Conception d'un système multiagents adaptatif : application à la gestion de crise

Hadhoum Boukachour
Alain Cardon
Stéphane Durand
Franck Lesage

Résumé

Nous présentons les caractères de la modélisation et de l'implémentation d'un Système d'Information et de Communication de gestion de crises civiles. Nous prenons en compte, dans la gestion de la connaissance du domaine, les avis et intentions des différents acteurs échangeant des informations. Nous introduisons la notion d'acte de communication pour représenter la valeur et le sens des informations échangées par les décideurs. Nous utilisons une modélisation par systèmes multiagents dynamiques, ayant pour propriété de se modifier et de se générer par eux-mêmes, suivant les caractères attribués aux informations échangées.

Mots clés

Système d'Information et de Communication, Système multiagent, morphologie d'un système d'agents, représentation et interprétation de la connaissance.

Abstract

We present the characters of the modelisation and implementation of an Information and Communication System for civil crisis management. We take into account, in the management of domain knowledge, the opinions and intentions of the different actors exchanging information. We introduce the notion of communication act in order to represent the value and the meaning of the exchanged information between the decision makers. We use a modelisation based on dynamic multiagent systems which have the capability of self-modification and auto-generation according to the characteristics of the exchanged information.

Keywords

Information and Communication System, Multiagent system, morphology of agents system, representation and interpretation of knowledge.

Table des matières

Introduction

Les Systèmes d'Information utilisables pour le traitement des phénomènes à fort degré de gravité et d'urgence, comme les crises industrielles ou les catastrophes climatiques, sont les SIC, Systèmes d'Information et de Communication. Ces systèmes doivent permettre de représenter et structurer la connaissance des acteurs décisionnels sur la situation en cours d'évolution, par échange d'informations sur des réseaux à haut-débit, dans un système distribué et fermé.

L'échange de connaissances objectives, factuelles, sur l'état d'une situation d'urgence est fondamental pour sa gestion, puisqu'il en permet la caractérisation puis l'analyse pour un traitement bien planifié. Mais ce type de connaissances s'avère insuffisant [Shrivastava, 1994] : il est également nécessaire de prendre en compte les représentations mentales des décideurs à propos du phénomène tel qu'ils le perçoivent [Borodzicz et al., 1993], [Toft and Reynolds, 94]. Ces représentations mentales de la situation telle qu'elle est perçue par les décideurs qui ont à la gérer sont en effet toujours altérées par rapport à des représentations habituelles, standards, sous la pression des événements, modifiant ainsi la reconnaissance objective des faits observés. Les caractères de ces représentations doivent être détectés et reconnus, et cette reconnaissance propagée par le Système d'Information comme une connaissance significative, modifiant les interprétations de toutes les autres connaissances, pour lever les ambiguïtés, détecter les incompréhensions ou les contresens entre acteurs, c'est-à-dire en fin de compte pour éviter une "crise dans la crise". La prise en compte explicite des caractères intentionnels conduisant les connaissances communiquées puis les décisions des acteurs est un élément essentiel dans le traitement du phénomène de crise lui-même [STEP, 1994] et il doit être explicitement apparent dans la représentation de la connaissance à propos du phénomène.

L'architecture du Système d'Information et de Communication que nous étudions doit donc permettre d'exprimer deux catégories de connaissances [Cardon and Durand, 1997]. Une architecture en réseau classique va permettre l'accès et l'échange de données factuelles sur la situation, par manipulation de bases de données et de Systèmes d'Information Géographiques. Une autre architecture, enveloppant la précédente, permettra d'exprimer la perception du phénomène par les acteurs, c'est-à-dire leur appréciation subjective de la situation telle qu'ils la perçoivent. Cette seconde catégorie de connaissances devra être représentée, en général graphiquement, et rendue disponible à tous les décideurs. Le système doit permettre des accès simultanés facilitant les prises de décision à plusieurs niveaux de responsabilités [Sfez, 1992], c'est-à-dire incluant la subjectivité des intentions des acteurs et des groupes d'acteurs.

Nous nous plaçons ainsi dans le domaine des Systèmes d'Information et de Communication dits complexes [Cardon, 1996], c'est à dire gérant des situations dont l'évolution n'est pas bien planifiée à l'avance et dont la couche cognitive, augmentée par les connaissances de perception et d'intention des acteurs, est à construire, augmenter et modifier pendant l'usage du système.

Nous exprimons les intentions des acteurs gérant le phénomène par la représentation de ce que nous appelons des actes de communications [Cardon and Durand, 1997]. Nous donnons au fait de communication entre acteurs un caractère d'action effective dans le sous-système de représentation de la connaissance de la situation à gérer. Toute communication se traduit alors par une génération d'entités adaptatives représentant les éléments de la variation de connaissance à propos de la situation dans leur contexte d'énonciation, que l'on nomme usuellement la pragmatique [Cordier, 1994]. Tout accès au système sera ainsi motivé par l'acteur, explicitant les tensions émotionnelles et les enjeux pour devenir, outre un échange d'informations, un acte effectif de communication. Nous représentons ces actes de communication, qui traduisent des variations de la connaissance sur la situation, par des systèmes dynamiques, à structure variable, des systèmes multiagents dans lesquels les agents représentent les différents traits sémantiques caractérisant à la fois les éléments du discours et de sa pragmatique [Cordier, 1994].

Nous nous appuyons sur la notion d'adaptation du système d'agents, c'est à dire sur l'expression de son organisation même, pour représenter la synthèse des faits, avis et intentions exprimés par les acteurs. L'usage des systèmes multiagents apparaît ici comme organiquement nécessaire. Tout acte de communication dans le SIC va engendrer la création effective de nombreux agents, cognitifs ou réactifs, la modification structurelle de certains autres, ou encore leur réduction. Nous pourrons alors prendre en compte et exprimer la morphologie du paysage d'agents [Cardon, 1997], son organisation, ce qu'il représente comme catégories cognitives réifiées, pour chaque contexte de communication et pour l'ensemble des communications. Cette expression sera une représentation du sens du phénomène perçu par les acteurs [Jacques et al., 1995]. Seul l'usage d'un système adaptatif et exprimant son adaptation permet cette approche. Nous nous plaçons donc dans le cadre de l'étude des systèmes multiagents hybrides [Strugeon, 1995] et dynamiques, en insistant particulièrement sur l'aspect conatif de ceux-ci [Hayes-Roth and Collinot, 1993].

Nous décrivons d'abord la problématique générale des SIC, l'architecture de la plate-forme multiagents que nous avons développée et allons utiliser puis les spécifications des différentes organisations d'agents. Nous présentons enfin les caractères de la distribution du système en CORBA2.

Chapitre 1

Architecture générale d'un système de gestion des situations de crise

Une situation d'urgence est caractérisée par une rupture dans une organisation socio-économique nécessitant une intervention et un traitement immédiats [Quarantelli, 1998]. Ce peut être, par exemple, un accident technologique ou écologique grave dont les traitements différés ou l'absence de traitement peuvent avoir des conséquences estimées très importantes et pouvant faire évoluer l'accident en catastrophe, puis éventuellement en crise. La crise sera la situation limite, la rupture organisationnelle où la gestion du phénomène se limitera à faire les meilleurs choix de sauvegarde de certaines fonctionnalités et la perte plus ou moins consentie de certaines autres [Borodzicz et al., 1993].

Les Systèmes d'Information et de Communication (SIC) sont usuellement les systèmes chargés d'aider les décideurs gérant les interventions sur les situations d'urgence à prendre les bonnes décisions [Cardon, 1997]. Ce sont les héritiers à la fois des Systèmes d'Information, fondés sur l'accès à des bases de données de grande taille, et des Systèmes de Commandement militaires, fondés sur la gestion hiérarchique d'organisations vastes et denses [Bares, 1996]. Ils utilisent toujours une certaine représentation des connaissances sur la situation, évidemment la plus conforme à l'idée que les institutions se font de la connaissance nécessaire, pour aider aux diverses prises de décision.

1.1 L'architecture distribuée d'un SIC

Un SIC de gestion de crises se base sur une composante communicationnelle mettant en relation les principaux acteurs décisionnels de la gestion de la situation. Dans le cas que nous étudions, il s'agit de la Préfecture de Région, du Samu, du Centre de Commandement des Sapeurs Pompiers, des Services de Police, des Services de gestion de crise de la Mairie. Tous ces points sont les noeuds géographiquement distants d'un réseau communicationnel, représentent les Postes de Commandement des différentes institutions, aux pratiques différentes.

La partie technique d'un SIC est donc constituée d'un système de communication classique entre les différents acteurs décisionnels. Chacun de ces acteurs est un élément du réseau communicationnel [Hayes-Roth and Collinot, 1993], représentant un décideur qui dialogue avec les autres via un réseau haut-débit, en disposant d'une interface personnalisée pour envoyer ses messages hypertextes et consulter ses Systèmes d'Information propres. Le réseau est fermé, dans le sens ou aucun nouvel acteur imprévu ne peut y être introduit pendant la gestion du phénomène (Fig 1.1).

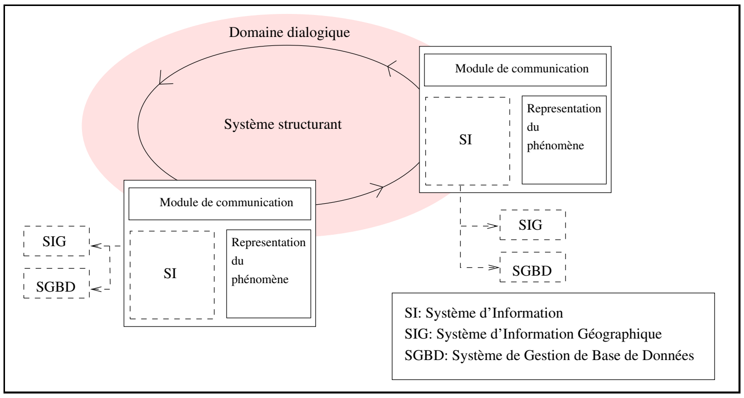

FIG. 1.1: *L'architecture distribuée d'un SIC*

Au niveau de cette composante communicationnelle, la connaissance nécessaire pour réaliser l'interconnexion et l'intéropérabilité revient à collecter le maximum de faits objectifs concernant la situation pour tenter de la caractériser, de la mettre en adéquation avec des situations connues, puis prévoir au plus tôt son évolution pour enfin engager les moyens appropriés. A ce niveau, le SIC opère donc comme un système d'aide à la planification dynamique. Il procède d'une démarche de collecte systématique de renseignements, certains étant fournis automatiquement, pour ensuite analyser, évaluer, envisager et agir en se basant sur des situations déjà prévues.

La composante renseignement, avec accumulation de connaissances préalables, y est particulièrement développée. Ces sources d'information s'appuient principalement sur les données de capteurs et des images venant de différentes sources. Les informations sont stockées dans des bases de données puis transférées dans des bases de connaissances symboliques. Ces connaissances sont synthétisées et représentées comme des entités objectives pour être utilisées dans les prises de décision stratégiques et tactiques, en assurant la cohésion des actions des différents unités sur le terrain. Les niveaux d'interconnexion et d'interopérabilité entre les diverses entités opérationnelles y sont bien pris en compte [Oziard and Ruhla, 1992]. Le problème de la coopération effective entre les opérateurs y constitue dorénavant un thème de recherche considéré comme fondamental [Bares, 1996] mais qui pose le problème de la prise en compte de la motivation et de l'intention des acteurs.

Chapitre 2

La connaissance situationnelle échangée entre les acteurs décisionnels d'un SIC

La gestion d'une situation d'urgence et de crise est un problème difficile dans le sens où le domaine de la connaissance à propos de la situation à gérer est typiquement mal structuré, les acteurs mal informés et la situation est à évolution rapide et peu prévisible [Winograd and Flores, 1986]. Cette connaissance, que nous appellerons connaissance situationnelle, exprime les représentations des acteurs décisionnels à propos de la situation que ceux-ci gèrent en commun, lors de leurs communications. Cette connaissance se construit par les messages échangés entre les acteurs distants. Il est donc nécessaire d'échanger des informations factuelles, objectives et exprimant des faits indéniables, et également de fournir une représentation des avis, appréciations et intentions des acteurs sur ces faits observés et sur les actions engagées.

2.1 Le caractère évolutif de la connaissance situationnelle

Dans un système de gestion de crise, de nombreux acteurs institutionnels communiquent simultanément sur le réseau, avec une fréquence éventuellement importante et ceux-ci énoncent des faits complexes, car mal perçus, mal reconnus par eux-mêmes, chargés de questionnements. Toute énonciation est donc chargée d'une intentionnalité non évacuable [Cardon, 1996]. Nous dirons que la situation de communication entre les acteurs est une situation dialogique [Jackendoff, 1983].

Chaque acteur interprète les faits du réel qu'il perçoit (un objet du réel) et les représente par des signes (le signe associé à l'objet), comme dans toute sémiotique triadique [Peirce, 1984]. Le système se doit, tant que faire se peut, de prendre en compte ces interprétations médiatisant objet et signe, pour représenter les actes d'interprétation (Fig 2.1).

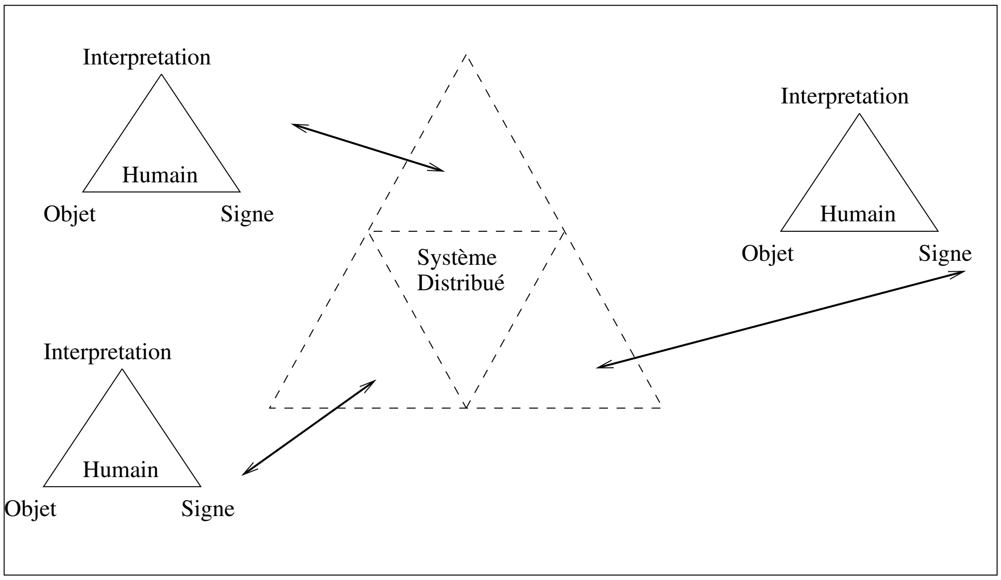

FIG. 2.1: *Le problème de coopération multiple*

Autant pour deux utilisateurs distants il est possible d'arriver à une compréhension mutuelle suffisante en un temps relativement court, autant le problème devient beaucoup plus difficile lors d'une session regroupant un plus grand nombre de participants. Il est impossible de prendre le temps nécessaire à une mise en adéquation dialogique de chacun des protagonistes du discours par rapport à tous les autres, si tant est que cela soit possible.

Le SIC doit donc représenter les intentions des acteurs à propos de toute connaissance objective manipulée, pour permettre de réaliser la cohésion et la coopération entre les groupes, en tenant compte des doutes et oppositions de point de vue qui engendrent éventuellement des "crises dans la crise" [STEP, 1994]. De telles ruptures ont été remarquées lors de nombreuses crises et notamment lors de la catastrophe aérienne de l'Airbus A320 le 20 janvier 1992 au Mont-Sainte- Odile, où les différentes institutions se sont physiquement opposées dans l'organisation des secours, conduisant à une crise dans la crise. Très souvent, les communications permettent la propagation de rumeurs et d'hypothèses non fondées, lorsque les faits transmis ne sont pas validés par l'expression de leur contexte d'énonciation et de leur portée. Cela entraîne des défauts de compréhension du phénomène et engagent l'organisation des secours dans des voies erronées. Le développement effectif de la représentation de la perception de la situation par tous les acteurs doit permettre de pallier à ces défauts, de prendre en compte la dimension humaine et sociale des informations produites par les opérateurs en situation de crise.

2.2 La fonction de représentation des connaissances d'un SIC

Les fonctions générales du SIC à construire, vu comme un système de gestion de connaissances situationnelles évolutives, se basent sur les deux caractères suivants :

1. On considère qu'il existe une connaissance explicite pour le traitement opérationnel des situations d'urgence, constituée de nombreux plans préétablis. C'est le cas des Plans Particuliers d'Intervention (PPI) mis en place dans les différentes Régions et permettant de conduire les actions coordonnées des services de protection civile dans un cadre défini à l'avance. Sous cette forme, le système de gestion de crise a un certain degré d'efficacité et malheureusement d'inefficacité, c'est-à-dire qu'il n'est opérant que dans les domaines limités déjà étudiés. La construction du SIC de gestion de crises fondée sur l'échange informatisé d'informations commentées et enrichies est une évolution sensible et une transformation des organisations en place, entraînant des conséquences éventuellement importantes pour celles-ci.

2. Le système de gestion de crise a au moins la complexité des organisations des institutions qui y interviennent. Sa construction nécessite à la fois une analyse des différentes structures des institutions et du champ géographique de la crise et également du comportement et de l'intentionnalité des différents acteurs. En situation de crise, la modification des processus mentaux des décideurs est très importante, comme cela a été montré par de nombreuses études [Borodzicz et al., 1993], [Toft and Reynolds, 94] et étudié dans un Contrat CEE STEP sur le sujet [STEP, 1994]. Dans le domaine des SIC civils, les architectures traditionnelles basées sur les données, et dont la tendance est la prévision exhaustive, sont considérées par les professionnels eux- mêmes comme insuffisantes. Le SIC a donc une composante d'aide à la décision de caractère non linéaire [Sfez, 1992] prenant en compte les comportements et intentions des acteurs et groupes d'acteurs. L'augmentation de la quantité et de la qualification des informations échangées va orienter le système vers une nécessaire adaptativité à ce flux de connaissances manipulées.

2.3 Les six niveaux d'organisation de la connaissance situationnelle

Nous représentons (Fig 2.2) les six niveaux d'organisation de la connaissance situationnelle dans un SIC. Cette décomposition correspond à celle de l'analyse systémique [Lapierre, 1992] [Le Moigne, 1990] des organisations.

Les trois premiers niveaux sont typiques des Systèmes d'Information classiques et des SIC militaires. Ils permettent de décrire une situation bien planifiée où l'information arrive au bon moment au bon endroit. Le quatrième assure la

1. Substrat physique, entités objectives observables
2. Espace de déploiement des entités observables
3. Déplacements, organisations, plans d'action
4. Communication de l'Information
5. Valeurs humaines, symboles, sens du phénomène, intentions
6. Règles du jeu social, relations de pouvoir, émergence du sens du phénomène

FIG. 2.2: *Les niveaux d'organisation d'un SIC*

mise en oeuvre des trois précédents, c'est-à-dire leur organisation dans l'espace et le temps. Les deux derniers niveaux ont des caractères typiquement sociaux et réfèrent aux valeurs qui fondent l'existence des formes sociales et le comportement intentionnel des acteurs. Ces niveaux sont ceux auxquels nous nous intéressons plus particulièrement pour prendre en compte, dans le processus de communication de l'information, les intentions et représentations mentales des acteurs. Remarquons que le niveau de Communication, qui relève typiquement de l'Informatique, a un rôle charnière essentiel entre les parties informationnelles et les parties référant à l'intentionnalité et aux valeurs sociales.

Chapitre 3

Les systèmes adaptatifs

La notion de système, dans l'approche usuelle de conception, est basée sur la création pièce par pièce, d'un ensemble d'algorithmes et de structures de données sur lesquelles ces algorithmes vont opérer. On se place toujours dans le cadre de la résolution de problèmes, le système ayant un état de départ, prenant ensuite des données en entrée, les traitant et donnant en sortie un ou plusieurs résultats pris dans l'ensemble des solutions possibles.

Cependant, le système de gestion de crises que nous avons à réaliser ne doit pas se contenter de résoudre un problème. Il doit concevoir une vision de la situation à gérer, et ce à partir des messages échangés entre ses utilisateurs: il doit adapter sa structure à ces messages afin que celle-çi soit l'expression du sens global de la situation. Un tel système fait partie de la famille des systèmes adaptatifs.

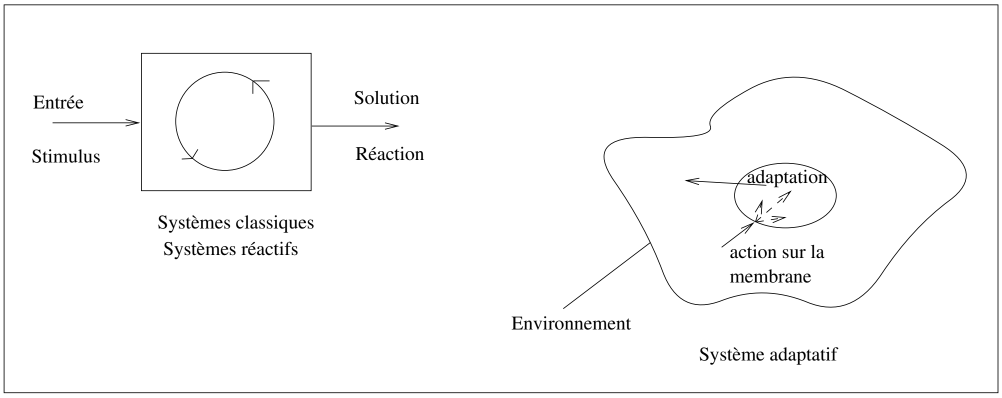

FIG. 3.1: *Systèmes adaptatifs vs Systèmes réactifs*

3.1 Définition

Les systèmes adaptatifs

Un système multiagents adaptatif est un ensemble d'organisations d'agents que limite une membrane, réelle ou virtuelle, et qui adapte dynamiquement son organisation aux sollicitations de l'environnement par clôture opérationnelle dense [Cardon, 1998].

La membrane est à la fois le limitateur des actions de réorganisation du système et l'interface avec l'extérieur. L'action de cloture opérationnelle est la capacité de réorganisation propre au système, pour adapter sa structure à la perception de l'environnement. L'adaptation consiste en une réorganisation des groupes d'agents et en une modification des agents, en genre et en nombre, par des processus de reproduction.

3.2 Les systèmes d'interprétation des informations

Les messages échangés entre les acteurs d'une situation d'urgence doivent servir à gérer des moyens, des hommes et à diffuser des jugements sur l'évolution des évenements. Cependant, ces messages ne sont pas émis au hasard mais sont motivés par une certaine intentionnalité et chargés d'une pragmatique qui leurs sont propres. Un système d'aide à la décision se doit de prendre en compte cette connaissance distribuée afin d'augmenter qualitativement les messages échangés. Cette augmentation doit permettre au système de fournir une vision synthétique aux différentes institutions devant prendre en charge la gestion de la situation. Nous pensons qu'une certaine forme de système adaptatif doit permettre une telle augmentation.

Chapitre 4

Présentation de l'architecture générale du système d'interprétation des messages échangés

Nous présentons ici l'architecture que nous avons développée afin de construire un système adaptatif d'interprétation des informations échangées pour la gestion de crise. Cette architecture repose sur quatre organisations d'agents qui auront pour but de mettre en place le processus d'interpretation des messages échangés. Nous allons voir quel est le rôle de chaque organisation et comment chacune d'entre elle s'articule avec les autres.

4.1 Un modèle à quatre organisations d'agents

Nous devons être capables de traiter sémantiquement, et plus précisement, d'interpreter les messages échangés entre les acteurs. Pour ce faire, le système sera à la fois le médium de communication entre les participants et le moyen d'interpretation de ces messages afin de donner en retour une lecture synthétique de la situation. Le système aura donc une composante fortement communicationnelle. Cette partie sera constituée par une organisation d'agents hybrides: le *SMA Aspectuel*. Cette couche aspectuelle est composée d'autant de SMA que d'utilisateurs. Leur rôle sera de rendre compte dans le système des différentes représentations de la situation des participants auxquels ils seront rattachés (Fig 4.1).

Le *SMA Aspectuel* sera composé d'agents aspectuels dont le rôle sera d'extraire des traits sémantiques des messages échangés qui, en quelque sorte, les nourirront. Les messages conditionneront donc l'état d'une certaine organisation des agents aspectuels. Nous faisons l'hypothèse que l'état de l'organisation d'agents se représente par une conformation morphologique [Cardon and Lesage, 1998] et que cette conformation particulière est l'interpretation que fait le

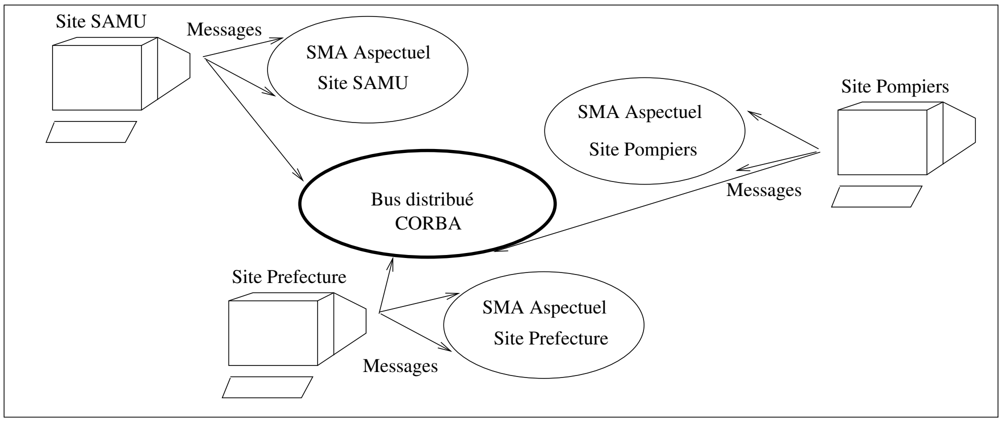

FIG. 4.1: *Les liens SMA Aspectuels - Acteurs*

système de la représentation que l'acteur a de la situation.

Les différents SMA aspectuels attachés à chaque acteur présentent chacun des organisations particulières de leurs agents. Ces agents tendent à s'agréger, s'affronter ou se renforcer afin de faire émerger la représentation que chaque acteur de la situation qu'il a à traiter [Cardon and Durand, 1997]. Cependant, pris tels quels, ces SMA ne sont pas compréhensibles directement pour un acteur. En effet, il est très difficile pour un humain, voire impossible, d'appréhender immédiatement tous les caractères émergents du SMA, sans même parler de les suivre au cours du temps.

Il est clair que l'ensemble complexe que représente tous les agents aspectuels liés à un acteur particulier est non appréhendable, même pour le système, sans un traitement bien particulier. Comme dit précédemment, nous pensons que cet ensemble d'agents possède de façon intrinsèque des caractéristiques de morphologie: il donne naissance à un "paysage d'agents", c'est-à-dire à une représentation géométrique de ses caractères.

Aussi, nous proposons un type particulier d'agents : les *Agents de Morphologie* qui feront apparaitre par émergence les traits morphologiques saillants d'un paysage d'agents. Ces agents sont regroupés dans un *SMA de Morphologie* qui aura donc pour rôle de fournir une vision synthétisée des caractères géométriques (et sémantiques) du SMA Aspectuel.

La mise en évidence de ces caractères va permettre en quelque sorte de donner les bases nécessaires au déclenchement du fonctionnement de la partie rationnelle du système. Comme expliqué dans [Cardon and Lesage, 1998], les faits saillants réifiés dans le SMA de Morphologie représentent les soucis possibles du système, des attracteurs d'attention, concernant le système lui-même ou son extérieur. Dans notre cas, les faits saillants seront les points importants du discours sur la situation à gérer. Une fois ces points identifiés, le système

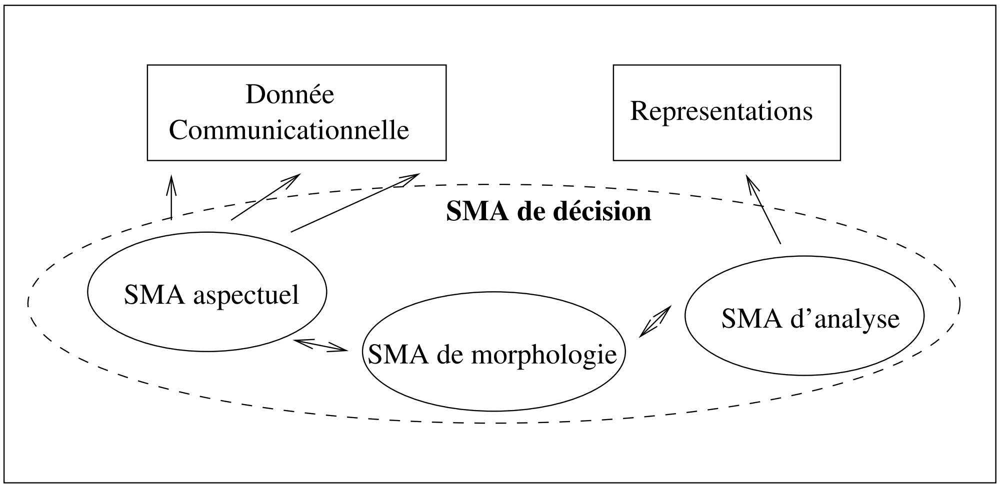

FIG. 4.2: *Les organisations d'agents dans le système*

doit opérer dessus au travers de sa partie rationnelle que constitue le *SMA d'Analyse*. Ce SMA est constitué d'agents dont le but va être de situer ces faits saillants dans l'espace et le temps par rapport à des plans pré-établis (plans d'intervention, plan ORSEC, etc ...) et relativement à des systèmes d'inférences divers et variés.

Enfin, pour être en mesure de maintenir une certaine cohérence dans le système, nous le dotons d'une organisation *d'agents de décision* qui vont opérer des choix sur les agents d'analyse actifs et sur les caractères morphologiques qui les alimenteront. Ces agents travailleront selon des principes relativement simples [Cardon, 1998].

Chapitre 5

Réalisation d'un environnement de développement et de mise au point de Systèmes MultiAgents

A chaque fois que l'on doit réaliser un SMA, on a le choix entre utiliser une plateforme déjà éxistante ou bien développer la sienne. Cependant, le point récurrent dans un tel choix est l'adéquation de la plateforme aux contraintes du modèle que l'on a à implémenter. Il n'existe pas à notre connaissance de plateforme offrant des agents de morphologie génériques. De plus, les contraintes fortes en besoins de simulation et débogage et les caractères des architectures d'agents disponibles ne sont pas tout à fait compatibles avec le grand nombre d'agents (plusieurs centaines) nécessaire au fonctionnement de notre système. Enfin, aucune plateforme, à notre connaissance, n'intègre CORBA dans sa version utilisable. Aussi, avons nous choisi de développer une mini-plateforme offrant les outils de base nécessaires au développement de notre modèle.

5.1 Définition des buts

Nous cherchons à mettre en place un moyen de développer un ensemble minimal d'outils cohérents devant permettre l'implémentation, le débogage et la simulation du fonctionnement d'une organisation d'agents. On peut appeler "plateforme SMA" un tel ensemble, bien que généralement cette notion suppose la mise à disposition non seulement d'une architecture d'agents de base mais aussi d'outils annexes tels qu'une librairie de comportements et de protocoles de communication entre les agents, ou encore un langage de définition de leurs compétences.

Ici, notre but n'est pas tant de fournir cet ensemble complet d'outils que de nous donner simplement les moyens de pouvoir tester nos agents, de faire des simulations et des tests de performance. En effet, un système d'aide à la gestion de crise doit être fiable et rapide. Plus clairement, les outils que nous nous proposons de développer pour nous permettre de construire notre système et le tester devront :

1. fournir un outil permettant de regrouper des agents de façon à pouvoir les contrôler facilement lors de simulations ou de débogage,

2. permettre la manipulation d'un sous-ensemble d'agents,

3. permettre une prise d'information minimale pour les agents.

Un système adaptatif est très difficile à mettre au point dans la mesure où il doit être en mesure de fournir des résultats compréhensibles aux utilisateurs mais aussi d'être capable d'avoir une certaine compréhension de ses actions afin de pouvoir évaluer l'interprétation qu'il a de la situation. C'est précisement cette boucle d'interprétation qui est compliquée à stabiliser, et la possibilité d'avoir un outil de développement permettant un suivi au plus près du fonctionnement du système est primordiale.

5.2 Une architecture d'agent ?

Quand on parle de plateforme SMA, on pense immédiatement à la structure d'agent que l'on va employer. La plateforme DIMA [Guessoum, 1997] ou d'autres systèmes basés sur d'AcTalk [Briot, 1994] auraient pu convenir, mais la nécessité d'integrer CORBA et les contraintes de taille de la population d'agents et de simulation/débogage risquaient de poser problème. Aussi, lors de l'implémentation de notre mini-plateforme nous avons mis l'accent sur le débogage, la simulation et la rapidité d'éxécution qui sont les principales préocupations qui ont guidé la façon dont nous avons codé nos agents. Bien que d'une façon générale nous ne proposons pas une architecture d'agent générique, le fonctionnement des agents que nous avons dû développer se rapproche assez d'Actalk et une fois que le prototype sera validé, nous travaillerons sur la migration de nos agents vers cette architecture, dans un souci de généricité, si toutefois cela ne dégrade pas trop les performances.

5.3 La classe MAS (MultiAgent System)

Dans la mesure où nous avons plusieurs organisations d'agents distinctes dans notre système, il nous faut un moyen de les identifier. Or une organisation d'agents est assimilable à un SMA et de plus nous devons pouvoir offrir des fonctionalités de simulation et de débogage.

Aussi, nous avons isolé un objet fondamental dans notre système : le SMA. Il est donc tout naturellement présent dans notre implémentation sous la forme d'un objet SMA qui propose un certain nombre de services.

5.3.1 Spécification objet d'un SMA

Un SMA dispose d'une horloge sur laquelle tous les agents peuvent se synchroniser. Il contient aussi le temps depuis la création du système (de l'objet SMA) et tout un ensemble de données utiles au développement et à la simulation. La majorité de celles-çi sont accessibles depuis l'interface du SMA ou de la plateforme (Fig 5.1 & 5.2).

Variables d'instance	
systemClock	Horloge système
globalClock	Horloge de référence
currentTime	heure courante
status	état du SMA
agentsSet	Ensemble des agents du SMA
smaType	Chaine de caractère décrivant le type de SMA
agentsCount	Nombre d'agents du SMA
runningTime	Temps d'activité du SMA
destroyedAgentsCount	Nombre d'agents detruits
createdAgentsCount	Nombre d'agents créés
messageServer	Médiateur avec l'environnement
messageServerWanted	Booléen indiquant l'obligation d'avoir un messageServer
identifier	Identificateur du SMA
log	Log du SMA
acceptNewAgents	Booléen indiquant si on peut créér manuellement des agents

5.4 La classe Agent

Nous allons exposer les points centraux qui nous ont guidé dans la conception de l'objet Agent tels qu'ils ont été suscités par les caractéristiques minimales communes aux différents types d'agents de notre système.

5.4.1 Fonctionnalités requises

Avant de commencer à définir les différents types d'agents dont nous allons avoir besoin, il est nécessaire de préciser les spécifications techniques minimales qui sont requises. Les agents devront :

1. fonctionner en parallèle avec d'autres agents de façon éventuellement distribuée,

2. être contrôlables pendant leur éxécution (débogage,simulation),

3. fournir un moyen de préciser leur niveau de scheduling (benchs, débogage),

4. être inspectables pendant leur éxécution (débogage),
5. pouvoir fonctionner sans protocole de communication (optionnel),
6. fournir des informations sur leur cycle de vie (débogage),
7. être augmentables à volonté au niveau de l'auto-programmation, c'est-à-dire que les agents devront pouvoir se recoder en cours de fonctionnement.

Le point 1 impose à l'agent de se comporter plus ou moins comme un processus Unix (et pourra même être effectivement un processus Unix si le langage d'implémentation le requiert), ou tout au moins comme une thread d'éxécution.

Le point 2 signifie que l'on doit pouvoir être en mesure d'interrompre, relancer ou tuer un agent, soit depuis une requête interagent ou bien depuis un contrôle extérieur (le développeur le plus souvent). Pour cela nous devons proposer des moyens d'agir sur leur moteur d'éxécution (processus, thread ou autre). C'est typiquement une fonctionnalité de débogage et de simulation.

Le point 3 requiert simplement de pouvoir fixer agent par agent le niveau de priorité de scheduling ainsi que le quantum de temps SMA entre deux réveils de l'agent. On pourra ainsi tester l'impact sur le fonctionnement du système que le niveau d'activité des agents peut avoir.

Le point 4 est essentiellement destiné au débogage des agents : il doit permettre de voir en temps réel certaines caractéristiques de l'agent. Il est clair que les caractéristiques choisies et le moyen de les atteindre sont laissés à la discrétion du développeur. Cependant nous devons fournir des outils de base destinés à cette tache, lesquels seront simplement utilisés tels quels ou encore augmentés.

Le point 5 spécifie que bien que l'agent fonctionne de façon intermitente entre deux périodes de sommeil, il doit etre capable de répondre immédiatement à certains messages. De plus, même s'il est dans une phase d'activité, il doit pouvoir répondre immédiatement à une requête. Cette façon de faire repose typiquement sur l'équivalence message $\leftrightarrow$ méthode.

Le point 6 impose à l'agent de pouvoir se recoder afin de donner éventuellement naissance à un nouveau type d'agent. C'est un point décisif dans le choix du langage d'implémentation : SmallTalk-80 en l'occurence.

Aussi, nous nous sommes limités dans un premier temps à définir une classe Agent minimale qui doit permettre d'ateindre les buts que nous venons de nous fixer.

5.4.2 Envoi de message

La présence de centaines d'agents couplée avec la nécessité d'avoir un système le plus rapide possible nous a contraints à choisir un système d'envoi de messages basé sur l'appel de méthodes d'objet. Il est clair que c'est beaucoup moins souple que l'utilisation d'un protocole de communication entre agents, mais c'est aussi beaucoup plus léger.

Il y a un inconvénient à cette solution : on doit connaitre la table d'équivalence entre message et méthode pour chaque agent. Il est donc hors de question d'avoir de façon simple un système de découverte de nouvelles compétences au cours du fonctionnement du système. Cependant, notre modèle n'éxige pas (encore) cette fonctionnalité. De plus, pour que l'agent ait une autonomie de décision, il faut qu'il puisse ne pas répondre ou mettre en attente une requête. Or l'appel de méthode est "bloquant" et synchrone, donc même si le processus animant l'agent est endormi, cet agent peut quand même répondre à une requête, ce qui est quasiment un non-sens. C'est un problème dont nous sommes pleinement conscient mais qui possède certains avantages. Tout d'abord, comme nous le montrerons plus tard, l'implémentation de la morphologie d'un SMA est très gourmande en messages, nécessite une réponse immédiate des agents et est quasiment incompatible avec un système asynchrone dans lequel les agents pourraient ne pas répondre immédiatement à une requête. Ensuite, elle rend la conception d'un agent beaucoup plus simple et rapide, bien que l'on se trouve alors sans une réelle architecture d'agents.

5.4.3 Comportements de base

La terminaison, le réveil et l'endormissement du processus animant un agent sont les trois seules exceptions au caractère obligatoire d'éxécution de la requête. En effet, bien que ces requètes soient toujours couplées à un appel de méthode de l'objet Agent, elles autorisent la mise en action d'un comportement spécifique à ces requètes. Ainsi, un agent peut tout à fait refuser de se plier à la demande et peut même entrainer la prise de certaines décision (mesure de représailles, etc ...).

Cependant, les besoins soulevés par la simulation ont nécessité l'implémentation de méthodes qui passent outre ces considérations.

5.4.4 Spécifications objet

Le code et les types utilisés ci-dessous sont basés sur l'implémentation du langage SmallTalk de VisualWorks^{tm}. Les variables en italique ne sont indispensables que pour le débogage et/ou la simulation.

L'agent, tel que nous le concevons et partant des considérations établies plus haut, se doit d'avoir un comportement, c'est à dire un ensemble de règles d'actions et de réaction de l'état ou en fonction de sollicitations venant de son environnement ou de son état. Il doit aussi avoir une représentation de cet environnement (donc des moyens de le percevoir) et des moyens d'agir dessus.

Comme finallement notre agent doit être implémenté et que nous avons choisi une modélisation par objets, les objets qui vont nous servir à réifier les agents dans le SMA devront posséder certaines méthodes et attributs devant servir les comportements, moyens de perception, de représentation et d'action.

Les variables qui ne sont pas en italique contiennent les informations minimales qu'un agent se doit d'avoir afin de pouvoir exister dans le système: un processus lui donnant vie, une référence sur son environnement, l'état de son processus et son identification dans le système.

Les autres variables ne sont utiles que pour la simulation et/ou le débogage et sont uniquement visibles depuis l'interface de l'agent (Fig 5.3).

process	processus visualworks
mas	réference au SMA hebergeant l'agent
processState	l'état d'activité du moteur de l'agent
id	entier referancant de facon unique l'agent dans le SMA
agentID	ValueHolder sur une String 'class name #id'
absoluteBirthDate	Date absolue (en sec.) de creation de l'agent
birthDate	Date au format lisible de la creation de l'agent
birthTime	Heure au format lisible de la creation de l'agent
absolutedeathDate	Date absolue (en sec.) de la mort de l'agent
deathDate	Date au format lisible de la mort de l'agent
deathTime	Heure au format lisible de la mort de l'agent
absoluteWakeDate	Date absolue (en sec.) du dernier reveil de l'agent
lifeTime	Temps en secondes de la durée de vie de l'agent
absoluteSleepDate	Date absolue (en sec.) de mise en sommeil de l'agent
elapsedAsAwake	Temps en secondes de la durée d'activté de l'agent
debugLevel	Niveau de débogage de l'agent
lastLoopEnteringAbsoluteTime	Date absolue (en sec.) du dernier cycle d'activité de l'agent

- Variables d'instance -

Variables d'instance de classe

Cette variable n'est utilisée que pour la simulation dans la mesure où elle contient la liste des types d'agents que l'on peut créer dans le système depuis l'interface de contrôle du SMA.

ValidAgentsTypes	Liste contenant les agents valides dans un type de SMA

Variables de classe

Ces variables servent à maintenir des statistiques sur la population d'agents de tous les SMA créés depuis l'initialisation de la plateforme. Elles sont visibles depuis l'interface de contrôle de la plateforme (Fig 5.2).

Progeny	Ensemble des agents créés
CreatedAgents	Nombre d'agents créés
FrozenAgents	Nombre d'agents stoppés
KilledAgents	Nombre d'agents tués
MaximumPopulation	Nombre maximum d'agents en activité
RunningAgents	Nombre d'agents actifs

Méthodes de contrôle

Ce sont les méthodes que l'on doit utiliser pour contrôler un agent depuis l'environnement (depuis un autre agent par exemple).

kill	Terminaison du moteur de l'agent
resume	Reprise de l'activité du moteur de l'agent
suspend	Arrêt de l'activité du moteur de l'agent

5.5 Contrôles et interfaces

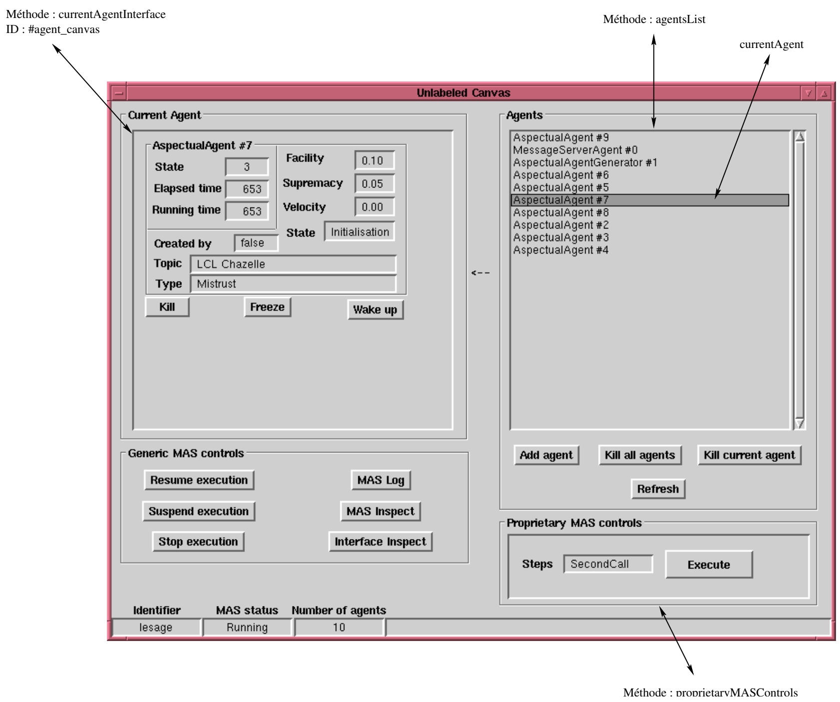

FIG. 5.1: *Interface de SMA*

Nous fournissons un ensemble d'interfaces destinées à permettre l'utilisation de la platerforme à des fins de débogage et de simulation. Nous y trouvons de quoi contrôler les agents, les SMA et plus généralement, toute la plateforme. Ces interfaces sont minimales et augmentables en fonction des besoins du développeur.

La figure 5.1 est l'interface d'un SMA aspectuel, on y retrouve la liste des agents le composant, ainsi que l'interface de l'agent courant. Tous les contrôles propres au SMA sont accessibles, en résumé, on peut, grâce à cette interface, contrôler le fonctionnement du SMA, ce qui est très utile pour la simulation et/ou le débogage.

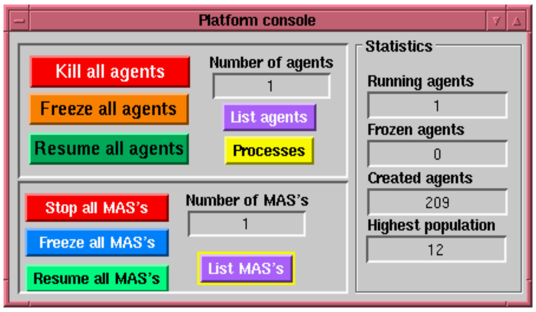

FIG. 5.2: *Console de la plateforme*

La figure 5.2 représente la console qui permet de contrôler tous les SMA et les agents. En fait, elle centralise toutes les commandes que l'on retrouve dans les interfaces de SMA. De plus, elle permet de consulter la liste de tous les agents présents et fournit des statistiques générales sur la charge du système.

Nous voyons dans la figure 5.3 l'interface typique d'un agent. Celle-çi permet de le contrôler et/ou de l'examiner en détail. Nous fournissons une interface

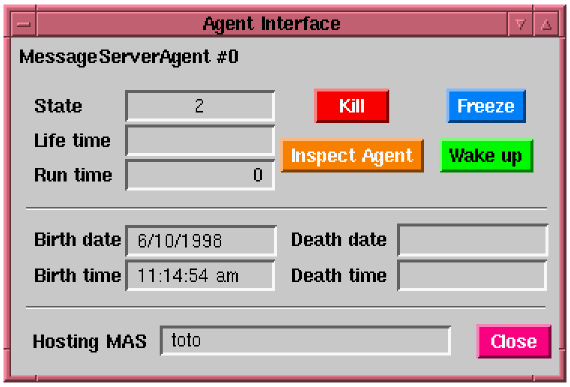

FIG. 5.3: *Interface d'un agent*

minimale, mais on peut bien sûr l'augmenter à volonté et y rajouter tout type d'outils.

Chapitre 6

Représentation du sens des mesages et de l'intentionnalité des acteurs

Nous présentons, dans ce chapitre, la façon dont le système représente la situation de communication dans une gestion de crise et la manière dont il exploite cette représentation. La représentation d'une situation comprend l'expression du sens perçu par les acteurs ainsi que leurs intentions et leurs motivations. Une partie de la représentation est générée par les acteurs eux-mêmes. Mais cette représentation n'est pas suffisante, le système l'augmente donc automatiquement de façon signifiante afin de palier ses incomplétudes.

Le point de vue d'un acteur sur une situation évolue en fonction du temps, des évènements et des communications qu'il peut avoir avec les autres acteurs impliqués dans la gestion de la crise. Il exprime son point de vue, ses intentions et ses motivations dans ses communications par des actes de langage [Austin, 1970] [Searle, 1969]. Nous devons donc utiliser une représentation fortement dynamique basée sur les actes de langage afin de pouvoir suivre les évolutions de la situation. Nous avons choisi une représentation par agent. Ces agents, reifiant les actes de langage, représentent différents aspects de la situation, nous les avons donc nommés Agents Aspectuels.

6.1 Description augmentée d'une situation

La description de la situation courante gérée par le SIC dépend de la personne qui la décrit. Selon son point de vue, un aspect de la situation sera perçu avec acuité alors qu'un autre sera négligé. Une autre personne décrivant la même situation pourra attribuer une importance totalement différente à ces deux aspects. Il faut donc une forme de description qui puisse permettre une description complète d'une situation selon tous les points de vue possibles. Nous avons introduit pour cela la notion de traits sémantiques.

Les traits sémantiques sont une représentation réduite et signifiante du discours. L'ensemble des traits semantiques émis lors d'une communication est appelé *donnée communicationnelle* [Cardon and Durand, 1997]. Ces données communicationnelles sont interprétées localement par le système de chaque émetteur de messages et par les systèmes locaux des destinataires. Ainsi elles donnent une image propre à chaque acteur de la façon dont il se représente la situation, des points qu'il estime importants, des relations qu'il entretient avec les autres acteurs, de la manière dont il les perçoit et de la façon dont cette connaissance évolue dans le temps.

Les données communicationnelles, et donc les traits sémantiques, sont les briques de base de notre système. Ils représentent la base de l'interprétation du système : une description objective et subjective de la situation.

6.1.1 Traits Sémantiques

Les traits sémantiques sont représentés par des triplets (qualification, sujet de la qualification, intensité de la qualification). Sous cette forme on peut décrire un grand éventail de situations.

Par exemple, pour décrire un commandant des pompiers, nommé Pierre Dupont, se trouvant au PC avancé, intervenant sur un incendie assez important d'un lycée type Pailleron contenant environ 600 élèves et doutant de la resistance du bâtiment, on génère les traits suivants :

1. (Identité, personne #1, Pierre Dupont)
2. (Fonction, personne #1, Commandant Pompier)
3. (Localisation spatiale, personne #1, PC avancé)
4. (Intervenants, phénomène #1, personne #1)
5. (Type, phénomène #1, incendie)
6. (Echelle, phénomène #1, assez important)
7. (Implication, phénomène #1, Bâtiment #1)
8. (Type, bâtiment #1, Lycée)
9. (Architecture, bâtiment #1, Pailleron)
10. (Habitants, bâtiment #1, 600)
11. (Précision, bâtiment #1 habitants, approximative)
12. (Doute, bâtiment resistance, Fort)

Ensuite, on ajoute à ces traits sémantiques l'identité de l'émetteur, la date et l'heure. On peut noter des traits de différents types : des traits descriptifs dans lesquels les valeurs sont spécifiques à la qualification (ex. localisation spatiale, type, identité, architecture, ...) et les traits expressifs dans lesquels la valeur est d'un genre moins bien défini et ressemble plus à une intensité (précision, doute, échelle,...). Pour les traits expressifs, l'intensité de la qualification est une valeur représentant une appréciation plus ou moins subjective de cette qualification.

Les traits sémantiques peuvent donc être de plusieurs types :

1. les traits descriptifs (hors traits de localisation spatiale)
2. les traits d'intensité
3. les traits de jugements
4. les traits de localisation temporelle (date, heure, durée, segment temporel, ...)
5. les traits de localisation spatiale (liés à des SIG)
6. les traits de potentialité (pouvant être généré par le système)

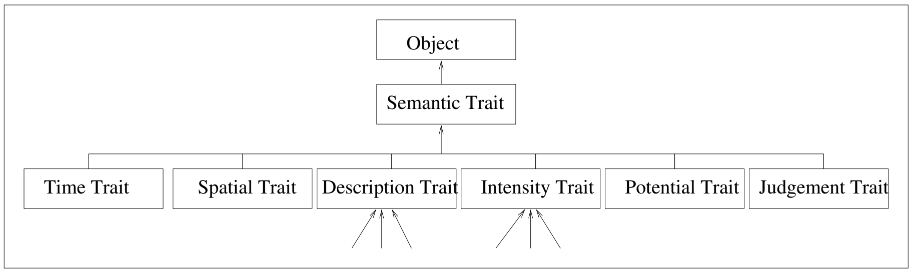

FIG. 6.1: *Graphe d'héritage des classes de traits sémantique*

Ces différents traits ayant chacun leur spécificité, ils sont représentés chacun par une classe à part entière dans le système, sous-classe de la classe générique "TraitSemantique".

6.1.2 Augmentation automatique

Le discours représenté par des traits sémantiques est une réduction d'un discours usuel. Tout n'est pas facilement exprimable, en particulier, tout ce qui s'exprime autrement que par les mots, par exemple : des sentiments ou des sous-entendus. Le discours est empli de non-dits et de sous-entendus et constitue ce que Searle appelle l'arrière-plan [Searle, 1992]. Le rôle de l'augmentation automatique des messages est d'exprimer explicitement une partie de cet arrière-plan. Par exemple, lors de la collision entre un véhicule et un train au niveau d'un passage à niveau, il est raisonnable de penser que le train a pu dérailler et qu'il y a des blessés. De même, dans un avenir proche, les trafics ferroviaire et routier risquent d'être fortement perturbés dans la zone de l'accident, en particulier si cet accident se situe sur une route très fréquentée et à une heure de pointe.

Par exemple, la notion de routes à traffic important selon des horaires ou des dates spécifiques, doit être présente dans le système, sinon celui-ci ne pourra pas évaluer correctement l'importance des conséquences de l'accident.

6.2 L'augmentation des messages

L'augmentation automatique des messages se fait grâce à un SMA réactif. Chaque Agent d'Augmentation représente un aspect de la situation (localisation temporelle, localisation spatiale, le trafic, la météorologie, les véhicules, les personnes, les bâtiments, les routes...). Ces agents regroupent toutes les informations dépendant typiquement de leur domaine. Ils les complètent en communiquant avec les autres agents.

Les Agents d'Augmentation communiquent entre eux grâce à un réseau d'accointance. Ce réseau représente le rapport que peuvent avoir les domaines entre eux. En effet, tout fait, objet ou sujet est localisé temporellement et spatialement. Donc tous les agents d'augmentation possèdent, dans leur réseau de communication, l'adresse des agents de localisation spatiale et de localisation temporelle. Ensuite, chaque agent possède l'adresse des agents susceptibles de compléter l'information qu'il détient. Par exemple, l'agent représentant les véhicules, outre la localisation spatiale et temporelle, possède l'adresse de l'agent représentant les personnes pour pouvoir donner des informations sur le conducteur, les passagers et le propriétaire du véhicule. Réciproquement l'agent chargé de représenter les personnes possède l'adresse de l'agent Véhicule afin de connaitre les véhicules conduits ou détenus par une personne.

Chaque agent est en plus chargé de faire une synthèse de l'état global du système selon son propre point de vue. En effet, tous les agents cherchent à déterminer dans quelle mesure les informations de leur domaine ont une influence sur l'état global de la situation. On peut, par exemple, représenter de cette manière, deux véhicules dangereusement proches, un délai trop court pour parcourir une distance ou une météo défavorable pour une certaine période. Chaque agent qui detecte une anomalie la communique aux agents concernés de son réseau.

Les agents analysent le nombre et le type des anomalies détectées et reçues et envoient un message d'alerte spécifique à leur domaine sous la forme d'une donnée communicationelle. Si de nombreux Agents d'Augmentation envoient des signaux d'alerte, alors l'ensemble des agents concernés envoie un signal d'alerte général.

Le comportement de ces agents est modélisé par un ATN (Augmented Transition Network) [Guessoum and Dojat, 1996].

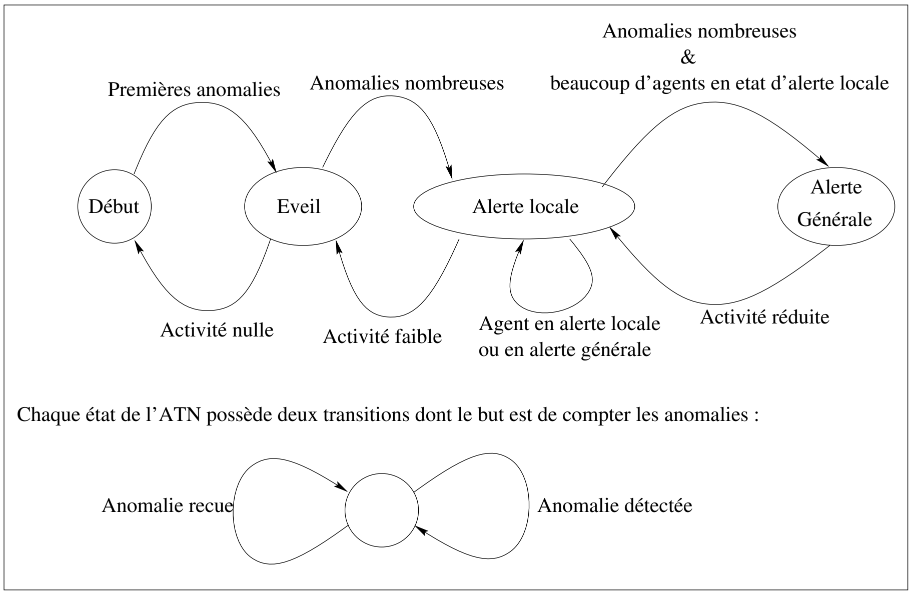

FIG. 6.2: *ATN des Agents d'Augmentation*

6.2.1 Les Agents d'Augmentation

L'Agent Calendar
Cet Agent est chargé de conserver tous les événement temporels survenus dans le système, tous ceux qui y ont été préalablement entrés (dates de travaux, manifestations,...) et tous ceux qui sont habituels ou recurents. Cet agent possède une représentation graphique sous la forme d'un diagrame de Gantt.

Agent Trafic
Cet agent représente deux choses : l'état courant du trafic et l'état du trafic habituel selon la date, le lieu et la météo en relation avec les agents adéquats (Weather, Calendar, Geography, Road).

Agent Géography
Cet agent sera directement connecté à un SIG, auquel il pourra faire des requêtes. Il est chargé de collecter les informations géographiques des objets présents dens le système.

Agent Weather
Connectée aux données Météo-France, cet agent fournit des informations météorologiques passées, courantes et prévues selon le lieu et la date (Calendar,

Geography).

Agent Vehicle
Cet agent collecte les informations concernant les véhicules : immatriculation, type, date d'entrée en service, couleur, propiétaire, ...

Agent People
Cet agent collecte les informations concernant les personnes : identité, état de santé, lieu d'habitation, lieu de travail,...

Agent Building
Cet agent collecte les informations concernant les bâtiments : nom, adresse, etages,...

Agent Road
Cet agent collecte les informations sur les routes : Nom, état général, tronçons,...

6.2.2 Schéma des relations entre les Agents d'Augmentation

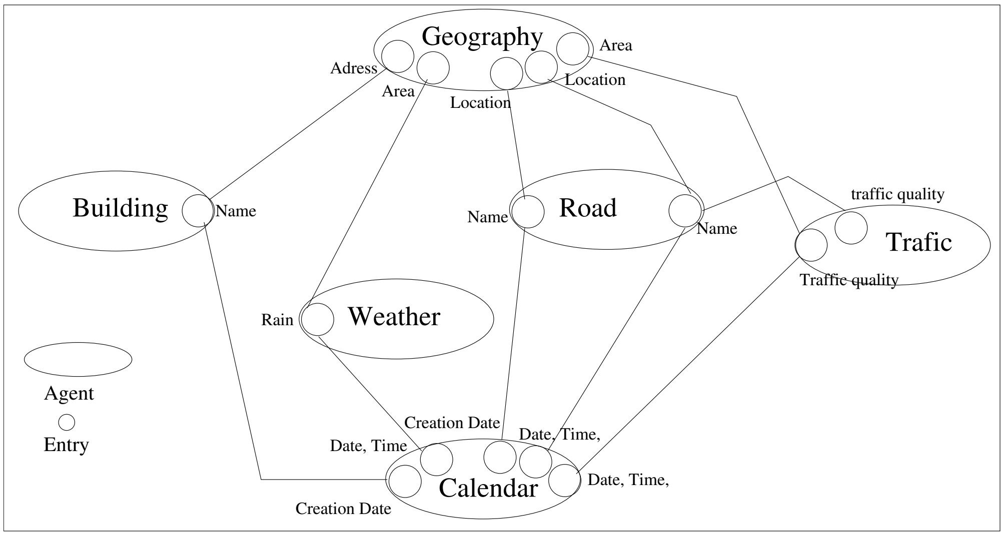

FIG. 6.3: *Relation entre les Agents d'Augmentation*

La figure 6.3 ci-dessus montre les relations entre les Agents d'Augmentation. Tous les objets du système sont décomposés en attributs représentés par les agents. Les informations concernant les objets sont donc répartis dans tout le système. Chaque agent garde une entrée pour cet objet et la relie à l'entrée correspondante chez les agents de son réseau de communication. Pour rassembler toutes les informations concernant un objet, il suffit de suivre le réseau avec l'entrée correspondante. Pour établir cette classification nous nous sommes, entre autre, basés sur la Méthode de Raisonement Tactique (MRT) [Malavieille, 1998] en vigueur chez les pompiers.

6.3 Le Système Multi-Agent Aspectuel

Le SMA aspectuel interpréte les traits sémantiques contenus dans les données communicationnelles en les réifiants sous forme d'agents. Les traits sémantiques représentent des aspects de la situation et les agents qui les réifient sont donc nommés Agents Aspectuels. Ces agents sont chargés de représenter l'évolution des traits sémantiques dans la perception de la situation que décrit l'acteur dans les messages envoyés et reçus.

La génération des agents aspectuels dans le SMA se fait grâce à un agent présent nativement dans le SMA. Cet agent, appelé Agent Générateur, analyse les traits sémantiques reçus par le SMA et génère les agents aspectuels correspondants s'ils ne sont pas déjà présents. Les traits sémantiques sont reçus par le SMA et sont tous transmis à un agent faisant office de messagerie, l'agent Serveur de Messages. Tous les autres agents le consultent afin de récuperer les traits sémantiques reçus.

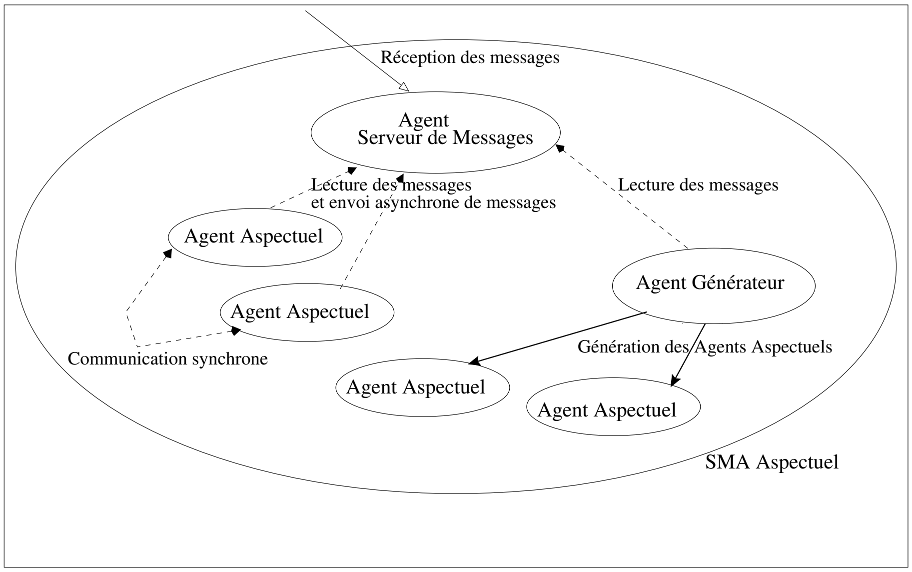

FIG. 6.4: *Les agents dans le SMA Aspectuel*

6.3.1 Le rôle du Système Multi-Agents Aspectuel

Le SMA aspectuel a pour rôle de représenter les aspects exprimés par les acteurs de la situation perçue. Cependant, ces aspects peuvent en infirmer certains et en confirmer d'autres. Donc chaque aspect n'a pas de sens en soi, son sens lui est donné par son immersion dans l'ensemble des aspects déjà exprimés.

Chaque agent du SMA, reifiant donc un aspect particulier de la situation exprimé par un acte de langage sous forme d'un trait sémantique, entre en relation avec les autres dés sa création. En effet, sa seule existence peut menacer l'existence de certains agents et consolider la prégnance d'autres dans le SMA. L'agent nouvellement créé s'insère donc dans une société plus ou moins hostile dans laquelle il devra se développer ou mourir.

Les règles de cette société comprennent des règles d'alliance, d'entre-aide et d'affrontement. Ces règles ont pour but de dynamiser la représentation des aspects. En effet, les aspects évoluent en fonction du temps, des communications et des autres aspects exprimés dans les messages. Ces règles permettent de modéliser les affrontements entre deux aspects contradictoires, l'entre-aide entre aspects alliés et les alliances entre aspects sémantiquement proches.

Il existe une relation très étroite entre le développement d'un agent dans le SMA et l'importance de l'aspect réprésenté par l'agent dans la situation perçue par les acteurs. Donc, plus un agent est avancé dans son développement, plus il est puissant et moins sujet aux attaques. Il devient nécessaire à ses ennemis de s'allier afin de pouvoir le contrer ou le ralentir.

6.3.2 L'agent Serveur de messages

Cet agent est purement réactif. Il se contente de collecter les traits sémantiques que le SMA de l'acteur local reçoit et les messages que peuvent éventuellement s'échanger les agents du SMA pour les mettre à disposition des autres agents.

La consultation se fait par l'appel de la méthode messageAt: de l'agent serveur de messages. Cette méthode prend en argument un nombre correspond au numéro d'ordre du dernier message lu par l'agent et renvoie le message suivant s'il existe, nil sinon.

6.3.3 L'agent Générateur d'agents aspectuels

Le rôle de cet agent générateur est de créer les agents aspectuels en fonction des messages reçus. Ainsi il scrute en permanence l'agent serveur de messages, récupère les nouveaux messages et crée les agents aspectuels correspondants si nécessaire. Il s'agit également d'un agent réactif.

La création d'agents dépend fortement du type de trait sémantique d'origine. S'il s'agit d'un trait sémantique de jugement, plusieurs agents aspectuels peuvent être générés. En effet, les qualifications de jugements (doute, certitude, peur, confiance, ...) ne sont pas totalement indépendantes. Le doute et la méfiance, par exemple, sont proches l'un de l'autre de même que chacun d'eux s'oppose respectivement à la certitude et à la confiance. Des relations de proximité existent aussi pour ces qualifications. Donc, lorsqu'un agent est généré, les agents représentant les qualifications proches de la qualification d'origine sont générés si besoin est. Cette notion de proximité sémantique est spécifique aux agents de jugements. La proximité sémantique des autres types d'agents s'effectue sur le sujet de la qualification.

Pour générer un agent aspectuel à partir d'un trait sémantique, on récupère le type du trait sémantique. S'il ne s'agit pas d'un trait sémantique de jugement, on verifie l'absence de l'agent aspectuel que l'on veut créer et s'il n'existe pas dans le SMA, on le génère.

Par contre s'il s'agit d'un trait de jugements, il ne suffit pas de créer le seul agent correspondant, mais aussi tous les agents dont la qualification est suffisament proche de celle du trait sémantique d'origine. C'est le rôle de la méthode agentGeneratingAtempt: aSemanticTrait.

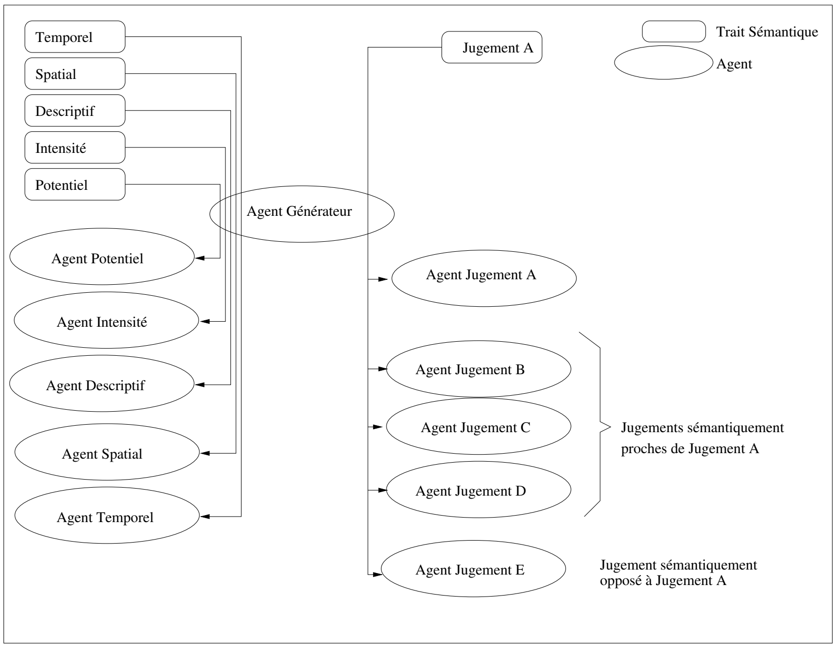

FIG. 6.5: *Génération des Agents Aspectuels à partir des Traits Sémantiques*

Les agents sont effectivement générés par la methode generateAgent: qui prend comme argument un trait sémantique. Cette méthode génère un agent du même type que les traits sémantiques. La génération des agents de jugements est un peu plus complexe, puisque pour chaque agent créé on doit générer son opposé, afin de refléter la multiplicité des états d'esprits possibles d'un acteur.

6.4 L'Agent Aspectuel

L'agent aspectuel est le premier maillon de la chaîne d'interpétation du système. Il est chargé de représenter un aspect ponctuel de la situation à un moment donné par un acteur. C'est par exemple une tendance ou un état d'esprit d'un acteur ou bien un fait. Comme chaque aspect de la situation perçue est exprimé sous forme d'agent, le developpement de l'agent reflète l'importance de cet aspect selon le point de vue des acteurs.

Chaque agent aspectuel, déjà présent dans le SMA, peut interpréter les traits sémantiques reçus par le SMA, afin d'évoluer. En effet, si un agent aspectuel représentant un incendie reçoit un trait sémantique à propos d'une explosion, cela le confortera. Certes, ce trait le confortera moins que s'il s'agissait d'un trait à propos d'un feu. L'interprétation des traits sémantiques par les agents aspectuels se fait grâce à une certaine proximité sémantique entre les aspects.

Cette proximité sémantique est représentée, dans le système, par une matrice de proximité dont les éléments sont numériques.

6.4.1 Structure générale

Un agent aspectuel possède un centre nerveux réflexe et un système d'analyse de la situation et de prise de décision. Le centre réflexe est modélisé par un automate à état dont chacun des états est un ATN. Cet automate est donc un automate d'automates et nous l'avons appelé macro-automate. Le comportement réflexe de l'agent est dirigé par les ATN. Il dépend donc de l'état dans lequel se trouve le macro-automate. Chacun des états de l'agent correspond à une phase de développement de l'agent. Ces quatre étapes vont de l'état d'éveil à la victoire de cet agent sur ses ennemis. Lorsqu'un agent atteint ce dernier état cela signifie que l'aspect qu'il représente est un aspect significatif de la représentation de la situation selon les messages échangés. Le centre d'analyse et de prise de décision est pour l'instant un petit système à règles, mais peut être remplacé par un système à base de connaissances plus complexe.

L'activité d'un agent aspectuel se mesure à l'aune de trois variables : la vélocité, la facilité et la suprémacie. La vélocité représente la rapidité de dévelopement de l'agent. Plus il est actif, plus il est véloce. La facilité représente, comme son nom l'indique, la facilité avec laquelle l'agent s'est développé jusque-là. Si l'agent a rencontré des obstacles lors de son développement, la facilité sera faible. Enfin la suprémacie représente l'importance de l'agent dans l'ensemble du SMA. Plus cet agent est prégnant dans le SMA plus sa suprémacie est importante.

La classe AspectualAgent possède plusieurs sous-classes :

– JudgementAgent

– PotentialAgent

– SpaceAgent

– TimeAgent

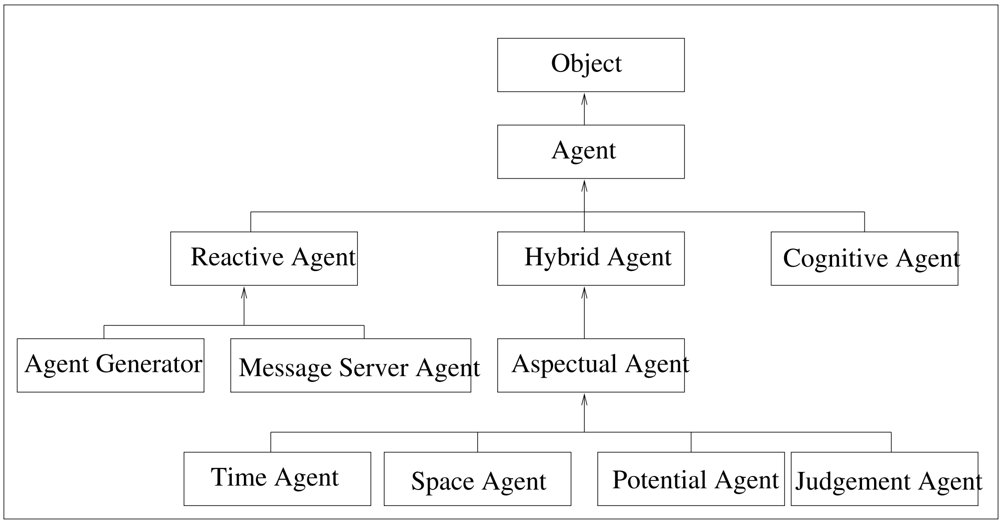

FIG. 6.6: *Graphe d'héritage des classes d'agents aspectuels*

Chacune de ses sous-classes possède un comportement spécifique et n'interfère pas avec les autre classes. Les JudgementAgents chercheront à se regrouper pour mieux résister aux attaques des autres. Les PotentialAgents se comporteront comme des agents aspectuels tant que les potentialités qu'ils représentent ne seront pas confirmées ou infirmées dans un message. Dans un tel cas, soit ils muteront en agents aspectuels, soit ils régresseront, ceci pouvant aboutir à la disparition de l'agent de potentialité. Les SpatialAgents scruteront toutes les données spatiales fournies dans les messages et tenteront de regrouper les faits ou des objets introduits dans l'ensemble des messages selon leur proximité spatiale en tenant compte des évolutions dans le temps des positions. Les TimeAgent effectueront le même travail pour les données temporelles, en tenant compte des problèmes de modifications de dates ou de durées.

6.4.2 Le macro-automate

La structure du macro-automate est basée sur le principe de la décision linéaire de Sfez [Sfez, 1992]. Les quatre états de cet automate correspondent aux quatre étapes de la décision linéaire, c'est-à-dire : initialisation, délibération, décision et action. L'état d'initialisation correspond à l'apparition d'un désir. L'état de délibération correspond à la réflexion en vue de la manière de satisfaire ce désir. L'état de décision correspond à la prise de décision de satisfaire ce désir. Dans l'état d'action, on agit afin de satisfaire ce désir.

On peut ainsi modéliser certains comportements typiques:

- Bestialité (Initialisation, Action)
- Intellectualisme (Initialisation, Délibération)

Chacun des états du macro-automate est un ATN qui pilote les actions réflexes de l'agent.

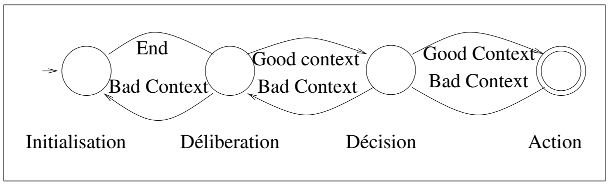

FIG. 6.7: *Macro-automate*

Lorsqu'un Agent Aspectuel est généré, son macro-automate est positionné dans l'état d'Initialisation. Les étiquettes des transitions du macro-automate sont les états finaux des ATN des états du macro-automate. Lorsqu'une transition du macro-automate est franchie, l'ATN de l'état d'arrivée est positionné dans l'état Start.

6.4.3 Les ATN

A chacun des ATN correpond un comportement particulier. Il existe quatre ATN par macro-automate et, à macro-état semblable, les ATN de deux agents de types distincts sont différents.

Il existe deux types de transitions dans ces ATN, les transitions chiffrées et les transitions nommées. Les transitions chiffrées sont des transitions correspondant à des stimuli d'une certaine intensité. La transition chiffrée dont le seuil est le plus grand nombre inférieur au stimulus sera franchie. Les transitions nommées ne sont franchies que lorsque la chaîne de caractères correspondant au nom est reçue par l'ATN.

Pour des raisons de simplicité, nous n'allons présenter que les quatre ATN des agents de jugements.

Initialisation

Cet ATN est le comportement d'éveil de l'agent, où l'agent attend d'être suffisamment stimulé pour se risquer dans l'arène. Pour l'éveiller, deux possibilités existent : une stimulation très intense ou plusieurs stimulations de faible intensité.

Chaque transition franchie renforce la vélocité et la facilité de l'agent proportionellement au seuil de la transition franchie.

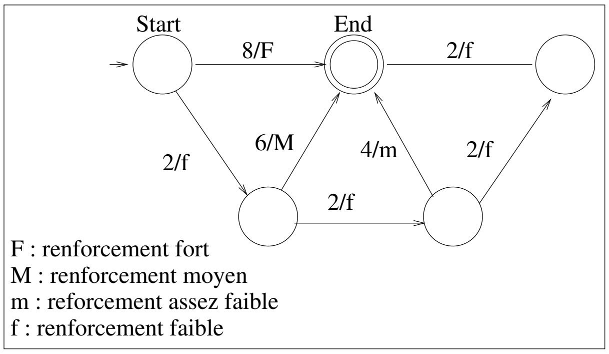

FIG. 6.8: *ATN d'initialisation*

Délibération

Cet ATN reste encore relativement simple : Il ne s'agit que d'analyser le contexte de l'agent afin de determiner si l'agent est viable. L'ATN ne fait que recevoir les stimuli. C'est l'agent qui décidera de franchir les transitions indiquant la nature du contexte.

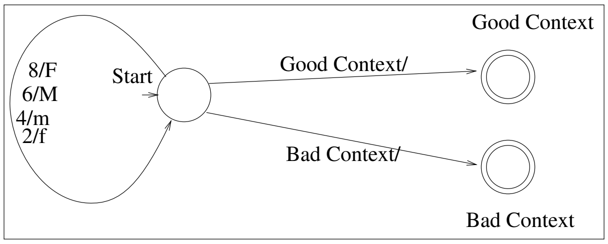

FIG. 6.9: *ATN de délibération*

Décision

L'ATN de décision est le plus complexe car c'est à partir des actions que l'agent va effectuer dans cet état que va se jouer sa victoire ou son échec dans son développement. Les trois états principaux de cet ATN reflétent les trois états d'un comportement belliqueux : l'attente, la défense, l'agression. Dans l'état d'attente, l'agent analyse plus son contexte et est très sensible aux stimuli. Dans l'état de défense, l'agent est moins sensible aux attaques, mais les stimuli les plus faibles ne sont pas pris en compte et il ne décide pas d'actions de combat excepté les actions d'aide. Dans l'état d'agression, l'agent effectue principalement des actions de combat envers les autres agents, ses défenses sont plus faibles et les stimuli les plus faibles ne sont pas pris en comptes.

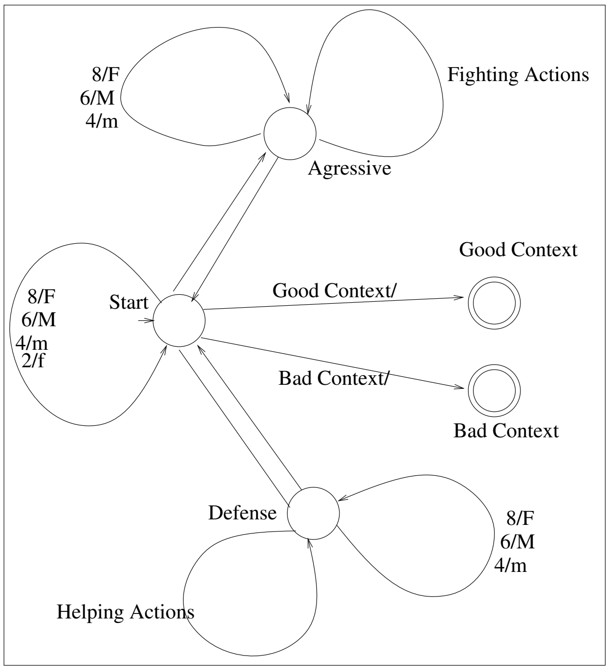

FIG. 6.10: *ATN de décision*

Action

Un agent qui a atteint l'état d'action n'effectue que peu d'actions reflexes, il intercepte les stimuli et attend les décisions du système d'analyse. L'ATN est donc simple et semblable à celui de délibération.

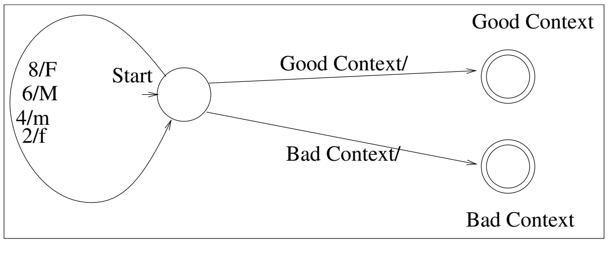

FIG. 6.11: *ATN d'Action semblable à l'ATN de délibération*

6.4.4 Le processus d'action

Le processus d'action est simple. L'agent analyse sa situation selon un calcul de valeur de contexte, comprenant sa force, celle de ses alliés et celle de ses ennemis, et il choisit éventuellement une action à effectuer. Il existe deux types d'actions : les actions sur soi et les actions dites de combat sur les autres agents.

Les actions personnelles de l'agent sont de trois types : le suicide, le franchissement d'une transition d'ATN et le franchissement d'une transition du macro-Automate.

L'action de suicide ne peut être prise que dans l'état de délibération du macro automate, l'état d'initialisation étant un état dans lequel aucune décision ne peut être prise et les états de décision et d'action sont des états soit proches de l'aboutissement, soit l'aboutissement lui-même. Dans ces deux derniers états, l'agent se bat, on peut le tuer, mais il ne peut pas décider de se suicider.

L'action de faire franchir une transition de l'ATN peut être prise dans n'importe lesquels des trois états de délibération, décision ou action. Cette action traduit une adaptation à l'analyse d'un changement dans la situation de l'agent.

L'action de faire franchir une transition du macro-automate peut être prise dans les deux derniers états décision et action. Cette action traduit le besoin d'une adaptation rapide à un important changement dans la situation de l'agent.

Les actions de combat sont plus nombreuses. Il en existe deux types, les actions d'aides et les actions néfastes. Les actions d'aides sont automatiques et consistent à aider des agents alliés qui sont moins développés. Les aides consistent à envoyer un stimulus qui peut les relancer ou à les doper en agissant directement sur leur ATN.

Les actions néfastes, par contre, ne sont pas automatiques. En effet, l'agent qui les émet doit réaliser l'envoi avec une intensité qui correspond à sa force et au nombre de ses alliés et de leur force. L'agent qui reçoit l'action néfaste compare cette force à sa propre force, calculée de la même manière que la force de l'agent émetteur. Plusieurs cas de figure peuvent se produire : le récepteur subit l'action, le récepteur subit l'action et tente de répondre, le récepteur subit une action atténuée, le récepteur subit une action atténuée et tente de répondre, le récepteur ne subit pas l'action et le récepteur ne subit pas l'action et tente de répondre.

Les seuils de réponse varient selon les types d'agent et les agents eux-mêmes. Certains agents sont très "susceptibles", d'autres amorphes. En règle générale, les agents répondent à une action par une action du même type. Mais plus l'action est néfaste, plus la réponse risque d'être violente. De plus, si l'agent subit l'action et répond, la force de la réponse est augmentée de façon d'autant plus importante que l'action est néfaste.

Les actions néfastes sont : tuer un agent, l'affaiblir, l'isoler, le paralyser ou le mutiler en agissant sur son macro-automate.

L'ensemble des agents aspectuels ainsi défini permet de représenter dynamiquement l'ensemble des communications entre les acteurs, au niveau du sens des messages. Cet ensemble d'agents représente, c'est-à-dire offre une forme exploitable, les connaissances, avis, jugements, certitudes et incertitudes des acteurs décisionnels de la crise, et ceci dans son évolution même. Nous associons à la dynamique des actes de langage [Searle, 1969] entre acteurs la dynamique des organisations des agents aspectuels.

Chapitre 7

Morphologies d'un paysage d'Agents Aspectuels

L'ensemble des agents composant chaque SMA aspectuel exhibe les tendances qu'ont certains de ces agents à se réunir entre eux ou au contraire à se détruire : le caractère agrégatif ou désagrégatif peut s'exprimer de façon morphologique. Cependant, ce comportement émergent reste au niveau de chaque SMA aspectuel sans qu'on l'utilise de façon explicite, afin de le traiter comme connaissance nouvelle et en le plaçant à un autre niveau conceptuel dans le système. C'est le rôle de ce que nous appelons l'organisation des *Agents de morphologie* de réifier les caractères morphologiques des différents SMA aspectuels. Nous introduisons ici la notion de paysage d'agents et plus généralement de morphologie d'un SMA, ainsi que l'implémentation de ces agents pour un certain type de morphologie.

7.1 Définitions

Soit E un ensemble contenant N agents. On définit une caractéristique commune à certains agents comme une mesure numérique μ comprise entre α et β.

$$\begin{aligned} \mu : \quad & \mathrm{E} \to R \\ & A_i \to \mu(A_i), \quad \alpha \leqslant \mu_k(A_i) \leqslant \beta \end{aligned}$$

$\mu_k(A_i)$ est la caractéristique k de l'agent aspectuel A_i. On considère un ensemble de telles mesures $\nu = \{\mu_k(A_i)\}$

Paysage d'agents

Un paysage d'agents est par définition une lecture géométrique de l'organisation de l'ensemble d'agents prenant en compte de façon indissociable les caractères suivants :

1. le graphe de couverture entre les agents et sa modification,

2. les états des agents et leurs modifications,

Un corollaire immédiat est le plongement des agents aspectuels dans un espace métrique grâce à leur transformation en vecteurs.

Agent de morphologie

Un agent de morphologie est la réification d'un caractère organisationnel émergent dans un paysage d'agents aspectuels. Il possède trois caractéristiques :

1. un objet : (variable focus) c'est l'aspect du discours des acteurs autour duquel le caractère organisationnel s'articule,
2. une force : (variable strength) c'est l'importance intrinsèque du caractère organisationnel,
3. une portée : (variable range) c'est une mesure de l'étendue de l'influence du caractère organisationnel

Imaginons par exemple les agents aspectuels suivants:

Agent	Sujet	Vélocité	Suprémacie	Facilité
AA1	Feu	0.2	0.8	0.3
AA2	Explosion	0.22	0.93	0.6
AA3	Dégats	0.8	0.5	0.9
AA4	Morts	0.3	0.4	0.5

AA1 et AA2 ont une vélocité de développement très proches, ce qui veut dire, dans le cas présent, qu'ils ont franchi des étapes de leur ATN assez peu rapidement. Ils vont donner naissance à un agent de morphologie qui réifiera le fait qu'ils se sont développés lentement jusqu'ici.

AA1 et AA2 par contre sont très prégnants vis-à-vis des autres agents (et selon leur point de vue), ce qui veut dire que les deux sujets auxquels ils font référence semblent être très importants pour l'acteur dont le SMA aspectuel rend compte. Aussi, un agent de morphologie de forte prégnance sera créé autour de ces sujets.

De la même manière AA3 et AA4 sont assez proches en terme de suprémacie, etc ... Nous aurons donc les agents de morphologie suivants:

Agent	Type	Agents agrégés	focus	force	range
Am1	Vélocité	AA1, AA2	0.21	2	0.07
Am2	Suprémacie	AA1, AA2	0.87	2	0.07
Am3	Suprémacie	AA3, AA4	0.45	2	0.07
Am4	Facilité	AA2, AA4	0.55	2	0.07

Chréode

Une chréode est le résultat de la réification de particularités géométriques émergentes autour d'un certain aspect entre des agents de morphologie.

7.1.1 Principes de la Morphologie Classique Unimodale

Cette morphologie d'un paysage d'agents est la premère que nous introduisons, et aussi la plus simple. Nous construisons un ensemble O d'agents de morphologie (Am) et pré-morphologie (Apm). Cet ensemble O est partionné entre le sous-ensemble P des Apm et le sous-ensemble M des Am. Il existe une bijection entre E, l'ensemble des agents aspectuels et l'ensemble P de telle sorte que tout agent de P est un représentant de $\mu_k(A_i)$ avec $A_i \in E$. Quand un nouvel agent A_i apparait dans E, son alter-ego Apm_i est créé dans P.

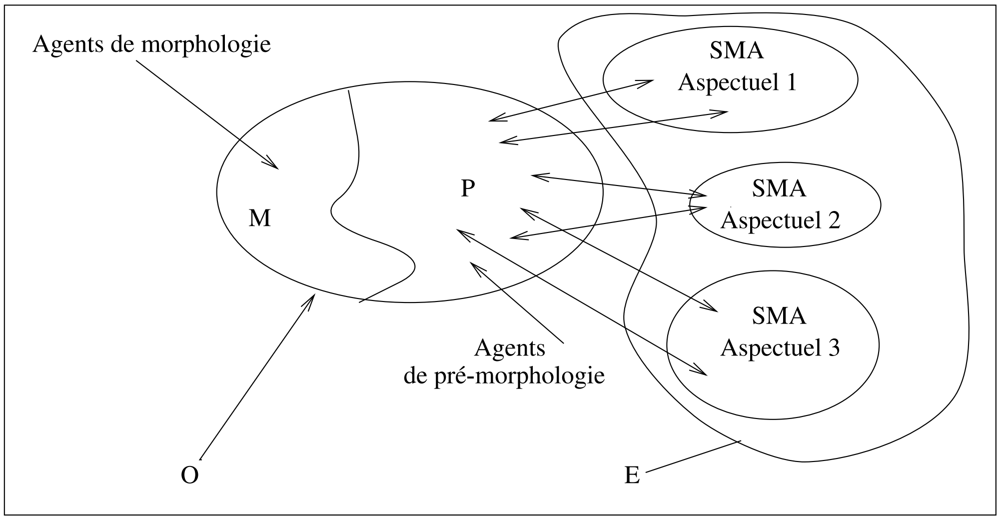

FIG. 7.1: *Les ensembles d'agents*

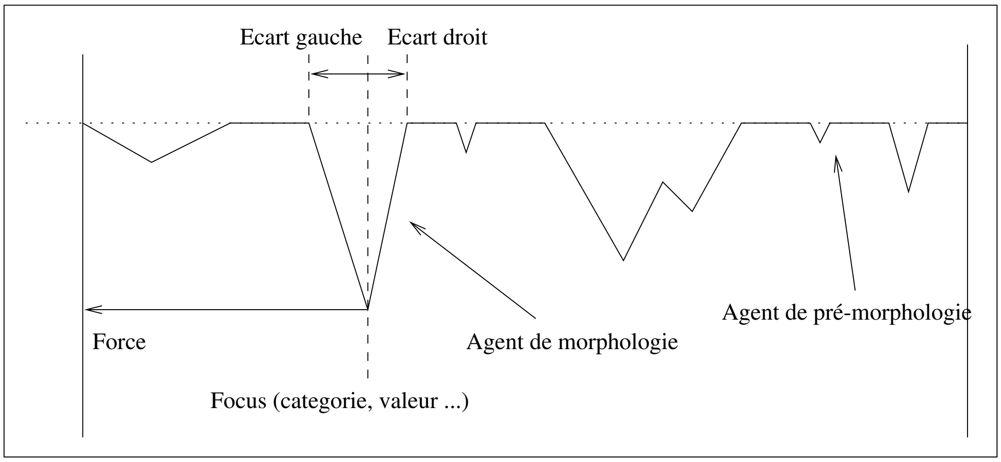

FIG. 7.2: *Morphologie classique*

Tout agent A_i de E possède au moins une mesure $\mu_k(A_i)$ qui correspond à chaque caractéristique que réifie cet agent. Nous organisons toutes ces caractéristiques dans un vecteur. Ce dernier sera la base qui servira aux Apm afin de pouvoir se positionner par rapport aux autres. En d'autres termes, nous considérerons (au niveau morphologique) qu'il y a équivalence entre le vecteur décrivant les caractéristiques d'un agent et cet agent. Nous utiliserons la notation suivante qui symbolise cette relation:

V_{A_i} **vecteur d'aspect de l'agent** A_i **:**

Soit $A_i \in$ E, $V_{A_i} = \{\mu_1(A_i), \mu_2(A_i), ..., \mu_n(A_i)\}$

Apm_i se trouve alors dans un univers, sous-espace de R^n régi par des lois qui donnent naissance à des puits de potentiel : tout Apm_i provoque une déformation de l'univers par le biais d'une force possédant un certain rayon d'action. Si Apm_i se trouve sous l'influence d'une telle force, alors il tombe, en quelque sorte, dans ce puits de potentiel. Il se passe alors l'une des deux choses suivantes :

1. le puits est causé par un Apm, les deux agents donnent naissance à un agent de morphologie Am_k,
2. le puits est causé par un Am, Apm_i renforce alors l'Am qui voit sa force et son rayon d'action augmenter.

Cependant, les Apm ne sont pas détruits pour autant lorsqu'il s'agrègent dans un agent de morphologie. Il faut se rappeler qu'il existe une bijection avec E et que celle-ci reste valide à tout instant. Ils continent à "vivre" au sein de leur Am hôte et cherchent au contraire à s'en détacher pour donner naissance à leur propre Am.

Il faut voir un Am comme une entité fondamentalement instable par nature: elle suit au plus près les changements morphologiques du SMA dont elle cherche à en réifier les caractères. L'instabilité est une notion intrinsèque à l'Am car il est constitué d'agents (les Apm) qui cherchent à remettre en cause leur dépendance vis-à-vis de celui-çi.

Sachant que les agents de morphologie sont la réification de l'agrégation de certains agents autour d'un aspect, il est possible de procéder à certaines opérations entre ces agents de morphologie. Le résultat de ces opérations s'appelle une *chréode*. On distingue au moins deux types de chréodes : celles d'agrégation et celles d'affrontement autour d'un aspect.

Chaque agent de morphologie peut à tout moment savoir sur quels aspects il a de influence : il lui suffit d'interroger ses agents de pré-morphologie qui eux-memes vont regarder les aspects des agents aspectuels qui leur sont connectés. Ainsi, en comparant les aspects qu'ils englobent, ils peuvent découvrir des aspects communs avec d'autres agents de morphologie. La forme émergente autour d'un aspect sera alors constituée par l'agent de morphologie le plus important et des autres agents de morphologies, le tout sous la forme d'une étoile (Fig 7.3).

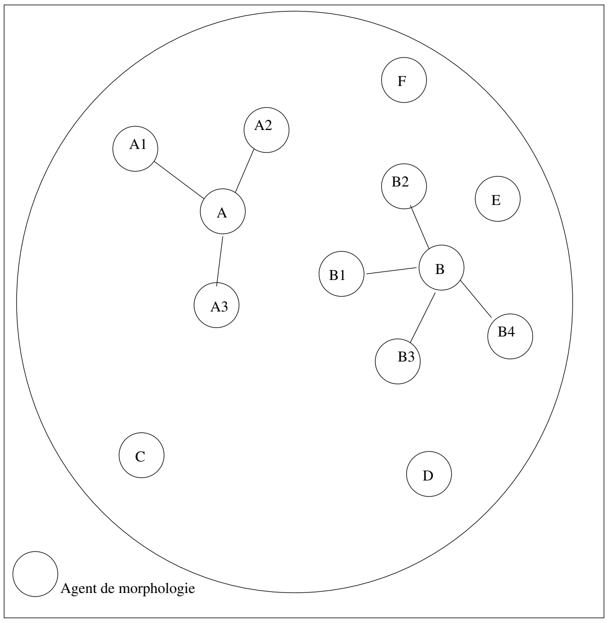

FIG. 7.3: *Un ensemble de chréodes*

7.2 Notion de rémanence morphologique

Jusqu'ici, dans le modèle d'agent de morphologie que nous avons décrit, nous faisons l'hypothèse que lorsqu'une caractéristique est suffisament éloignée d'une agrégation autour d'une valeur d'attraction, elle s'en détache aussitôt sans laisser de trace. En d'autres termes, l'agent de pré-morphologie correspondant coupe son lien avec l'agent de morphologie. Il pourrait être interessant de pouvoir laisser cette trace pendant un certain temps et avec une certaine intensité décroissante avec le temps. Pour y arriver, nous devons modifier les agents de morphologie afin que lorsqu'ils recoivent un message d'un agent de pré-morphologie les informant qu'il se détache, ils gardent en mémoire la présence de ce constituant morphologique maintenant parti.

Plus concretement, il s'agit de considerer que la déconnexion effective vue de l'extérieur de l'agent de morphologie sera retardée. L'agent de morphologie va donc, en quelque, sorte "tricher" sur le nombre d'agents de pré-morphologie composant l'attracteur qu'il réifie.

Dans un certain sens, cela peut constituer une sorte de mémoire à court terme sur les aspects morphologiques saillants du paysage d'agents. Le délai est paramétrable par l'agent ou par une demande extérieure. Il ne faut pas oublier que c'est le nombre d'agents de pré-morphologie qui constitue le seuil d'existence d'un agent de morphologie. Aussi, par le biais de la rémanence morphologique, c'est la vision globale du système qui est modifiée, cela pouvant mener à des interprétations complètement déformées par rapport à la réalité extérieure au système, et que celui-çi tente de représenter.

Chapitre 8

Implémentation de la morphologie classique

Nous avons présenté la notion de morphologie d'un SMA et les agents dont le rôle va être de réifier les caractères morphologiques d'un SMA. Il nous reste encore à étudier les détails de l'implémentation des agents de morphologie et quelles sont les techniques que nous avons utilisées pour y parvenir. Nous allons tout d'abord nous interesser aux agents de pré-morphologie qui sont les initiateurs des agents de morphologie, ce qui constituera le deuxième volet de ce chapitre.

8.1 Implémentation des Agents de Pré-Morphologie

Avant de présenter le code ou même les algorithmes qui vont animer les agents de pré-morphologie, nous devons concevoir un agent dont les rôles seront de :

1. réifier une caractéristique $\mu_k(A_i)$ d'un agent aspectuel A_i,
2. chercher un Am_j satisfaisant aux règles d'agrégation en fonction du focus et de la portée,
3. chercher un Apm satisfaisant au mêmes règles,
4. chercher à échaper à l'influence de l'Am dont il dépend.

Le comportement d'un Apm est donc clairement instable, en effet, ses buts 2 et 3 sont agrégateurs alors que son but 4 est déstructurant. C'est au prix d'un tel fonctionnement que nous pouvons suivre au plus près les évolutions émergentes du système d'agents aspectuels dont nous cherchons à réifier les caractères morphologiques.

8.1.1 Le moteur de l'agent de pré-morphologie

L'agent de pré-morphologie se comporte comme une bascule instable qui oscille entre deux états. S'il est stabilisé dans un agent de morphologie, il va alors chercher à remettre en cause cette stabilité. Dans le cas contraire, il va se mettre en quête d'un agent de morphologie ou bien d'un autre agent de pré-morphologie avec lequel il va tenter de donner naissance à un nouvel agent de morphologie.

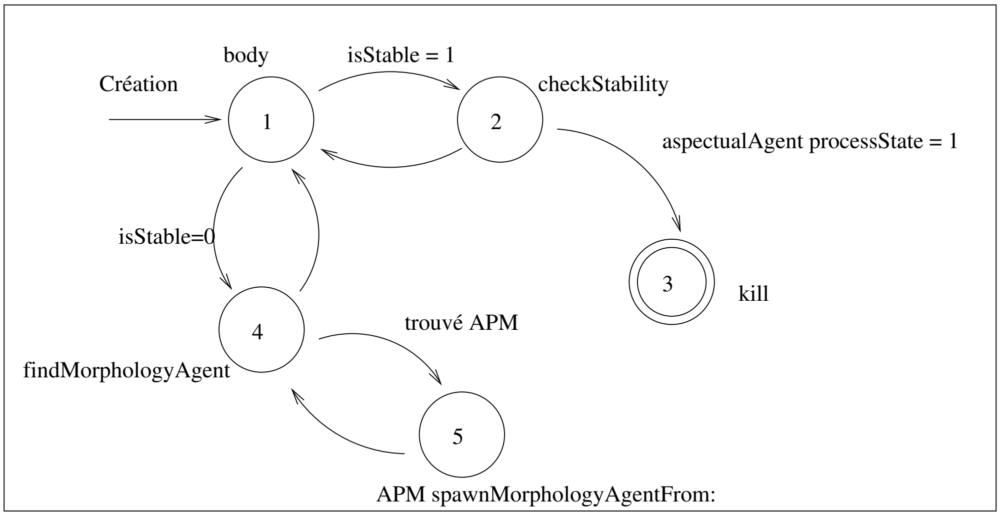

FIG. 8.1: *Principe de fonctionnement d'un Apm*

```
body
    | morph_agent |
    isStable = 1
        ifTrue: [self checkStability]
        ifFalse: [
            morph_agent := self findMorphologyAgent.
            morph_agent isNil
                ifFalse: [
                    morphologyAgent := morph_agent.
                    self isStable: 1]]
```

Activité principale d'un agent de pré-morphologie

8.1.2 Implémentation du rôle 4 - déstructuration

L'agent de pré-morphologie se suicide si son agent aspectuel est mort, emportant avec lui la caractéristique qu'il réifiait. Sinon, il regarde s'il tombe toujours sous l'influence de l'agent de morphologie auquel il est déjà rattaché. Si ce n'est pas le cas, alors il se déconnecte de cet agent, se marque comme instable et change alors d'état.

```
checkStability
    aspectualAgent processState = 1
        ifTrue: [
            morphologyAgent notNil ifTrue:
                [morphologyAgent disconnect: self].
            self aspectualAgent: nil.
            ^self].
    (morphologyAgent focus - focus)  abs <= morphologyAgent range
        ifFalse: [
            morphologyAgent disconnect: self.
            isStable := 0.
            morphologyAgent := nil]
```

Remise en cause de la stabilité

8.1.3 Implémentation des rôles 2 et 3 - agrégation

L'agent commence par regarder tous les agents du SMA selon l'ordre croissant des focus. S'il trouve un agent satisfaisant, il vérifie s'il s'agit d'un agent de morphologie ou de pré-morphologie. Si c'est un agent de pré-morphologie il lui demande de créer un agent de morphologie rendant compte de leur agrégation. Si c'est un agent de morphologie, il s'y agrège tout simplement.

```
findMorphologyAgent
    | morphology_agent |
    morphologyAgents do: [:anAgent |
        (anAgent focus - focus) abs <= anAgent range
            ifTrue: [
                anAgent agentType = #PreMorphologyVFSAgent
                    ifTrue: [
                        (anAgent id <> id)
                            ifTrue: [
                                morphology_agent := anAgent
                                    spawnMorphologyAgentFrom: self.
                                ^morphology_agent]]
                    ifFalse: [
                        morphology_agent := anAgent.
                        morphology_agent connect: self.
                        ^morphology_agent]]].
        ^nil
```

Recherche d'un agent de morphologie

8.2 Implémentation des Agents de Morphologie

Un agent de morphologie n'étant que la réification d'une caractéristique émergente du paysage d'agents, il ne fait que peu de choses. Il doit uniquement se contenter de recalculer sa portée et sa force en fonction du va-et-vient des agents de pré-morphologie qu'il héberge. Enfin, il doit vérifier s'il doit effectivement se suicider au cas où le nombre d'agents de pré-morphologies deviendrait trop faible.

8.2.1 Mise à jour des valeurs intrinsèques à l'agent de morphologie

L'activité principale d'un Am consiste à recalculer sa portée. La création de chréodes nécessitera d'introduire dans cette méthode le broadcast de recherche des agents ayant des aspects en commun.

La force d'un agent de morphologie décroissant avec la distance par rapport au focus, nous avons choisi une loi de décroissance logarithmique. On peut cependant imaginer toute autre loi en fonction du but recherché. Cette loi peut même être modifiée dynamiquement (point 6 des caractéristiques d'un agent).

```
body
self range: self computeRange

computeRange
        ^strength ln / 10.
```

Calcul du range

Gestion des agents de pré-morphologie

La déconnexion d'un agent de pré-morphologie correspond au cas où cet agent a pu s'extraire de l'influence de l'agent de morphologie. La force de cet agent de morphologie diminue donc en conséquence. Si le nombre d'agents de pré-morphologie agrégés tombe en dessous du seuil de viabilité, l'agent de morphologie se suicide. C'est ici que doit s'insérer le mécanisme de rémanence morphologique.

La connexion d'un agent de pré-morphologie correspond au cas où celui-çi s'est stabilisé. Les modifications sont l'inverse de celles qui se produisent lors de la déconnexion.

```
disconnect: anAgent
	agregatedAgents remove: anAgent.
	self strength: strength - 1.
	self agregatedAgentsCount: agregatedAgents size.
	self range: self computeRange.
	agregatedAgents size <= thresold ifTrue:
						[^self crumbling].

connect: unAgent
	agregatedAgents add: unAgent.
	self agregatedAgentsCount: agregatedAgents size.
	self strength: strength + 1.
	self range: self computeRange.
```

Ajout/soustraction d'un Apm

Prise en compte du seuil de viabilité

C'est ce que fait l'agent lorsqu'il se suicide: il prévient les agents de pré-morphologie restants. Dans le modèle courant, le seuil de viabilité est fixé à 1.

```
crumbling
	| last_agent |
	agregatedAgents do: [:lastAgent |
				last_agent morphologyAgentDied].
	agregatedAgents := nil.
```

Suicide de l'Am si en dessous du seuil de viabilité

Chapitre 9

Les Agents d'Analyse du Système d'Interprétation

Nous n'avons présenté jusqu'ici que les processus de prise en compte des traits sémantiques dans les messages échangés par les acteurs et les outils de lecture morphologique des SMA aspectuels qui évoluent tout au long des échanges de messages. Il n'est fait nulle part mention d'une quelconque analyse rationnelle des messages. Cependant, un outil d'aide à la gestion de crise se doit de donner des informations sur la situation en cours. Ces informations peuvent être quantitatives (il reste 3 minutes avant l'arrivée de l'ambulance X, etc ...) ou mieux encore, qualitatives: la situation se dégrade et on risque tel ou tel évenement. C'est le rôle des agents d'analyse que de fournir de telles informations. Ils s'appuient sur des règles et plans typiques des institutions, mais aussi sur des archives de cas.

9.1 Définition et principes

Jusqu'au niveau morphologique, le comportement du système est dirigé par la perception : les agents aspectuels fournissent une appréhension du monde selon plusieurs catégories sémantiques mais au travers d'un filtre intentionnel fort. Pour que le système puisse être utilisable, il doit être capable, au minimum, de placer les faits saillants dégagés par le SMA de morphologie dans le temps et relativement à des plans. Il doit analyser rationnellement sa représentation du monde. Les agents d'analyse sont typiquement des mini-systèmes à base de connaissances (CBR, Système expert, etc ...) ou des réseaux de neurones par exemple.

Les roles d'un agent d'analyse sont les suivants:

1. Prédire l'évolution du SMA aspectuel
2. Donner les faits saillants et les relier/expliquer

L'état 1 est l'état d'éveil de l'agent d'analyse durant lequel il va chercher dans les chréodes des informations déclenchant des règles dans le cas d'un sys-

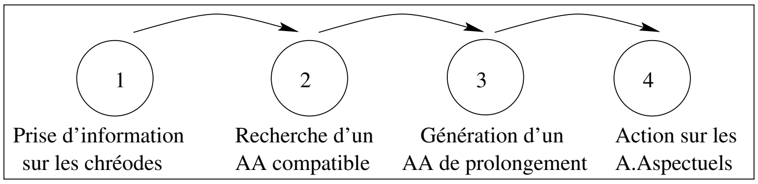

FIG. 9.1: *Fonctionnement d'un agent d'analyse*

tème expert, ou des formes reconnues par un réseau de neuronnes

L'état 2 est l'état de scrutation pendant lequel il va se mettre en quête d'un autre agent d'analyse qui travaille sur la même chréode ou sur une chréode duale.

L'état 3 correspond au moment où, ayant trouvé un agent d'analyse compatible, il vont créer un nouvel agent d'analyse mais qui sera cette fois-çi focalisé sur ce qui aura permi les deux agents géniteurs de passer à l'état 2. Ce nouvel agent sera au minimum créé dans son état initial 2.

L'état 4 représente le cas où l'agent d'analyse cherche à influencer les agents aspectuels pour renforcer les informations qui servent de base à son travail.

Les différentes catégories d'agents d'analyse sont actuellement en cours d'implémentation.

Chapitre 10

Distributed Smalltalk de ParcPlace

Distributed Smalltalk^{tm} (DST) est un ensemble de classes ajouté à VisualWorks^{tm} pour le développement des applications réparties. DST permet au programmeur de créer des applications distribuées facilement et rapidement comme n'importe quelle application Smalltalk. DST est basé sur la norme CORBA2.0 (Common Object Request Broker Architecture). C'est une norme à objets répartis qui permet l'interaction entre des composants objets. CORBA fournit une plate-forme d'exécution à base de technologies orientées objet pour l'intégration et l'interopérabilité d'applications distribuées et hétérogènes. Nous présentons dans ce chapitre les principaux caractères de DST.

10.1 La création d'une application DST

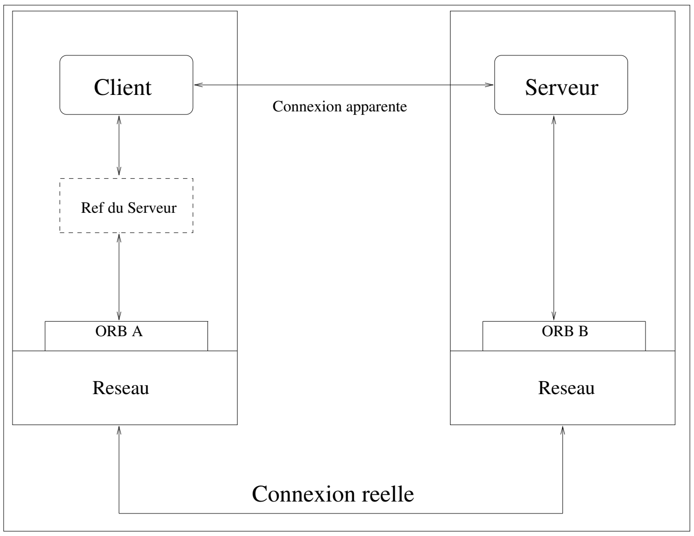

FIG. 10.1: *Modèle client-serveur DST*

Pour créer une application répartie (Fig 10.1), il faudra au minimum deux objets, l'un jouant le rôle du client et l'autre du serveur. Le développement d'une application DST nécessite de suivre les étapes suivantes:

1. création et test de l'application non distribuée,

2. création de code IDL : une fois que l'objet serveur est créé, il faut créer le code IDL correspondant à l'objet. Ce code décrit les interfaces que le client peut appeler. DST permet la généreration automatique du code IDL correspondant à un objet.

Remarque: L'activation de l'ORB (Object Request Broker) est nécessaire, pour pouvoir démarrer l'IDL Generator :

– Cliquer sur le menu Tool de DST fenêtre.

– Cliquer ensuite sur la commande " IDL Generator ".

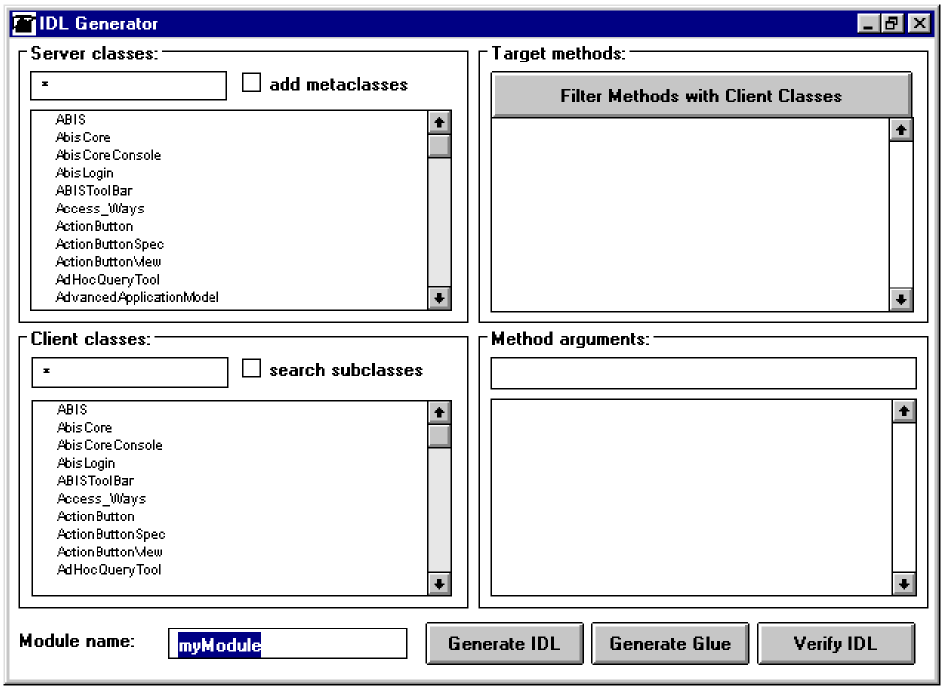

FIG. 10.2: *Générateur d'IDL*

Une boîte de dialogue apparaît (Fig 10.2) :

– La saisie de gauche, en haut, permet de sélectionner les objets pour lesquels vous souhaitez créer le code IDL.

– A droite, pour chaque objet sélectionné, on doit sélectionner les méthodes à offrir.

– Dans la saisie de gauche et en bas, on a le choix pour indiquer les clients potentiels des objets serveurs.

- A droite, en bas, on a une liste avec les types de sortie et des paramètres pour chaque méthode saisie.
- On dispose d'un champ de saisie correspondant au module à créer et il est nécessaire de lui donner un nom.

Une fois que tous les champs sont remplis, on clique sur le bouton " Generate IDL " pour obtenir une fenêtre avec l'Interface Repository, on retrouve ainsi le code généré par DST . En faisant un **accept** le code sera stocké dans le Repository. L'étape suivante consiste à lier le code généré avec l'implémentation de l'objet serveur. Il suffit de cliquer sur " Generate Glue " qui ajoute deux méthodes dans la classe implémenté de l'objet serveur qui sont **CorbaName** et **AbstractClassId**.

- **CorbaName**: indique le module IDL auquel l'implémentation de l'objet appartient.
- **AbstractClassId**: garde un identificateur unique de classe qui permet à l'ORB de reconnaître l'implémentation à utiliser à l'appel d'une méthode d'un objet.

La dernière étape est la vérification du code IDL généré par rapport à l'implémentation de l'objet. Pour enregistrer l'application, il suffit de sauvegarder l'image.

L'objet serveur doit s'enregistrer dans le Naming Service pour que l'objet client puisse consulter ce Naming Service et avoir une référence sur l'objet serveur afin d'appeler localement les méthodes de ce dernier. Le code qui permet à l'objet serveur de s'enregistrer dans le Naming Service, ainsi que le code qui permet a l'objet client d'avoir la référence de l'objet serveur, sont simples à retrouver dans la documentation DST ParcPlace.

10.2 Lancement de l'application

Une fois que le serveur et le client sont implémentés, et avant de démarrer toute application distribuée, il faut configurer les settings des deux objets client et serveur, puis lancer les ORB correspondants.

10.2.1 Les settings

Pour permettre la communication intra-ORB, il faut configurer ce qu'on appelle les settings. Les settings constituent l'environnement qui permet l'interaction entre différents ORB. Cet environnement contient:

- **IIOP Transport**: le protocole IIOP défini par l'OMG permet la communication entre différents ORB. On peut fixer manuellement le numéro de port ou laisser le système le faire dynamiquement.

- **le Naming Service**: permet la localisation locale ainsi que distante des objets. Si dans la saisie " Naming Service " on choisit l'option " local ", alors l'ORB créera une instance du Naming Service local. Si nous indiquons un IdAdapter, l'ORB regardera les valeurs indiquées dans le champ de saisie pour trouver le Naming Service.

- **Repository**: c'est le service utilisé pour partager l'interface repository dans le système distant.

10.2.2 Configuration de l'objet serveur

Pour la configuration de l'objet serveur, il faut sélectionner l'option :

- **configuredTo** pour le IIOP Transport

- **Local** pour le Naming Service

- **Local** pour le Repository

10.2.3 Configuration de l'objet client

Pour la configuration de l'objet serveur, il faut sélectionner l'option:

- **Dynamically Allocated** pour le IIOP Transport

- **Hostname** pour le Naming Service, en tapant le hostname du Naming Service de l'objet Serveur

- **Local** pour le Repository

10.3 Conclusion

Cette représentation de DST est une simple introduction qui permettra de créer des applications distribuées. Les détails d'implémentation seront vus dans le chapitre suivant qui traite la distribution du système.

Chapitre 11

Distribution du système avec CORBA

Nos travaux actuels consistent à distribuer le prototype de système de gestion de crises sur un réseau privé. En effet, ce système doit fonctionner entre différentes institutions réparties géographiquement et utilisant un réseau informatique afin de communiquer et de coopérer. L'implémentation est réalisée en Distributed Smalltalk^{tm}, la version de Smalltalk qui réalise la norme CORBA.

Cette norme fournit une plate-forme d'exécution à base de technologie orientée objet pour l'intégration et l'interopérabilité d'applications distribuées et hétérogènes. Cela signifie qu'elle permet à des systèmes fonctionnant sous des systèmes d'exploitations différents et utilisant des langages de programmation différents, de s'échanger des services et des données dans un langage commun. L'avantage par rapport à des systèmes classiques est indéniable puisque la mise en place des futures machines du système de gestion de crise ne pose plus aucun problème d'adaptation.

La distribution sur plusieurs machines, se fait de façon transparente en ce qui concerne l'utilisation des objets. Celle-çi s'opèrera toujours de la même façon, que l'objet utilisé soit local à l'image ou qu'il soit sur une image distante (une image Smalltalk contient le code qui est éxécuté par la machine virtuelle). De façon transparente, c'est toute l'architecture de CORBA qui permet un tel fonctionnement.

11.1 Le Naming Service

Concrètement, le service de CORBA que nous utilisons principalement est le *Naming Service*. Celui ci permet d'enregistrer les objets de façon centralisée pour l'ensemble des images du système. Il s'agit en fait d'un processus qui fonctionne sur l'une des images et dont les autres images possèdent la référence. Les objets ainsi enregistrés y sont désignés par un nom qui permet de les retrouver depuis n'importe quelle image.

On peut récupérer une référence sur ce service en invoquant la méthode *namingService* de l'objet **ORBObject** :

```
| aNamingService |
    aNamingService := ORBObject namingService.
```

La démarche pour la distribution d'un objet est simple. Elle peut pratiquement se résumer à l'utilisation de trois opérations :

1. Enregistrement d'un objet dans le naming service avec un nom donné

```
ORBOBject namingService contextBind: anObject to: aName
```

Cette opération permet de donner un nom à l'objet dans le naming service. Le paramètre *aName* doit être un objet **DSTName**

2. Récupération de la référence d'un objet à partir de son nom

```
| anObjectRef |
    anObjectRef := ORBObject namingService contextResolve: aName.
```

Le nom de l'objet connu, nous pouvons récupérer la référence de cet objet. Cette référence équivaut, au moment de l'utilisation, à une référence sur un objet local. Nous pouvons donc invoquer ses méthodes de la même façon que nous le ferions pour un objet local.

3. Suppression du nom d'un objet du naming service.

```
ORBObject namingService contextUnBind : aName.
```

L'objet n'est plus référencé dans le naming service. On ne peut donc plus le retrouver par ce moyen. En revanche, il n'est pas détruit et donc toutes les références sur lui restent valides.

L'utilisation du naming service, bien que pratique, n'est pas indispensable. Elle offre simplement un moyen aisé de retrouver des objets distants. On l'utilise dans l'unique but d'obtenir une référence sur une liste (un objet de la classe **Set** par exemple). Cette liste contient alors les références des objets distribués, on peut ensuite parcourir cette liste de récupérer ainsi les références au fur et à mesure des besoins.

11.2 Bases pour la distribution

Nous avons vu que le fait de distribuer ou non les objets à travers le réseau ne change rien la façon de les utiliser. Les seules différences résident essentiellement dans l'enregistrement et la récupération de la référence d'un objet par le naming service. Au niveau des interfaces utilisateurs, nous avons rajouté les configurations nécessaires à l'utilisation en réseau (adresses IP de la machine contenant le naming service).

Au niveau de l'implémentation, nous avons ajouté deux classes afin de faciliter la distribution des objets. Nous en donnons ci-dessous les principales méthodes.

11.3 La classe ServicesDST

```
Object subclass: #ServicesDST
        instanceVariableNames: ''
        classVariableNames: ''
        poolDictionaries: ''
        category: 'ABIS-DST'
```

Cette classe regroupe et fournit les fonctions de bases des différents services de CORBA implémenté en Distributed Smalltalk.

Les méthodes

namingService	retourne une référence sur le naming service
getName: aName	retourne un objet DSTName construit à partir d'une string
getObjRef: aName	retourne la référence d'un objet à partir de son nom
listBindings: aValueHolder	retourne la liste de tous les objets enregistrés
registration: anObject as: aName	enregistre un objet sous un nom donné
reRegistration: anObject as: aName	change le nom d'un objet déjà enregistré
unregistration: aName	efface l'objet de nom donné du naming service

Toutes les opérations d'enregistrement des objets concernent l'enregistrement de ceux-ci dans le naming service. Les différentes méthodes de la classe ServicesDST existent en tant que méthodes de classes et méthodes d'instances et ne nécessitent donc pas d'instanciation d'objets pour être utilisées.

11.4 La classe DSTObject

```
Object subclass: #DSTObject
        instanceVariableNames: 'NSRefName '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'ABIS-DST'
```

Les différentes classes à distribuer hériterons, pour la plupart, de cette classe. Celle ci possède les méthodes qui lui permettent de s'auto-gérer par rapport au naming service.

Les méthodes

nsRefName	retourne le nom donné pour cet objet dans le naming service
nsRefName: aName	fixe le nom de cet objet
registrationAs: aName	enregistre cet objet sous un nom donné
reRegistrationAs: aName	change le nom de cet objet
unregistration	efface cet objet du naming service

11.5 Architecture distribuée du système

La section suivante traitant de l'architecture du système mais ceci de façon distribuée, nous renverrons ici le lecteur au chapitre 3 de ce rapport. L'architecture distribuée de notre système est basée sur une topologie en anneau où chaque noeud représente un site hébergeant un ou plusieurs utilisateurs appartenant à une même organisation (Fig 1.1). Une image Distributed Smalltalk s'exécute sur chacun de ces sites.

Les utilisateurs se logent sur des machines connectées à un noeud de la boucle et accèdent au système en ouvrant une session qui permet de réaliser des envois de messages et une lecture de la situation actuellement traitée. L'ensemble des informations émanant des différentes sessions fait l'objet d'un traitement par une session dite "superviseur". Les sessions et le superviseur sont connectés et communiquent via un bus (un bus objet qui s'appuie sur le bus CORBA) qui est un objet unique et partagé, contenant un certain nombre de renseignements sur chaque site connecté.

11.6 Les machines et leur rôle

11.6.1 Le 'Boot Server'

Le 'Boot Server' est la machine clé de notre système : elle héberge le *Naming Service*, ainsi que la session superviseur. Comme nous l'avons dit le superviseur réalise un traitement des données émanant des différentes sessions. Ce traitement est assuré par les SMA de morphologie.

11.6.2 Les serveurs de sessions

Les serveurs de sessions sont hébergés chacun par une machine différente qui logiquement correspond à une institutions unique (pompier, SAMU, etc). Avec le 'Boot Server', ces machines forment l'anneau physique (au sens des réseaux informatiques) de notre système. Elles servent surtout à la mise en route des sessions et permettent de maintenir une facilité d'accès de certaines informations sur :

1. les sessions pour le 'Boot Server' : nombre de sessions ouvertes, statistiques diverses sur les informations ressortant de cette institution, etc.
2. le 'Boot Server' pour les sessions : accès au bus de communication principalement.

11.6.3 Les sessions

Les sessions constituent les éléments utilisateurs les plus importants. C'est la partie visible du système. A la connexion, l'utilisateur a juste à savoir quelle est la machine 'Boot Server' et quelle est le nom de l'institution à laquelle il se rattache. Ensuite, tout se fait de façon transparente pour lui.

Nous décrirons la procédure de login d'un nouvel utilisateur tout en présentant les différentes fenêtres qui le permettent.

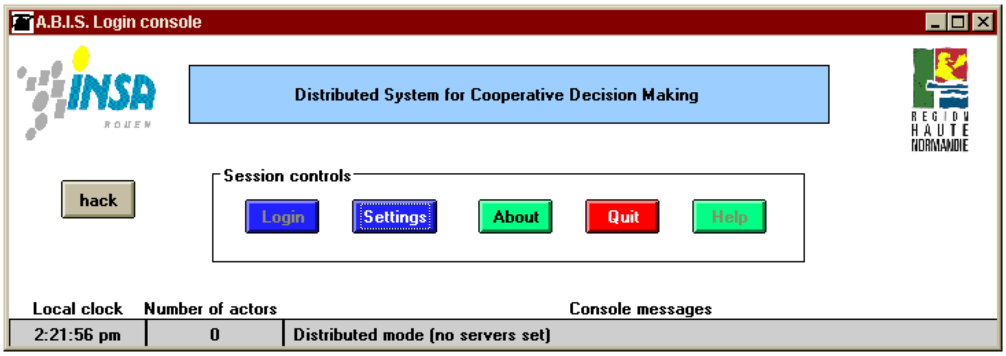

FIG. 11.1: *Interface utilisateur : fenêtre de démarage*

Si les configurations (Settings) pour l'utilisation en mode distribué ne sont pas faites le login n'est pas possible.

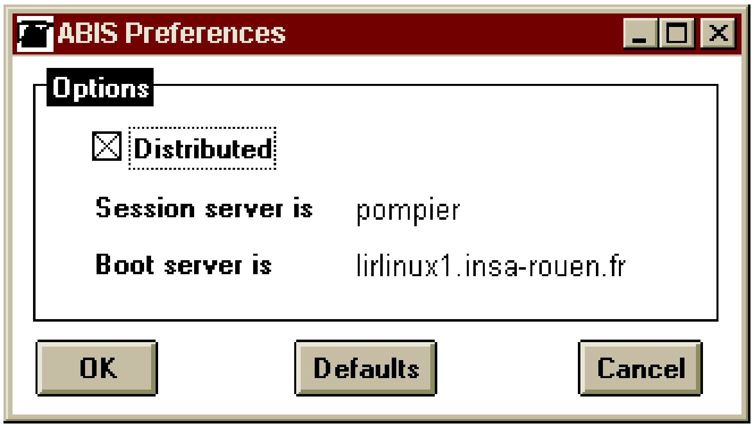

FIG. 11.2: *Fenêtre des settings*

La fenêtre des settings est très simple : une case à cocher permet de choisir si l'on travaille en mode distribué ou non (en situation réelle cela sera le cas) et deux champs nous permettent de saisir, l'un le nom du serveur de session et l'autre celui du 'Boot Server'.

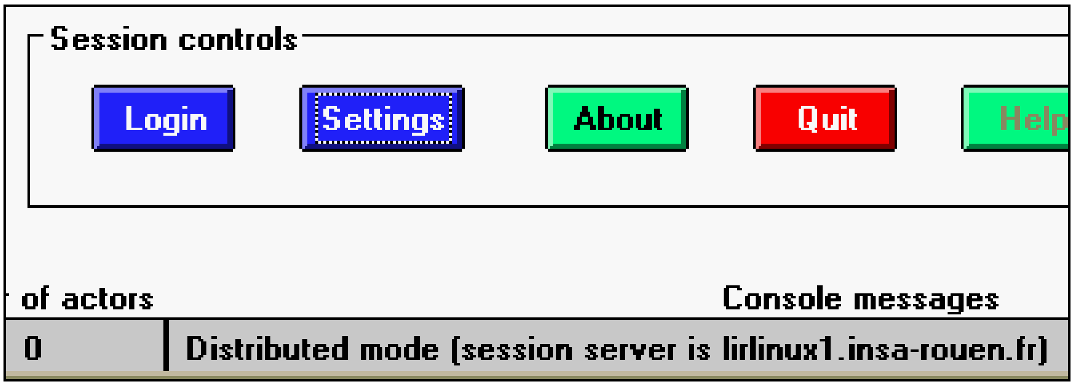

FIG. 11.3: *Extrait de la fenêtre de démarage après configuration des Settings*

Après avoir procédé à ces configurations, et si elles sont correctes, l'utilisateur peut se loger pour accéder au système.

11.7 Principe de fonctionnement des SMA en mode distribué

Dans notre modélisation, chaque session possède un SMA aspectuel et chaque SMA aspectuel est traité par un SMA de morphologie qui est, lui, situé au niveau du superviseur. Chacun des agents du SMA aspectuel est tenu par un

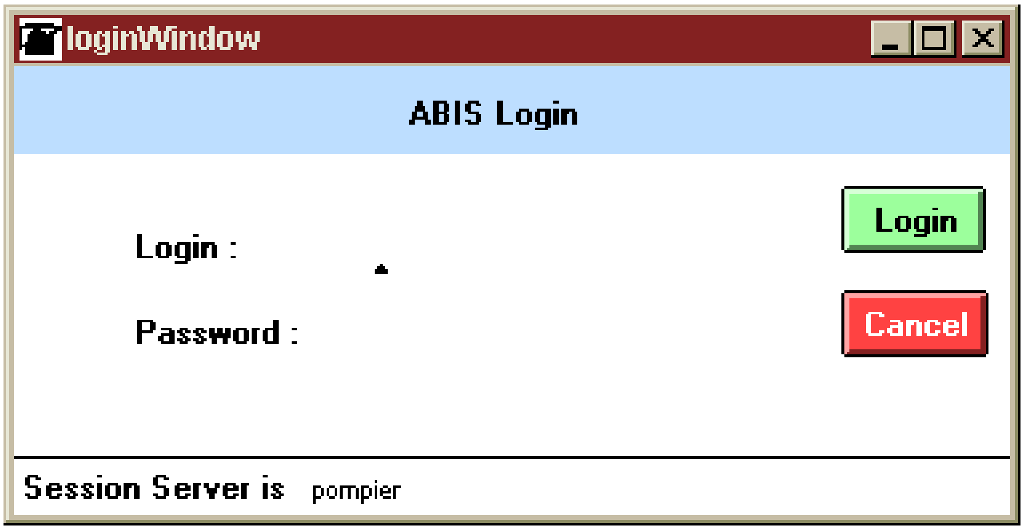

Fig. 11.4: *Fenêtre de login*

agent de pré-morphologie dans le SMA de morphologie.

Au niveau de l'implémentation chaque SMA répertorie ses agents dans un objet nommé *agentsSet* (un objet de la classe **Set**). En mode non distribué, le SMA de morphologie les prend en compte en parcourant simplement cet ensemble et en récupérant pour chaque agent de pré-morphologie, la référence de son agent aspectuel correspondant.

Pour le fonctionnement en mode distribué nous n'avons que deux contraintes supplémentaires :

1. Enregistrer le SMA aspectuel dans le *Naming Service*,
2. Récupérer pour le SMA de morphologie, une référence sur ce SMA aspectuel. A partir d'ici le fonctionnement est exactement le même qu'en mode non distribué,
3. Parcourir la liste des agents du SMA aspectuel et récupérer leur référence pour les passer aux agents de pré-morphologie.

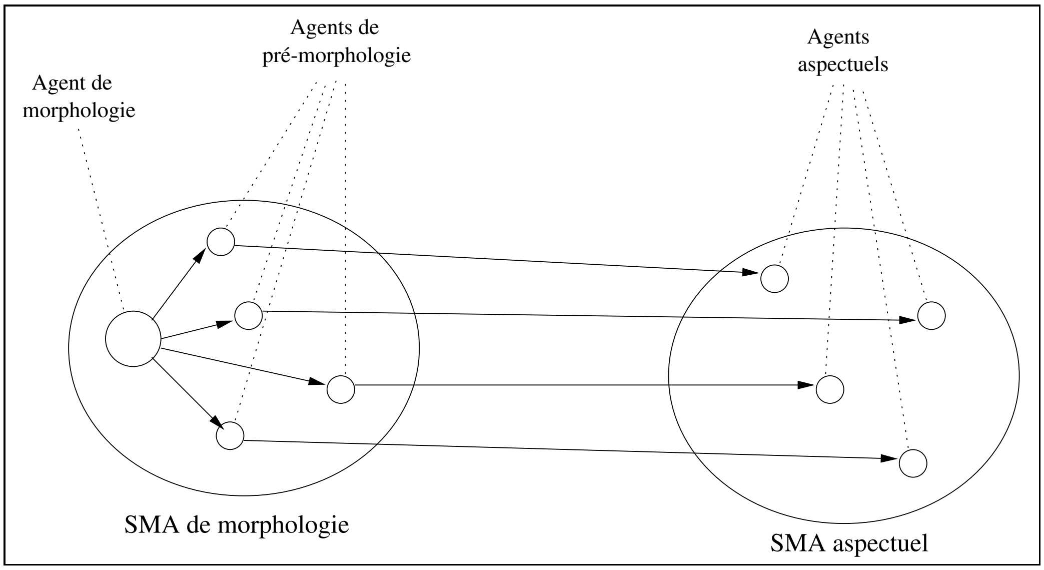

Fig. 11.5: *Lien SMA aspectuel - SMA de morphologie en mode non distribué*

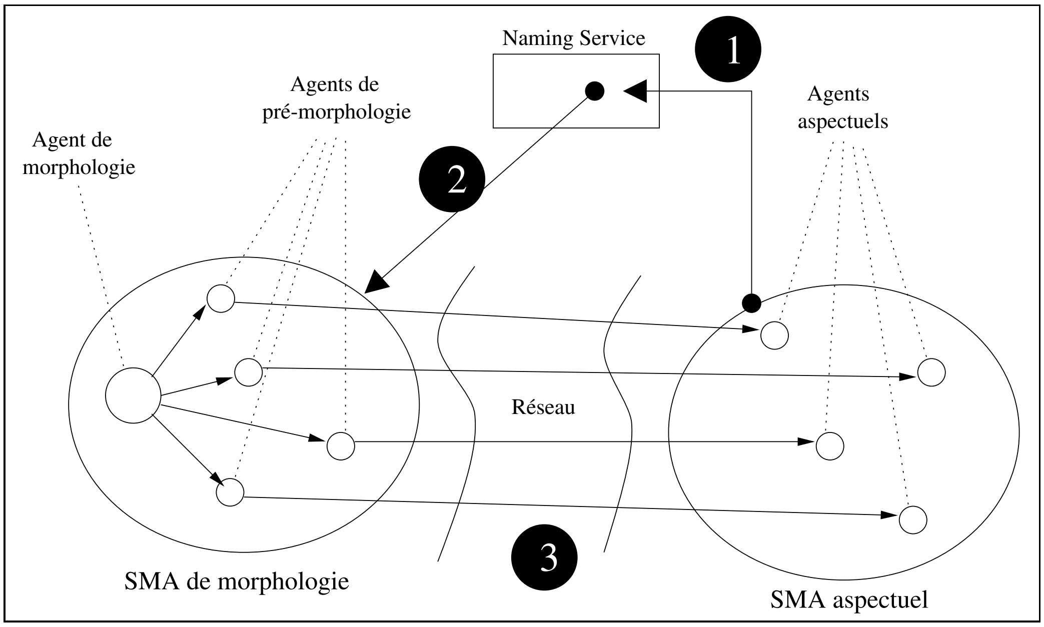

Fig. 11.6: *Lien SMA aspectuel - SMA de morphologie en mode distribué*

Conclusion

Les systèmes adaptatifs semblent ouvrir une voie nouvelle pour la réalisation de SIC d'aide à la decision pour la gestion de crises. En effet, ces situations mettent en jeu différentes institutions qui ont des procédures, des usages et des motivations qui leur sont propres. Le rapport européen STEP [STEP, 1994] ayant clairement établi que bien que les informations factuelles soient indispensables dans de tels systèmes, la prise en compte de la pragmatique des messages est nécessaire afin d'éviter des incompréhensions, des conflits qui mènent à une "crise dans la crise". Aussi, le défi qu'un système d'aide à la décision doit relever est important : il doit conserver les fonctionnalités des systèmes classiques et apporter un traitement de la pragmatique des messages échangés [Eco, 1993] afin de fournir aux acteurs une lecture compréhensible et synthétique de la situation perçue par <u>tous</u> les acteurs.

Notre approche consiste en l'incorporation des modules classiques des SIC au sein d'un système plus complèxe devant fournir une vue synthétique de la situation en fonction des messages échangés entre les différents acteurs. Le système d'interprétation de ces messages est basé sur un modèle s'appuyant sur des organisations adaptatives d'agents.

L'implémentation de notre modèle a nécessité la réalisation d'une mini-plateforme SMA intégrant la norme CORBA afin de permettre la distribution de notre système sur les différents sites susceptibles d'être impliqués dans une situation de crise. Les choix d'implémentation et du fonctionnement de nos agents ont été dirigés par des contraintes évidentes de performances et de mise au point du système d'interprétation.

Un prototype de notre système est actuellement en phase de test et de validation auprès des Sapeurs Pompiers du Havre, de la Préfecture de région, du SAMU du Havre. Nous esperons obtenir un retour d'expérience afin de vérifier les hypothèses à la base de notre modèle et enrichir l'ontologie du domaine.

Cependant, il nous reste encore à terminer l'implémentation des agents d'analyse et de décision qui devront permettre de tester effectivement la boucle d'interprétation des messages. Bien que ces deux catégories d'agents ne soient pas encore implémentées dans le système, la lecture produite par les agents de morphologie est, malgré tout, interessante du point de vue des institutions. Un travail important sur l'ergonomie du système est en cours.

Le modèle actuel de système adaptatif est toujours en évolution et les travaux déjà menés laissent entrevoir des persepctives nouvelles qui font l'objet de recherches sur les systèmes auto-adaptatifs de génération de sens.

Bibliographie

J.L. Austin. *Quand dire c'est faire*. Le Seuil, 1970.

M. Bares. *Systèmes de commandement, Elements pour une prospective*. Polytechnica, 1996.

E. Borodzicz, J. Aragones, and N. Pidgeon. Risk communication in crisis: meaning and culture in emergency response organizations. In *European Conference on Technology & Experience in Safety Analysis and Risk Management, Rome*, 1993.

J.-P. Briot. Modlisation et classification de langages de programmation concurrente objets : l'exprience Actalk. In *LMO'94*, pages 103–125, Grenoble, 1994.

A. Cardon. *La complexité des systèmes d'expression du sens*. IBP, 1996.

A. Cardon. A multi-agent model for co-operative communications in crisis management system: the act of communication. In *Proceedings of the 7th European-Japanese Conference on Information Modelling and Knowledge Bases*, pages 111–123, May 1997.

A. Cardon. *Les systèmes adaptatifs à architecture d'agents dynamiques : une approche de la conscience artificielle*. LIP6, Fevrier 1998.

A. Cardon and S. Durand. A model of crisis management system including mental representations. In *Proceedings of the AAAI Spring Symposium*, March 1997.

A. Cardon and F. Lesage. Toward adaptive information systems : considering concern and intentionality. In *Proceedings of KAW'98*, April 1998.

F. Cordier. Armand Colin, 1994.

H. Eco. *Sémiotique et philosophie du langage*. PUF, 1993.

Z. Guessoum. Dima, une plate-forme multi-agents en smalltalk. *Objet*, 3(4): 393–409, 1997.

Z. Guessoum and M. Dojat. A real-time agent model in an asynchronous object environment. In *MAAMAW'96*, pages 190–203, Eindhoven, The Netherlands, January 1996. Springer Verlag.

B. Hayes-Roth and A. Collinot. *A Satisficing Cycle for Real-Time Reasoning in Intelligent Agents.* 1993.

R. Jackendoff. *Semantics and Cognition.* Cambridge, M.I.T. Press, 1983.

J.M. Jacques, I. Adant, F. Gaspart, and L. Gatot. Aggravating factors and organizational learning: a multidisciplinary approach of crisis modelling. In *Workshop for Greening of Management, EIASM, Bruxelle*, 1995.

J.W. Lapierre. *L'analyse des systèmes.* Syros, 1992.

J.-L. Le Moigne. *La Modélisation des Systèmes Complexes.* Dunod, Paris, 1990.

D. Malavieille. *Etude et réalisation d'un système à base de connaissances pour l'aide à l'intervention sur un accident majeur.* PhD thesis, Thèse de l'Université Paris XI Orsay, 1998.

P. Oziard and J. Ruhla. *Méthodes et outils pour le développement des SIC.* Dunod, 1992.

C.S. Peirce. *Textes anticartésiens.* Aubier, 1984.

E.L. Quarantelli. *Disaster Crisis Management: a summary of research findings.* 1998.

J.R. Searle. *Speech Acts.* Cambridge University Press, 1969.

J.R. Searle. *La redécouverte de l'esprit.* Gallimard, 1992.

L. Sfez. *Critique de la décision.* Presses de la fondation nationale de sciences politiques, 1992.

P. Shrivastava. *Technological and organisational roots of industrial crisis: lessons from Exxon Valdez and Bhopal.* 1994.

STEP. *Peut-on modifier les processus mentaux de représentation et de décision en situation d'urgence.* Contrat CEE STEP 90-0094, 1994.

E. Le Strugeon. *Une méthodologie d'auto-adaptation d'un système multi-agents cognitifs.* 1995.

B. Toft and S. Reynolds. *Learning from Disaster: a Management Approach.* Butterworth-Heinneman,Oxford, 94.

T. Winograd and F. Flores. *Understanding Computers and Cognition: A new Fundation for Design.* New-Jersey, Abley Press, 1986.