduire (explicitement ou non) de nouvelles connaissances. La conception de nos agents devra donc prendre en compte l'utilisateur pour que ce dernier puisse agir aussi bien avec l'ensemble du système qu'avec les agents, individuellement ou en groupe.

4.2 Agents réactifs, éthologie et organisation de données

Les agents, tels qu'ils sont définis jusqu'à présent, ont à leur charge une donnée ou un ensemble de données, ils réagissent dynamiquement à la mise à jour ou à l'arrivée des informations et ils possèdent des caractéristiques graphiques permettant à un observateur de les regrouper visuellement. Il est maintenant nécessaire de les doter de capacités d'interactions locales afin qu'ils puissent profiter des capacités d'organisation des systèmes multi-agents.

Les modèles des termites et du tri collectif peuvent, par exemple, offrir à un système de visualisation de données dynamiques un modèle distribué et dynamique permettant de regrouper l'information à l'écran. Les données correspondent ainsi aux morceaux de bois qu'il faut trier et les termites à des agents chargés d'effectuer ces tris. Le nombre d'agents «trieurs» dépend de la vitesse à laquelle un observateur veut voir se former les groupes. Rien n'empêche alors de ne laisser dans la simulation qu'un seul agent trieur. En effet, les comportements définis dans l'ALGORITHME 3.1 restent tout aussi valables pour trier les données avec un seul agent (il lui faudra simplement plus de temps). Le qualificatif de tri *collectif* n'est donc plus forcément le mieux choisi.

De plus, la différence entre une entité «à trier» et une entité «trieuse» repose simplement sur leur capacité à se déplacer. Ainsi, en s'éloignant du modèle éthologique, il est possible de faire abstraction des agents termites en dotant les morceaux de bois (virtuels) de capacité de déplacement aléatoire. L'ALGORITHME 3.1 peut alors être simplifié en ALGORITHME 4.1

ALGORITHME **4.1** Comportement d'un morceau de bois virtuel

```
déplacement aléatoire
Tant Que un autre morceau de bois n'a pas été trouvé Faire
  déplacement aléatoire
se poser à côté du morceau de bois trouvé
```

Dans notre approche, l'idée est de supprimer les agents correspondants aux termites et de fournir des mécanismes de déplacement aux agents-données eux-mêmes. Mais disposer d'agents-données capables de se déplacer ne suffit pas si ces agents ne sont pas dotés de mécanismes d'interactions leur permettant de traiter et de comparer simplement les éléments d'informations qu'ils portent.

Peu de travaux ont été menés sur l'utilisation de systèmes multi-agents dans le domaine du traitement et de la représentation de données. Kuntz [Kuntz et Snyers 1994] a utilisé un modèle agent afin de traiter le problème d'optimisation lors du partitionnement de graphes. Une société d'agents de différentes espèces est créée afin que l'évolution de cette société conduise à la colonisation du territoire selon les espèces, cette répartition devant alors correspondre à la solution du problème. Lumer *et al.* [Lumer et Faieta 1994] ont proposé un modèle de regroupement de données à partir d'agent-fourmi. Ce modèle repose sur le tri collectif de couvains par les fourmis. L'environnement est donc constitué d'une grille à deux dimensions. Comme dans le modèle de Resnick (Chapitre 3.2.1), les agents-fourmis sont distincts des données et agissent sur elles pour les regrouper. En plus d'être l'une des premières applications de tri collectif à des données, ce travail a l'originalité d'avoir testé de nouvelles stratégies afin d'améliorer les performances du système : diversifier la population des fourmis, rajouter des capacités temporaires de mémorisation aux individus afin qu'ils se souviennent des dernières actions effectuées, etc.

L'intérêt majeur de ces travaux est de mettre en évidence la possibilité de concevoir des modèles multi-agents basés sur des agents réactifs pour traiter des informations (contenues dans un graphe ou dans des données). À la différence des travaux de Dorigo, ces modèles soulignent le rôle topologique de l'environnement sur l'organisation des agents ainsi que le lien que peut avoir cet environnement avec les données. En effet, qu'il s'agisse des graphes ou du tri, l'environnement devient porteur des données et permet de les positionner de la même façon qu'il permet aux agents d'être situés. Ces deux modèles se rapprochent ainsi des logiciels de cartographie (Chapitre 1.3.1). Cependant, ils ne fournissent pas d'interface de visualisation à proprement parler. En effet, si la visualisation repose bien sur le suivi visuel de l'environnement des agents, aucun moyen n'est donné à l'utilisateur pour interagir avec eux. De plus, contrairement à l'interface que nous souhaitons développer, ces modèles reposent sur une analyse de données statiques et non sur des flux de données. Enfin, comme nous l'avons déjà signalé, ces modèles distinguent d'une part les données et d'autre part les agents devant agir sur elles, alors que nous souhaitons, dans notre conception, faire l'économie de deux types d'agents pour ne conserver que des agents-données.

Un dernier système multi-agent ayant des capacités d'analyse d'information peut être cité. *Amalthaea* [Moukas 1996] est un environnement virtuel dans lequel évoluent des agents «semi-intelligents». Ils interagissent selon des modèles de coopérations et de compétitions. Les agents de filtrage de l'information ont pour rôle de s'adapter aux changements progressifs des intérêts de l'utilisateur et de personnaliser ainsi le système. Les agents de découverte d'information permettent, quant à eux, d'adapter le système aux nombreuses sources d'informations analysées et de proposer de nouveaux URL à l'utilisateur. Les agents de filtrage indiquent le profil de l'utilisateur aux agents de découverte, qui vont rechercher des URL pertinentes. Ils soumettent ensuite ces URL aux agents de filtrage. Ces agents fournissent les réponses à l'utilisateur qui, en fonction de l'intérêt qu'il y trouve, va renforcer positivement ou négativement la fonction de fitness des agents. Seuls les agents ayant une fonction d'évaluation relativement importante pourront se reproduire. L'intérêt de ce modèle est d'une part de pouvoir traiter des données dynamiques issues de recherche de pages Web, mais surtout de disposer d'agents capables de construire un profil de l'utilisateur évoluant dynamiquement.

Ces trois derniers modèles nous permettent de présenter les comportements que nous souhaitons mettre en œuvre dans notre interface afin de répondre aux exigences de structuration de la visualisation. À la différence des travaux de Kuntz où chaque espèce correspond à un critère, nous avons choisi de représenter les ensembles de données par le même type d'agents. Les différents critères de ces données sont alors interprétés en différentes émissions de stimuli chimiques dans l'environnement. Les émissions de message et les possibilités de déplacement fournies aux agents-données leurs permettent ainsi d'interagir afin de s'attirer ou de se repousser en fonction de leur similarité ou de leur dissimilarité. Ce sont ces mécanismes d'attraction et de répulsion qui doivent permettre de structurer l'environnement.

4.3 Organisation et filtrage de l'information par des sociétés d'agents réactifs

En résumé, notre approche consiste à concevoir une interface de visualisation basée non pas sur un système ou un agent complexe mais sur une société d'agents-données hétérogènes capables de s'organiser, par le biais des interactions d'attraction et de répulsion portant sur leur similarité. L'interface de visualisation, en représentant l'organisation des agents dans leur environnement, aboutira à une visualisation structurée des agents-données.

Le tableau donné dans (Fig. 4.3 - 4.4) récapitule les caractéristiques des différents types de systèmes abordés jusqu'à présent. Ce tableau situe LEA, l'application que nous présentons au chapitre 6, par rapport à ces systèmes. LEA (Learning E-mail Agents) est une interface de visualisation de boîtes aux lettres électroniques, connectée en temps réel à un serveur de messagerie. Chaque agent est en charge d'un message et interagit avec les autres messages selon les similitudes qu'ils comportent. Par le biais de mécanismes d'attraction et de répulsion, les agents-messages ont alors la possibilité de se regrouper dans leur environnement et donc à l'écran. L'utilisateur possède lui aussi des moyens d'interagir avec les agents. En se situant à mi-chemin entre les systèmes de visualisation de données et les systèmes multi-agents, LEA intègre dans une même application les caractéristiques des deux domaines afin de mettre en œuvre une interface multi-agent de visualisation de données dynamiques.

Avant de présenter l'application LEA plus en détail dans le chapitre 6, nous allons introduire, dans le chapitre suivant, la plate-forme multi-agent OSCAR ayant servi à sa réalisation.

Systèmes de Visualisation — **SMA**

	SYSTÈMES et RÉFÉRENCES / CARACTÉRISTIQUES	Cartographie : – Landscapes [Wise et al.1995] – WEBSOM [Honkela et al. 1997]	Treemaps / Arbres coniques – [Shneiderman 1992] – [Robertson et al. 1991]	Mur perspectif [Mackinlay et al. 1991]	Chat Circle [Viegas et Donath 1999]
	type de données	articles / news ensemble de documents	arborescence disque dur / données boursières	sites Web	interface de *Chat*
PROPRIÉTÉ 1	– hétérogènes	non	non	non	non
PROPRIÉTÉ 1	– dynamiques	possible	possible	??	oui
PROPRIÉTÉ 1	– sources distribuées	possible	possible	??	oui
PROPRIÉTÉ 1	– hiérarchiquement structurées	non	oui	non	non
PROPRIÉTÉ 2	**techniques d'exploration**				
PROPRIÉTÉ 2	– situation topologique des données	cartographie		sur un "plan"	interface abstraite
PROPRIÉTÉ 2	– arbres / structures arborescentes		oui		
PROPRIÉTÉ 2	**mécanisme de structuration**				
PROPRIÉTÉ 2	– regroupement d'informations	zones sémantiques	par niveau hiérarchique	non	– rapprochement des dialogues
PROPRIÉTÉ 3	– synthèse d'informations	regroupement sémantique	non	non	
PROPRIÉTÉ 3	– hiérarchie visuelle	possible	oui	non	– suivi historique de la conversation
	techniques d'interaction				
PROPRIÉTÉ 5	– zoom	simple zoom possible	simple zoom possible	focus + contexte	– dialogue utilisateur
PROPRIÉTÉ 5	– requetes	non	possible	oui	– type focus+contexte avec évolution des cercles
PROPRIÉTÉ 5	– réorganisation possible par l'utilisateur	non	non	non	en fonction de l'état du dialogue
	systèmes à agents				
	– nombre d'agents				autant que d'utilisateurs
	– type de comportements				graphique – pas d'interaction
	– communication entre agents				non
PROPRIÉTÉ 4	**adaptation du système**				
PROPRIÉTÉ 4	– aux fluctuations des flux de données	possible (news) ?	oui	??	oui
PROPRIÉTÉ 4	– à l'utilisateur (profil utilisateur)	non	non	non	
	représentation graphique				
	– visualisation de mots–clés	oui	oui		texte de *Chat*
	– autres	paysage 2D / 3D	<<étiquette>>	représentation directe de l'information	cercles variant en couleur, taille et position

FIG. 4.3 – Tableau récapitulatif des différents systèmes

Systèmes de Visualisation

Systèmes (Multi–) Agents

	LEA [Renault 2001]	[Proctor et Winter 1998]	Agents autonomes d'interfaces : Letizia/Let's Browse [Lieberman et al 1998]	[Lumer et Faieta 1994]	Tri collectif / Modèles des termites [Resnick 1994]	Artificial Fishes [Terzopoulos 1994]
	courriers électroniques	centre d'intéret d'utilisateurs	sites Web	données quelconques	"objet"	modèles de poissons artificiels
	oui (mots–clés, dates, etc.)	oui ?	non	oui (variation de critères)	oui	possible
	oui	oui	oui (recherche info)	non	non	oui
	possible	oui	oui	possible	oui	
	non	non	non	non	non	
	environnement agent (proche cartographie)	environnement agent (proche cartographie)		environnement agent (proche cartographie)	environnement agent (proche cartographie)	monde virtuel
	non					
	oui	oui		oui	oui	
	formation de groupes	non		non	non	
	parcours des groupes	non		non	non	
	focus + contexte					
	sur les agents–données	sur les agents	possible			
	oui	??				
	égal au nombre de données	égal au nb d'utilisateurs	1	nb objets + trieurs	nb objets + trieurs	nombre de poissons
	réactifs / interactions	réactifs / interactions	cognitifs	réactifs	réactifs	cognitifs
	via l'environnement	via l'environnement	non	non : détecte si objet au meme endroit qu'eux	non : détecte si objet au meme endroit qu'eux	perception de l'environnement
	oui	oui	oui	non		
	profil utilisateur distribué	profil utilisateur	profil utilisateur	non		
	possible	possible	oui			
	iconographique	iconographie = poissons virtuels		ensemble de points	points de couleurs	poissons virtuels

FIG. 4.4 – Tableau récapitulatif des différents systèmes (suite)

Chapitre 5

Conception multi-agent de systèmes de visualisation de données dynamiques

Sommaire

En 1998, le sujet de mon stage de DEA IARFA [Renault 1998] concernait les «*processus distribués de hiérarchisation pour la visualisation dynamique de données numériques*». Il s'agissait d'étudier comment des mécanismes de hiérarchisation au niveau de systèmes multi-agents de visualisation pouvaient fournir une représentation intuitive mais structurée de l'état du système représenté. C'est à partir de ce stage et du projet des *Jardins de Données*, sur lesquels il reposait, qu'ont découlé mes travaux de thèse ainsi que la plate-forme OSCAR.

Après avoir décrit les principes des *Jardins de Données*, nous montrons pourquoi il a été nécessaire de développer une nouvelle plate-forme multi-agent pour la simulation et la visualisation de données. Nous présentons ensuite l'architecture de la plate-forme OSCAR à l'aide de la description du modèle des termites regroupant des morceaux de bois (Chapitre 3.2.1).

5.1 Jardin des Hasards et Jardins de Données

Jardin des Hasards

Le dessein général du *Jardin des Hasards*, développé par Guillaume Hutzler et Bernard Gortais [Hutzler et Drogoul 1996, Hutzler *et al.* 1998a, Hutzler 2000], était de proposer un nouveau paradigme de représentation de données issues de systèmes complexes. Cette interface permettait de représenter en temps réel un flux de données numériques sous la forme d'une interface graphique abstraite et sonore traduisant l'ambiance du système de données (Fig. 5.1).

Fig. 5.1 – Les Jardins des Hasards en automne [Hutzler 2000]

Cette formulation ne prétendait pas remplacer les techniques habituelles de traitement de données, mais de proposer une interface beaucoup plus intuitive. À la différence des méthodes statistiques, par exemple, qui stockent puis étudient des données «inertes», l'idée développée dans le Jardin des Hasards était de les traduire directement dans une représentation évoluant dans le temps. Il s'agissait alors de passer d'une «photographie» du système à un moment précis à une animation en temps réel traduisant les flux de données.

Le Jardin des Hasards permettait de générer automatiquement un monde virtuel, où vivaient et se développaient des entités caractérisées par leur forme et leur couleur. La particularité du projet résidait dans l'intégration de données dynamiques (des données météorologiques) obtenues en temps réel. Cela permettait aux «créatures» virtuelles de «vivre» et d'interagir les unes avec les autres, comme le font les différents éléments d'un jardin réel, en fonction des données climatiques en provenance de l'environnement. Les plantes virtuelles se décomposaient en familles, chacune d'elle se caractérisant par sa forme, sa couleur, sa taille et son mouvement.

Jardins de Données

Le projet des *Jardins de Données* [Hutzler *et al.* 1998b, Renault et Hutzler 2000] avait pour but d'étendre le Jardin des Hasards à d'autres données complexes, sans nécessairement conserver la dimension artistique existant dans le projet original. Initialement, nos travaux ont été développés sur l'architecture des Jardins de Données. Cette architecture a été conçue par Guillaume Hutzler [Hutzler 2000] et traduite en Java par Antoine Melki and Jérôme Cailly lors de leur stage de DESS GLA en 1998.

Limites des systèmes

Comme nous l'avons déjà précisé aux chapitres précédents, un système de visualisation doit posséder des mécanismes de traitement, d'organisation et de synthèse de l'information. Ils doivent, de plus, fournir une interface interactive avec l'utilisateur lui permettant d'interroger les agents sur l'information qu'ils portent. La version originale des Jardins de Données ne possédant pas ces mécanismes, il était nécessaire de revoir la conception du système. De plus, l'architecture même des Jardins de Données s'est révélée trop complexe pour un système multi-agent basé sur des agents réactifs.

Une nouvelle architecture a donc été conçue. OSCAR est ainsi une plate-forme simple définie pour prendre un charge des agents réactifs hétérogènes, pouvant évoluer dynamiquement avec des données. Ce noyau répond aussi aux exigences d'interactivité d'une interface. Enfin, comme nous le montrons, l'utilisation de XML évite d'avoir recours à un langage de script propre au système comme c'était le cas dans les Jardins de Données.

5.2 OSCAR : Outil de Simulation Comportementale par Attraction-Répulsion

OSCAR[1] est une plate-forme générique permettant de concevoir des systèmes multi-agents.

L'architecture de cette plate-forme repose sur un noyau multi-agent à base d'agents réactifs ayant la possibilité de communiquer via leur environnement. La généricité du noyau permet cependant de complexifier les agents et leurs comportements.

5.2.1 Problématique

Le développement d'une architecture multi-agent propre à notre cadre de recherche provient de plusieurs contraintes. Tout d'abord, il était nécessaire que les agents puissent prendre en charge simplement et rapidement des flux de données dynamiques. De plus, nos travaux portent sur le moyen d'organiser une société d'*agents-données* afin que cette organisation se traduise à l'écran par une structuration visuelle des données. Il était donc nécessaire que nous puissions maîtriser la prise en charge d'informations ou de groupe d'informations par les agents ainsi que la traduction visuelle de l'organisation de notre société d'agents à l'écran. Enfin, l'application finale devait pouvoir prendre en compte des interactions plus ou moins complexes entre l'utilisateur et les agents.

5.2.2 Architecture générale

Dans la suite de ce mémoire, nous emploierons le terme de *session* pour désigner aussi bien une simulation telle qu'une simulation proies-prédateurs que pour désigner une application telle que la visualisation de boîtes aux lettres électroniques. En effet, OSCAR a été conçu afin de pouvoir traiter aussi bien des modèles de simulations multi-agents que pour permettre de développer des interfaces multi-agents de visualisation de données.

La figure 5.2 présente l'architecture générale d'OSCAR. Le noyau contient les classes minimales de l'architecture multi-agent et prend en charge les nouveaux agents, leurs comportements et leurs paramètres définis lors de la *création d'agents* et lors de l'*initialisation de sessions*. L'ensemble du programme est écrit en Java (jdk1.2.2)

[1]OSCAR (I) fut aussi le nom donné à une maquette proies-prédateurs développée au cours d'un stage que j'ai effectué en juillet-août 1997 au LabRI à Bordeaux 1, sous la direction de Christophe Schlick, en collaboration avec Alice Barsse.

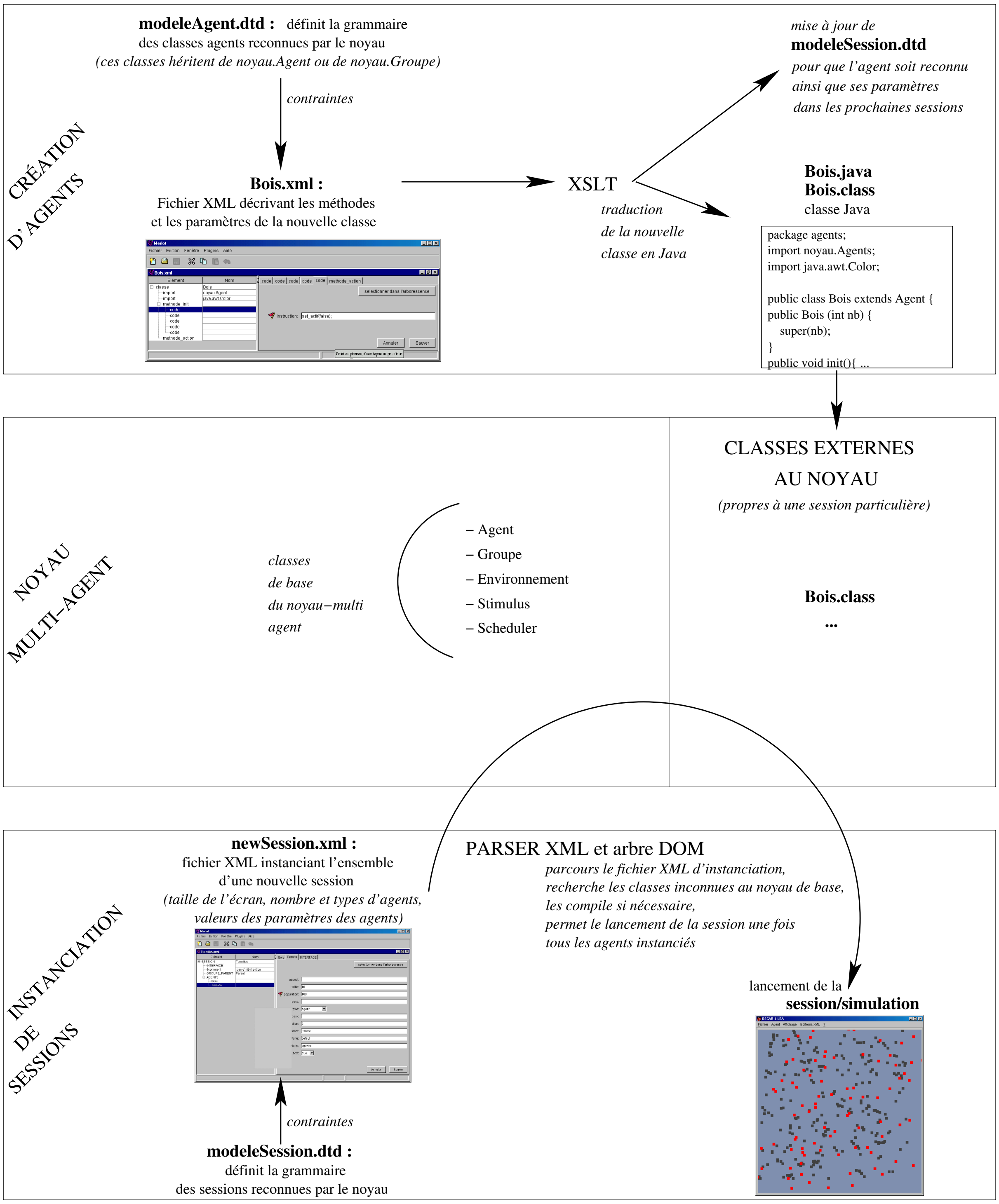


FIG. 5.2 – Architecture générale d'OSCAR

[Horstmann et Cornell 2000a, Horstmann et Cornell 2000b]. Le paquetage Xerces est utilisé pour lire les fichiers XML. Xalan traduit ensuite les fichiers XML à l'aide d'une base de règles au format XSLT. L'éditeur Merlot a été couplé au noyau afin de permettre la création et la visualisation des fichiers XML. Le paquetage Javamail sera utilisé pour LEA (chapitre suivant) pour permettre la connection à un serveur et la prise en charge en temps réel du courrier électronique.

XML (*eXtensible Markup Language*) est issu du langage SGML (*Standard Generalized Markup Language*) qui spécifie les règles de création de nouveaux langages de balisage [Daconta et Saganich 2001, Eckstein et Casabianca 1999, Bray *et al.* 1998]. Le balisage permet de séparer la structure d'un document de son contenu. À chaque langage est associé un ensemble de balises correspondant aux fonctions des données et une structure d'arbre d'éléments qui permet de décrire le type de document. La *définition du type de document* ou *DTD, Document Type Definition,* est un fichier spécifiant l'ensemble des balises et la structure de l'arbre, qui permet ainsi de spécifier les règles du langage. Un document sera *valide* s'il est conforme aux déclarations des éléments et des attributs de la DTD. Un document XML est *bien formé* s'il suit toutes les règles syntaxiques spécifiées dans la spécification XML.

Afin que notre noyau multi-agent soit le plus générique possible, nous avons choisi d'écrire deux langages de balisage qui interviennent à deux niveaux différents dans le noyau. Le premier langage permet de représenter une classe générique d'agent, les balises correspondant alors aux différentes caractéristiques d'un agent (paramètres, stimuli, comportements, etc.). Le second permet de structurer la description d'une session multi-agent (taille de l'interface, fichiers de données, type et nombre d'agents, etc.). XML permet ainsi l'indépendance du noyau multi-agent par rapport aux classes d'agents possibles, mais aussi par rapport à l'agencement des données et des agents dans une session.

L'utilisateur a alors la possibilité de créer de toutes pièces ses propres types d'agents, puis de les utiliser dans une session en leur donnant la possibilité d'interagir ou simplement de lancer une session déjà existante. Le déroulement de l'ensemble d'une session (de la création des agents à leur utilisation) est donc le suivant :

- création de chaque classe d'agent (types de paramètres, comportements) en XML, la création étant guidée par la DTD qui fournit le modèle d'un agent dans l'éditeur XML ;
- création du fichier XML de session indiquant les types d'agents à inclure et leurs paramètres initiaux ;
- ce fichier de session est «chargé» par le noyau lorsqu'il commence une session ou une simulation ;
- lors du chargement, lorsque le système rencontre une classe d'agent inconnue du noyau multi-agent d'origine, il recherche la classe correspondante dans les fichiers XML d'agents connus, si nécessaire la compile, la charge et instancie les agents de

cette classe.

5.2.3 Noyau multi-agent

Le noyau multi-agent lui-même ne comporte que très peu de classes :
- la classe Scheduler ordonne l'activation des agents au cours de la session ;
- les classes Agent et Groupe correspondent aux deux types d'agents du système ;
- la classe Environnement permet de situer les agents et diffuse leurs stimuli (classe Stimulus).

La figure 5.3 résume l'architecture de ces classes.

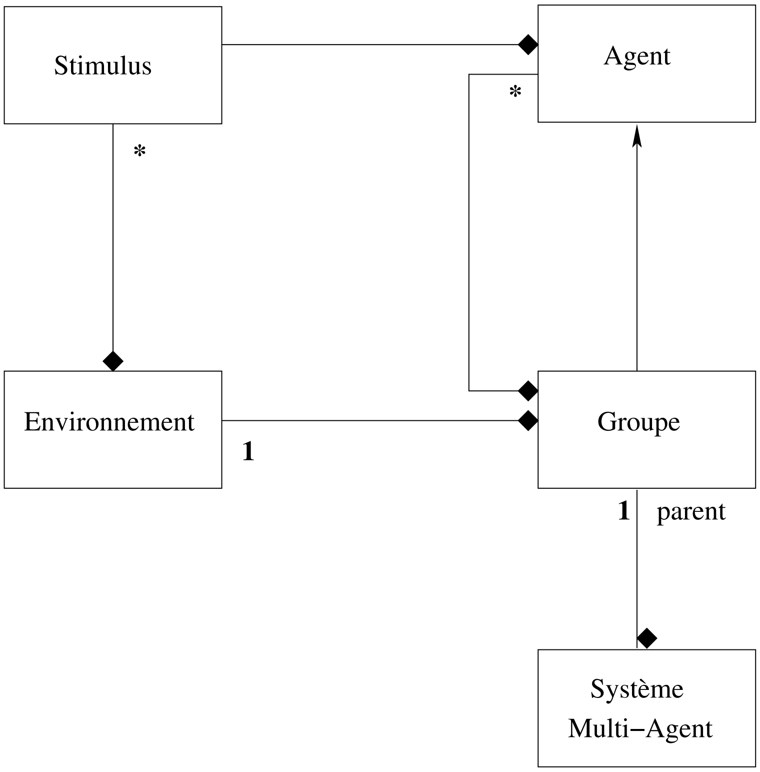


FIG. 5.3 – Schéma UML d'OSCAR

Scheduler

Le *Scheduler* parcourt successivement chaque agent de la liste des agents de la simulation et les active s'ils sont *actifs*. Chaque nouveau parcours de la liste des agents peut ainsi être considéré comme un cycle de la session.

Agent

La classe Agent est la classe «mère» de toutes les classes d'agents d'une session, qu'il s'agisse d'agents visibles à l'écran ou non. Cette classe contient donc des méthodes et des paramètres communs à tous les agents. Les paramètres-clés sont la position de l'agent dans son environnement, la connaissance de son parent, son tableau de stimuli, son numéro d'agent et une valeur booléenne indiquant si l'agent est actif ou non. Les méthodes publiques reconnues par le Scheduler sont : la méthode *init()* qui permet d'initialiser les paramètres et les comportements d'un agent au début de chaque session et la méthode *action()* qui s'exécute à chaque cycle de la session. D'autres méthodes sont définies dans cette classe et peuvent être redéfinies par les agents d'une session particulière. C'est, par exemple, le cas de la méthode *get_ ToolTipText()* et des méthodes appelées par la classe *MyMouseMotionListener* qui permettent de définir les actions de l'agent en fonction des actions exécutées avec la souris.

Groupe

Un groupe est un agent particulier qui possède son propre environnement. D'autres agents (et donc d'autres groupes) peuvent évoluer dans son environnement. La classe Groupe hérite de la classe Agent, car un groupe est avant tout un agent ayant la possibilité d'interagir avec les autres agents, comme n'importe quel agent, au niveau de l'environnement de son parent.

Chaque session comporte un *agent-groupe* particulier qui est l'*agent-parent*. Cet *agent-groupe* est le plus haut dans la «hiérarchie» des groupes, c'est-à-dire qu'il contient tous les autres agents et tous les autres groupes de la simulation. Ainsi, c'est dans l'environnement de cet *agent-parent* que les autres agents vont évoluer. Il se distingue des autres seulement du fait qu'il soit le seul à ne pas avoir lui-même de parent. Son environnement est l'*environnement-parent* de la session et il correspond au fond de la fenêtre de visualisation. Ce mécanisme a un double avantage. D'une part, il permet de ne pas faire de différences entre l'*environnement-parent* et les autres environnements contenus dans les groupes et ainsi de pouvoir y appliquer les mêmes méthodes (par exemple, pour la diffusion de stimuli ou le déplacement des agents). D'autre part, il devrait permettre d'étendre plus facilement une application. Par exemple, dans le cas de LEA, l'*agent-parent* représente le contenu de la boîte aux lettres d'un utilisateur. Élargir le système à la visualisation d'un ensemble de boîtes aux lettres d'un groupe d'utilisateurs ne modifiera pas les agents d'un utilisateur. L'*agent-parent* d'un utilisateur aura alors un parent et ses comportements seront identiques à ceux de n'importe quel *agent-groupe*.

Environnement et stimuli

> *«En matière de télépathie, il n'y a pas encore l'automatique ! Alors... ou la pensée est mal émise, ou elle est mal reçue, ou c'est l'esprit de votre correspondant qui est occupé, ou alors - et c'est ce qui arrive le plus souvent - c'est votre propre esprit qui est en dérangement !»*
>
> — Raymond Devos

Nous avons choisi d'utiliser un environnement continu et non discret comme c'est le cas par exemple dans StarLogo [Resnick 1994b]. La discrétisation, ou pavage, de l'environnement est le plus souvent utilisé pour diminuer les coûts de calcul lors de la recherche des voisins d'un agent donné. En effet, afin de connaître les agents avec lesquels il doit interagir, chaque agent parcourt la liste des stimuli présents dans l'environnement et il n'interagit alors qu'avec les agents émetteurs de stimuli significatifs pour lui. Cette structure d'environnement nécessite de parcourir l'ensemble des stimuli significatifs pour un agent et donc même ceux qui ne sont pas présents dans son voisinage immédiat. Le pavage permet habituellement de diminuer cette recherche en ne laissant parcourir à l'agent qu'une zone restreinte de son environnement. Reynolds [Reynolds 2000], par exemple, présente un modèle d'environnement permettant de construire des sous-régions pour permettre aux agents de faire des requêtes locales afin de connaître leurs voisins. Dans OSCAR, chaque *agent-groupe* ayant lui même son propre environnement, seuls les agents contenus dans ce groupe interagissent entre eux. Le groupe interagit alors avec les agents présents dans le même environnement que lui. Cette modélisation d'*agents-groupes* qui possèdent leur propre environnement, permet ainsi de construire un équivalent à la notion de régions développée par Reynolds et de réduire ainsi le temps de calculs que doit effectuer chaque agent.

L'environnement a un double rôle. D'une part, il sert d'espace topologique et il permet ainsi aux agents de se *situer* à la fois dans leur environnement et par rapport aux autres. D'autre part, il sert de support de communication en véhiculant l'information émise par les agents. La notion de propagation de stimulus (vision éthologique) ou de champ de potentiel (vision physique) ou encore de gradient de potentiel (vision mathématique) est souvent utilisée pour faire communiquer des agents réactifs [Ferber 1995]. Chaque agent peut propager dans son environnement un nombre variable de stimuli, comme le font des fourmis lorsqu'elles diffusent des phéromones différentes selon leur état interne.

Chaque stimulus est caractérisé par sa *valeur* (maximale à l'endroit où se trouve l'émetteur) et sa *portée* (distance de propagation). Lors de l'initialisation d'un stimulus, il est possible de choisir la fonction caractérisant la propagation de cette valeur dans l'environnement de l'agent émetteur. La fonction choisie peut alors permettre de diminuer l'intensité de la valeur proportionnellement à l'éloignement de l'émetteur ou de maintenir constante cette intensité tant que la distance à l'agent est inférieure à la portée, etc.

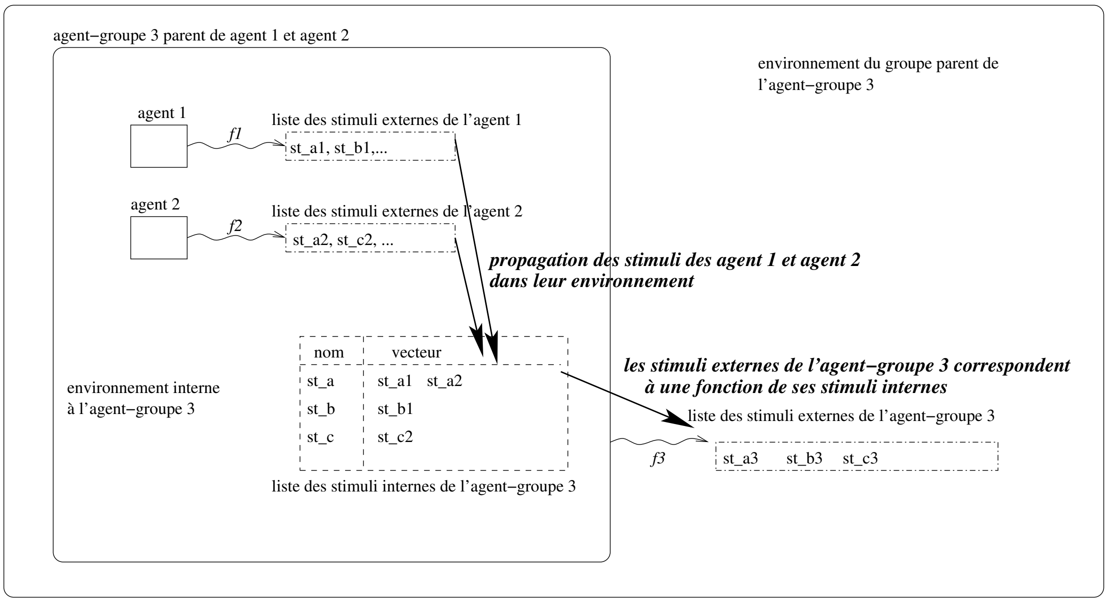


FIG. 5.4 – Diffusion des stimuli sur plusieurs niveaux d'environnement agent

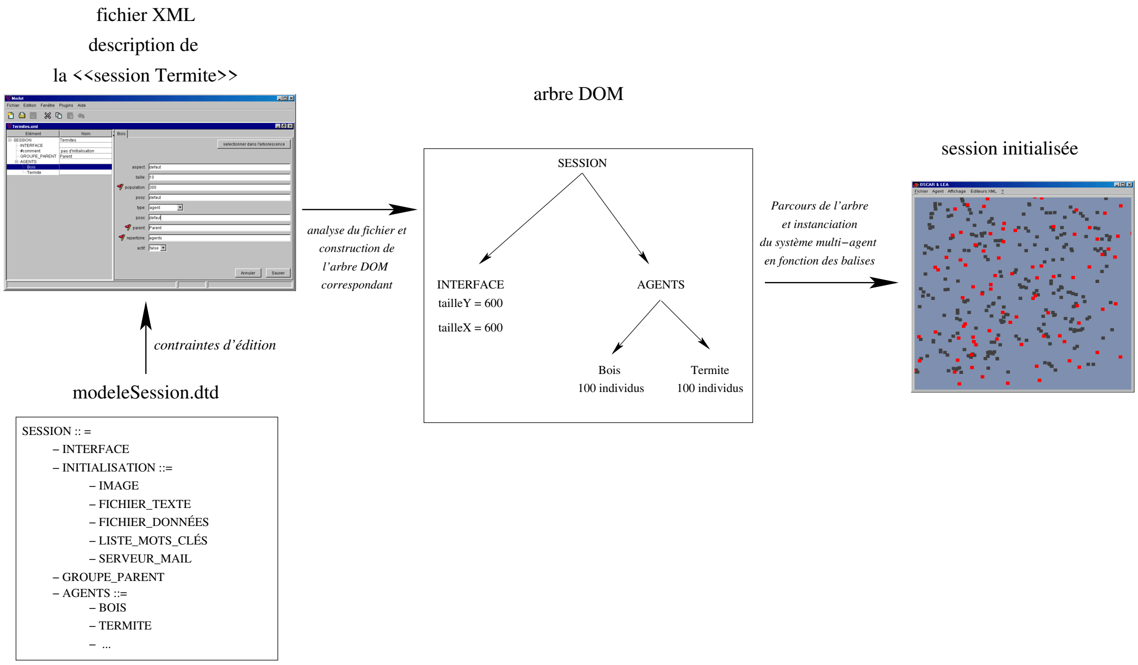
Lorsque des agents appartiennent à un groupe, leurs stimuli sont propagés dans l'environnement interne à ce groupe. L'*agent-groupe* construit alors ses stimuli externes en fonction de ces stimuli internes (FIG. 5.4). Selon la session, les stimuli externes peuvent correspondre à la moyenne, à la somme, à la valeur maximale ou minimale des stimuli internes.

Dans la fenêtre de l'interface, l'*environnement-parent* correspond alors au fond d'écran de la session. L'environnement est donc non seulement un moyen de communiquer entre agents mais aussi un moyen d'interagir avec l'utilisateur.

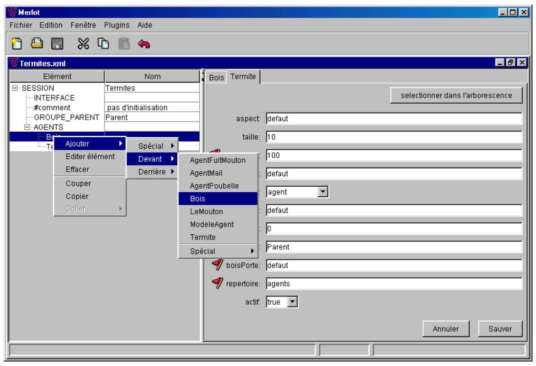
5.2.4 Initialisation de session

La DTD *modeleSession.dtd* (Annexe B) fournit la grammaire permettant de construire des fichiers de session valides et bien formés qui pourront être interprétés par le noyau multi-agent. Ce langage peut ainsi être considéré comme un *métalangage de description de session* (simulation et application) multi-agent. Des parseurs standard, tels que *Xerces* fournit par *Apache Software Foundation*, existent en Java afin de lire des documents XML et de fournir à l'application leur contenu et leur structure. L'analyse du fichier XML consiste à construire l'arbre DOM (Document Object Model) du fichier d'initialisation de la session. Une fois cet arbre construit, chaque balise est récupérée avec ses arguments et elle est donnée au noyau multi-agent afin d'initialiser la session.

Processus d'initialisation de la «session des Termites» :



Fichier d'initialisation de la «session des Termites» vu sous Merlot :



Des éditeurs existent afin d'aider à la création des fichiers XML. Ils permettent à l'utilisateur de visualiser l'arbre XML, de lui proposer les balises qu'il peut rajouter et de visualiser ses erreurs. L'intérêt de ces éditeurs sont qu'ils ne sauvegardent un fichier XML que si ce dernier respecte la grammaire (DTD) et les règles syntaxiques de XML. L'application qui lira ces fichiers est donc garantie de la validité du fichier qu'elle reçoit en entrée.

Fichier d'initialisation de la «session des Termites» :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE SESSION SYSTEM "..\\sessions\\modeleSession.dtd">
<SESSION name="Termites" date="septembre2001"
         commentaire="modèles des termites qui regroupent des morceaux de bois"
         auteur="Valérie Renault">

  <!-- Définition de la taille de la fenêtre à l'écran ->
  <INTERFACE tailleY="600" tailleX="600"/>

  <!-- Définition de l'agent Parent, la taille correspond à la taille de l'environnement Parent ->
  <!-- dans lequel vont évoluer les agents. ->
  <GROUPE_PARENT tailleY="600" tailleX="600" population="1" name="Parent" aspect="none" actif="true"/>

  <!-- Description des agents de la session : les attributs des balises ne comportent->
  <!-- que les paramètres définis dans la classe d'agent à instancier. ->
  <AGENTS>
      <!-- Création de 300 agents Bois, de taille 10, placés dans l'environnement Parent. ->
      <Bois type="agent" taille="10" repertoire="agents" population="300" parent="Parent" actif="false"/>

      <!-- Création de 100 agents Termites, de taille 10, placés dans l'environnement Parent. ->
      <!-- La valeur "defaut" indique au système qu'il faut qu'il utilise la valeur
           définie dans la Classe Termite. ->
      <Termite type="agent" taille="10" repertoire="agents" population="100" parent="Parent"
               direction="0" boisPorte="defaut" actif="true"/>
  </AGENTS>

</SESSION>
```

Afin de pouvoir relancer une session à partir d'un état donné, il est possible de la sauvegarder. Cette sauvegarde consiste à écrire un fichier équivalent au fichier XML d'initialisation de session avec les nouvelles valeurs des paramètres. Pour cela, il est nécessaire de parcourir les paramètres de la session, qu'il s'agisse des paramètres de l'interface ou des paramètres des agents. À la différence de l'exemple de session présenté ci-dessus, le fichier comporte une ligne par agent afin de décrire séparément ses paramètres (la valeur de population étant alors de 1).

Sauvegarde de l'initialisation de la «session des Termites» :

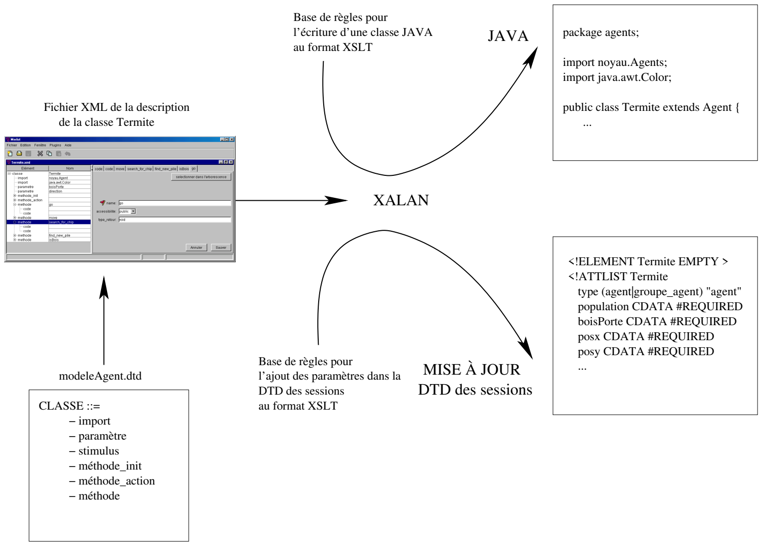
```
...

 <AGENTS>
      <!-- Création de 5 agents Bois, de taille 10, placés dans l'environnement Parent. ->
      <Bois type="agent" taille="10" repertoire="agents" posx="50" posy= "76"
            population="1" parent="Parent" actif="false"/>
      <Bois type="agent" taille="10" repertoire="agents" posx="46" posy= "258"
            population="1" parent="Parent" actif="false"/>
      <Bois type="agent" taille="10" repertoire="agents" posx="327" posy= "75"
            population="1" parent="Parent" actif="false"/>
      <Bois type="agent" taille="10" repertoire="agents" posx="121" posy= "324"
            population="1" parent="Parent" actif="false"/>
      <Bois type="agent" taille="10" repertoire="agents" posx="383" posy= "458"
            population="1" parent="Parent" actif="false"/>

      <!-- Création d'un agent Termite, de taille 10, placés dans l'environnement Parent. ->
      <Termite type="agent" taille="10" repertoire="agents" posx="109" posy= "541"
               population="1" parent="Parent"
               direction="0" boisPorte="defaut" actif="true"/>
...
 </AGENTS>
...
```

5.2.5 Création d'agent

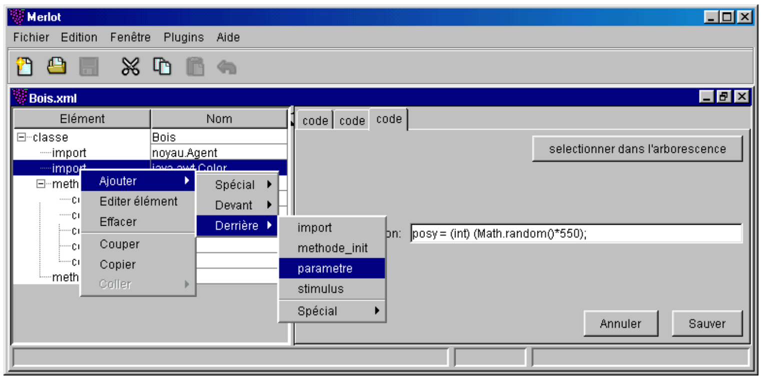
Comme pour la création de la session, il a été nécessaire de fournir une grammaire adaptée à la conception d'un nouveau type d'agent (Annexe C). Ce langage de création d'agents peut être considéré comme un *métalangage de description d'agents*.

XSLT (*eXtensible Stylesheet Language Transformation*) est un langage de transformation de documents XML. Il permet, d'une part de *re-structurer* un document XML en une autre arborescence et d'autre part de *formater* le document dans le nouveau format. Nous l'utilisons pour transformer les fichiers XML de description des classes agents en classes Java. Comme pour la description des sessions, l'avantage d'utiliser XML est de fournir à l'utilisateur une interface le guidant et le contraignant dans la construction des agents, de leurs méthodes et de leurs paramètres. Le code à l'intérieur de chaque méthode est écrit directement en Java, ce qui évite à l'utilisateur d'apprendre un nouveau langage de script. XSLT sert ensuite à traduire le code de ces agents en code interprétable par le noyau. Par exemple, pour chaque paramètre décrit en XML, XSLT génère automatiquement les méthodes *get*_NOM_PARAMETRE() et *set*_NOM_PARAMETRE() permettant ainsi au noyau multi-agent de retrouver facilement les paramètres.

Processus de création d'un nouveau type d'agent :



Fichier de création de l'agent Bois sous l'éditeur Merlot :



XSLT permet aussi de mettre à jour le fichier *modeleSession.dtd* afin que cette DTD accepte et reconnaisse les nouveaux agents que l'utilisateur vient de décrire et que le noyau puisse initialiser les paramètres de ces agents lors de l'ouverture d'une session. Une fois le code Java de la classe écrit, le système gère lui-même la compilation de cette classe.

L'utilisation de XML et de XSLT, à partir d'une grammaire donnée, par l'intermédiaire d'éditeurs spécifiques tels que Merlot, assure à l'utilisateur que le document qu'il vient d'écrire est à la fois bien formé et valide. L'éditeur informe l'utilisateur en cas contraire.

Les DTD étant adaptées au noyau multi-agent, l'utilisateur sera ainsi informé s'il construit une classe ou une session non conforme. Ces mécanismes offrent plusieurs avantages. Tout d'abord, cela permet de séparer la conception des agents et des sessions, en dissociant l'implémentation des méthodes de la gestion des valeurs de paramètres des agents. De plus, l'utilisateur peut modifier les paramètres des agents sans avoir besoin de modifier les classes Java et donc sans avoir besoin de recompiler le code des agents. Cette modification peut se faire très rapidement en modifiant les valeurs dans la session et en la relançant. Ces mécanismes offrent aussi la possibilité d'obtenir des fichiers de sauvegarde de la session qui ne nécessitent pas de traitement particulier lorsqu'il faut relancer la session à partir de l'état enregistré. Enfin, ces mécanismes ne nécessitent pas de la part de l'utilisateur l'apprentissage d'un langage de script spécifique à la plate-forme.

5.3 Conclusion

L'architecture d'OSCAR nous a permis de concevoir une plate-forme multi-agent générique et ouvert.

L'utilisation du langage XML permet d'aider l'utilisateur lors de la conception de nouveaux agents et de nouvelles sessions. L'introduction de nouveaux agents dans une simulation est guidée par la grammaire *modeleAgent.dtd*. La saisie de ce fichier à l'aide d'un éditeur XML, prenant en compte les règles de la grammaire, permet de ne générer que des agents valides pour la plate-forme. La construction d'une session est aussi facilitée et validée par des principes similaires.

La généricité autorise la construction de simulations comportant des agents réactifs et hétérogènes, comme par exemple la simulation des termites effectuant un tri collectif. Ce noyau est aussi suffisamment ouvert pour s'adapter à des interfaces de visualisation de données, comme nous allons le voir maintenant avec l'exemple du système LEA.

Chapitre 6

Learning E-mail Agents : visualisation de boîtes aux lettres électroniques

Sommaire

Le chapitre 4, en synthétisant les chapitres précédents, nous a permis de définir ce que nous attendons d'une *interface multi-agent de visualisation de données dynamiques*. Ce chapitre a ainsi donné lieu à la présentation de recherches menées d'une part sur l'introduction de sociétés d'agents dans des interfaces de visualisation, d'autre part sur le regroupement d'information par des agents réactifs. Cependant, dans le premier cas, les études portent sur le regroupement visuel de l'information par l'utilisateur au détriment des capacités d'interactions des agents. À l'inverse, les travaux menés sur le regroupement des données tirent profit des capacités d'interaction et d'organisation des agents, mais n'offrent pas à l'utilisateur les moyens nécessaires pour interagir avec les agents.

Nous allons maintenant, au travers de l'application LEA, proposer une interface multi-agent de visualisation de données dynamiques mise en œuvre à partir de la plate-forme OSCAR décrite au chapitre précédent. Cette application se base sur un double postulat. D'une part, l'organisation globale de l'interface repose sur les mécanismes d'interactions entre *agents-données*. D'autre part, l'interface, résultant de la visualisation de l'environnement des agents, permet à l'utilisateur de structurer et de synthétiser l'information tout en lui donnant la possibilité d'interagir avec les agents.

6.1 Choix d'une application

Afin de confronter nos hypothèses à des exemples concrets, les données doivent respecter les contraintes suivantes :

- être suffisamment nombreuses pour faire intervenir un nombre d'agents important (Chapitre 1 - PROPRIÉTÉ 1) ;
- posséder des relations entre elles pour permettre aux agents d'avoir des éléments d'information sur lesquels baser leurs interactions (Chapitre 1 - PROPRIÉTÉS 2 & 3) ;
- être issues de flux d'information dynamique et permettre ainsi de mettre à l'épreuve la réactivité des agents face à des modifications externes (Chapitre 1 - PROPRIÉTÉ 4) ;
- être manipulables par un utilisateur afin d'expérimenter les interactions entre l'utilisateur et les agents (Chapitre 1 - PROPRIÉTÉ 5) ;
- s'intégrer dans le cadre des recherches menées dans mon laboratoire d'accueil (France Telecom R&D) ;
- être facilement intelligibles en vue de pouvoir tester et valider notre approche auprès d'utilisateurs ;
- et surtout, être facilement disponibles.

Ainsi, plusieurs applications ont été envisagées au cours de mes trois années de thèse. La première application envisagée avec l'INRETS, reposait sur la mise en œuvre d'un système de suivi de l'état du trafic routier [Renault 1998]. L'idée était d'obtenir une interface destinée à deux types d'utilisateurs. D'une part, utilisée comme un système embarqué à l'intérieur d'un véhicule, elle aurait permis de guider et d'informer le conducteur sur les conditions de circulation et l'état de son véhicule. D'autre part, elle aurait eu un rôle d'écran de contrôle pour un contrôleur surveillant le trafic routier. Cependant, cette application se rapprochait plus d'un système d'alarme et devenait particulièrement «intéressante» lors de la survenue de problèmes dans le trafic. Nos travaux reposant sur la traduction de modifications progressives de flux de données en évolution d'organisation de sociétés d'agents, la détection de problèmes ponctuels lors du suivi de trafic ne se révélait pas tout à fait adaptée à notre problématique.

D'autres applications ont été envisagées lors des réunions de travail au Studio créatif de France Telecom R&D : le suivi des stocks dans une bibliothèque, la visualisation à distance de l'ambiance d'une maison (température, humidité, consommation d'électricité, etc.), la visualisation de données boursières ou encore la gestion d'un agenda virtuel.

Nous avons finalement convergé vers la visualisation de boîtes aux lettres électroniques. Cette application permet, en effet, de répondre à toutes les contraintes décrites précédemment et, de plus, s'intègre relativement bien aux recherches menées au sein du laboratoire.

6.2 Problématique

De nombreuses interfaces de messagerie existent dans le commerce (Microsoft Outlook, Eudora, Netscape Messager, etc.). Nous ne prétendons pas ici reproduire l'ensemble des fonctionnalités de ces interfaces. En effet, nous focalisons nos travaux sur la partie interface de visualisation des courriers reçus. Les fonctionnalités telles que la réponse à un courrier ou l'envoi automatique de messages d'absence ne font donc pas partie de notre propos.

Les interfaces existantes se composent principalement de trois parties (Fig. 6.1). La première partie représente l'arborescence des dossiers de l'utilisateur dans lesquels celui-ci peut conserver ses messages. La deuxième partie permet de lister les messages appartenant à un dossier. En général, le comportement par défaut de l'application est de permettre à l'utilisateur de consulter les messages reçus récemment. Enfin, l'interface est composée d'une dernière partie qui permet de lire un message sélectionné parmi ceux de la partie précédente.

Avec LEA (Learning E-mail Agents), nous proposons une alternative à ces interfaces

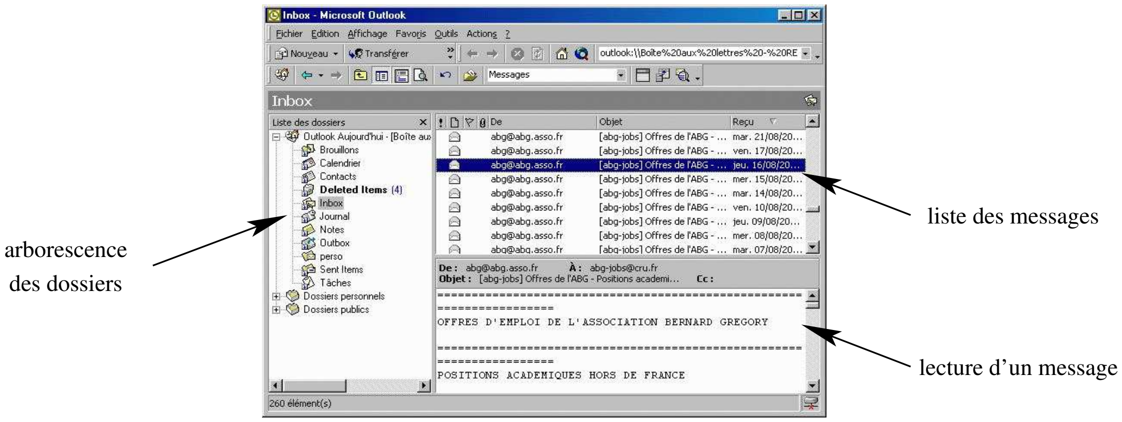


FIG. 6.1 – Photo d'écran d'Outlook

textuelles en visualisant les messages et les dossiers sous forme iconographique (FIG. 6.2). La partie «arborescence de dossiers» et «liste de messages» des interfaces habituelles se trouvent réunies dans une seule fenêtre. La partie permettant la lecture du message conserve, quant à elle, une interface plus standard. LEA a pour rôle d'aider l'utilisateur lors de la consultation de ses messages en lui fournissant un moyen de filtrer, d'organiser et de synthétiser l'information.

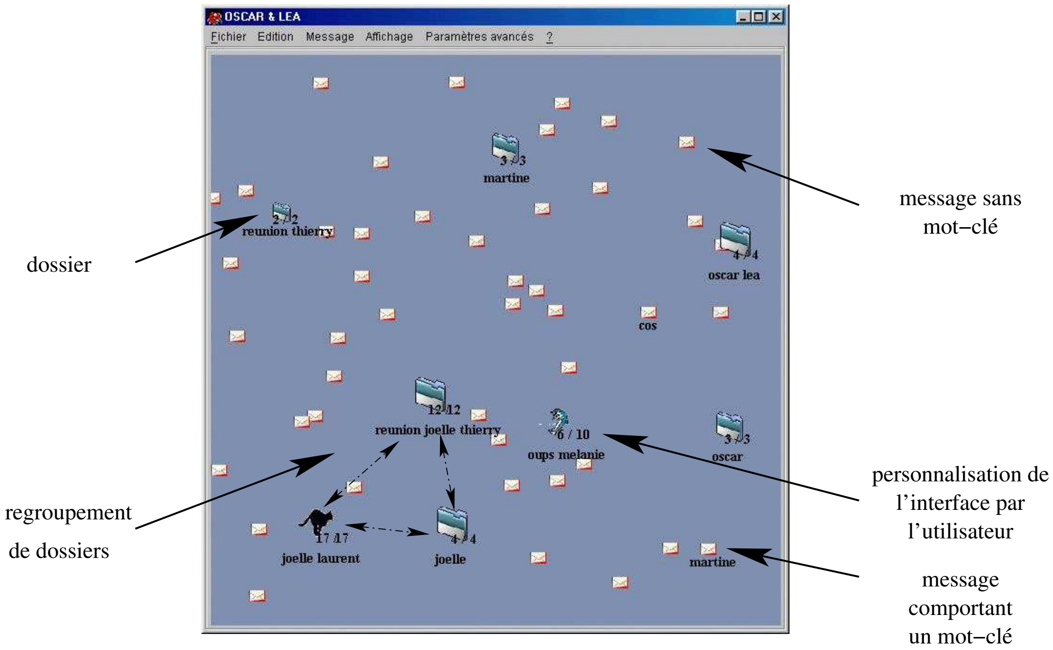


FIG. 6.2 – Photo d'écran de LEA

Nous allons maintenant présenter l'application LEA. Dans un premier temps, nous montrons comment les nouvelles classes d'agents de LEA s'intègrent à la plate-forme

OSCAR. Nous détaillons ensuite les comportements de ces agents en nous appuyant sur le déroulement d'une session de consultation de messages relativement simple.

6.3 Architecture : de OSCAR à LEA

LEA a besoin de certaines informations spécifiques liées au domaine de la messagerie électronique. À partir du noyau générique OSCAR, il est donc nécessaire de définir de nouvelles classes d'agents pour permettre à LEA d'accéder au serveur de messagerie, de prendre en charge les messages ou encore de regrouper certaines informations.

La figure 6.3 représente l'agencement des classes d'agents de l'application LEA par rapport aux classes de la plate-forme OSCAR (Fig. 5.3).

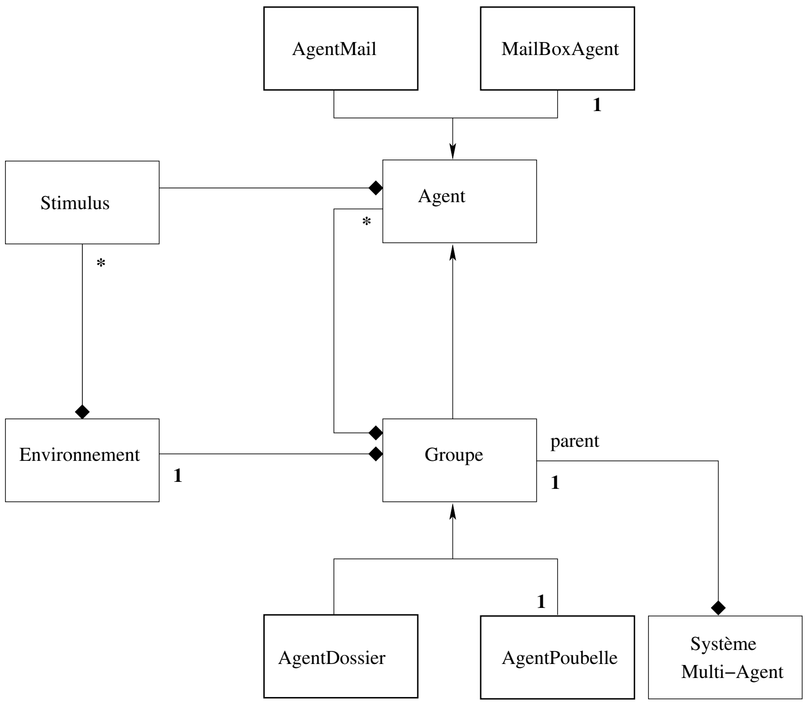


Fig. 6.3 – Schéma UML de LEA

Quatre nouvelles classes ont été ajoutées :

- la classe MailBoxAgent gère la connexion au serveur de messagerie et surveille l'arrivée de nouveaux courriers. Une seule instance, l'*agent-boîte-aux-lettres*, est nécessaire pour l'ensemble de la session ;
- la classe AgentMail prend en charge un message électronique. Il y a autant d'instances de cette classe que de messages. Ces agents sont désignés, dans la suite de ce mémoire, sous le nom d'*agent-message* ;

- la classe AgentDossier représente un dossier. Les *agents-dossiers* permettent de regrouper et de conserver certains messages ;
- la classe AgentPoubelle est un dossier particulier qui stocke les messages que l'utilisateur souhaite supprimer. Un seul *agent-poubelle* est généralement utilisé dans une session.

Nous allons maintenant détailler comment les comportements relativement simples de ces agents, issus des classes précédentes, permettent d'aboutir à une interface structurée et interactive de messagerie.

6.4 Comportements des agents

Ces comportements peuvent être décrits sous l'angle de la méthodologie Cassiopée [Collinot *et al.* 1996] qui permet de définir de manière incrémentale l'organisation des agents en considérant les niveaux suivants :
- le rôle individuel ;
- le niveau topologique ;
- le rôle social ;
- les groupes ;
- le niveau collectif.

Cette décomposition en plusieurs niveaux s'adapte parfaitement à un système de visualisation de données.

Le *rôle individuel* des *agents-messages* est de retranscrire chaque message à l'écran tel qu'il arrive sur le serveur, sans interprétation de sa signification. Chaque message est alors associé à un agent. Ce niveau correspond ainsi à l'arrivée d'un message en temps réel dans la session.

Au *niveau topologique*, le comportement de l'agent est de rendre le message visible à l'écran en s'affichant lui-même et en se positionnant dans son environnement. La méthodologie Cassiopée permet de prendre en compte les liens qui existent entre les messages. Ces liens construisent les relations qui pourront exister entre les agents et préfigurent ainsi les interactions futures. Les agents possèdent ainsi des mécanismes de communication avec lesquels ils peuvent émettre dans leur environnement des informations sur les mots-clés contenus dans leur message.

Le *rôle social* des agents est d'interagir avec les autres agents en fonction des similitudes ou des divergences entre les informations qu'ils portent et celles qu'ils reçoivent. Ainsi, par des mécanismes d'attraction et de répulsion, les *agents-messages* établissent progressivement des regroupements en fonction des informations qu'ils portent.

Les *groupes* sont alors créés à partir de ces regroupements, autour d'un mot-clé commun à tous les *agents-messages*. La formation des *agents-dossiers* permet ainsi, en visualisant un seul agent à la place d'un ensemble d'*agents-messages*, de réduire l'information à l'écran. Cet *agent-dossier* constitue une synthèse de l'information.

Au *niveau collectif*, les *agents-dossiers* peuvent, à leur tour, avoir leur propre comportement comme par exemple fusionner entre eux autour d'un mot-clé commun. Ils peuvent aussi interagir avec les *agents-messages* ou encore avec l'utilisateur.

Nous illustrons la suite de la description de LEA au moyen d'un exemple très simple composé de six messages. Cet exemple permet de visualiser l'ensemble des fonctionnalités présentées. Nous réduisons volontairement l'information en présentant un message comme étant composé seulement d'un expéditeur et d'un sujet.

Les pseudo-messages sont :

- *message 1 :*
  - *expéditeur : oscar <oscar@lip6.fr>*
  - *sujet : réunion de demain confirmée*
- *message 2 :*
  - *expéditeur : oscar <oscar@lip6.fr>*
  - *sujet : ordre du jour de la réunion*
- *message 3 :*
  - *expéditeur : lea <lea@lip6.fr>*
  - *sujet : présence confirmée*
- *message 4 :*
  - *expéditeur : lea <lea@lip6.fr>*
  - *sujet : réclamations*
- *message 5 :*
  - *expéditeur : oscar <oscar@lip6.fr>*
  - *sujet : nouveaux locaux*
- *message 6 :*
  - *expéditeur : lea <lea@lip6.fr>*
  - *sujet : réunion du 22*

6.4.1 Prise en charge de nouveaux courriers en temps réel

Accés à la boîte aux lettres

L'*agent-boîte-aux-lettres* a pour rôle de se connecter au serveur de la messagerie et de récupérer les messages qui s'y trouvent. Il ne fait que consulter les messages pour permettre leur visualisation, sans les supprimer du serveur. LEA peut ainsi fonctionner en parallèle avec n'importe quel autre système de messagerie (cela peut être utile puisque l'application actuelle ne permet pas de répondre aux messages).

L'*agent-boîte-aux-lettres* dispose de compétences spécifiques comme, par exemple, demander et vérifier le mot de passe de l'utilisateur. C'est aussi cet agent qui, lorsqu'il détecte un nouveau courrier, crée l'*agent-message* approprié puis rajoute ce nouvel agent à la session.

Détection des mots-clés

L'utilisateur donne une liste de mots-clés à LEA pour lui signifier ses centres d'intérêt. Cette liste est donnée dans le fichier XML de paramétrage de la session. Lors de l'ouverture d'une session, cette liste contient pour chaque mot-clé :

- le mot ;
- un poids servant à moduler son importance relative par rapport aux autres ;
- un icône permettant de représenter les messages dans lesquels il se trouve ;
- un icône permettant de représenter les dossiers dont il constitue le critère de formation.

La valeur du poids a un double rôle. Tout d'abord, c'est à partir de cette valeur qu'est calculée l'intensité du stimulus émis dans l'environnement. Ensuite, c'est en modulant cette valeur que des mécanismes d'apprentissage sont mis en place pour permettre à LEA de s'adapter aux centres d'intérêt de l'utilisateur.

Définition des mots-clés :

- *mot-clé 1 :*
  - *mot :* ***oscar***
  - *valeur : 250*
  - *icône de message : enveloppe*
  - *icône de dossier : dossier simple*
- *mot-clé 2 :*
  - *mot :* ***lea***
  - *valeur : 150*

- *icône de message : enveloppe*
- *icône de dossier : dossier simple*

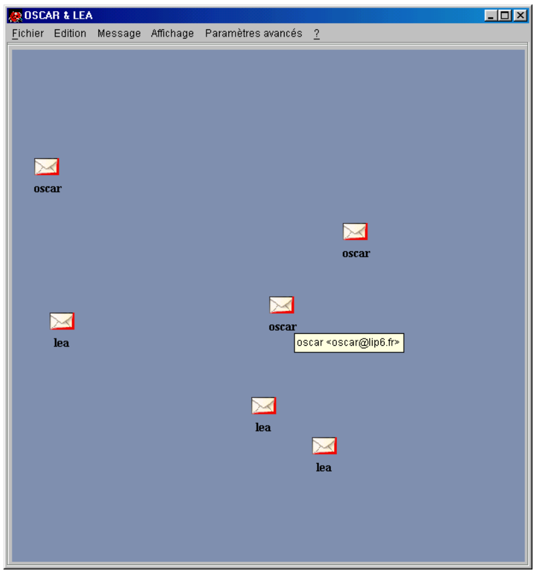
À tout moment dans la session, l'utilisateur a la possibilité de modifier «en ligne» les paramètres de ces mots-clés. Il peut aussi supprimer ou rajouter des nouveaux mots-clés. Ces possibilités font partie des moyens d'interactions avec l'utilisateur décrits dans la section 6.5.

Avant de placer un nouvel *agent-message* dans l'*environnement-parent* de la session, l'*agent-boîte-aux-lettres* parcourt le message et vérifie si un des mots-clés définis par l'utilisateur est présent dans le message. La détection de mot-clé peut se faire aussi bien dans l'expéditeur, dans le sujet, dans la date ou dans le corps du message. Par contre, dans la version actuelle du système, les fichiers attachés ne sont pas pris en compte et il n'existe pas de mécanisme d'extraction automatique de mots-clés.

À chaque mot-clé trouvé dans le message, un nouveau stimulus correspondant est rajouté à l'*agent-message*. Lorsque l'agent est visualisé, il est représenté par l'icône correspondant au mot-clé le plus important contenu dans le message ou bien par un icône par défaut lorsqu'il n'y a pas de mot-clé dans le message.

Visualisation des messages

Dès que l'*agent-message* est placé aléatoirement dans l'*environnement-parent* par l'*agent-boîte-aux-lettres*, il peut commencer à interagir avec les autres agents. À partir de ce moment-là, il est aussi visible à l'écran. Par défaut, chaque *agent-message* est représenté graphiquement par une enveloppe.



Visualisation des messages : *Chacun des six messages, envoyés par Lea et Oscar, est pris en charge par un* agent-message. *Graphiquement, ce dernier est représenté par une enveloppe sous laquelle le mot-clé s'affiche. L'adresse d'Oscar apparaît lorsque la souris est déplacée sur un des messages qu'il a envoyé.*

Des informations supplémentaires peuvent être représentées telles que la présence ou non de mots-clés dans le message ou le fait que le message ait été lu on non. Le passage de la souris sur l'agent peut aussi permettre de visualiser l'expéditeur.

6.4.2 Mécanismes d'attraction et de répulsion des agents et regroupement d'informations

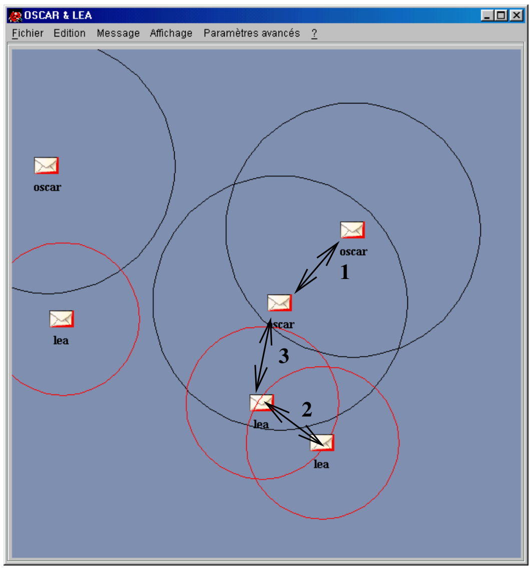
Comme nous l'avons montré dans les chapitres 3 et 4, il existe de nombreux modèles multi-agents de regroupement d'éléments, comme par exemple celui des termites. Ce modèle nécessite cependant deux types d'agents : des agents «à trier» et des agents «trieurs». D'autre part, les modèles proies-prédateurs ont montré l'intérêt que peuvent avoir les phéromones virtuelles dans un environnement simulé. Comme nous l'avons montré au chapitre 4, nous proposons donc ici des modèles d'organisation de données reposant d'une part, sur le déplacement des *agents-données* eux-mêmes et d'autre part, sur les interactions entre ces *agents-données*.

Pour chaque mot-clé qu'il possède, l'agent émet un stimulus local dont l'intensité correspond à l'intérêt que porte l'utilisateur à ce mot. Plus l'intérêt de l'utilisateur est important (i.e. le poids du mot-clé est important), plus le stimulus va être propagé dans l'environnement. Chaque agent peut donc détecter, dans son voisinage, les agents porteurs de messages ayant des mots-clés identiques aux siens. L'agent calcule ainsi son déplacement en fonction des stimuli qu'il reçoit : il est attiré par les agents qui propagent les mêmes stimuli et repoussé par les stimuli différents (ALGORITHME 6.1).

ALGORITHME **6.1** Comportement au niveau individuel des agents

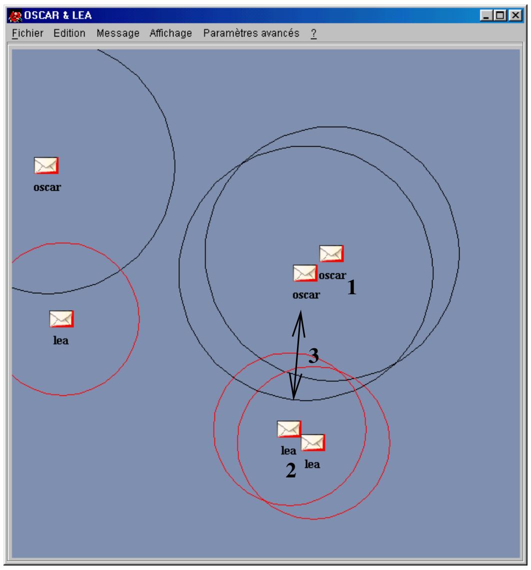
```
Si l'agent détecte des stimuli identiques au sien Alors
    remonter le gradient
Si l'agent détecte des stimuli différents au sien Alors
    descendre le gradient
```

Ainsi, les messages ayant des points communs s'attirent mutuellement à l'écran. Inversement, les messages émettant des stimuli différents se repoussent et s'éloignent donc à l'écran.



Propagation des stimuli et regroupement : *Le message 1 contient le mot-clé «oscar» dans l'adresse de son expéditeur. L'agent, ayant en charge ce message, diffuse le stimulus correspondant à ce mot-clé dans son environnement. Ce stimulus est émis avec une intensité et une portée de valeur équivalente au poids du mot-clé. Les stimuli émis par les* agents-messages *dont les messages sont expédiés par Oscar ont une plus grande intensité et une plus grande portée que ceux de Lea. (À noter : Les cercles autour des agents permettent de visualiser la limite de propagation des stimuli. Cette visualisation est optionnelle.)*

- ***1*** *: indique l'attraction entre les deux messages émis par Oscar ;*
- ***2*** *: indique l'attraction entre les deux messages émis par Lea ;*
- ***3*** *: indique la répulsion entre deux messages ayant des stimuli différents.*



Regroupement par attraction et répulsion : *Les agents suivent les gradients d'attraction et de répulsion :*

- ***1*** *: les deux messages émis par Oscar se rapprochent ;*
- ***2*** *: les deux messages émis par Lea se rapprochent ;*
- ***3*** *: les deux messages émis respectivement par Oscar et Lea se repoussent.*

6.4.3 Formation de groupes d'agents et synthèse d'informations

Formation d'agents-dossiers

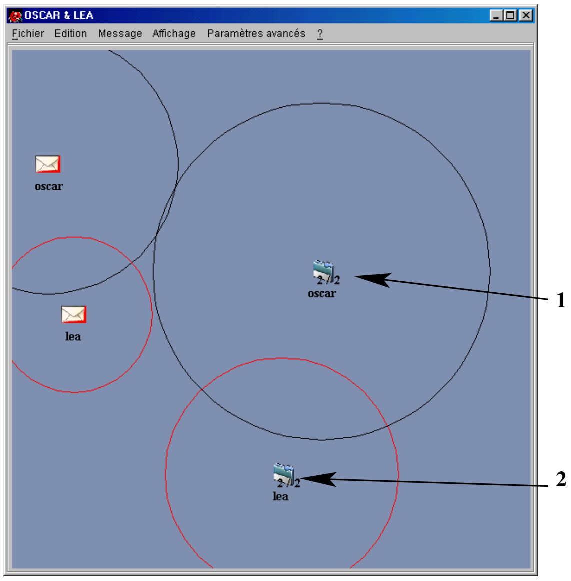
Lorsque le nombre de messages devient trop important, le nombre d'agents à visualiser à l'écran augmente, il n'est alors plus possible, pour des raisons d'ergonomie et de lisibilité, de les représenter tous sur le même écran. Il est donc nécessaire de doter le système de mécanismes de synthèse d'informations. Notre solution est de créer des groupes d'agents autour de stimuli communs. Lorsque deux agents sont suffisamment proches et qu'ils possèdent des mots-clés communs, ils construisent un groupe dans lequel ils vont alors se déplacer et évoluer (ALGORITHME 6.2). L'architecture d'OSCAR permet de considérer les groupes comme des agents. Le groupe va diffuser dans son environnement parent le stimulus commun aux agents ayant conduit à sa création. L'intensité de ce stimulus augmente alors proportionnellement avec le nombre d'agents contenus dans le répertoire. Lorsque de nouveaux agents, ayant le même mot-clé, arrivent suffisamment près du groupe, ils y entrent à leur tour.

ALGORITHME **6.2** Formation d'un *agent-dossier* par un *agent-message*

Si l'*agent-message* détecte un stimulus identique au sien **Alors**
  Si la distance à l'*agent-émetteur* $<$ seuil-formation-dossier **Alors**
    Si l'*agent-émetteur* est un *agent-message* **Alors**
      création d'un *agent-dossier* ayant pour critère le stimulus commun
      déplacement de l'*agent-émetteur* dans l'*agent-dossier*
      se déplacer dans l'*agent-dossier*
    Sinon
      <!– l'*agent-émetteur* est un *agent-dossier* ->
      se déplacer dans *l'agent-dossier*
  Sinon
    <!– la distance à l'*agent-émetteur* $>$ seuil-formation-dossier ->
    remonter le gradient
Si l'agent détecte des stimuli différents au sien **Alors**
  descendre le gradient

Cependant, avec la plate-forme OSCAR, tous les stimuli des agents contenus dans un groupe sont propagés dans l'environnement du groupe. Si ce mécanisme est appliqué tel quel, il conduit à la formation récursive de groupes dans chaque groupe selon le critère servant au regroupement des deux messages. Pour éviter cette boucle infinie, le stimulus permettant la formation du groupe n'est pas diffusé à l'intérieur du groupe, ni dans les niveaux inférieurs de ses fils.

Au niveau de l'application à la messagerie, un groupe correspond à la création d'un répertoire permettant de stocker les messages possédant un point commun. Graphiquement, ces agents sont représentés par un dossier. Cette représentation graphique peut contenir d'autres informations telles que le nombre de messages présents dans le groupe ainsi que le nombre de messages non lus. L'augmentation du nombre de messages dans le groupe provoque une augmentation relative de sa taille graphique à l'écran jusqu'à atteindre une limite maximale.



Formation d'agents-dossiers :

- ***1*** *: Les deux* agents-messages *se sont trouvés suffisamment proches pour former un* agent-dossier *correspondant au mot-clé «oscar». «2/2» indique que ce dossier contient alors deux* agents-messages *(les deux messages écrits par Oscar) et que aucun de ces messages n'a été lu ;*
- ***2*** *: De la même façon, les deux agents ayant à leur charge les messages émis par Lea ont créé leur propre dossier. Il contient deux messages, tous les deux ayant été lus (0/2).*

Grâce à ce mécanisme, les messages sont organisés dans une structure hiérarchique visuelle, de la même façon que les fichiers des ordinateurs sont organisés dans une arborescence. Le répertoire constitue alors une représentation synthétique de l'ensemble des éléments qu'il contient. Représenté visuellement par une seule entité graphique, il permet ainsi de réduire et de structurer l'information à l'écran.

Fusion de groupes

La diffusion de stimuli locaux peut conduire à la formation de deux groupes suivant le même critère à deux endroits différents de l'environnement. Lorsque ces deux *agents-dossiers* se détectent, ils se rapprochent l'un de l'autre et fusionnent en un seul dossier selon l'ALGORITHME 6.3.

De même que la formation de groupes, la fusion de groupes permet à son tour de structurer et de synthétiser l'information visuelle.

Il n'est pas nécessaire que l'ensemble des agents possèdent des stimuli globaux. En effet, pour un type de mot-clé donné et donc de stimulus donné, chaque agent peut avoir un stimulus local. Lorsqu'il y a peu d'agents, l'association des agents entre eux se fait alors visuellement par l'utilisateur qui associe les messages ayant la même représentation

ALGORITHME **6.3** Comportement de fusion lors de la rencontre de deux groupes

Si l'*agent-groupe* détecte un stimulus identique au sien **Alors**
  Si la distance au *groupe-émetteur* $<$ seuil-fusion-groupe **Alors**
    prendre tous les agents internes au *groupe-émetteur*
    supprimer ce *groupe-émetteur*
  Sinon
    <!– la distance au *groupe-émetteur* $>$ seuil-fusion-groupe ->
    remonter le gradient

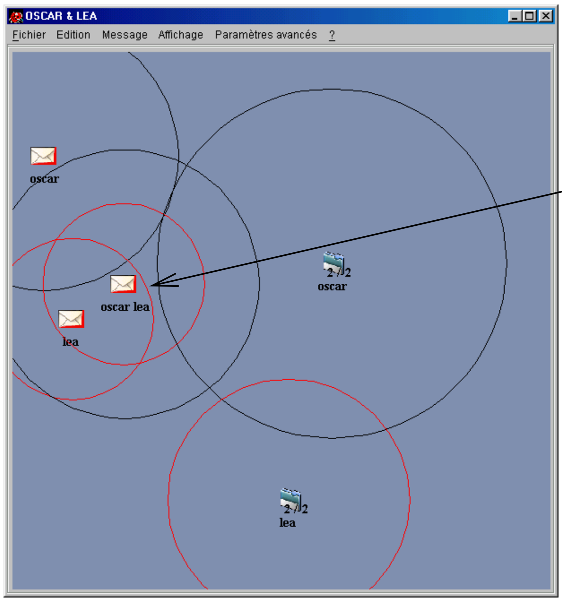
graphique. Comme dans [Proctor et Winter 1998], le système profite donc ici des capacités humaines à associer des informations de même forme ou de même couleur. Lorsque le nombre d'agents augmente, les agents ont plus de chance d'interagir et donc de former des groupes. L'intensité et la portée des stimuli d'un groupe étant proportionnelles au nombre d'agents contenus dans ce groupe, les groupes auront à leur tour plus de «chance» d'interagir et de fusionner.

6.4.4 Cas des messages ayant plusieurs mots-clés

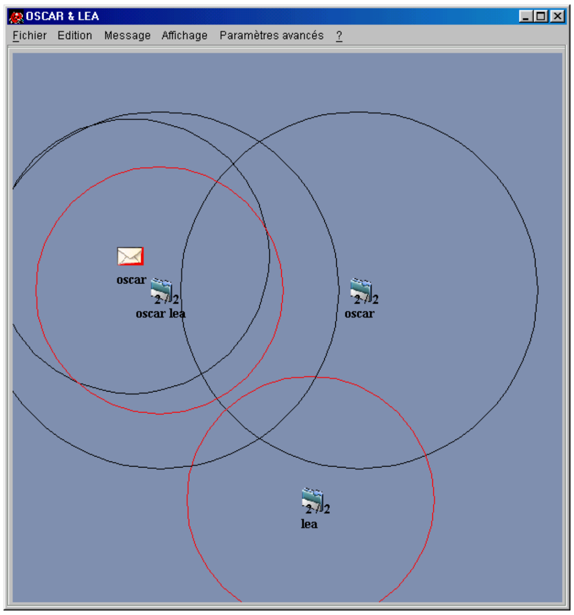
Nous avons jusqu'à présent décrit essentiellement les comportements des *agents-messages* lorsqu'ils ne comportent qu'un seul mot-clé. Il nous faut aussi envisager le cas des agents ayant plusieurs mots-clés. Lorsque un agent possède le mot-clé «lea» et le mot-clé «oscar», il peut être amené à choisir entre se déplacer vers le répertoire «lea» ou le répertoire «oscar». Ce choix pose le problème de la signification de l'appartenance d'un agent à plusieurs groupes distincts en même temps. En faisant abstraction d'une application particulière, un agent, capable d'appartenir à plusieurs groupes en même temps, peut soit :

- choisir un groupe aléatoirement ou en fonction d'un critère de priorité ;
- appartenir à l'un des groupes en ayant un clone de lui-même dans l'autre groupe, la suppression de l'un entraînant alors la suppression de l'autre ;
- se dupliquer, chaque copie allant dans un groupe, la suppression de l'un n'entraîne alors pas la suppression de l'autre ;
- enfin, créer un nouveau groupe ayant un critère correspondant à l'union des deux critères.

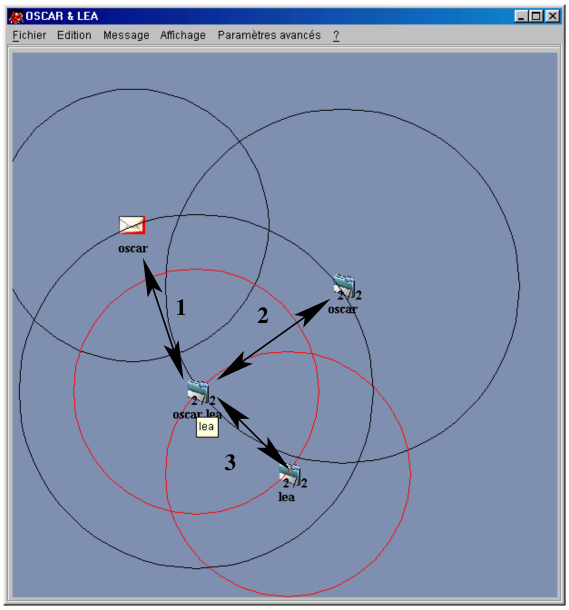
Aucune de ces possibilités n'est *a priori* la meilleure dans toutes les circonstances. Même dans le cadre d'une application particulière, les utilisateurs peuvent vouloir regrouper leurs informations différemment. Dans le cadre d'une messagerie, certains vont préférer dupliquer les messages, d'autres vont créer un nouveau répertoire. Dans le cas de LEA, nous avons choisi de ne pas dupliquer un agent : lorsque un agent rentre dans un groupe, il propage ses stimuli dans son nouvel environnement, c'est-à-dire dans l'environnement de l'*agent-dossier* dans lequel il se trouve. Ce dernier effectue une *synthèse* des stimuli contenus dans son environnement et propage les stimuli issus de cette synthèse dans son *environnement-parent* (FIG. 5.4). Ainsi, l'information que l'*agent-dossier* contient aussi d'autres stimuli que celui ayant conduit à sa création, est conservée.



Introduction d'un message ayant deux mots-clés : *Un nouveau message arrive sur le serveur.*
message 7 :
– *expéditeur : oscar* <*oscar@lip6.fr*>
– *sujet : lea, viens-tu demain ?*
*Il est détecté par l'*agent-boîte-aux-lettres*, puis est pris en charge par un *agent-message* et il apparaît à l'écran. Il possède les deux mots-clés «oscar» et «lea». Il diffuse donc les deux stimuli dans l'environnement.*

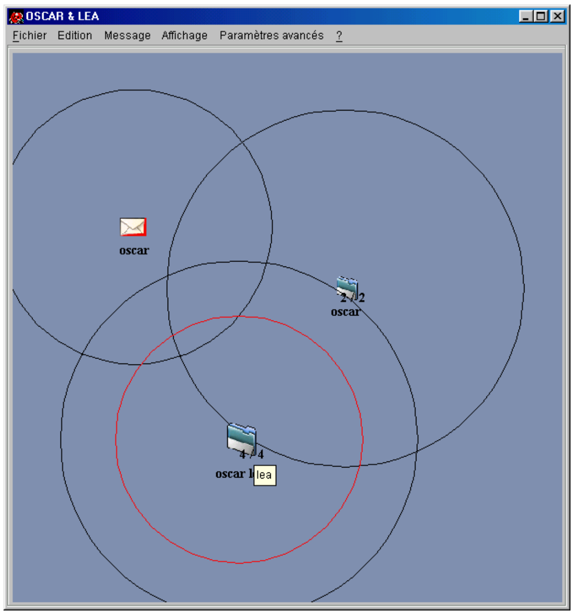


Formation d'un groupe : *À son arrivée, le nouvel agent est très proche de l'agent contenant le mot-clé «lea», il forme alors un* agent-dossier *avec lui. Ce groupe a comme critère de formation «lea», mais il diffuse dans son environnement aussi bien les stimuli de «lea» et de «oscar». Seuls les* agents-messages *ayant le mot-clé «lea» pourront se déplacer dans ce groupe. De même, seuls les* agents-dossiers *ayant le mot-clé «lea» comme critère de formation pourront fusionner avec ce groupe.*



Recherche de stabilité :

- ***1* :** *Les agents se repoussent car ils émettent respectivement des stimuli «oscar» et «lea». Mais en même temps, ils s'attirent car ils émettent aussi tous les deux le mot-clé «oscar». Ce processus d'attraction-répulsion introduit donc un déplacement des agents jusqu'à ce qu'ils trouvent une meilleure position ;*
- ***2* :** *(idem que 1) ;*
- ***3* :** *Les situations d'attraction et de répulsion sont pratiquement les mêmes que dans les deux cas précédents, seulement le critère «lea» a conduit à la formation de ces deux groupes. Ces deux* agents-dossiers *ont donc la capacité de fusionner.*



Fusion de groupes : *Les groupes «lea» se sont rapprochés suffisamment pour fusionner et ne plus former qu'un seul dossier.*

6.5 Interaction avec l'utilisateur

Les sections précédentes nous ont permis d'organiser simplement nos agents sur la base de comportement d'attraction et de répulsion pour qu'émergent à l'écran une structuration et une organisation de l'information. Comme nous l'avons déjà signalé, l'application actuelle ne prétend pas répondre à toutes les demandes d'un utilisateur. Les fonctionnalités que nous avons choisi de développer offrent cependant un éventail assez large des actions possibles que peuvent effectuer des agents réactifs dans une interface de visualisation. Il est maintenant nécessaire de doter nos agents de capacités supplémentaires d'interaction pour qu'ils puissent aussi interagir avec l'utilisateur et qu'ils s'organisent ainsi en fonction de ses centres d'intérêt.

6.5.1 Actions élémentaires

L'utilisateur dispose de plusieurs possibilités d'interactions à l'aide de la souris sur les *agents-messages* et les *agents-groupes*.

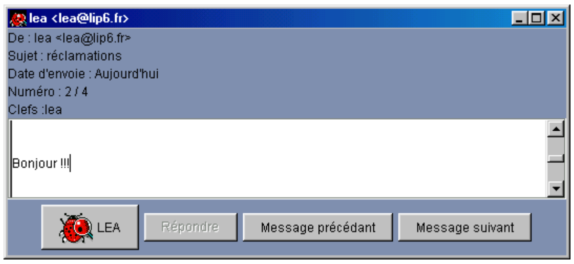
Il peut, par exemple, déplacer un agent (message ou groupe) avec la souris afin de le rapprocher d'autres informations ou de le positionner à un endroit particulier. Ensuite, s'il ne veut plus que cet agent se déplace, il peut le bloquer en le *figeant* à un endroit précis de l'écran. L'agent continuera ainsi à émettre des stimuli mais ne réagira pas lui-même aux stimulations des autres agents. Les autres agents pourront toujours s'approcher de lui pendant qu'il restera immobile. Ce mécanisme permet ainsi d'*ancrer* une information à l'écran. Lorsque tous les mots-clés possèdent un groupe ancré dans l'environnement, LEA ressemble alors à un logiciel de cartographie (chapitre 1.3.1).

Lorsque l'utilisateur déplace sa souris sur un agent (message ou dossier), un *tooltip* indique la liste des mots-clés ou, à défaut, l'adresse de l'expéditeur.

L'utilisateur, pour comprendre les liens entre les messages, peut aussi demander à visualiser les stimuli émis par les agents. Cependant, cette fonctionnalité devient rapidement inutilisable lorsque le nombre de stimuli émis devient important.

Chaque agent possède également un menu contextuel. Dans le cas d'un *agent-message*, ce menu permet de demander la lecture du message ou de l'envoyer à la poubelle. Dans le cas d'un *agent-dossier*, il offre la possibilité d'ouvrir le groupe ou de supprimer dynamiquement le mot-clé ayant conduit à la formation du groupe.

L'utilisateur a bien sûr la possibilité de lire un message en cliquant dessus ou en passant par le menu contextuel. Une nouvelle fenêtre apparaît alors afin de visualiser le corps du message.



Lecture d'un message : *L'utilisateur peut afficher un message envoyé par Lea en cliquant dessus. Le mot-clé «lea» contenu dans ce message est indiqué.*

À partir de cette fenêtre, l'utilisateur a aussi la possibilité de sélectionner des mots du corps du message afin de les rajouter comme mot-clé de la session. Nous allons voir maintenant comment ces ajouts sont gérés.

6.5.2 Introduction et suppression de mots-clés

Comme nous l'avons déjà signalé, LEA ne possède actuellement pas de système de détection automatique de mots-clés. Ceux qui sont présents à l'initialisation de l'interface, sont fournis dans le fichier XML d'initialisation. L'ajout et la suppression de mots-clés peuvent se faire dynamiquement sur demande de l'utilisateur. Celui-ci peut accéder à la fenêtre de configuration soit à partir de la sélection de mots dans le corps d'un message, soit par les menus.

Fenêtre de modification de mots-clés : *L'utilisateur peut modifier les poids des mots-clés «oscar» et «lea», ainsi que leur apparence à l'écran.*

Cette fenêtre permet donc de supprimer des mots-clés ou d'en rajouter de nouveaux, mais aussi de spécifier leur poids et la représentation graphique des *agents-messages* et des *agents-dossiers* qui les contiennent.

Introduction de nouveaux mots-clés

Lorsqu'un utilisateur rajoute dynamiquement un mot-clé, il peut choisir entre deux alternatives :

- soit il conserve les dossiers déjà existants et la formation de nouveaux dossiers ou sous-dossiers contenant ce mot se fait alors *dans* les groupes déjà existants ;
- soit il réinitialise l'ensemble de sa session : les agents sont repositionnés aléatoirement sur l'écran et ils forment des dossiers comme lors de l'initialisation d'une nouvelle session.

Le choix de l'une ou l'autre de ces solutions repose sur le nombre de mots-clés à rajouter en une seule fois et sur le nombre de messages comportant ces mots. En effet, si l'ajout de mots-clés risque d'entraîner de nombreuses modifications, l'utilisateur a tout intérêt à réinitialiser l'ensemble de son système.

Lorsqu'un agent possède plusieurs mots-clés, sa représentation graphique correspond au mot-clé ayant le plus fort poids.

Suppression de mots-clés

La boîte de dialogue que nous venons de voir pour ajouter un mot-clé peut aussi être utilisée pour modifier les poids des mots-clés ou les représentations graphiques des agents. L'utilisateur peut, par exemple, choisir d'associer un icône particulier pour représenter certains messages.

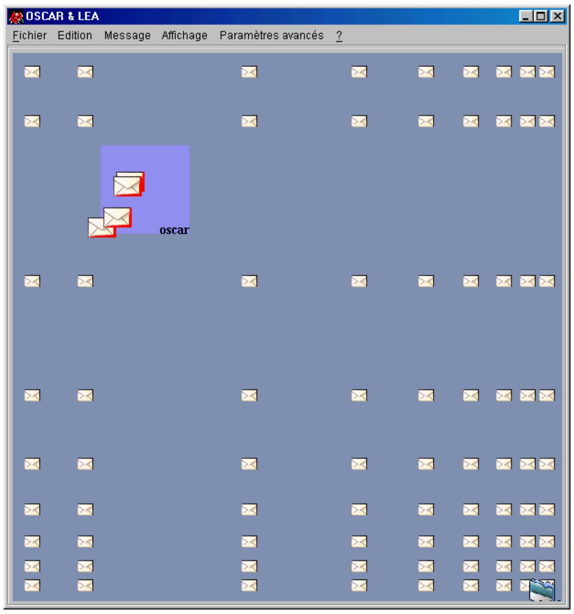
La suppression des mots-clés entraîne la disparition des dossiers basés sur ce mot-clé. Avant de «mourir» et donc ne plus être représenté à l'écran, un dossier déplace tous les agents qu'il contient et les positionne autour de lui dans son *environnement-parent*. Lorsque l'*agent-dossier* disparaît, l'utilisateur peut retrouver facilement les agents qu'il contenait. La suppression d'un mot-clé peut aussi se faire directement, sans passer par la boîte de dialogue, en utilisant le menu contextuel d'un des répertoires portant ce mot-clé.

6.5.3 Focus sur un groupe d'agents

L'utilisateur peut aussi cliquer sur un dossier afin de l'*ouvrir*. L'*agent-dossier* possédant son propre environnement, l'ouvrir revient donc à faire apparaître les agents qu'il contient. Ces agents peuvent à leur tour être organisés en sous-groupes. Visuellement, l'ouverture d'un groupe est associée à un mécanisme de Fisheye Views (section 1.4.1 et Annexe A). Les messages situés dans le même environnement que le dossier ouvert restent visibles. Ce mécanisme permet ainsi à l'utilisateur de garder le contexte présent autour du dossier ouvert. Les messages et les dossiers contenus dans le dossier sont visibles et l'utilisateur a la possibilité d'interagir avec eux (lecture, suppression).

Le principe de distorsion par Fisheye Views est particulièrement bien adapté à une vision individu centré : il suffit de considérer le point de focus comme l'*agent-dossier* à ouvrir. L'avantage de l'approche agent du Fisheye Views est que chaque groupe peut moduler sa distorsion et l'affichage de son environnement en fonction du nombre d'agents qu'il contient.

Ce mécanisme permet de se déplacer graphiquement dans l'arborescence visuelle formée par les répertoires. Le répertoire fermé est associé à un icône constituant une représentation synthétique de l'ensemble des éléments qu'il contient. À l'inverse, un répertoire ouvert permet de visualiser tous les messages contenus dans le groupe et ainsi d'accéder à un niveau de détail supplémentaire.



Mécanisme de distorsion avec le Fisheye View : *Le mécanisme de distorsion devient intéressant lorsqu'il y a beaucoup d'information affichée à l'écran. Dans cet exemple, à l'origine tous les* agents-enveloppes *sont alignés régulièrement. Lorsque le dossier Oscar s'ouvre, l'environnement est déformé afin d'afficher les agents contenus dans ce groupe, tout en gardant visible l'ensemble des* agents-enveloppes. *Dans cet exemple, le cœfficient de distorsion est de 1,5 (l'exemple est détaillé dans l'Annexe A).*

6.5.4 Agent «poubelle»

Nous avons vu jusqu'à présent comment arrive un message et comment il s'intègre à l'organisation déjà existante. L'*agent-poubelle* permet de supprimer des *agents-messages* et des *agents-dossiers*. Comme les poubelles de n'importe quel bureau virtuel, l'*agent-poubelle* est un répertoire particulier dans lequel l'utilisateur stocke les messages avant de les supprimer définitivement.

Cet agent émet en permanence un stimulus «poubelle» dans l'ensemble de la simulation. En fonctionnement normal, les autres agents ne sont pas sensibles à ce stimulus. Lorsque l'utilisateur sélectionne un message et indique qu'il veut le supprimer, l'agent porteur de ce message devient alors sensible au stimulus émis par la poubelle et seulement à celui-ci. Il se déplace ainsi jusqu'à l'atteindre. L'*agent-poubelle* a comme comportement de «récupérer» tous les agents qui se trouvent à la même position que lui. Ainsi, lorsque un message atteint la poubelle, il y est automatiquement déplacé.

Ce comportement de «récupération» a deux conséquences pour l'utilisateur : d'une part, il peut lui-même déplacer un message vers la poubelle, comme il a l'habitude de le faire sur son bureau d'ordinateur ; d'autre part, il peut aussi déplacer la poubelle elle-même. Dans ce deuxième cas, la poubelle récupérant tous les messages sur lesquels elle passe, acquiert alors le rôle de «gomme».

L'*agent-poubelle* étant un groupe, il possède lui-aussi son propre environnement. Comme pour les *agents-dossiers*, il est alors possible de l'ouvrir afin de visualiser les messages qu'il contient.

6.5.5 Apprentissage face aux actions de l'utilisateur

Les actions que peut faire un utilisateur traduisent, implicitement ou explicitement, ses centres d'intérêt à un moment donné. Par exemple, les changements dans les poids des mots-clés indiquent des changements d'intérêts de l'utilisateur.

Ces changements peuvent être faits soit directement par l'utilisateur lorsqu'il modifie les valeurs des mots-clés, soit par le système lui-même en introduisant des mécanismes de renforcement [Sutton et Barto 1998] à partir de certaines actions de l'utilisateur. Ainsi, lorsque l'utilisateur ouvre la fenêtre de lecture d'un message, cela signifie que le message est important. Le poids des mots-clés qu'il contient est augmenté de façon inversement proportionnelle à son importance (un mot-clé faible augmente plus son poids qu'un mot-clé important). Au contraire, lorsqu'un *agent-message* ou un *agent-dossier* est mis à la poubelle (dans l'*agent-poubelle*), le poids des mots-clés qu'il contient est diminué de façon proportionnelle à son importance (un poids faible est diminué faiblement, alors qu'un poids important est largement diminué). Ainsi, les poids de mots-clés associés à un message lu puis mis à la poubelle diminuent très faiblement. À l'inverse, ceux d'un message lu et gardé augmentent et ceux d'un message directement mis à la poubelle sont plus fortement diminués. Cette diminution du poids lorsqu'un mot-clé n'est plus utilisé ou qu'il reçoit des renforcements négatifs est proche du mécanisme d'effacement graphique progressif d'un icône à l'écran [Minar et Donath 1999] lorsque l'agent n'est plus sollicité.

6.5.6 Profil utilisateur et cartographie personnalisée

Les mécanismes d'apprentissage effectués lors de la lecture ou de la suppression de messages permettent de modifier dynamiquement les poids des mots-clés en fonction des actions de l'utilisateur à tout moment. Ces modifications permettent ainsi de suivre l'évolution des centres d'intérêt de l'utilisateur. Les mots-clés et leur poids constituent ainsi le profil de l'utilisateur.

À partir de l'évolution de ce profil, les agents mettent donc plus ou moins en évidence certaines informations. En effet, même avec une session initialisée par des mots-clés aléatoires, le système peut apprendre des actions de l'utilisateur, améliorer son organisation et personnaliser ainsi la représentation qu'il propose.

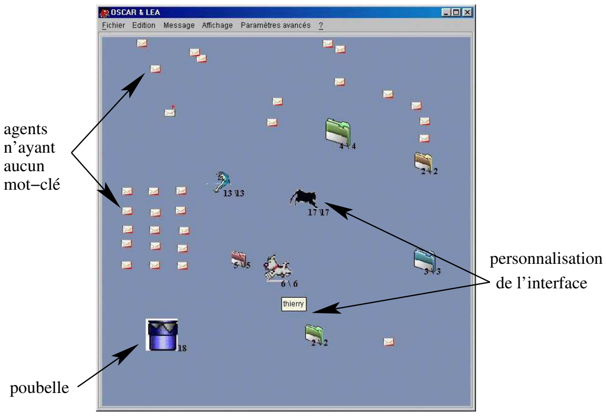
L'utilisateur peut, par exemple, positionner les agents à certains endroits de l'écran selon des repères qui lui sont propres. Il construit ainsi des zones thématiques sans que ces zones aient nécessairement un sens pour un autre utilisateur. À la différence des systèmes de cartographie, le système accepte les positions des agents telles qu'elles sont données par l'utilisateur. LEA permet à l'utilisateur de construire une sorte de cartographie dynamique et personnalisée de sa messagerie.

6.5.7 Sauvegarde de la simulation

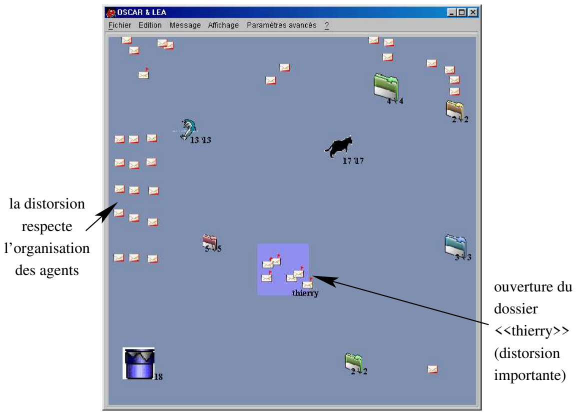
La personnalisation de l'interface ne trouve son intérêt que si, d'une session à l'autre, l'utilisateur peut retrouver ses messages et ses dossiers tels qu'il les avait laissés. Il faut donc que les agents mémorisent les messages dont ils ont la charge, ainsi que leur position dans l'environnement, etc. Ils ne doivent pas être réinitialisés à chaque ouverture de la session. LEA profite ainsi des fonctionnalités de sauvegarde d'OSCAR afin de construire un fichier XML de sauvegarde qui servira à initialiser la prochaine session. Lors de l'ouverture de cette nouvelle session, l'*agent-boîte-aux-lettres* n'a alors plus qu'à rajouter les messages arrivés depuis la session précédente.

6.6 Résultats

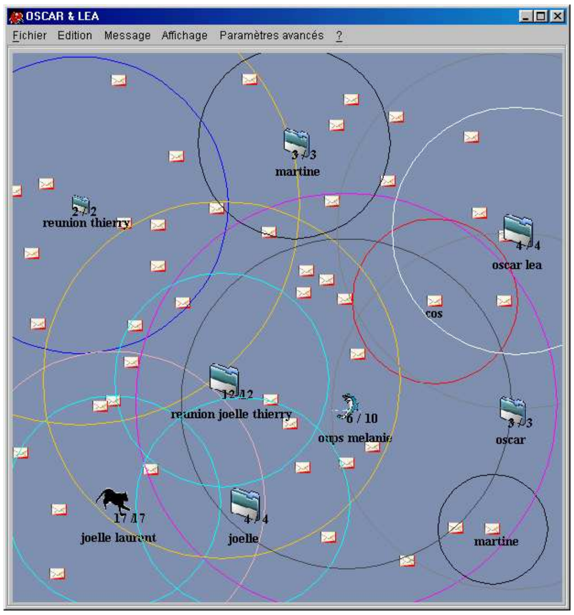
Les photos d'écran suivantes présentent deux exemples de résultats obtenus avec 100 messages électroniques.



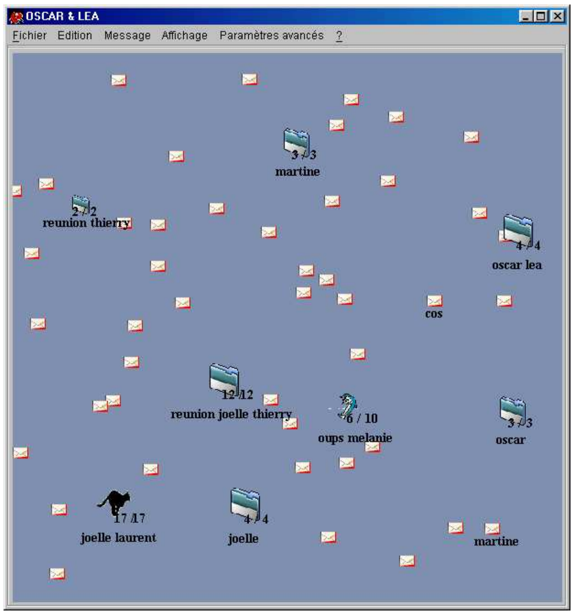
Exemple 1 (1) : *Les mots-clés ne sont pas affichés. L'utilisateur a remplacé les icônes par des images : ces images peuvent avoir une signification personnelle pour l'utilisateur, par rapport à un mot-clé. Les* agents-messages *sans icône particulier traduisent la nécessité d'introduire un système de détection automatique de mot-clé.*



Exemple 1 (2) : *L'ouverture du dossier «thierry» a entraîné une distorsion homogène de l'environnement.*



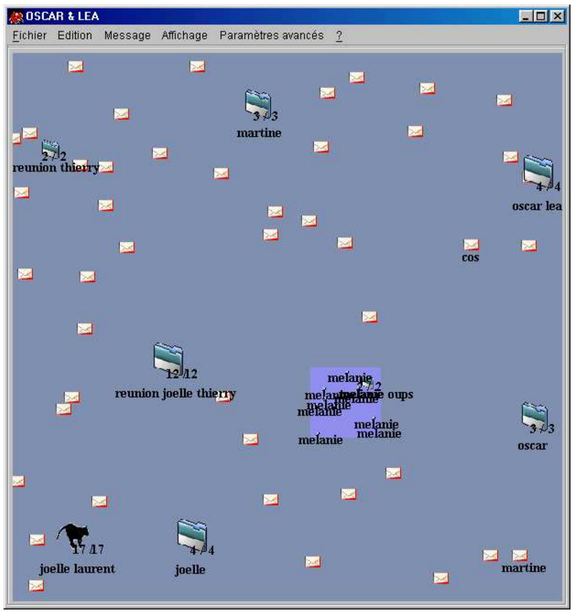
Exemple 2 (1) : *Cet exemple présente la visualisation d'une interface comportant 100 messages et 10 mots-clés, avec visualisation des stimuli.*



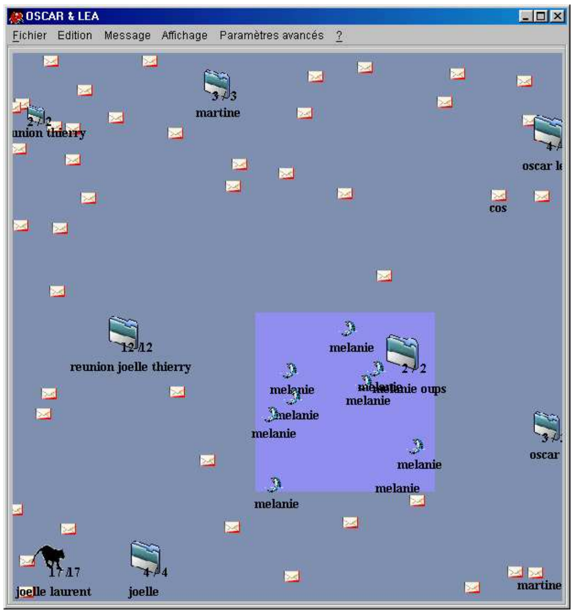
Exemple 2 (2) : *Même interface que l'image précédente mais sans visualisation des stimuli.*

Comme nous l'avons déjà signalé au cours de ce chapitre, LEA ne se prétend pas être une application complète pour la visualisation d'une messagerie électronique. En effet, il faudrait pour cela qu'elle traite d'autres informations spécifiques à ces systèmes telles que :

- les informations venant des fichiers attachés (type de fichier, taille, etc.) ;
- les informations indiquant si le destinataire est unique ou s'il s'agit d'un message collectif ;
- la priorité du message ;
- la taille du message ;



Exemple 2 (3) : *L'ouverture du groupe «mélanie» est effectuée avec un faible cœfficient de distorsion. À noter : la présence d'un autre groupe («oups») dans le groupe «mélanie».*



Exemple 2 (4) : *Même exemple avec un cœfficient de distorsion plus important.*

– la possibilité de savoir si le message fait partie du fil d'une discussion comportant plusieurs échanges entre les expéditeurs.

Cependant, LEA a été construite afin d'étudier l'émergence d'organisations visuelles grâce à des systèmes multi-agents et les caractéristiques ci-dessus n'apportent pas d'intérêt supplémentaire par rapport aux mécanismes agents sous-jacents. Les résultats obtenus sont tout à fait encourageants. En effet, en dehors de l'application propre aux messages électroniques, cette interface de visualisation permet de montrer que les agents sont capables de s'adapter aux modifications de leur environnement qu'il s'agisse de fluctuations des données (introduction de nouveaux messages, de nouveaux mots-clés, etc.) ou de modifications induites par l'utilisateur (lecture ou suppression de messages). Il nous faut maintenant poser le problème de la validation d'un tel système.

Conclusion

«Plus tard, à cette joie d'inventeur se mêle un peu de tristesse, le regret de n'avoir pas dit tout ce que l'on voulait dire. L'œuvre qu'on portait en soi paraît toujours plus belle que celle qu'on a faite. Tant de choses se perdent en ce voyage de la tête à la main !»

— Alphonse Daudet, *Contes du Lundi*, 1871

1 Synthèse des objectifs et résultats obtenus

Notre objectif principal était de construire une architecture multi-agent afin d'obtenir un certain nombre de propriétés aptes à constituer le noyau d'une interface de visualisation «idéale». Le chapitre 1 de ce mémoire nous a conduit à extraire certaines propriétés dont la présence nous a semblé nécessaire dans une interface de visualisation. Les chapitres 2 et 3 ont montré pourquoi les systèmes multi-agents nous semblaient être des candidats adaptés pour fournir ces propriétés, en particulier grâce aux mécanismes d'auto-organisation qui émergent des interactions des agents. Avec OSCAR et LEA, nous avons ensuite étudié comment mettre en place une société d'agents. Nous avons également montré comment, en fournissant à ces agents des mécanismes rudimentaires de communication et de prise en charge d'informations, ces agents étaient effectivement capables de construire une interface multi-agent de visualisation d'informations. Les résultats indiquent que les systèmes multi-agents réactifs se révèlent être des outils particulièrement pertinents pour résoudre des problèmes aussi complexes que la prise en compte en temps réel de données hétérogènes et distribuées, la mise en place d'une représentation visuelle structurée et synthétique de l'information, l'adaptation aux modifications de l'environnement ou la mise en place de mécanismes d'interactions. Ces résultats ont été possibles aussi bien grâce au noyau OSCAR que grâce à l'application LEA. En effet, la conception de la plate-forme générique multi-agent OSCAR permet de mettre en œuvre des simulations multi-agents mais aussi d'autres applications à la visualisation de données. Les simulations nous ont permis d'expérimenter des modèles réactifs d'organisations de sociétés d'agents, en particulier grâce aux modèles établis en éthologie autour des sociétés d'insectes. LEA, l'application de nos hypothèses à la visualisation dynamique de boîtes aux lettres électroniques, a démontré que ces hypothèses étaient opérationnellement réalisables. Nous allons maintenant analyser l'interface fournie sous l'angle des cinq propriétés initiales afin de voir comment la méthodologie agent se propose de répondre à ces critères.

2 Résumé des propriétés recherchées pour notre interface

2.1 Propriété 1 : données hétérogènes et distribuées issues de flux dynamiques d'informations

La propriété 1 concerne la capacité de l'interface à recevoir et à prendre en charge des données hétérogènes, distribuées et dynamiques. La réification des données et la répartition de l'information dans les agents permettent d'obtenir des *agents d'information*. Ces derniers peuvent posséder des paramètres internes ou des comportements propres aux données qu'ils portent sans que cela les handicape pour communiquer avec leurs congénères. L'hétérogénéité et l'origine distribuée des données se traduisent ainsi simplement dans l'hétérogénéité des agents et dans la diversité de leurs comportements. Un agent ne portant qu'une partie restreinte de l'information, la modification de cette information entraîne seulement des changements au niveau de cet agent. Les conséquences de cette modification sont ensuite répercutées sur l'ensemble du système par le jeu des interactions de cet agent avec les autres.

Les agents de LEA reçoivent, en temps réel, des informations du serveur de messagerie (i.e. de nouveaux messages) ou de la souris (i.e. des modifications de leur environnement par l'utilisateur). Ces informations sont intégrées dans l'ensemble de la dynamique du système multi-agent mais ne se répercutent que sur les quelques agents concernés (les nouveaux agents créés pour prendre en charge les messages ou la nouvelle position d'un agent indiquée par la souris). Les nouveaux *agents-messages* trouvent dynamiquement leur place dans l'organisation de la société et donc, dans la représentation visuelle, par le biais des interactions avec les autres. Les mécanismes permettant d'atteindre cette nouvelle organisation dépendent de la propriété 2.

2.2 Propriété 2 : mécanismes de structuration et d'organisation visuelle de l'information

Les modèles éthologiques fournissent aux systèmes multi-agents réactifs des modèles d'organisations pour leurs sociétés. Ces modèles permettent notamment de structurer cette société en se basant sur des comportements individus centrés, sans qu'il soit nécessaire de recourir à des processus centralisés de contrôle sur la structure de l'organisation. L'organisation de la société induit l'organisation des agents dans leur environnement. En représentant alors visuellement cet environnement, il est possible de visualiser la structuration des agents et donc celle des données qu'ils portent.

La visualisation de l'environnement des agents de LEA rapproche alors l'interface d'un logiciel de cartographie de données. La proximité des *agents-données* à l'écran s'adapte à la proximité sémantique des messages. La position d'un agent n'est alors pas dépendante du choix du concepteur à un moment donné, mais elle provient des comportements d'attraction et de répulsion des agents en fonction de leur similarité. Ainsi, les agents sont eux-mêmes «aptes» à permettre l'émergence d'une organisation visuelle. Cependant, lorsque les *agents-messages* sont trop nombreux, il n'est alors plus possible de les distinguer à l'écran et il devient alors nécessaire de mettre en œuvre la propriété 3.

2.3 Propriété 3 : mécanismes de synthèse (visuelle) de l'information

Comme nous l'avions indiqué au chapitre 1.5, cette propriété peut être obtenue par création d'une «hiérarchie visuelle» de l'information. La structuration de l'environnement des agents aboutit au regroupement d'éléments d'information proches et à l'éloignement d'éléments distincts. Ces processus de regroupements peuvent alors être considérés comme favorisant la formation de zones sémantiques dans l'écran. Lorsque ces zones deviennent trop nombreuses, il peut être nécessaire de les représenter en tant que nouvelle entité et, ainsi, de ne plus faire apparaître individuellement les éléments d'information. Ces changements dans le niveau de visualisation sont tout à fait comparables avec les changements de niveau organisationnel dans les sociétés d'agents : remplacer un ensemble d'informations par *une* information plus abstraite revient alors, en terme agent, à créer des *agents-groupes*, c'est-à-dire à passer d'un niveau micro à un niveau macro dans l'organisation. Chaque *agent-groupe* peut ensuite interagir avec d'autres agents et aboutir, de nouveau, à la formation de zones. Ainsi récursivement, les groupes peuvent être contenus dans de nouveaux groupes, chaque niveau de groupes permettant d'abstraire l'information contenue dans les agents. Visuellement, cela correspond alors à la création d'une hiérarchie visuelle.

Lorsque les *agents-groupes* de LEA sont créés, ils aboutissent à la création d'un *agent-dossier* contenant un ensemble de messages ayant un mot-clé en commun. Le dossier constitue alors visuellement une représentation synthétique d'un ensemble de messages. Représenté visuellement par une seule entité graphique, il permet de réduire et de structurer l'information à l'écran. L'ensemble des répertoires de LEA peut ainsi construire dynamiquement des arborescences telles qu'elles existent dans les messageries plus classiques.

Les trois propriétés vues jusqu'à présent reposent sur la dynamique du système mais aussi sur sa capacité à s'adapter aux modifications de son environnement.

2.4 Propriété 4 : capacité d'adaptation de l'interface due à sa relative autonomie

L'interface doit être capable de s'adapter aux modifications dynamiques de son environnement, que ces modifications proviennent des données ou de changements dans les centres d'intérêt de l'utilisateur. En effet, la propriété 1 requiert que les agents puissent prendre en charge des flux de données dynamiques, que des données issues de nouvelles sources d'information puissent être intégrées ou au contraire que d'autres disparaissent, etc. Afin de satisfaire la propriété 1, le système doit s'adapter à ces modifications sans que l'utilisateur ait besoin d'intervenir. De plus, les centres d'intérêt de l'utilisateur peuvent évoluer dans le temps. Les agents, grâce à leur autonomie, peuvent constamment remettre à jour les informations dont ils ont la charge mais aussi apprendre des actions de l'utilisateur.

L'entrée de nouveaux messages, leur prise en compte par l'*agent-boîte-aux-lettres* et leur intégration dans la structure de la société d'agents confèrent à LEA une réactivité et une adaptabilité en temps réel aux fluctuations de l'environnement. La limite actuelle de LEA repose sur l'absence de détection automatique de mots-clés, ce qui lui permettrait d'avoir une autonomie beaucoup plus grande face à l'utilisateur. Par exemple, LEA n'aurait plus besoin de ce dernier lorsque de nouvelles thématiques interviennent dans les messages. Cette limite est en partie compensée par une adaptation des paramètres des mots-clés déjà existants. En effet, la construction dynamique du profil de l'utilisateur au travers de ses actions sur les agents, à partir essentiellement de la lecture et de la suppression de messages, permet à LEA de mettre en évidence les messages auxquels s'intéresse l'utilisateur et, au contraire, de mettre les autres en arrière plan. S'adapter à l'utilisateur nécessite que l'interface puisse extraire suffisamment d'information de ses actions et donc qu'il existe une interaction constante entre l'utilisateur et la société d'agents.

2.5 Propriété 5 : mécanismes d'interactions avec l'utilisateur

L'interface se traduisant en la visualisation de la société d'agents, chaque agent possède donc une représentation graphique indiquant l'information qu'il porte. C'est par le biais de cette représentation que l'utilisateur peut agir avec un agent et accéder ainsi à certains éléments d'information. Le regroupement des agents ayant réduit l'information présentée à l'écran, des mécanismes de zoom centré sur chaque groupe permettent d'accéder à l'ensemble de l'information.

Avec LEA, l'utilisateur a évidemment la possibilité de demander à un agent d'afficher le message qu'il contient en cliquant sur cet agent. Il peut aussi, à l'aide de la technique de «Fisheye Views centré agent», parcourir l'arborescence des dossiers et retrouver les messages qu'ils contiennent.

Cette synthèse nous permet ainsi de montrer que les propriétés des systèmes multi-agents s'intègrent aux propriétés requises dans les interfaces de visualisation. L'énoncé de ces propriétés est venu d'une part de l'analyse des mécanismes perceptifs de l'être humain et de ses limites, d'autre part d'un état de l'art sur les systèmes existants actuellement dans le domaine de la visualisation d'information. Il nous faut donc maintenant revenir sur l'interface elle-même en temps qu'interface de visualisation afin d'étudier ce qu'elle apporte dans le domaine de la visualisation.

3 Bilan

3.1 Quels apports pour les systèmes de visualisation ?

Nos travaux se sont focalisés sur le rôle organisationnel que pouvaient apporter les systèmes multi-agents dans les processus de structuration d'un ensemble dynamique d'information à l'écran. Par la construction d'une «cartographie» dynamique et personnalisable d'une boîte de messagerie, LEA est une interface originale par rapport aux systèmes généralement utilisés.

Des expérimentations devront cependant être mises en œuvre afin de comprendre comment des utilisateurs peuvent intuitivement réagir face à une application comme LEA. Plusieurs hypothèses ont été envisagées au cours de ce mémoire sur la compréhension intuitive que peut avoir un utilisateur face à LEA.

L'utilisation de métaphores éthologiques lors de la conception du système permet de rendre sa description beaucoup plus intuitive. Nous pouvons ainsi nous interroger sur la nécessité ou non de présenter le système sous l'angle de cette métaphore. Comme nous l'avons vu au chapitre 3.3, cette métaphore peut avoir un aspect pédagogique non négligeable, mais l'utilisateur, plus habitué à un système global, centralisé et «intelligent», peut être gêné par ce type d'approche.

Comme dans les systèmes de cartographie, l'un des postulats est que la position topologique des *agents-données* dans leur environnement est interprétée en terme de similitude entre les données. Il est alors légitime de s'interroger sur la façon dont des agents qui se déplacent pour trouver une «meilleure» place ou pour former un groupe avec d'autres va être interprétée. Les utilisateurs vont-ils alors «figer» leurs *agents-dossiers* afin de créer des zones sémantiques prédéfinies ou sont-ils prêts à laisser une certaine autonomie au système ?

La question de l'autonomie d'une interface, et plus généralement d'un système informatique, et de son acceptation par un utilisateur se pose de plus en plus, notamment, avec l'essor des agents intelligents. Selon Lieberman [Lieberman 1997], les agents d'interface autonomes sont plus adaptés dans les situations où il n'y a pas de décisions critiques à prendre. En effet, dans ce dernier cas, les gens peuvent être inquiets de savoir si l'entité informatique ne risque pas de prendre la mauvaise décision. Les agents sont plus attendus pour faire des suggestions dans une situation donnée que pour prendre des décisions.

3.2 Tests utilisateurs

LEA a été testée et utilisée par des membres de l'équipe MIRIAD et ACASA du LIP6 et des membres de France Telecom R&D de juin 2001 à octobre 2001. Ces tests informels, sur un système en évolution constante, ont permis de mettre en évidence un certain nombre de points pertinents.

Une des premières réactions était de savoir si le système pouvait être couplé avec un système plus «classique» de messagerie : le système était alors vu comme un système complémentaire apportant des informations différentes d'un système habituel ainsi que de nouveaux moyens d'interactions. Par exemple, LEA apporte une vision globale de l'état général de la messagerie. Cette application permet aussi d'établir des «filtres» dynamiques grâce aux mécanismes d'apprentissage.

L'aspect graphique et l'introduction d'agents autonomes donnent un aspect «jeu» au système qui a séduit les utilisateurs. Cet aspect est accentué par le déplacement des agents lorsqu'ils recherchent une position stable et par la possibilité qu'a l'utilisateur de déplacer lui-même les messages.

Les limites soulevées reposent d'une part sur la non-prise en compte de la date dans l'interface actuelle mais surtout sur le besoin de coupler le système avec un système de détection automatique de mots-clés. Ce dernier besoin se ressent notamment pour éviter de laisser «traîner» dans l'interface des messages qui ne seraient pas reliés aux autres.

Il reste bien évidemment à valider les propriétés que nous avons réussi à obtenir dans le cadre d'un usage quotidien de ce type d'interface, ce qui ne pourra être fait que suite à des tests utilisateurs poussés (prévus à France Telecom R&D).

3.3 Retour sur les hypothèses

Nous pouvons cependant nous interroger sur les hypothèses initiales sur lesquelles reposent nos travaux et nous demander si elles sont valables pour l'ensemble des applications de visualisation d'informations. En effet, toutes ces propriétés ne sont peut-être pertinentes que dans des domaines applicatifs très complexes [Pavard 1999], c'est-à-dire trop complexes pour être traités par l'être humain en temps réel. En effet, peu d'applications ont été développées pour coupler les caractéristiques des systèmes multi-agents aux besoins des systèmes de visualisation. Il était donc nécessaire de commencer avec

LEA sur un domaine applicatif simple avec lequel nous pouvions contrôler l'ensemble des paramètres. LEA a donc constitué un premier pas vers des applications plus complexes telles que celles qui peuvent se développer autour du Web : visualisation de sites internet, visualisation d'informations circulant sur un réseau, etc.

La simplicité de l'application de LEA et la concurrence des outils existants dans le domaine de la messagerie posent problème lorsqu'il s'agit d'effectuer des tests sur le système. En effet, d'une part les utilisateurs ont déjà intégré les mécanismes de fonctionnement des outils de messagerie plus classiques, mais surtout ces outils sont généralement suffisants pour traiter des informations peu complexes telles qu'une liste de messages.

Dans le temps imparti, il ne m'a bien entendu pas été possible d'aborder simultanément la conception d'un système permettant d'obtenir les propriétés décrites ci-dessus et le développement d'une application dans un domaine où les données sont beaucoup plus complexes à traiter.

L'implémentation de LEA a conduit à vérifier et à expérimenter nos hypothèses. Cela nous permet donc d'envisager cette implémentation et ces résultats comme une base pour de nouvelles perspectives de travail.

4 Perspectives

Dans un premier temps, et comme nous venons de le signaler, des applications plus complexes devront être envisagées avec la même plate-forme OSCAR. En effet, d'une part ce noyau est suffisamment générique pour supporter diverses applications, d'autre part les systèmes multi-agents, de par leur modularité, permettent une conception incrémentale qui favorise la mise en œuvre de systèmes d'une grande complexité. D'un autre côté, il sera intéressant de diffuser largement la plate-forme OSCAR afin de mettre à l'épreuve la généricité de ce noyau et des grammaires sur lesquelles il repose. Cela devrait nécessiter quelques reformulations de ces grammaires et permettre ainsi d'aboutir à un métalangage de description XML de classe d'agent et de simulation multi-agent afin d'aboutir à un système encore plus ouvert à d'autres applications.

La complexité du système devrait aussi nécessiter un développement des capacités d'apprentissage, d'autonomie et de pro-activité des agents afin de pouvoir faire des suggestions à l'utilisateur à partir de certaines informations.

Enfin des expérimentations devront être menées à long terme afin d'analyser comment les utilisateurs *s'adaptent* à un système autonome qui *s'adapte* lui-même à eux.

Bibliographie

1 Références bibliographiques complètes

[Ahlberg *et al.* 1992] Christopher Ahlberg, Christopher Williamson et Ben Shneiderman. Dynamics queries for information exploration : an implementation and evaluation. In *Proceedings of the ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 619–626, Monterey, California. ACM Press, 1992.

[Ahlberg et Wistrand 1995a] Christopher Ahlberg et Erik Wistrand. Ivee : An environment for automatic creation of dynamic queries applications. In *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems, Demo Program*, Denver, CO. ACM Press, 1995.

[Ahlberg et Wistrand 1995b] Christopher Ahlberg et Erik Wistrand. Ivee : An information visualization and exploration environment. In *Proceedings International Symposium on Information Visualization*, Atlanta, GA. 1995.

[Andrews 2000] Keith Andrews. Information visualization. Technical report, Lecture Notes, Graz University of Technology, Mars 2000.

[Atkinson et Shiffrin 1968] Richard C. Atkinson et Richard M. Shiffrin. Human memory : A proposed system and its control processes. In K. W. Spence et J. T. Spence, editors, *The Psychology of Learning and Motivation : Advances in Research and Theory*, volume 2, pages 89–195. New York : Academic Press, 1968.

[Baddeley 1992] Alan Baddeley. Working memory : the interface between memory and cognition. *Journal of Cognitive Neuroscience*, 4(3) : 281–288, 1992.

[Baddeley 1994] Alan Baddeley. Les mémoires humaines. *La Recherche*, 25(267) : 730–735, Juillet-Août 1994.

[Bagot 1999] Jean-Didier Bagot. *Information, sensation et perception*. Collection Cursus, Psychologie. Armand Colin, Paris, France, 2ème édition, 1999.

[Beaudoin *et al.* 1996] Luc Beaudoin, Marc-Antoine Parent et Louis C. Vroomen. Cheops : a compact explorer for complex hierarchies. In *Proceedings on Visualization'96*, pages 87–92, San Francisco, CA. IEEE Computer Society, 1996.

[Berliner *et al.* 1964] C. Berliner, D. Angell et D.J. Shearer. Behaviors, measures and instruments for performance evaluation in simulated environments. In *Symposium and Workshop on the Quantification of Human Performance*, Albuquerque, New Mexico. 1964.

[Birk 1997] A. Birk. Autonomous Recharging of Mobile Robots. In *Proceedings of the 30th International Sysposium on Automative Technology and Automation*. Isata Press, 1997.

[Boesch et Boesch-Achermann 1991] Christophe Boesch et Hedwige Boesch-Achermann. Les chimpanzés et l'outil. *La Recherche*, 22(233) : 724–731, Juin 1991.

[Bonabeau *et al.* 1997] Éric Bonabeau, Guy Theraulaz, Jean-Louis Deneubourg, Nigel R. Franks, Oliver Rafelsberger, Jean-Louis Joly et Stéphane Blanco. The emergence of pillard, walls and royal chambers in termite nests. Technical report, Santa Fe Institute, working paper 97-04-033, 1997.

[Bonabeau et Theraulaz 1994] Éric Bonabeau et Guy Theraulaz, editors. *Intelligence collective*. Éditions Hermès, Paris, 1994.

[Boudes et Amaldi 1996] Nicole Boudes et Paola Amaldi. Le filtrage des informations dans le contrôle d'un environnement dynamique. In *8ième journées sur l'ingénierie de l'Interaction Homme-Machine*, pages 9–14, 1996.

[Bray *et al.* 1998] Tim Bray, Jean Paoli et C.M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 , W3C Recommendation. Technical Report REC-xml-19980210, W3C, 1998.

[Breton *et al.* 1999] Laurent Breton, Jean-Daniel Zucker et Éric Clément. Une approche multi-agents pour la résolution d'équations en physique des milieux granulaires. In *Ingénierie des systèmes multi-agents - JFIADSMA'99*, pages 281–293, La Réunion. Éditions Hermès, 1999.

[Breton *et al.* 2000] Laurent Breton, Jean-Daniel Zucker et Éric Clément. GranuLab : un laboratoire virtuel d'expérimentations pour la découverte scientifique en physique granulaire. In *RFIA'2000*, Paris, France. 2000.

[Breton *et al.* 2001] Laurent Breton, Jean-Daniel Zucker, Cécile Le Pape, Anneli Lénica et Fabrice Mérault. GranuPart : Des composants XML pour un laboratoire virtuel d'expérimentations en physique granulaire. *L'Objet, numéro spécial XML*, soumis, 2001.

[Breton 2002] Laurent Breton. *GranuLab : un système d'aide à la découverte scientifique pour la physique des milieux granulaires.* Thèse de Doctorat, Université de Paris VI - à paraître, 2002.

[Brooks 1991] Rodney Brooks. Intelligence without reason. In *Proceedings of IJCAI'91*, pages 569–595, Sydney, Australie. Morgan-Kaufmann, 1991.

[Camps 1998] Valérie Camps. *Vers une théorie de l'auto-organisation dans les systèmes multi-agents basée sur la coopération : application à la recherche d'information dans un système d'information répartie.* Thèse de Doctorat, Université Paul Sabatier, Toulouse, 1998.

[Card *et al.* 1999] Stuart K. Card, Jack D. Mackinlay et Ben Shneiderman, editors. *Readings in information visualization, using vision to think.* Morgan Kaufmann Publishers, Inc, San Francisco, USA, 1999.

[Carrière et Kazman 1995] Jeromy Carrière et Rick Kazman. Interacting with huge hierarchies : beyond cone trees. In *Proceedings in Symposium on Information Visualization*, pages 90–96, Atlanta, GA. 1995.

[Chalmers 1993] Matthew Chalmers. Using a lanscape metaphor to represent a corpus of documents. In A. Frank et I. Caspari, editors, *Proceedings of the European Conference on Spatial Information Theory*, pages 377–390, Elba. Springer-Verlag LNCS 716, 1993.

[Chauvin 1989] Rémy Chauvin. *Des hommes et des animaux.* Seghers, Paris, France, 1989.

[Chavez et Maes 1996] Anthony Chavez et Pattie Maes. Kasbah : An Agent Marketplace for Buying and Selling Goods. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK. 1996.

[Chernoff 1973] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal Amer. Statistical Association*, 68 : 316–368, 1973.

[Cliff et Miller 1996] Dave Cliff et Geoffrey Miller. Co-evolution of pursuit and evasion 2 : Simulation methods and results. In P. Maes, M. Mataric, J-A. Meyer, J. Pollack et S.W. Wilson, editors, *From Animals to Animats 4, Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviors, SAB96.* MIT Press Bradford Books, 1996.

[Collinot *et al.* 1996] Anne Collinot, Laurent Ploix et Alexis Drogoul. Application de la méthodologie Cassiopée à l'organisation d'une équipe de robots. In Jean-Pierre Müller et Joël Quinqueton, editors, *Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, pages 136–152, Paris, France. Éditions Hermès, 1996.

[Congjun Yang 2001] King-Ip Lin Congjun Yang. An index structure for efficient reverse nearest neighbor queries. In *Proceedings of the Seventeenth IEEE International Conference on Data Engineering*, April 2001.

[Cremer *et al.* 2000] James Cremer, Joan Severson, Shayne Geloand Joseph Kearney, Marise McDermott et Rick Riccio. "This old digital city" : virtual historical Cedar Rapids, Iowa circa 1900. In *Proceedings of the 6th Int. Conf. on Virtual Systems and Multimedia (VSMM 2000)*, pages 27–34, Japan. October 2000.

[Daconta et Saganich 2001] Michael C. Daconta et Al Saganich. *Développer en XML avec Java 2*. CampusPress, 2001.

[de Bra et Post 1994] Paul M.E. de Bra et Reinier D.J. Post. Information retrieval in the world-wide web : making client-based searching feasible. In *First International Conference on the World-Wide Web*, Geneva. May 1994.

[de Waal et Lanting 1999] Frans de Waal et Frans Lanting. *Bonobos, le bonheur d'être singe*. Librairie Arthème Fayard, 1999.

[Deneubourg *et al.* 1992] Jean-Louis Deneubourg, Guy Theraulaz et R. Beckers. Swarm-made architectures. In *Towards a Practice of Autonomous Systems*, pages 123–133. MIT Press, 1992.

[Denis et Quinqueton 1992] Michel Denis et Joël Quinqueton. Mémoire et représentation. *Le courrier du CNRS - dossiers scientifiques : sciences cognitives*, 79 : 56–57, Octobre 1992.

[Denis 1994] Michel Denis. *Image et cognition*. Presses Universitaires de France, 2ème édition, 1994.

[Di Caro et Dorigo 1998] Gianni Di Caro et Marco Dorigo. AntNet : Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9 : 317–365, 1998.

[Dijkstra et Timmermans 2000] Jan Dijkstra et Harry J.P. Timmermans. A multi-agent systems approach for visualizing simulated behavior to support the assessment of design performance. In *Fourth International Conference on Autonomous Agents, workshop on Autonomous Agents in Traffic Modelling*, Barcelona, Spain. 2000.

[Donath *et al.* 1999] Judith Donath, Karrie Karahalios et Fernanda Viegas. Visualizing conversation. In *Hawai International Conference on System Sciences*, 1999.

[Dorigo *et al.* 1998] Marco Dorigo, Gianno Di Caro et Luca Maria Gambardella. Ant algorithms for distributed discrete optimization. Technical report, Tech. rep. 98-10, IRIDIA, Université Libre de Bruxelles, 1998.

[Dorigo et Gambardella 1997] Marco Dorigo et Luca Maria Gambardella. Ant colony system : a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1 : 53–66, 1997.

[Doron et Parot 1991] Roland Doron et Françoise Parot, editors. *Dictionnaire de psychologie*. Presses Universitaires de France, 1991.

[Dortier 1999] Jean-François Dortier. *Le cerveau et la pensée, la révolution des sciences cognitives*. Éditions Sciences Humaines, 1999.

[dossier Pour La Science 2001] dossier Pour La Science. La mémoire, le jardin de la pensée. 31, Avril-Juillet 2001.

[dossier Sciences et Avenir 2001] dossier Sciences et Avenir. L'invasion des Robo sapiens. 655 : 40–51, Septembre 2001.

[Drogoul *et al.* 1992] Alexis Drogoul, Bruno Corbara et Dominique Fresneau. Applying ethomodeling to social organization in ants. *Biology and Evolution of Social Insects*, pages 375–383, 1992. Leuven University Press, Leuven.

[Drogoul *et al.* 1995] Alexis Drogoul, Bruno Corbara et Dominique Fresneau. MANTA : New experimental results on the emergence of (artificial) ant societies. In Nigel Gilbert & R. Conte, editor, *Artificial Societies : the computer simulation of social life*, London. UCL Press, 1995.

[Drogoul et Collinot 1997] Alexis Drogoul et Anne Collinot. Entre réductionnisme méthodologique et stratégie intentionnelle, l'éthologie, un modèle alternatif pour l'i.a.d. ? In *Proceedings of JFIADSMA'97*, pages 307–322, Paris, France. Éditions Hermès, 1997.

[Drogoul 1993] Alexis 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*. Thèse de Doctorat, Université de Paris VI, 1993.

[Drogoul 2000] Alexis Drogoul. *Systèmes multi-agents situés*. Mémoire d'habilitation à diriger des recherches, Université de Paris VI, 2000.

[Désesquelles 2001] Annie-Claire Désesquelles. *La représentation*. Collection Philonotions. Ellipses Éditions Marketing S.A., Paris, 2001.

[Dunn et Everitt 1982] Graham Dunn et Brian Everitt. *An introduction to mathematical taxonomy*. Cambridge University Press, Cambridge, MA, 1982.

[Eckstein et Casabianca 1999] Robert Eckstein et Michel Casabianca. *XML, précis et concis*. O'Reilly, 1999.

[Eibl-Eibesfeldt 1972] Irenäus Eibl-Eibesfeldt. *Éthologie, biologie du comportement*. Éditions Scientifiques Naturalia et Biologia, Paris, 1972.

[Errard *et al.* 1990] Christine Errard, Catherine Vienne et Bruno Corbara. Ouverture et cohabitation chez les fourmis : les sociétés mixtes. *La Recherche*, 21(222) : 780–782, Juin 1990.

[Fabre 1913] Jean-Henri Fabre. *Les Merveilles de l'Instinct chez les insectes, Morceaux choisis, Extraits des Souvenirs entomologiques et Histoires inédites du Ver luisant et de la Chenille du chou.* Delagrave, Paris, 1913.

[Faloutsos et Lin 1995] Christos Faloutsos et King-Ip Lin. Fastmap : a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of ACM SIGMOD, International conference on management of data*, pages 163–174, San Jose, CA. 1995.

[Ferber 1995] Jacques Ferber. *Les systèmes multi-agents, vers une intelligence collective.* InterÉditions, 1995.

[Ferber 1997] Jacques Ferber. Les systèmes multi-agents : un aperçu général. *Technique et science informatiques*, 16(8) : 979–1012, 1997.

[Fortin et Rousseau 1997] Claudette Fortin et Robert Rousseau. *Psychologie cognitive : une approche du traitement de l'information.* Télé-Université, Presses de l'Université du Québec, Sainte-Foy, Québec, 2ème édition, 1997.

[Francès 1992] Robert Francès. *La perception.* Number 1076 in Que sais-je ? Presses Universitaires de France, Paris, France, 8ème édition, 1992.

[Furnas 1981] George W. Furnas. The fisheye view : a new look at structured files. Technical report, Bell Laboratories technical memorandum #81-11221-9, October 1981.

[Furnas 1986] George W. Furnas. Generalized fisheye views. In *Proceedings of the ACM CHI'86 Conference on Human Factors in Computing Systems*, pages 16–23, Boston, MA. ACM Press, 1986.

[Gasser et Huhns 1989] Les Gasser et Michael N. Huhns, editors. *Distributed Artificial Intelligence*, volume II. Pitman/Morgan Kaufmann, 1989.

[Gasser 2001] Les Gasser. Perspectives on organizations in multi-agent systems. In Michael Luck, Vladimír Mařík, Olga Štěpánková et Robert Trapp, editors, *ACAI 2001*, Lecture Notes in Artificial Intelligence 2086, pages 1–16. Springer-Verlag Berlin Heidelberg, 2001.

[Grassé 1959] Pierre Paul Grassé. La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis et cubitermes sp.* La théorie de la stigmergie : essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6 : 41–81, 1959.

[Gregory 1993] Richard L. Gregory. *Le cerveau, un inconnu - dictionnaire encyclopédique*. Bouquins. Robert Laffont, Paris, France, 1993.

[Harman 1967] Harry Horace Harman. *Modern factor analysis*. University of Chicago Press, 1967.

[Healy 1994] Sue D. Healy. La mémoire et l'adaptation animale. *La Recherche*, 25(267) : 768–773, Juillet-Août 1994.

[Hofstadter et Dennett 1987] Douglas Hofstadter et Daniel Dennett, editors. *Vues de l'Esprit, Fantaisies et réflexions sur l'être et l'âme*. InterÉditions, Paris, 1987.

[Hogeweg et Hesper 1979] Pauline Hogeweg et B. Hesper. Heterarchical, selfstructuring simulation systems : concepts and applications in biology. In B.P. Zeigler, M.S. Elzas, G.J. Klir et T.I. Ören, editors, *Methodology in systems modelling and simulation*, pages 221–232. North-Holland, 1979.

[Hogeweg et Hesper 1983] Pauline Hogeweg et B. Hesper. The ontogeny of the interaction structure in Bumble Bee Colonies : A MIRROR Model. *Behavioral Ecology and Sociobiology*, 12 : 271–283, 1983.

[Honkela *et al.* 1997] Timo Honkela, Samuel Kaski, Krista Lagus et Teuvo Kohonen. WEBSOM, self-organizing maps of document collections. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps*, pages 310–315, Helsinki University of Technolohy, Finland. 1997.

[Horstmann et Cornell 2000a] Cay S. Horstmann et Gary Cornell. *Au cœur de Java 2, fonctions avancées*, volume 2. Campus Press France - Sun Micosystems, Inc, 2000.

[Horstmann et Cornell 2000b] Cay S. Horstmann et Gary Cornell. *Au cœur de Java 2, notions fondamentales*, volume 1. Campus Press France - Sun Micosystems, Inc, 2000.

[Hutzler *et al.* 1998a] Guillaume Hutzler, Bernard Gortais et Alexis Drogoul. Data Gardens : an artistic proposal towards the representation of distributed and dynamic data using multi-agent systems. In *Proceedings of the Third International Conference on Multi-Agents Systems, ICMAS'98*, 1998.

[Hutzler *et al.* 1998b] Guillaume Hutzler, Bernard Gortais et Valérie Renault. Point et ligne sur plan : des agents qui communiquent localement. In J.-P. Barthès, V. Chevrier et C. Brassac, editors, *JFIADSMA'98*, pages 191–204, Pont-à-Mousson. Éditions Hermès, 1998.

[Hutzler et Drogoul 1996] Guillaume Hutzler et Alexis Drogoul. Le Jardin des Hasards, peinture abstraite et IAD réactive. Technical Report 96/04, LOFORIA, 1996.

[Hutzler et Renault 1999] Guillaume Hutzler et Valérie Renault. Vers une conception multi-agent des interfaces hommes-machines. In M.-P. Gleizes et P. Mercenac, editor, *JFIADSMA'99*, pages 339–340, Saint-Gilles, Ile de La Réunion. Éditions Hermès, 1999.

[Hutzler et Renault 2000] Guillaume Hutzler et Valérie Renault. Les Jardins de Données : Sociétés d'agents pour la visualisation de Systèmes complexes. In *ERGO-IHM'2000*, Biarritz. 2000.

[Hutzler 2000] Guillaume Hutzler. *Du Jardin des Hasards aux Jardins de Données : une approche artistique et multi-agent des interfaces homme/systèmes complexes.* Thèse de Doctorat, Université de Parix 6, 2000.

[Ishizaki 1996] Suguru Ishizaki. Multi-agent model of dynamic design, visualization as an emergent behavior of active design agents. In *Proceedings of the ACM CHI'96 Conference on Human Factors in Computing Systems*, pages 347–354, Vancouver, Canada. ACM Press, 1996.

[Jeong et Pang 1998] Chang Sung Jeong et Alex Pang. Reconfigurable Disc Trees for Visualizing Large Hierarchical Information Space. In *Proceedings of the IEEE Symposium on Information Visualization'98*, 1998.

[Johnson et Shneiderman 1991] Brian Johnson et Ben Shneiderman. Treemaps : a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd International IEEE Visualization Conference*, pages 284–291, San Diego. 1991.

[Johnson 1993] Brian Johnson. *Visualizing hierarchical and categorical data.* PhD thesis, University of Maryland, 1993.

[Keim *et al.* 1993] Daniel A. Keim, R. Daniel Bergeron et Ronald M. Pickett. Test data sets for evaluating data visualization techniques. In *Database issues for visualization, IEEE Visualization'93 Workshop*, Berlin. Springer-Verlag, 1993.

[Keim 2000] Daniel A. Keim. An introduction to information visualization techniques for exploring large databases. In *IEEE Visualization 2000, Tutorial Notes*, 2000.

[Kohonen 1995] Teuvo Kohonen. *Self-organizing maps.* Springer, Berlin, 1995.

[Kolski 1997] Christophe Kolski. *Interfaces homme-machines, application aux systèmes industriels complexes.* Éditions Hermès, 2ème édition, 1997.

[Koza 1991] John R. Koza. *Genetic Programming.* MIT Press, Cambridge, Massachusetts, 1991.

[Kuntz et Snyers 1994] Pascale Kuntz et Dominique Snyers. Emergent colonization and graph partitioning. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer et Stewart W.

Wilson, editors, *From animals to animats 3, proceedings of the third international conference on simulation of adaptive behavior (SAB94)*, pages 494–500, 1994.

[Lamping *et al.* 1995] John Lamping, Ramana Rao et Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems*. ACM Press, 1995.

[Langton 1994] Chris Langton, editor. *Journal of Artificial Life*, volume 1, 1994.

[Leung et Apperley 1994] Ying K. Leung et M.D. Apperley. A review and taxonomy of distortion-orientation presentation techniques. In *Proceedings of the ACM CHI'94 Conference on Human Factors in Computing Systems*, pages 126–160, Boston, MA. ACM Press, 1994.

[Lieberman *et al.* 1999] Henry Lieberman, Neil Van Dyke et Adriana Vivacqua. Let's browse : a collaborative web browsing agent. In *International Conference on Intelligent User Interfaces*, 1999.

[Lieberman 1995a] Henry Lieberman. Letizia : An agent that assists web browsing. In *International Joint Conference on Artificial Intelligence*, Montreal. 1995.

[Lieberman 1995b] Henry Lieberman. The visual language of experts in graphic design. In *IEEE Symposium on Visual Language*, Darmstadt, Germany. September 1995.

[Lieberman 1997] Henry Lieberman. Autonomous interface agents. In *Proceedings of the ACM CHI'97 Conference on Human Factors in Computing Systems*, Atlanta. ACM Press, March 1997.

[Locher 2000] J.L. Locher. *The magic of M.C. Escher.* Thames and Hudson, 2000.

[Lorenz 1969] Konrad Lorenz. *L'agression, une histoire naturelle du mal.* Flammarion, 1969.

[Lorenz 1984] Konrad Lorenz. *Les fondements de l'éthologie.* Flammarion, 1984.

[Lumer et Faieta 1994] Erik D. Lumer et Baldo Faieta. Diversity and Adaptation in Populations of Clustering Ants. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer et Stewart W. Wilson, editors, *From animals to animats 3, proceedings of the third international conference on simulation of adaptive behavior (SAB94)*, pages 501–508, 1994.

[Mackinlay *et al.* 1991] Jack D. Mackinlay, G.G. Robertson et Stuart K. Card. The Perspective Wall : detail and context smoothly integrated. In *Proceedings of the ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 173–179, New Orleans, LA. ACM Press, 1991.

[McFarland 1994] D. McFarland. Towards Robot Cooperation. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer et Stewart W. Wilson, editors, *From animals to animats 3. Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior*, pages 440–444. MIT Press, 1994.

[McFarland 1995] D. McFarland. Autonomy and Self-Sufficiency in Robots. In Luc Steels, editor, *The Artificial Life Route To Artificial Intelligence. Building Embodied, Situated Agents*, pages 187–213. Lawrence Erlbaum Ass., USA, 1995.

[Mephu-Nguifo 1993] Engelbert Mephu-Nguifo. *Concevoir une abstraction à partir de ressemblances.* Thèse de Doctorat, Université de Montpellier II, 1993.

[Microsoft Corporation 2001] Microsoft Corporation. Microsoft Agent. Technical report, http ://www.microsoft.com/products/msagent/datasheet.htm, 2001.

[Minar et Donath 1999] Nelson Minar et Judith Donath. Visualizing the crowds at a Web site. In *CHI'99, Late Breaking Papers.* ACM Press, 1999.

[Minsky 1988] Marvin Minsky. *The Society of Mind.* First Touchstone Edition, 1988.

[Müller 1996] Jean-Pierre Müller. Un modèle de systèmes d'agents autonomes situés : application à la déduction automatique. In *JFIADSMA'96.* Éditions Hermès, 1996.

[Müller 2000] Jean-Pierre Müller. Modélisation organisationnelle en systèmes multi-agents. In *Septième École d'été de l'ARCo*, 2000.

[Morin 1977] Edgar Morin. *La Méthode : la nature de la nature*, volume 1. Éditions du Seuil, 1977.

[Morris et Maes 2000] Joan Morris et Pattie Maes. Negotiating beyond the Bid price. In *Proceedings of the ACM CHI 2000 Conference on Human Factors in Computing Systems*, The Hague, Netherlands. 2000.

[Moukas 1996] Alexandros Moukas. Amalthaea : Information discovery and filtering using a multiagent evolving ecosystem. In *Proceedings of the Conference on Practical Application of Intelligent Agents and Multi-agent Technology*, London. 1996.

[Mucchielli 2000] Alex Mucchielli. *Les motivations.* Number 1949 in Que sais-je ? Presses Universitaires de France, Paris, France, 5ème édition, 2000.

[Muñoz *et al.* 2001] Angélica Muñoz, François Sempe et Alexis Drogoul. Sharing Resources without explicit communication in collective robotics. In *soumis à AAMAS 2002*, 2001.

[Munzner 1997] Tamara Munzner. H3 : Laying out large directed graphs in 3D hyperbolic space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization*, pages 2–10, Phoenix, AZ. October 1997.

[Naville 1963] Pierre Naville. *La Psychologie du comportement : le behaviorisme de Watson.* Gallimard, Paris, 1963.

[Nehaniv 1999] Chrystopher L. Nehaniv. *Computation for Metaphors, Analogy, and Agents*, volume 1562 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, Berlin, 1999.

[Nguyen-Xuan 1992] Anh Nguyen-Xuan. L'analogie dans l'apprentissage des dispositifs de commande. *Le courrier du CNRS - dossiers scientifiques : sciences cognitives*, 79 : 34, Octobre 1992.

[Nilsson 1998] Nils J. Nilsson. *Artificial Intelligence : a new Synthesis.* Morgan Kaufmann Publishers, Inc, San Francisco, California, 1998.

[Nishimura et Ikegami 1997] Shin I. Nishimura et Takashi Ikegami. Emergence of collective strategies in a prey-predator game model. *Artificial Life*, 3(4) : 243–260, 1997.

[Pavard 1994] Bernard Pavard. *Systèmes coopératifs : de la modélisation à la conception.* Octares Éditions, Toulouse, 1994.

[Pavard 1999] Bernard Pavard. Complexity paradigm as a framework for the study of cooperative systems. In *COTCOS Summer School*, Brighton, England. 1999.

[Picault et Collinot 1998] Sébastion Picault et Anne Collinot. Designing Social Cognition Models for Multi-Agent Systems Through Simulating Primate Societies. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98).* IEEE Press, 1998.

[Poirier 2001] Hervé Poirier. Création assitée par ordinateur - Génial, non ? *Science & Vie*, 1001 : 64–69, Février 2001.

[Poulantzas 2001] Ariane Poulantzas. Comment pensent les génies - Enquête dans les coulisses de l'intelligence créative. *Science & Vie*, 1001 : 52–63, Février 2001.

[Proctor et Winter 1998] Glenn Proctor et Chris Winter. Information flocking : data visualisation in virtual worlds using emergent behaviours. In J.-C. Heudin, editor, *Proceedings of the First International Conference on Virtual Worlds*, volume LNAI 1434, pages 168–176. Springer-Verlag, 1998.

[Rao et Card 1994] R. Rao et S.K. Card. The table lens : merging graphical and symbolic representation in an interactive focus+context visualization for tabular information. In *Proceedings of the ACM CHI'94 Conference on Human Factors in Computing Systems*, pages 318–322, Boston, MA. ACM Press, 1994.

[Renault et Hutzler 2000] Valérie Renault et Guillaume Hutzler. Data Gardens : Agent Societies for Visualization of Complex System. In H. R. Arabnia, editor, *IC-AI'2000*, pages 167–173, Las Vegas, Navada. CSREA Press, 2000.

[Renault 1998] Valérie Renault. Jardins de Données et Parc Automobile, processus distribués de hiérarchisation pour la visualisation dynamique de données numériques. Technical report, Mémoire de DEA IARFA, Intelligence Artificielle, Reconnaissance des Formes et Applications, 1998.

[Renault 2001] Valérie Renault. Computation for metaphors, analogy, and agents, edited by chrystopher l. nehaniv (reviewed of). *The Journal of Artificial Socities and Social Simulation*, 4(1), 2001.

[Resnick 1994a] Mitchel Resnick. Learning About Life. *Artificial Life*, 1(1-2), 1994.

[Resnick 1994b] Mitchel Resnick. *Turtles, termites, and traffic jams : explorations in massively parallel microworlds.* MIT Press, 1994.

[Resnick 1996] Mitchel Resnick. Beyond the centralized mindset. *Journal of the Learning Sciences*, 5(1) : 1–22, 1996.

[Reuchlin 1993] Maurice Reuchlin. *Psychologie.* Presses Universitaires de France, Paris, France, 10ème édition, 1993.

[Reynolds 1987] Craig W. Reynolds. Flocks, herds and schools : a distributed behavioral model. *Computer Graphics, SIGGRAPH'87 Conference Proceedings*, 21(4) : 25–34, 1987.

[Reynolds 1999] Craig W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, 1999.

[Reynolds 2000] Craig W. Reynolds. Interaction with groups of autonomous characters. In *Proceedings of Game Developers Conference 2000*, pages 449–460, San Francisco, California. CMP Game Media Group, 2000.

[Robert 1990] Le Robert, editor. *Le Petit Robert, 2.* Paris, France, 1990.

[Robertson *et al.* 1991] George G. Robertson, Jock D. Mackinlay et Stuart K. Card. Cone trees : animated 3D visualizations of hierarchical information. In *Proceedings of the ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 189–194, New Orleans, LA. ACM Press, 1991.

[Robertson et Mackinlay 1993] George G. Robertson et Jock D. Mackinlay. The Document Lens. In *Proceedings of UIST '93*, pages 101–108, Atlanta, Georgia. 1993.

[Roitblat 1994] Herbert L. Roitblat. Mechanism and process in animal behavior : models of animals, animals as models. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer et Stewart W. Wilson, editors, *From animals to animats 3, proceedings of the third international conference on simulation of adaptive behavior (SAB94)*, pages 12–21, 1994.

[Russell et Norving 1995] Stuart Russell et Peter Norving. *Artificial Intelligence, a modern approach*. Prentice Hall Series in Artificial Intelligence, 1995.

[Ruwet 1975] Jean-Claude Ruwet. *L'éthologie : biologie du comportement*. Pyschologie et sciences humaines. Pierre Mardaga, Bruxelles, Belgique, 3ème édition, 1975.

[Sarkar et Brown 1992] Manojit Sarkar et Marc H. Brown. Graphical Fisheye Views of Graphs. *Proceedings of the ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 83–91, 1992.

[Scaife et Rogers 1996] Mike Scaife et Yvonne Rogers. External cognition : How do graphical representations work. *International Journal of Human-Computer Studies*, 45 : 185–213, 1996.

[Schaffer *et al.* 1993] Doug Schaffer, Zhengping Zuo, Lyn Bartram, John Dill, Shelli Dubs, Saul Greenberg et Mark Roseman. Comparing fisheye and full-zoom techniques for navigation of hierarchically clustered networks. In *Proceedings of Graphics Interface (GI'93) in Canadian Information Processing Soc.*, pages 87–96, Toronto, Ontario. Morgan Kaufmann Publishers, Inc, 1993.

[Schaffer *et al.* 1996] Doug Schaffer, Zhengping Zuo, Saul Greenberg, Lyn Bartram, John Dill, Shelli Dubs et Mark Roseman. Navigating Hierarchically Clustered Networks Through Fisheye and Full-Zoom Method. *ACM Transactions on Computer-Human Interaction*, 3(2) : 162–188, June 1996.

[Scullin *et al.* 1995] Will H. Scullin, Thomas T. Kwan et Daniel A. Reed. Real-time visualization of the World Wide Web traffic. In *Symposium on visualizing time-varying data*, September 1995.

[Searle 1996] John R. Searle. Deux biologistes et un physicien en quête de l'âme : Crick, Penrose et Edelman passé au scalpel de la critique philosophique. *La Recheche*, 287 : 62–77, Mai 1996.

[Servat *et al.* 1998] David Servat, Edith Perrier, Jean-Pierre Treuil et Alexis Drogoul. When Agents Emerge From Agents : Introducing Multi-scale Viewpoints in Multi-agent Simulations. In Conte Sichman et Gilbert, editors, *Multi-agent Systems and Agent-based Simulation*, volume 1534 of *LNAI series*, Berlin. Springer-Verlag, December 1998.

[Servat 2000] David Servat. *Modélisation de dynamiques de flux par agents. Application aux processus de ruissellement, infiltration et érosion*. Thèse de Doctorat, Université de Paris VI, 2000.

[Seyfarth et Cheney 1993] Robert Seyfarth et Dorothy Cheney. La pensée chez les singes. *Pour la Science*, 184 : 46–52, Février 1993.

[Shepard *et al.* 1972] Roger N. Shepard, A.K. Romney et S.B. Nerlove. *Multidimensional scaling.* Seminar Press, New-York, 1972.

[Shepard 1967] Roger N. Shepard. Recognition memory for words, sentences and pictures. *Journal of Verbal Learning and Verbal Behavior*, pages 156–163, 1967.

[Shneiderman 1992] Ben Shneiderman. Tree visualization with treemaps : a 2D space-filling approach. *ACM Transactions on Graphics*, 11(1) : 92–99, 1992.

[Shneiderman 1997] Ben Shneiderman. Designing information-abundant web sites : issues and recommandations. *International Journal of Human-Computer Studies*, 47 : 5–29, 1997.

[Shortliffe 1976] Edward H. Shortliffe. *Computer-based Medical Consultations : MYCIN.* North-Holland, New York, 1976.

[Siéroff 1992] Éric Siéroff. L'attention sélective. *Le courrier du CNRS - dossiers scientifiques : sciences cognitives*, 79 : 59, Octobre 1992.

[Skinner 1969] Burrhus Frederic Skinner. *L'analyse expérimentale du comportement.* Mardaga (Psychologie et Sciences Humaines), 1969.

[Standing 1973] L. Standing. Learning 10 000 pictures. *Quarterly Journal of Experimental Psychology*, 25 : 207–222, 1973.

[Steels 1991] Luc Steels. Towards a theory of emergent functionality. In *From Animals to Animats*, pages 451–461. MIT Press, 1991.

[Sutton et Barto 1998] Richard S. Sutton et Andrew G. Barto. *Reinforcement Learning : An Introduction.* MIT Press, A Bradford Book, Cambridge, MA, 1998.

[Terzopoulos *et al.* 1994] Demetri Terzopoulos, Xiaoyuan Tu et Radek Grzeszczuk. Artificial fishes with Autonomous locomotion, perception, behavior, and learning in a simulated physical world. In Rodney A. Brooks et Pattie Maes, editors, *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 327–351. MIT Press, A. Bradford Book,, 1994.

[Tesler 1981] L. Tesler. The Smalltalk environment. *Byte*, 6 : 90–147, 1981.

[Tinbergen 1967] Niko Tinbergen. *La vie sociale des animaux.* Payot, 1967.

[Tisseau 1996] Gérard Tisseau. *Intelligence artificielle, problèmes et méthodes.* Psychologie et sciences de la pensée. Presses Universitaires de France, Paris, France, 1996.

[Travers 1996] M. Travers. *Programming with Agents : new metaphors for thinking about computation.* PhD thesis, MIT, 1996.

[Tufte 1983] E.R. Tufte. *The visual display of quantitative information.* Graphics Press, Cheshire, CT, 1983.

[van Honk et Hogeweg 1981] Cor van Honk et Pauline Hogeweg. The ontogeny of the social structure in a captive Bombus terrestris Colony. *Behavioral Ecology and Sociobiology*, 9 : 111–119, 1981.

[Varela 1996] Francisco J. Varela. *Invitation aux sciences cognitives.* Number S111 in Collection Points, Série Sciences. Éditions du Seuil, 2ème édition, 1996.

[Vauclair 1992] Jacques Vauclair. *L'intelligence de l'animal.* Éditions du Seuil, 1992.

[Vauclair 2000] Jacques Vauclair. À quoi ressemble le monde des primates ? *Science et Avenir*, 119, Hors-série 2000.

[Vetter et Reed 2000] Jeffrey S. Vetter et Daniel A. Reed. Real-time performance monitoring, adaptive control, and interactive steering of computational grids. *The International Journal of High Performance Computing Applications*, 14(4) : 357–366, Winter 2000.

[Veuille 1986] Michel Veuille. *La sociobiologie.* Number 2284 in Que sais-je ? Presses Universitaires de France, Paris, France, 1ère édition, 1986.

[Viegas et Donath 1999] Fernanda B. Viegas et Judith S. Donath. Chat circles. In *Proceedings of the ACM CHI'99 Conference on Human Factors in Computing Systems*, 1999.

[Vignaux 1991] Georges Vignaux. *Les sciences cognitives, une introduction.* Number 4193 in biblio, essais. Éditions La Découverte, Le livre de Poche, 1991.

[Vignaux 1999] Georges Vignaux. *Le démon du classement, penser, organiser.* Le temps de penser. Éditions du Seuil, 1999.

[Wattenberg 1999] M. Wattenberg. Visualizing the stock market. In *CHI'99 Extended Abstracts*, pages 188–189, Pittsburgh, Philadelphia. ACM Press, 1999.

[Wise *et al.* 1995] J.A. Wise, J.J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur et V. Crow. Visualizing the non-visual : spatial analysis and interaction with information from text documents. In *Proceedings Symp. on Information Visualization*, pages 51–58, Atlanta, GA. 1995.

[Wolfram 1986] Stephen Wolfram, editor. *Theory and Applications of Cellular Automata.* World Scientific, Singapore, 1986.

[Worden 1996] Robert P. Worden. Primate social intelligence. *Cognitive Science*, 20 : 579–616, 1996.

[Yoshida *et al.* 1998] Sen Yoshida, Koji Kamei, Makoto Yokoo, Takeshi Ohguro, Kaname Funakoshi et Fumio Hattori. Visualizing potential communities : a multiagent approach. In Yves Demazeau General Chair, editor, *Proceedings of the Third International*

Conference on Multi-Agents Systems (ICMAS'98), pages 477–478, Paris, France. IEEE Computer Science, 1998.

[Young et Shneiderman 1993] D. Young et B. Shneiderman. A graphical filter/flow model for boolean queries : an implementation and experiment. *Journal of the American Society for Information Science*, 44(6) : 327–339, 1993.

Les références suivantes, déjà signalées dans la section ci-dessus, récapitulent les ouvrages ou les articles-clés selon les domaines rencontrés dans ce mémoire.

2 Références-clés dans le domaine de la perception et des IHM

[Bagot 1999] Jean-Didier Bagot. *Information, sensation et perception.* Collection Cursus, Psychologie. Armand Colin, Paris, France, 2ème édition, 1999.

[Doron et Parot 1991] Roland Doron et Françoise Parot, editors. *Dictionnaire de psychologie.* Presses Universitaires de France, 1991.

[Reuchlin 1993] Maurice Reuchlin. *Psychologie.* Presses Universitaires de France, Paris, France, 10ème édition, 1993.

3 Références-clés dans le domaine de la visualisation d'informations

[Andrews 2000] Keith Andrews. Information visualization. Technical report, lecture notes, Graz University of Technology, Mars 2000.

[Card *et al.* 1999] Stuart K. Card, Jack D. Mackinlay et Ben Shneiderman, editors. *Readings in information visualization, using vision to think.* Morgan Kaufmann Publishers, Inc, San Francisco, USA, 1999.

[Furnas 1986] George W. Furnas. Generalized fisheye views. In *Proceedings of the ACM CHI'86 Conference on Human Factors in Computing Systems*, pages 16–23, Boston, MA. ACM Press, 1986.

[Leung et Apperley 1994] Ying K. Leung et M.D. Apperley. A review and taxonomy of distortion-orientation presentation techniques. In *Proceedings of the ACM CHI'94*

Conference on Human Factors in Computing Systems, pages 126–160, Boston, MA. ACM Press, 1994.

[Sarkar et Brown 1992] Manojit Sarkar et Marc H. Brown. Graphical fisheye views of graphs. *Proceedings of the ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 83–91, 1992.

4 Références-clés dans le domaine des systèmes multi-agents

[Brooks 1991] Rodney Brooks. Intelligence without reason. In *Proceedings of IJCAI'91*, pages 569–595, Sydney, Australie. Morgan-Kaufmann, 1991.

[Ferber 1995] Jacques Ferber. *Les systèmes multi-agents, vers une intelligence collective.* InterÉditions, 1995.

[Minsky 1988] Marvin Minsky. *The Society of Mind.* First Touchstone Edition, 1988.

[Russell et Norving 1995] Stuart Russell et Peter Norving. *Artificial Intelligence, a modern approach.* Prentice Hall Series in Artificial Intelligence, 1995.

5 Références-clés dans le domaine de l'éthologie

[Chauvin 1989] Rémy Chauvin. *Des hommes et des animaux.* Seghers, Paris, France, 1989.

[Errard *et al.* 1990] Christine Errard, Catherine Vienne et Bruno Corbara. Ouverture et cohabitation chez les fourmis : les sociétés mixtes. *La Recherche*, 21(222) : 780–782, Juin 1990.

[Grassé 1959] Pierre Paul Grassé. La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis et cubitermes sp.* la théorie de la stigmergie : essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6 : 41–81, 1959.

[Tinbergen 1967] Niko Tinbergen. *La vie sociale des animaux.* Payot, 1967.

[Vauclair 1992] Jacques Vauclair. *L'intelligence de l'animal.* Éditions du Seuil, 1992.

6 Références-clés dans le domaine de l'éthologie et des systèmes multi-agents

[Bonabeau et Guy 1994] Éric Bonabeau et Theraulaz Guy, editors. *Intelligence collective.* Éditions Hermès, Paris, 1994.

[Caro et Dorigo 1998] Gianni Di Caro et Marco Dorigo. Antnet : Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9 : 317–365, 1998.

[Drogoul 1993] Alexis 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.* Thèse de Doctorat, Université de Paris VI, 1993.

[Resnick 1994] Mitchel Resnick. Learning about life. *Artificial Life*, 1(1-2), 1994.

[Reynolds 2000] Craig W. Reynolds. Interaction with groups of autonomous characters. In *Proceedings of Game Developers Conference 2000*, pages 449–460, San Francisco, California. CMP Game Media Group, 2000.

7 Références-clés en programmation Java et XML

[Bray *et al.* 2000] Tim Bray, Jean Paoli et C.M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 , W3C Recommendation. Technical Report REC-xml-19980210, W3C, 2000.

[Daconta et Saganich 2001] Michael C. Daconta et Al Saganich. *Développer en XML avec Java 2.* CampusPress, 2001.

[Eckstein et Casabianca 1999] Robert Eckstein et Michel Casabianca. *XML, précis et concis.* O'Reilly, 1999.

[Horstmann et Cornell 2000a] Cay S. Horstmann et Gary Cornell. *Au cœur de Java 2, fonctions avancées*, volume 1. Campus Press France - Sun Micosystems, Inc, 2000.

[Horstmann et Cornell 2000b] Cay S. Horstmann et Gary Cornell. *Au cœur de Java 2, notions fondamentales*, volume 2. Campus Press France - Sun Micosystems, Inc, 2000.

Annexe A

Techniques de distorsion de «Fisheye Views»

A.1 Principes généraux développés par G. Furnas

Lorsqu'un utilisateur manipule un grand nombre de données, il doit parfois focaliser son attention sur certains détails. Ces détails peuvent se situer à différents niveaux de précision. Les techniques de «Fisheye Views» ont été développées afin de permettre la visualisation d'une aire d'intérêt relativement large et détaillée tout en conservant une vision synthétique du contexte global [Leung et Apperley 1994]. Ces transformations divisent l'interface en deux parties. La première partie, où se trouve le point d'intérêt (le point de focus), est représentée de façon à pouvoir visualiser et manipuler les objets pertinents. La deuxième partie plus «élastique» permet de représenter les autres éléments en périphérie, leur taille converge alors vers zéro en fonction de l'éloignement par rapport au point de focus.

Furnas [Furnas 1981, Furnas 1986] a introduit les principes de base dans un cadre général de représentation d'informations ayant une structure *hiérarchique*. Cet article développe les principes autour des exemples de la représentation d'arbres et de la visualisation du texte d'un programme informatique. Il montre qu'il est ainsi possible d'orienter la représentation de la structure d'un arbre, par exemple, en fonction du centre d'intérêt de l'utilisateur. Il suffit alors de définir :

- le *point de focus* «.» : un nœud de l'arbre intéressant à un moment donné;
- la *distance du point x au point de focus* $D(.,x)$: elle représente la contribution spécifique du point au point de focus courant; dans le cas d'un arbre, cette distance $d(.,x)$ peut correspondre au nombre de liens intervenant dans le chemin qui relie le

point au focus ;

- la *mesure d'importance a priori* ou *niveau de détail* ou *LOD («level of detail»)* : cette valeur est statique et indépendante de l'interaction en cours, elle reflète l'influence du point sur la structure globale à représenter ; dans le cas des arbres, $LOD(x) = -d(r, x)$ avec r la racine de l'arbre.

Le *degré d'intérêt, DOI («degree of interest»)* est alors une fonction des deux mesures précédentes :

$$DOI(x|.) = F(LOD(x), D(., x)) \tag{A.1}$$

Dans le cas de l'arbre, le degré d'intérêt du nœud x par rapport au nœud «.» est :

$$\begin{cases} DOI(x|.) & = & f(g(LOD(x)) - h(D(., x))) \\ & = & -(d(., x) + d(r, x)) \end{cases} \tag{A.2}$$

où f, g et h sont des fonctions monotones croissantes. Un nœud n'est ensuite représenté à l'écran que si son degré d'intérêt est supérieur à un certain seuil. Ainsi cette technique permet d'établir un équilibre entre le besoin de détails locaux et le besoin de maintenir un contexte global.

A.2 Fisheye Views graphiques de Sarkar & Brown

Sarkar et Brown [Sarkar et Brown 1992] ont fourni une interprétation graphique des Fisheye Views pour représenter des graphes.

Le calcul des coordonnées d'un nœud v dans la Fisheye View dépend des coordonnées de ce point dans sa position normale P_{norm} («normale» correspond à la situation sans distorsion) et de la position P_{focus} du point de focus f :

$$P_{feye}(v, f) = F_1(P_{norm}(v), P_{norm}(f)) \tag{A.3}$$

Lors de l'implémentation de la fonction F_1 de l'équation A.3, les coordonnées x et y peuvent être traitées indépendamment comme suit :

$$\left\{ \begin{array}{rcl} P_{feye} & = & \{G(\frac{D_{norm_x}}{D_{max_x}}) \times D_{max_x} + P_{focus_x}, \\ & & G(\frac{D_{norm_y}}{D_{max_y}}) \times D_{max_y} + P_{focus_y}\} \end{array} \right. \tag{A.4}$$

avec

$$G(x) = \frac{(d+1)x}{dx+1} \tag{A.5}$$

où :

- D_{max_x} est la distance horizontale entre le bord de l'écran et le point de focus en coordonnées normales ;
- D_{norm_x} est la distance horizontale entre le point à transformer et le point de focus, en coordonnées normales ;
- d est une constante appelée *facteur de distorsion*. L'amplitude de la distorsion augmente avec d.

La fonction $G(x)$ est monotone croissante continue pour $0 \leq x \leq 1$, avec $G(0) = 0$ et $G(1) = 1$. La dérivée de $G(x)$ est :

$$G'(x) = \frac{d+1}{(dx+1)^2} \tag{A.6}$$

L'équation (A.6) indique que pour une valeur de d donnée, la distorsion d'un point x, lorsque x est proche de 0 (près du focus), est importante, d'où un effet de grossissement. Inversement, le point est faiblement décalé lorsque x se rapproche de 1 (près des bords), ce qui a pour effet de «tasser» les points sur le bord de l'écran (Fig. A.1). À la différence des travaux de Furnas où les éléments sont soit représentés en détail, soit non représentés, il y a ici une continuité entre ces deux extrêmes.

Leung [Leung et Apperley 1994] définit ainsi les fonctions $G(x)$ et $G'(x)$: la fonction $G(x)$ est la *fonction de transformation* qui définit comment l'image originale va être reconstruite dans une image déformée. La fonction $G'(x)$ est appelée *fonction de grossissement* («magnification function»). Elle fournit le profil du facteur de grossissement (ou de réduction) associé à l'ensemble de l'image.

L'*importance a priori* (*API* pour «a priori importance») d'un nœud permet, comme dans [Furnas 1981], d'indiquer l'importance relative du nœud dans son contexte global. La *taille* d'un nœud correspond à la taille de la forme le représentant à l'écran. Dans la

Fisheye View, elle est ainsi fonction de la distance du nœud au point de focus, de sa taille normale et de son API.

$$S_{feye}(v, f) = F_2(S_{norm}(v), P_{norm}(v), P_{norm}(f), API(v)) \tag{A.7}$$

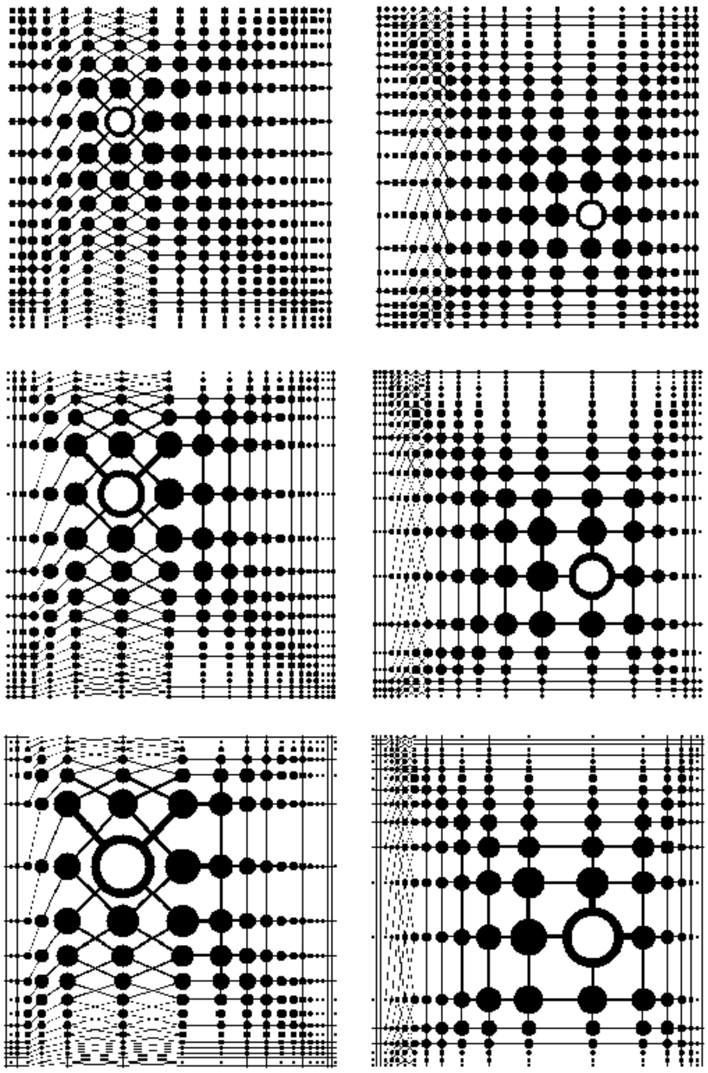


FIG. A.1 – Effet des Fisheye Views sur un graphe [Sarkar et Brown 1992].
La colonne de gauche utilise une distorsion en haut à gauche, celle de droite utilise une distorsion en bas à droite. La valeur d de distorsion augmente de haut en bas, avec respectivement les valeurs $d = 1,46$, $d = 2,92$ et $d = 4,38$.

L'utilisateur peut ensuite sélectionner un élément du point de focus avec la souris, le déplacement de celle-ci entraîne un changement du point de focus en temps réel et permet ainsi à l'utilisateur de se déplacer dans le graphe en fonction des informations qui l'intéressent.

A.3 Approche agent des Fisheye Views

Ces techniques sont particulièrement pertinentes dans le cadre de nos travaux, d'une part parce qu'elles s'appliquent à des informations hiérarchiquement structurées et d'autre part car elles sont adaptées à un point de vue *individu centré*. Il suffit alors de considérer le point de focus comme l'agent sur lequel nous voulons avoir des informations. Si cet agent est un groupe, lui appliquer une technique de Fisheye Views correspond alors à «ouvrir» le groupe pour visualiser le niveau de détail inférieur, ce qui revient à afficher les informations (les agents) qu'il contient. La distorsion permet alors de modifier la position visuelle des agents présents autour du groupe afin qu'ils soient toujours visibles. Ainsi, il est aussi possible de se déplacer graphiquement dans une hiérarchie de groupe, la distorsion pouvant être variable selon certains critères du groupe (nombre d'agents, taille des agents, nombre d'agents-groupes «ouverts», etc.).

Un autre avantage d'associer les techniques agents et la technique de Fisheye View est la possibilité que chaque agent ait son propre cœfficient de distorsion, il devient ainsi relativement simple d'adapter la distorsion lors de l'ouverture d'un groupe au nombre et à la taille des informations contenues dans ce groupe.

Cette approche agent rejoint les travaux de Schaffer [Schaffer *et al.* 1996] sur la navigation dans des réseaux comportant des agrégations de nœuds et créant ainsi des niveaux hiérarchiques de visualisation. Certains nœuds sont regroupés ensemble afin de former un groupe, puis chaque groupe peut à son tour se regrouper avec d'autres et former un nouvel ensemble. En dehors de la différence conceptuelle entre l'approche graphe et l'approche agent, nous pouvons noter plusieurs points de convergence et plusieurs autres points de divergence. Les points de convergence reposent sur la visualisation finale obtenue. Dans les deux cas, chaque groupe (nœud ou agent) peut être «ouvert» ou «fermé». Un groupe fermé est associé à un icône constituant une représentation abstraite de l'ensemble des éléments du groupe. À l'inverse, un groupe ouvert permet de visualiser tous les éléments contenus dans le groupe et ainsi de se présenter un niveau de détail supplémentaire. Les points de divergence reposent d'une part sur le processus de création de la hiérarchie et d'autre part, sur les types d'interactions qui sont offerts à l'utilisateur. Dans les travaux de Schaffer, la hiérarchie est «surimposée» à la structure des graphes en fonction des relations entre les nœuds et elle est stable au cours de la visualisation. La construction de la hiérarchie est une étape supplémentaire et indépendante à la visualisation. Dans notre cas, les agents peuvent se déplacer et interagir dans leur environnement, c'est de cette interaction que vont naître les groupes et par là, la hiérarchisation. Cette hiérarchisation n'est donc pas stable dans le temps mais au contraire, elle peut évoluer en fonction des interactions ou de nouvelles informations entrant dans l'environnement des agents. Ainsi, pour un groupe donné, le cœfficient de distorsion peut évoluer en fonction des entrées et des sorties d'agents dans ce groupe. De plus, l'approche agent permet aussi à l'utilisateur de construire ses propres groupes en temps réel ou de déplacer un agent d'un groupe à l'autre et ainsi de modifier dynamiquement les groupes et la hiérarchisation.

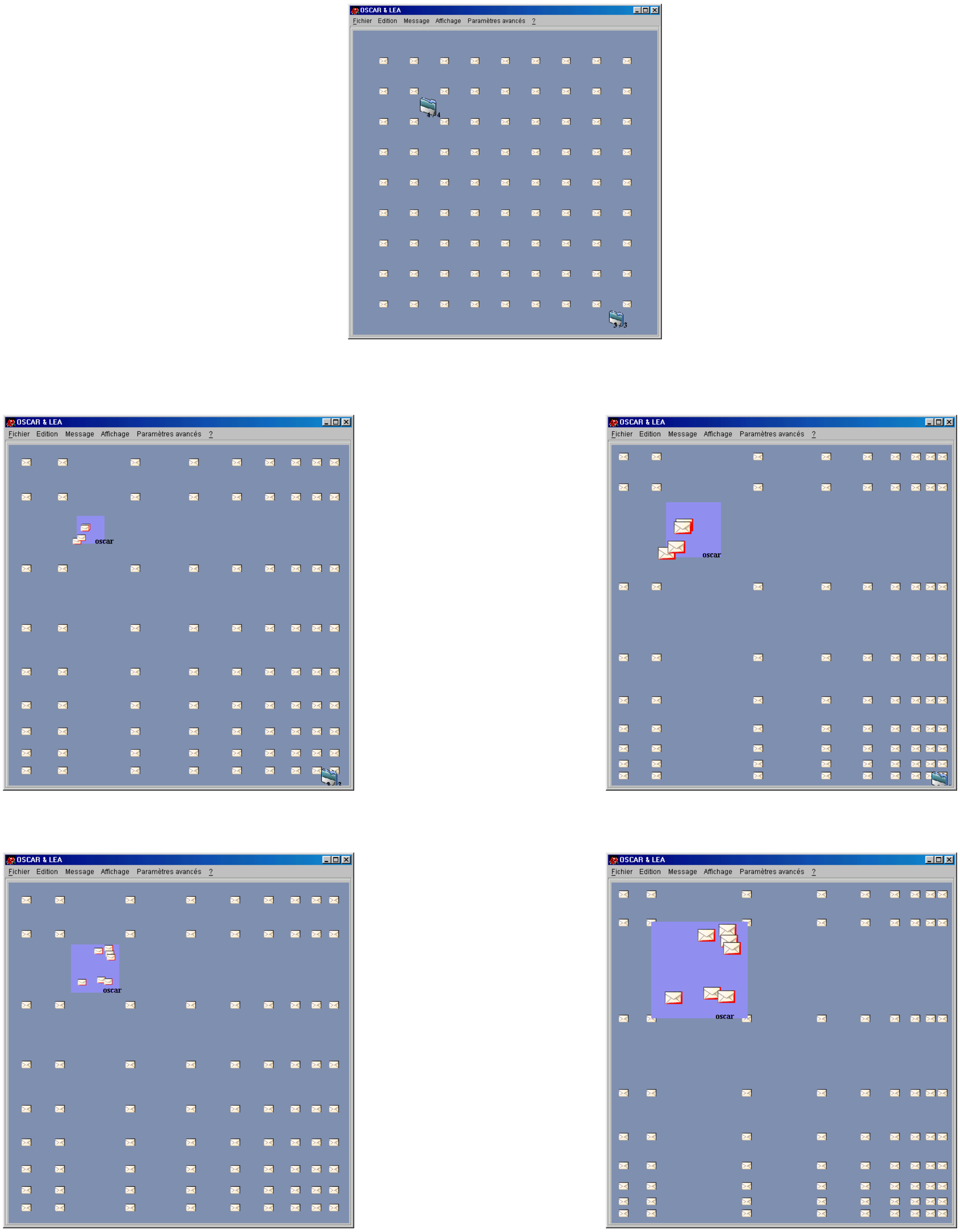


FIG. A.2 – Approche agent de la distorsion de Fisheye Views
La première figure représente les positions initiales des agents. La colonne de gauche utilise une distorsion de valeur $d = 1,5$, celle de droite utilise une distorsion de valeur $d = 3$. Sur la deuxième ligne, l'*agent-groupe* contient 4 agents, sur la dernière ligne, il contient 7 agents.

Annexe B

Grammaire des sessions multi-agents

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!ELEMENT SESSION (INTERFACE, INITIALISATION?, GROUPE_PARENT, AGENTS)>
<!-- une session comporte une interface, des paramètres gérénaux d'initialisation, -->
<!-- un groupe Parent dont l'environnement contient tous les agents visibles de la session -->
<!-- et les agents à instancier -->
<!ATTLIST SESSION name CDATA #REQUIRED
                    date CDATA #IMPLIED
                    auteur CDATA #IMPLIED
                    commentaire CDATA #IMPLIED>

<!ELEMENT INTERFACE EMPTY>
<!-- initialisation de la taille de l'interface à l'écran -->
<!ATTLIST INTERFACE tailleX CDATA #REQUIRED
                   tailleY CDATA #REQUIRED>

<!ELEMENT INITIALISATION (IMAGE*, FICHIER_TEXTE*, FICHIER_DONNEES*, LISTE_MOT_CLEF*, SERVEUR_MAIL?)>

<!ELEMENT IMAGE EMPTY>
<!-- charger et donner un nom à une image, le nom pourra être donné à l'aspect d'un agent -->
<!-- afin qu'il soit représenté par cette image -->
<!ATTLIST IMAGE name CDATA #REQUIRED
                source CDATA #REQUIRED>

<!ELEMENT FICHIER_TEXTE EMPTY>
<!-- charger un fichier texte -->
<!ATTLIST FICHIER_TEXTE name CDATA #REQUIRED>

<!ELEMENT FICHIER_DONNEES EMPTY>
<!-- charger un fichier de données -->
<!ATTLIST FICHIER_DONNEES name CDATA #REQUIRED
                canal CDATA #REQUIRED
                nb_parametre CDATA #REQUIRED
                nb_agent CDATA #REQUIRED >

<!ELEMENT LISTE_MOT_CLEF (MOT_CLEF)*>
<!ELEMENT MOT_CLEF EMPTY>
<!-- définir un mot-clé et lui donner un poids -->
```

```
<!-- peut être utilisé, par exemple, pour initialiser les mots-clés dans LEA -->
<!ATTLIST MOT_CLEF name CDATA #REQUIRED
                   poids CDATA #REQUIRED>

<!ELEMENT SERVEUR_MAIL (SERVEUR, TABLEAU_CLEF_IMG?)>

<!ELEMENT SERVEUR EMPTY>
<!-- spécifique à LEA : paramètre de connexion au serveur de messagerie -->
<!ATTLIST SERVEUR protocole CDATA #REQUIRED
                serveur CDATA #REQUIRED
                user CDATA #REQUIRED
                box CDATA #REQUIRED >

<!ELEMENT TABLEAU_CLEF_IMG (LIEN_CLEF_IMG)*>
<!ELEMENT LIEN_CLEF_IMG EMPTY>
<!-- spécifique à LEA : associer un mot-clé avec une image pour un message simple et une autre pour un dossier -->
<!ATTLIST LIEN_CLEF_IMG name CDATA #REQUIRED
                        nom_img_message CDATA #REQUIRED
                        nom_img_dossier CDATA #REQUIRED >

<!ELEMENT GROUPE_PARENT EMPTY>
<!-- groupe Parent de toute simulation, contient tous les autres agents de la session -->
<!ATTLIST GROUPE_PARENT name CDATA #REQUIRED
                        population CDATA #FIXED "1"
                        actif CDATA #FIXED "true"
                        aspect CDATA #FIXED "none"
                        tailleX CDATA #REQUIRED
                        tailleY CDATA #REQUIRED >

<!ELEMENT AGENTS ((LeMouton  | AgentFuitMouton  | Bois  | Termite )*)>
<!-- lorsqu'une classe est créée, ses paramètres sont rajoutés automatiquement ici afin -->
<!-- que la DTD des sessions reconnaisse cette classe et qu'il soit possible d'instancier -->
<!-- les paramètres de l'agent en externe, par un éditeur XML -->

<!ELEMENT AgentFuitMouton EMPTY >
<!-- description des paramètres de l'agent AgentFuitMouton -->
<!ATTLIST AgentFuitMouton
        type (agent|groupe_agent) "agent"
        repertoire CDATA #REQUIRED
        population CDATA #REQUIRED
        parent CDATA #REQUIRED
        actif (true|false) "true"
        aspect CDATA #IMPLIED
        taille CDATA #IMPLIED
        posx CDATA #IMPLIED
        posy CDATA #IMPLIED
>

<!ELEMENT LeMouton EMPTY >
<!-- description des paramètres de l'agent LeMouton -->
<!ATTLIST LeMouton
        type (agent|groupe_agent) "agent"
        repertoire CDATA #REQUIRED
        population CDATA #REQUIRED
        parent CDATA #REQUIRED
```

```
        actif (true|false) "true"
        aspect CDATA #IMPLIED
        taille CDATA #IMPLIED
        posx CDATA #IMPLIED
        posy CDATA #IMPLIED
        seuilFaim CDATA #REQUIRED
>

<!ELEMENT Bois EMPTY >
<!-- description des paramètres de l'agent Bois -->
<!ATTLIST Bois
        type (agent|groupe_agent) "agent"
        repertoire CDATA #REQUIRED
        population CDATA #REQUIRED
        parent CDATA #REQUIRED
        actif (true|false) "true"
        aspect CDATA #IMPLIED
        taille CDATA #IMPLIED
        posx CDATA #IMPLIED
        posy CDATA #IMPLIED
>

<!ELEMENT Termite EMPTY >
<!-- description des paramètres de l'agent Termite -->
<!ATTLIST Termite
        type (agent|groupe_agent) "agent"
        repertoire CDATA #REQUIRED
        population CDATA #REQUIRED
        parent CDATA #REQUIRED
        actif (true|false) "true"
        aspect CDATA #IMPLIED
        taille CDATA #IMPLIED
        posx CDATA #IMPLIED
        posy CDATA #IMPLIED
        boisPorte CDATA #REQUIRED
        direction CDATA #REQUIRED
>
```

Annexe C

Grammaire des agents

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!ELEMENT classe (import*, parametre*, stimulus*, methode_init, methode_action, methode*)>
<!-- une classe agent comporte des fichiers à importer, des paramètres, -->
<!-- un tableau de stimuli pouvant être vide,
<!-- une méthode d'initialisation lancée en début de session,-->
<!-- une méthode d'action lancée à chaque cycle du scheduler, -->
<!-- et des nouvelles méthodes pouvant soit redéfinir des méthodes existantes dans la classe Agent, -->
<!-- soit définir des méthodes nouvelles -->
<!ATTLIST classe package CDATA #REQUIRED
                 name CDATA #REQUIRED
                 herite_de CDATA #IMPLIED>

<!ELEMENT import EMPTY>
<!ATTLIST import name CDATA #REQUIRED>

<!ELEMENT parametre EMPTY>
<!ATTLIST parametre name CDATA #REQUIRED
                    type CDATA #REQUIRED
                    valeur_defaut CDATA #IMPLIED>

<!ELEMENT stimulus EMPTY>
<!ATTLIST stimulus name CDATA #REQUIRED
                   valeur CDATA #REQUIRED
                   valeur_min CDATA #REQUIRED
                   valeur_max CDATA #REQUIRED
                   portee CDATA #REQUIRED
                   portee_min CDATA #REQUIRED
                   portee_max CDATA #REQUIRED
                   fonction_diffusion (proportionnelle|constante|gaussienne) "proportionnelle" >

<!ELEMENT methode_init (code*)>

<!ELEMENT methode_action (code*)>

<!ELEMENT methode (parametre_methode*, code*)>
<!ATTLIST methode name CDATA #REQUIRED>
<!ATTLIST methode accessibilite (public|private) "public">
```

```
<!ATTLIST methode type_retour CDATA #IMPLIED>

<!ELEMENT code EMPTY>
<!ATTLIST code instruction CDATA #REQUIRED>

<!ELEMENT parametre_methode EMPTY>
<!ATTLIST parametre_methode type CDATA #REQUIRED
                            name CDATA #REQUIRED>
```

Annexe D

Fichiers XML et Java pour l'exemple des Termites

Fichier Bois.XML décrivant la classe des agents Bois en respectant la grammaire des agents :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE classe PUBLIC "oscar" "modeleJava.dtd">
<classe package="agents" name="Bois" herite_de="Agent">
<import name="noyau.Agent"/>
<import name="java.awt.Color"/>

<methode_init>
    <code instruction="couleur = Color.darkGray;"/>
    <code instruction="posx = (int) (Math.random()*550);"/>
    <code instruction="posy = (int) (Math.random()*550);"/>
    <code instruction="taille = 10;"/>
    <code instruction="set_actif(false);"/>
</methode_init>

<methode_action>
</methode_action>

</classe>
```

Fichier Bois.java après traduction du fichier XML par XSLT :

```
package agents;
import noyau.Agent;
import java.awt.Color;

public class Bois extends Agent {

/**************************/
/****    Constructeur   ****/
/**************************/
public Bois () {
}

public Bois (int nb) {
        super(nb);
} //fin du constructeur

/*************************/
/****   Methode init   ****/
/*************************/
public void init() {
        couleur = Color.darkGray;
        posx = (int) (Math.random()*550);
        posy = (int) (Math.random()*550);
        taille = 10;
        set_actif(false);
}// fin de la methode init

/**************************/
/****  Methode action   ****/
/**************************/
public void action() {
}// fin de la methode action

} //fin de la classe
```

Fichier Termites.XML initialisant une session comportant des agents Termites et des agents Bois :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE SESSION SYSTEM "..\\sessions\\modeleSession.dtd">
<SESSION name="Termites" date="aout 2001" commentaire="modèles des termites qui
                        regroupent des morceaux de bois" auteur="Valérie Renault">

  <INTERFACE tailleY="600" tailleX="600"/>
  <GROUPE_PARENT tailleY="600" tailleX="600" population="1" name="Parent" aspect="none" actif="true"/>

  <AGENTS>
      <Bois type="agent" taille="10" repertoire="agents" population="300" parent="Parent" actif="false"/>
      <Termite type="agent" taille="10" repertoire="agents" population="100" parent="Parent"
               direction="0" boisPorte="defaut" actif="true"/>
  </AGENTS>

</SESSION>
```

Résumé

La diversité des données auxquelles un utilisateur est confronté est en continuelle augmentation. Le développement d'outils d'organisation et de représentation visuelle est donc de plus en plus nécessaire, notamment lorsque ces données sont dynamiques. Nous proposons d'utiliser des systèmes multi-agents pour concevoir ces outils. La visualisation des données se traduit par la visualisation d'agents réactifs et de leurs interactions dans leur environnement. Chaque agent prend en charge une partie de l'information et est doté de capacités de communication rudimentaire via son environnement. L'objet de cette thèse est de montrer comment, en dotant ces agents de comportements issus de modèles éthologiques (insectes sociaux), il est possible d'aboutir à des mécanismes de représentation visuelle de l'information. Nous montrons ainsi comment l'introduction de comportements d'attraction-répulsion entre agents peut induire une organisation et un filtrage des données et comment la formation de groupes d'agents peut aboutir à une synthèse et à une hiérarchisation de l'information. La conception de la plate-forme OSCAR a été nécessaire pour aboutir à un noyau multi-agent générique répondant aux exigences des simulations multi-agents (définition de comportements de regroupement, émission de phéromones virtuelles, etc.) mais aussi à celles des interfaces de visualisation d'informations (prise en charge et visualisation de données dynamiques, interactions avec l'utilisateur, etc.). LEA est une application développée à partir du noyau OSCAR. Elle visualise le contenu de boîtes aux lettres électroniques. Chaque agent prend en charge un message et se positionne dans son environnement en fonction des interactions qu'il a avec les autres agents. Ces interactions évoluent selon l'intérêt que porte l'utilisateur à certains mots-clés. L'aspect dynamique provient d'une part de l'arrivée de nouveaux messages et d'autre part de la personnalisation en temps réel des agents par l'utilisateur.

Mots-clés: Systèmes multi-agents, agents réactifs, visualisation d'informations dynamiques, organisations, éthologie

Abstract

As volume of data is growing every day, the development of data visualization and organization tools becomes more and more useful, especially when data are dynamic. We propose to use multi-agent systems to develop such tools. Data visualization is performed as visualization of reactive agents and of their interactions in their environment. Each agent takes charge of a part of information and has rudimentary means of communication via its environment. The aim of this thesis is to show how by using agents with behaviors based on ethological models (social insects), information is organized on the screen. First, we show how the introduction of attractive and repulsive behaviors succeeds in data organizing and screening. Next, we present a new way of information synthesis and hierarchy thanks to agent groups. The OSCAR framework has been implemented in order to have a generic multi-agent core. It complies with requirements of both multi-agent simulations (definition of aggregation behavior, virtual pheromone, etc.) and information visualization interfaces (take charge of and visualize dynamic data, user interactions, etc.). LEA is an application based on OSCAR. It visualizes an electronic mailbox. Each agent takes charge of an e-mail and it finds a place in its environment according to its interactions with the other agents. These interactions evolve with the user's interests. Dynamics come from two elements : first, new messages arrival modifies the organization ; second, the user may personalize the agents, in real time.

Keywords: Multi-agent system, reactive agent, dynamic information visualisation, organization, ethology