Des Systèmes Multi-Agents temporels pour des systèmes industriels dynamiques

Thibault Carron

▶ **To cite this version:**

Thibault Carron. Des Systèmes Multi-Agents temporels pour des systèmes industriels dynamiques. Intelligence artificielle [cs.AI]. Ecole Nationale Supérieure des Mines de Saint-Etienne; Université Jean Monnet - Saint-Etienne, 2001. Français. NNT : 2001EMSE0026 . tel-00818319

HAL Id: tel-00818319

https://theses.hal.science/tel-00818319v1

Submitted on 26 Apr 2013

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

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

Ecole nationale supérieure des mines de Saint-Etienne

Université Jean Monnet

Des Systèmes Multi-Agents temporels pour des systèmes industriels dynamiques

THÈSE

présentée et soutenue publiquement le 20/12/2001

pour l'obtention du

Doctorat de l'école Nationale Supérieure des Mines de Saint-Etienne et de l'université Jean Monnet

(spécialité informatique)

par

Thibault CARRON

Composition du jury

Président :	F. Oquendo
Rapporteurs :	Y. Demazeau
	P. Enjalbert
Examinateurs :	F. Jacquenet
	C. Sayettat
	O. Boissier

Ecole nationale supérieure des mines de Saint-Etienne

Université Jean Monnet

Des Systèmes Multi-Agents temporels pour des systèmes industriels dynamiques

THÈSE

présentée et soutenue publiquement le 20/12/2001

pour l'obtention du

Doctorat de l'école Nationale Supérieure des Mines de Saint-Etienne et de l'université Jean Monnet

(spécialité informatique)

par

Thibault CARRON

Composition du jury

Président :	F. Oquendo
Rapporteurs :	Y. Demazeau P. Enjalbert
Examinateurs :	F. Jacquenet C. Sayettat O. Boissier

Remerciements

Depuis notre enfance jusqu'à la "vie active", nous rencontrons parfois quelques personnes exceptionnelles qui se révèlent décisives pour notre orientation et notre vie future. Aujourd'hui, j'aimerai remercier deux personnes en particulier à qui, il me semble devoir beaucoup : Jean-charles Marty et Olivier Boissier. Directement par leurs conseils, leurs encouragements, leur soutien ou indirectement par leur motivation pour leur métier, leur franchise, leur honnêteté, ils m'ont permis de mener à bien cet objectif et de ne pas choisir la voie de la facilité là où cela aurait été tellement plus...soulageant. Aujourd'hui, ils sont naturellement des amis.

Ensuite, dans le cadre de la thèse, j'aimerais également remercier Claudette Sayettat qui était toujours présente dans les moments importants et m'a montré sa confiance dès le DEA. Je souhaite remercier Patrice Enjalbert et Yves Demazeau pour avoir accepté d'être les rapporteurs et surtout pour la qualité de leur remarques et le sérieux de leur avis : tous deux, chacun sur des plans différents, m'ont ouvert des perspectives et permis d'envisager ce mémoire sous des angles différents, très enrichissants. Je remercie Flavio Oquendo et François Jacquenet pour avoir accepté de faire partie du jury et leur soutien amical.

Pour terminer, je suis persuadé que l'environnement joue un rôle très important (cf. manuscrit...), je tiens donc à remercier l'Ecole de Mines de Saint-Etienne et la région Rhône-Alpes pour les conditions excellentes offertes pour mener cette thèse jusqu'à son terme. Au sein de l'Ecole des Mines, un excellent souvenir et ma gratitude vont à l'équipe SMA, à Liliane Brouillet pour sa gentillesse et sa disponibilité, et plus généralement tout le 5ième étage ainsi que le personnel "système" pour sa compétence et sa sympathie : ce sont principalement Bernard Kaddour, Florence Dujardin, Michel Chatard et Jean-françois Tchebanoff. De manière plus ludique mais tout aussi importante et équilibrante : je remercie et salue tous les thésards et personnels de l'école qui ont contribués à créer cette ambiance sympathique et dépourvue de tout stress...Parmi ceux-ci, l'équipe des "Bombermen" (Jojo, Nico, Lolo et invités), l'équipe de foot du foyer du personnel et leur inénarrable "coach"', les amateurs de starcraft, tous les "amis de Chambéry" qui ne nous ont pas oubliés dans notre "exil"', le groupe de moniteurs de Grenoble et tous ceux qui ne sont cités ici mais savent qu'ils sont là et pour beaucoup comme ma famille, ...

Et enfin, "last but not least", et loin de là : le soutien et le réconfort de tous les instants,

- à ma douce et tendre Karine...d'une patience angélique pour donner naissance quatre jours avant la soutenance...
- à Aloys, petite merveille à qui je souhaite de vivre autant de moments intenses et de rencontrer tant de personnes laissant un excellent souvenir.

Table des matières

II Modèle de système multi-agents temporel

III Applications temporelles

Table des figures

Table des figures

Introduction

Problématique

Depuis quelques années, notre société est marquée par un certain nombre d'évolutions sur le plan industriel. Parmi celles-ci, trois faits fondamentaux nous intéressent particulièrement : ce sont le développement de nouvelles technologies de communication, l'arrivée de l'informatique à tous les niveaux dans les systèmes industriels et l'évolution dans des mesures importantes, de la complexité de ces systèmes. La concomitance de ces trois faits impose dorénavant pour un certain nombre de ces systèmes, la prise en compte au niveau logiciel des aspects liés à la *distribution*, à la *décentralisation* ou à l'*autonomie* comme nous allons le voir (cf. chapitre 1) mais aussi des aspects liés à la *dynamique*, à l'*évolutivité* et à la *flexibilité* (cf. chapitre 2) pour pouvoir répondre à de nouveaux problèmes relatifs au contrôle, à la surveillance ou à la simulation, par exemple. Plus précisément, le développement d'applications informatiques coopératives (réseaux d'entreprises, entreprises virtuelles, e-business...) prend de plus en plus d'importance. Il devient ainsi crucial de disposer de méthodes et d'outils logiciels bien adaptés aux caractéristiques de tels systèmes afin de pouvoir les modéliser aussi fidèlement et précisément que nécessaire. De plus, dans le cadre d'environnements et/ou de systèmes industriels très dynamiques, il s'avère souvent nécessaire de pouvoir prendre en compte de manière explicite les aspects temporels à l'intérieur des différentes composantes de telles applications logicielles.

Motivations

La résolution de ces problèmes se révélera cruciale pour les entreprises du XXI^{ieme} siècle qui évolueront dans un environnement en constant changement et deviendront encore plus dépendantes du *temps*. Cette nécessité croissante pour rester compétitives face au monde entier (et non plus seulement "localement"), d'être très réactives (voire même innovantes) donc *de maîtriser la dimension temporelle*, situe notre cadre applicatif au sein de ces entreprises. En effet, pour celles-ci, le besoin d'outils adaptés capables de prendre en compte tous ces aspects, va devenir une priorité. Cette problématique intéresse fortement l'industrie. Nos recherches sont d'ailleurs liées à deux projets issus du milieu industriel. En effet, comme nous allons le voir, bien que les systèmes multi-agents (SMA) soient bien adaptés pour

représenter ce type de systèmes, peu de travaux sur les SMA offrent une prise en compte explicite du temps, encore moins différencient plusieurs niveaux dans cette prise en compte. Enfin, il n'existe pas à notre connaissance, de proposition ou de modèle de SMA spécifiquement temporel permettant de modéliser de manière satisfaisante les aspects temporels des systèmes qui nous intéressent.

Objectifs

L'objectif principal de nos travaux est la représentation explicite de la composante temporelle au sein de systèmes multi-agents évolutifs. L'objectif secondaire est ensuite d'essayer de tirer profit de cette prise en compte du temps au sein des divers niveaux que nous différencions dans un système multi-agents pour mettre en œuvre une coordination temporelle. Nos travaux ont donc porté sur deux domaines très distincts avec d'une part, les *systèmes multi-agents* qui offrent des techniques pour développer des applications coopératives pour des systèmes décentralisés et ouverts, et d'autre part, le *raisonnement temporel* qui permet de modéliser le temps en informatique.

Les différents résultats ont été implémentés sur la plate-forme multi-agents existante (MAST) au sein du laboratoire SMA de l'ENSM.SE. Deux applications issues du milieu industriel illustrent ces résultats ainsi que le fonctionnement des outils développés et proposent des possibilités de validation de l'approche choisie.

Organisation du document

Le présent document est organisé en trois parties. La première est consacrée à un état de l'art relatif au temps dans les SMA. La seconde partie décrit nos propositions issues de l'analyse faite en première partie. Enfin la troisième et dernière partie concerne la réalisation de ces propositions.

Première partie : Temps dans les systèmes multi-agents

Dans cette première partie, nous nous intéressons à une problématique particulière qui est la prise en compte des aspects temporels dans les systèmes multi-agents.

Chapitre 1 : Systèmes multi-agents

Ce chapitre présente les systèmes multi-agents à travers l'approche *Voyelles* (appelée également *AEIO*) sur laquelle nous nous appuyons. Cette présentation est organisée selon deux visions que nous distinguons pour un même système multi-agents, une vision "macro" concernant le SMA lui-même et une vision "micro" au niveau de l'agent.

Nous complétons cette vision par une description plus précise des concepts d'interaction et d'organisation utiles pour appréhender la notion de coordination au sein d'un SMA que nous abordons ensuite.

Chapitre 2 : Raisonnement temporel
Nous consacrons ce chapitre à l'étude de la dimension temporelle en informatique. Nous présentons différentes représentations et techniques de raisonnement temporel. Ce chapitre présente notamment les notions temporelles évoquées ou utilisées dans le cadre de ce mémoire.

Chapitre 3 : Temps dans les systèmes multi-agents
Nous nous intéressons ici à la prise en compte du temps dans les SMA. Cette analyse porte sur des travaux existants et pour la plupart bien connus. Elle s'appuie sur l'approche *Voyelles* permettant de mettre en exergue un certain nombre d'aspects temporels propres à chacune des dimensions susceptibles d'apparaître au sein d'un SMA.

Deuxième partie : Modèle de SMA temporel

Nous présentons dans cette partie, notre modèle de SMA temporel permettant la prise en compte du temps dans les différentes dimensions d'un SMA, exhibées par l'analyse AEIO effectuée dans la partie précédente.

Chapitre 4 : Outils pour représenter et manipuler le temps
Ce chapitre présente les fondements sur lesquels s'appuient les autres chapitres. Nous décrivons pour commencer, le langage d'expressions temporelles permettant d'expliciter des contraintes liées au temps ; ce chapitre présente également un gestionnaire de relations entre intervalles (GRIS) qui nous permet ensuite de manipuler les différentes expressions temporelles explicitées.

Chapitre 5 : Interactions temporelles dans les SMA
Ce chapitre décrit le langage d'interaction temporel TACL permettant aux agents de communiquer en exprimant des contraintes temporelles ; il présente également une structure de protocoles temporels TIP et la gestion des conversations associées permettant notamment de contrôler de manière temporelle les communications entre agents.

Chapitre 6 : Organisations temporelles dans les SMA
Nous proposons également la prise en compte de la dimension temporelle au sein de l'organisation par le biais d'un langage de structures organisationnelles temporelles TOSL décrit dans ce chapitre.

Chapitre 7 : Modèle d'agent temporel : TAG
Nous proposons dans ce chapitre, un modèle d'agent temporel TAG offrant la possibilité de prendre en compte les différents aspects temporels et d'intégrer les modèles décrits dans les chapitres 5 et 6 précédents.

Troisième partie : Applications temporelles

Dans cette troisième partie, nous commençons en abordant l'implémentation du modèle d'agent et des outils développés. Nous présentons ensuite deux instanciations du modèle précédent : l'une dans un objectif de simulation et l'autre dans une optique de résolution distribuée de problèmes.

Chapitre 8 : Implémentation des modèles
Nous commençons cette partie en présentant une implémentation du modèle d'agent temporel (décrit au chapitre 7), des langages TACL et TOSL (décrits aux chapitres 5 et 6) ainsi que du Gestionnaire de Relations entre Intervalles GRIS.

Chapitre 9 : Gestion d'alliance d'ateliers d'impression
Ce premier exemple concerne le développement d'une infrastructure logicielle pour la gestion d'alliances et pour l'aide à la négociation et à l'exécution de contrats entre différents ateliers d'impression. Nous décrivons les aspects temporels de ce projet et donnons des exemples illustrant un fonctionnement temporel centré interaction.

Chapitre 10 : Simulation d'entreprise fonctionnant par projet
Ce deuxième exemple concerne la simulation d'une entreprise fonctionnant par projet. Ce chapitre décrit l'entreprise, son fonctionnement par projet ainsi qu'un exemple de projet issu d'un cas réel. L'instanciation du SMA temporel et des exemples sont ensuite présentés illustrant un fonctionnement temporel centré organisation.

Première partie

Temps dans les systèmes multi-agents

Introduction

L'objectif de cette partie est d'analyser la prise en compte du temps dans les systèmes muti-agents. Cette partie comportera trois chapitres :

- le premier chapitre décrit les systèmes multi-agents en mettant l'accent sur les aspects qui sont importants dans le cadre de notre problématique tels que l'approche *AEIO*, la coordination...
- le second chapitre est consacré à une présentation de l'étude de la dimension temporelle dans le domaine de l'Intelligence Artificielle. Ce chapitre s'attachera à présenter différentes techniques liées au raisonnement temporel permettant de représenter et de manipuler le temps.
- enfin, le dernier chapitre s'appuiera sur les deux précédents pour définir une grille d'analyse permettant d'étudier comment le temps est actuellement pris en compte dans les SMA. Cette analyse nous permettra de faire apparaître une vue globale de la prise en compte du temps au sein des systèmes multi-agents.

1

Systèmes multi-agents

Sommaire

1.1 Introduction

Avec le développement de nouvelles technologies de communication, la distance, l'éloignement entre différentes entités devant communiquer, ne se posent plus comme des problèmes insurmontables comme ils pouvaient l'être auparavant. Les systèmes industriels suivant évidemment cette évolution, se révèlent de plus en plus *complexes* et leurs composants deviennent de plus en plus *hétérogènes* et *distribués*. A ce stade d'évolution, il est devenu nécessaire de développer des applications informatiques coopératives capables de prendre en compte les caractéristiques de tels systèmes. Le domaine de recherche "systèmes multi-agents" (SMA) [Wa98] [BDa01] propose des modèles et des techniques bien adaptés pour modéliser et développer de telles applications.

1.1.1 Définitions

Comme dans de nombreux domaines, il est difficile d'obtenir une définition consensuelle. La définition reconnue au sein de notre laboratoire est la suivante : **"Un système multi-agents est un ensemble de processus autonomes organisés appelés *agents* plongés dans un environnement commun et capables d'interagir afin de réaliser des tâches complexes".**

Il faut reconnaître que cette définition est assez fortement connotée par les recherches menées au sein de notre laboratoire (voir section 1.2 pour plus de détails) aussi nous donnerons quelques autres définitions :

- *Research in multiagent systems (MAS) is concerned with the behavior of a collection of (possibly preexisting) autonomous agents aiming at solving a given problem*[1] [MCd96].
- un système multi-agents peut être défini comme *a loosely-coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities*[2] [DLC89].
- *On appelle système multi-agents (ou SMA), un système composé des éléments suivants [Fer95] :*
  1. *Un environnement* **E**, *c'est-à-dire un espace disposant généralement d'une métrique.*
  2. *Un ensemble d'objets* **O**. *Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans* **E**. *Ces objets sont passifs, c'est à dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*
  3. *Un ensemble* **A** *d'agents, qui sont des objets particuliers (*$\mathbf{A} \subseteq \mathbf{O}$*), lesquels représentent les entités actives du système.*
  4. *Un ensemble de relations* **R** *qui unissent des objets (et donc des agents) entre eux.*
  5. *Un ensemble d'opérations* **Op** *permettant aux agents de* **A** *de percevoir, produire, consommer, transformer et manipuler des objets de* **O**.
  6. *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.*

Nous remarquerons seulement à propos de ces définitions que la dernière est très orientée "conception" voire même "spécification de génie logiciel". L'embryon d'algèbre opérationnelle esquissé ici, laisse imaginer l'utilité d'un langage de spécification *multi-agent* qui permettrait éventuellement à la communauté de se mettre d'accord sur une définition suffisamment "ouverte" pour englober les particularités de chacun et suffisamment "coercitive" pour cadrer les notions utilisées. Cette

[1] La recherche dans les systèmes multi-agents concerne le comportement d'un ensemble d'agents autonomes (éventuellement préexistants) visant à résoudre un problème donné.

[2] Un réseau faiblement couplé de "résolveurs de problème" travaillant de concert pour résoudre des problèmes au-delà de leurs capacités individuelles.

volonté d'uniformisation est également présente dans les efforts de standardisation entrepris par la *FIPA* (*Foundation for Intelligent Physical Agents*, [FIP00]), organisme de normalisation qui vise à l'interopérabilité d'agents logiciels hétérogènes entre eux et également de systèmes à agents.

Le terme *agent* (ou *problem solver*) apparaissant dans ces définitions nécessite également d'être défini de manière précise. Le problème est identique à celui de la définition du SMA puisque les définitions sont également nombreuses et variées[3]. Là encore, l'objectif n'est pas de disserter sur chacune des définitions proposées par l'un ou l'autre pour trouver *la meilleure* mais plutôt de donner un aperçu de ce que peut-être un *agent*. Par conséquent, il nous paraît plus intéressant de donner quelques définitions et celle que nous retenons dans le cadre de notre travail.

Un agent peut donc être défini par :

- *un processus informatique ayant un unique mécanisme de contrôle et/ou d'intention* [BG88].
- *une entité réelle ou abstraite qui est capable d'agir sur elle-même et son environnement, qui dispose d'une représentation partielle de cet environnement, qui, dans un univers multi-agents peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec des autres agents* [Fer87].
- *a computer system, **situated** in some environment, that is capable of **flexible autonomous** action in order to meet its design objectives*[4] [JSW98].

Pour notre part, nous reprenons comme définition d'agent, celle énoncée dans [Boi01], qui décrit un agent comme :
"Un programme informatique autonome, plongé dans un environnement (médium commun aux agents du système) qu'il est capable de percevoir et sur lequel il est capable d'agir". La définition s'affinera avec la description de l'approche sur laquelle nous nous appuyons (cf. section 1.2 pour plus de détails).

Deux types d'agent sont classiquement différenciés :

Les agents dits "*réactifs*" qui possèdent uniquement des mécanismes basiques de réaction aux événements. Généralement, un unique agent de ce type n'est pas considéré comme *intelligent* mais le système constitué d'un ensemble de ces agents se révèle avoir un comportement émergent intelligent [DF94].

Les agents "*cognitifs*" ou "*délibératifs*" qui possèdent des connaissances, sont capables d'adapter leur réaction et de choisir un comportement parmi plusieurs possibilités. Ils peuvent ainsi poursuivre un ensemble de buts ou faire de la planification de leur tâches par exemple.

Nous trouvons aussi parfois les agents dits "*hybrides*" qui possèdent à la fois des

[3] Le lecteur intéressé pourra trouver dans [Ld95] [FG96] une revue de différentes définitions du terme agent.

[4] Un système informatique situé dans un environnement, qui est capable d'action flexible autonome afin de satisfaire ses objectifs conceptuels.

caractéristiques ou des comportements relevant des agents réactifs et délibératifs permettant éventuellement de passer de l'un à l'autre. Ce troisième type d'agents recouvre évidemment toutes les gradations possibles entre agent réactif et agent délibératif. Là encore, les définitions peuvent être sujettes à caution ; aussi, nous nous contenterons de préciser que **dans le cadre de notre travail, nous nous intéressons à des agents de type délibératifs**.

1.1.2 Domaines d'application et problématiques

Les systèmes multi-agents (modèles et techniques) sont appliqués à de nombreuses branches de l'informatique (robotique, systèmes experts, systèmes distribués, commerce électronique, télécommunications, Internet, etc.) et concerne notamment les systèmes industriels complexes comme entreprises en réseaux ou les entreprises virtuelles par exemple. Mais nous retrouvons également les SMA dans des applications plus spécifiques : en médecine [Doj94] [HRWHH89] [HJF95] [MCHR97], en contrôle de processus [All98] [Gab96] [PHR87], etc. Plus précisément, les SMA s'intéressent à des systèmes présentant généralement les caractéristiques (non exclusives et non exhaustives) suivantes :

- *distribution spatiale et fonctionnelle* : les différentes composantes du système ainsi que leurs fonctionnalités associées peuvent être situées à des endroits séparés.
- *décentralisation* : un contrôle centralisé permettant de gérer tous les traitements du système n'existe pas (ou n'est pas souhaité).
- *hétérogénéité* : les données qui sont traitées et les décisions qui sont prises peuvent concerner différents domaines complètement différents et indépendants les uns des autres.

et éventuellement :

- *ouverture* : des entités peuvent s'insérer ou se retirer du système à tout moment en cours de fonctionnement.

La complexité d'un système est généralement liée à sa taille et ainsi à la quantité d'interactions qui peuvent exister entre les différents éléments qui le composent. La distribution peut être un facteur qui complexifie (communication, localisation, etc) alors que la décentralisation généralement simplifie la modélisation ou l'exécution des traitements. Cependant ces deux facteurs sont à l'origine d'un thème central de recherches dans les SMA : la coordination d'applications réparties. Nous reviendrons sur ce sujet par la suite.
De manière générale, les grandes problématiques abordées par les SMA, sont la coopération [DL87] [DL91], la négociation [RG85] [ZR89] [Far00], le contrôle distribué comme la supervision de systèmes industriels [All98], l'apprentissage [Wei93] [HLS96], la planification distribuée [Pol86] [GK93] [Gab96], etc.

1.1.3 Objectifs de développement d'un SMA

Les SMA peuvent être utilisés pour poursuivre différents objectifs. Actuellement, nous en dénombrons trois principaux :

1. la résolution distribuée de problèmes (RDP) qui est issue de l'Intelligence Artificielle. En effet, certaines tâches nécessitent l'intervention d'un ensemble de spécialistes pour être résolues : le concept "Multi-Agents" permet de mettre en œuvre une solution logicielle adaptée pour ce type de problèmes où la distribution est justement l'une des clés de leur résolution .

2. la simulation décentralisée de systèmes complexes. Les SMA permettent notamment de prendre en compte les aspects liés à la distribution et la décentralisation (cf. section 1.1.2) et ainsi d'être très proche du modèle réel. Nous pouvons citer, par exemple, à ce sujet, le système *DASCh* (Dynamic Analysis of Supply Chains) [PSRC98] qui s'intéresse aux moyens de modéliser une chaîne de production.

3. l'intégration de systèmes préexistants comme par exemple *l'exploitation et la gestion de systèmes d'informations* [Fer97]. Avec le regroupement, l'évolution des entreprises ou de différents systèmes centralisés, des logiciels existants sont amenés à coopérer. Ainsi, la définition d'une infrastructure grâce à laquelle ils peuvent coopérer s'avère nécessaire. Cette infrastructure devra être capable de gérer et coordonner le fonctionnement simultané de différentes entités : les SMA par leurs méthodes et leurs techniques peuvent être vus comme un moyen bien adapté pour réaliser ce genre d'intégration.

Les deux premiers cas présentent des approches radicalement différentes puisque le premier postule la donnée d'un problème avec un système a priori inconnu alors que le second part de la donnée d'un système existant et opérationnel. Le troisième cas peut appartenir à l'une ou l'autre des approches selon le contexte :

- L'intégration de systèmes existants peut être vue comme le problème distribué à résoudre et il peut également être spécifié de *résolution distribuée de problème distribué* [Fer95].
- Le système est considéré comme existant et le SMA à réaliser est sensé reproduire aussi fidèlement que possible le fonctionnement de chacune de ses entités autonomes et ainsi être vu dans un esprit "simulation".

Les systèmes multi-agents relatifs aux applications décrites dans la troisième partie de ce rapport ont été développés chacun avec une finalité différente et permettront d'illustrer les deux premiers objectifs.

Comme nous l'avons vu, il existe différentes façons de définir un SMA et de présenter un SMA. Notre travail s'appuie sur une approche particulière que nous allons décrire à présent : *l'approche Voyelles.*

1.2 Approche Voyelles

L'approche *Voyelles* [Dem95] [Dem01] distingue 4 dimensions **Agent**(A), **Environnement**(E), **Interaction**(I) et **Organisation**(O) pour analyser, concevoir et décrire un SMA. Par la suite, nous nous référerons à cette approche *voyelles* également avec le terme *AEIO* qui dénote chacune des dimensions que nous allons décrire à présent.

1.2.1 Dimension Environnement

La dimension E (Environnement) concerne la représentation de l'environnement dans lequel sont plongés les agents. Les modèles manipulés dans cette dimension représentent et gèrent par exemple, l'ensemble des ressources, ainsi que les possibilités de perception et d'action disponibles pour un agent dans l'environnement. Comparativement aux autres, cette dimension a été peu étudiée hormis quelques travaux particuliers comme [Mag96], [DL93] ou [PSRC98]. Diverses raisons peuvent être à la base de ce désintérêt :

- Cette dimension est très dépendante du domaine d'application et de l'application elle-même. Il est donc très difficile d'en avoir une vue générale et de proposer des modèles d'environnements.
- Les applications étudiées et modélisées jusqu'à présent, possédaient un environnement très pauvre ou peu intéressant à modéliser. Les domaines où l'environnement est important voire fondamental, utilisent surtout des agents réactifs comme par exemple, en aménagement du territoire, écologie, éthologie.
- L'environnement parce qu'il est trop simple ou trop complexe, a plutôt été intégré dans le raisonnement interne de l'agent sous forme de connaissances, comme paramètre par exemple.

Nous reviendrons au chapitre 3 sur quelques exemples au sujet de l'environnement dans les SMA.

1.2.2 Dimension Interaction

La dimension I (Interaction) représente l'ensemble des possibilités et moyens offerts aux agents pour interagir. Les modèles manipulés dans cette dimension concernent à la fois les langages d'interaction et les messages associés, les protocoles d'interaction disponibles pour les agents, ainsi que les mécanismes liés aux conversations et à leur gestion. Ces notions étant fondamentales dans le cadre de notre travail, nous allons essayer d'expliciter ce que peut recouvrir ce terme d'interaction.

Nous commencerons par en donner une définition parmi le grand nombre que l'on peut trouver dans la littérature. Celle qui est proposée ci-après, reste générale mais a le mérite d'être axée sur les systèmes multi-agents. Ferber définit l'inter-

action comme *la mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques* [Fer95].

On peut distinguer deux classes d'interactions : avec ou sans communication. **Dans notre étude, nous nous intéressons plus précisément aux interactions avec communication.**

Niveaux de communication

Il existe essentiellement 3 types d'interaction avec communication :

1. Les *communications primitives* de type signaux. Dans celles-ci, nous retrouvons par exemple les travaux relatifs aux agents réactifs où l'interaction correspond à un comportement stimulus-réponse [Dro93] [DF94] [Dem95]. D'autres travaux [Geo83] utilisent les signaux uniquement à des fins de synchronisation. Le rôle des signaux est déterminé à la conception au sein des agents.
2. La *communication par échange de messages*, de plans. Ce type de communication, très utilisé en Intelligence Artificielle Distribuée (DVMT [LC83], Langages acteurs [Hew77]), tend avec les travaux actuels et les nouvelles techniques disponibles à s'orienter vers les communications sophistiquées (en complétant le message en lui-même ("brut") par des informations annexes plus complexes).
3. Les *communications sophistiquées.* Ces travaux issus au départ des recherches sur le traitement de la langue naturelle, sur le dialogue homme/machine et sur les intentions dans les communications sont maintenant repris dans de nombreux domaines. En effet, le développement des réseaux de communication et les possibilités croissantes offertes par les communications de type sophistiquées incitent des branches de l'informatique axées jusqu'alors sur des solutions "centralisées" à se tourner vers des solutions "distribuées". Plus concrètement, la fiabilité ainsi que le débit des media de communication actuels ne sont plus un frein au développement de solutions basées sur des systèmes communiquant de manière complexe.

Dans le cadre de ce travail, nous nous intéressons à ce dernier type de communication.

Langages

Les communications sophistiquées nécessitent l'élaboration d'un langage évolué. Les langages d'interaction sont basés pour la plupart sur la théorie des actes de langage que l'on présentera plus tard. Il en existe un grand nombre, chacun avec ses propres caractéristiques liées à son utilisation et au contexte applicatif dans lequel il sera utilisé. Pour illustrer cette diversité, nous pouvons en citer quelques uns :

1. le langage du système *ASIC* [BD94] qui offre une implantation des actes de langage dans le cadre d'un système multi-agents pour un système intégré de vision.
2. Le langage du système *DASEDIS* [BHS93] est également basé sur des états mentaux avec une forte étude sur la notion d'engagement et une approche très orientée "protocole".
3. *KQML* (Knowledge Query and Manipulation Language) [LF94] qui est l'un des langages dont on entend le plus parler. Les nombreux travaux qui le font évoluer ou qui s'en inspirent l'impose un peu comme le précurseur d'un standard. Il existe une évolution de KQML appelé COOL (COOrdination Language)[BF95] qui permet en plus de gérer des conversations.
4. ACL FIPA (*Agent Communication Langage*) [FIP00] s'appuyant sur *ARCOL* [Sad91] pour la sémantique et étendant *KQML*. Ce langage utilise des états mentaux issus de la logique BDI.

Comme nous le voyons, diverses solutions ont été imaginées et développées afin d'utiliser un langage riche, puissant ou rapide et simple...Une tentative d'uniformisation est en cours de réalisation par la FIPA [FIP00] avec le langage ACL mais la grande diversité des langages (KQML [LF94] [MLF96], ARCOL [Sad91], ASIC [BD94], COOL [BF95], etc) existants et leurs liens avec des applications ou des plates-formes internes aux laboratoires laissent penser que l'apparition d'un langage d'interaction universel n'est pas encore pour demain.

Protocoles

Ces langages permettent la mise en place de conversations qui font souvent apparaître des schémas typiques. Ainsi, à certains moments d'une conversation, certaines séquences de message sont attendues. Ces schémas typiques de messages sont appelés "protocoles" [FIP97]. Dans ces langages, l'enchaînement des messages est géré par des *protocoles d'interaction.*

Semblablement aux langages, on trouve une multitude de protocoles d'interaction qui ont été proposés lors de divers travaux sur l'interaction.

Afin de mieux comprendre, cette notion de protocole, nous allons présenter le "Fipa-ContractNet-Protocol" [FIP00] (cf. figure 1.1).

Ce protocole est spécifié en langage AUML (Agent UML) [BMO01]. Chaque côté correspond à un agent différent : la partie gauche correspond à un agent qui demande une information et la partie droite sont les réponses possibles de l'agent ayant reçu le message.

Dans ce protocole, nous constatons que l'envoi du message *cfp* ("call for proposal") pour proposer un contrat, ne peut être suivi en retour que de trois possibilités :

1. *not-understood* si l'agent recevant le message ne peut en déchiffrer le sens,

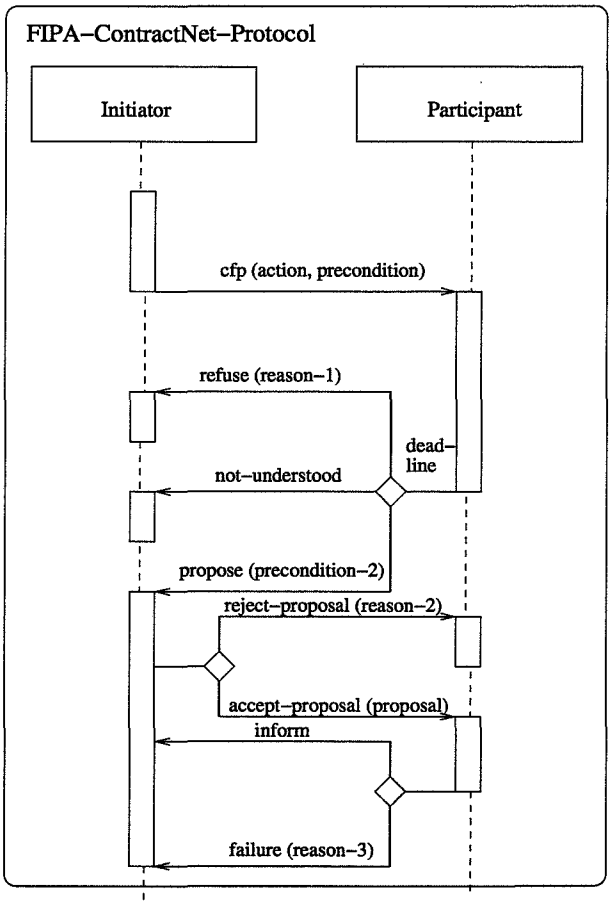


FIG. 1.1 – FIPA-ContractNet Protocol

2. *propose* si l'agent fait une proposition avec de nouvelles préconditions.
3. *refuse* si l'agent recevant le message refuse de répondre avec la raison pour laquelle il refuse.

On peut citer divers protocoles mis en place selon des objectifs bien spécifiques : la coopération, l'échange de connaissances [CD90] ou encore la négociation avec, par exemple, le "contract net protocol" [Smi80].

Comme nous le verrons, langage et protocole sont étroitement liés mais dans le cadre de notre travail, nous nous sommes focalisés principalement sur l'aspect langage.

Le paradigme majeur que l'on retrouve dans les communications avec les systèmes multi-agents est la *théorie des actes de langage*. En effet, cette théorie permet une modélisation précise de l'interaction par communication.

Théorie des actes de langage

La plupart des travaux existants cités précédemment prennent pour base une théorie de la communication appelée *théorie des actes de langage* (ou "Speech Acts

Theory") [Sea69a]. Cette théorie, très utilisée, reprise et enrichie par le monde informatique bien après sa formulation, est l'objet et l'origine de nombreuses recherches actuelles. Elle mérite donc d'être présentée ou rappelée.

Historique En 1962, le philosophe Austin se penche sur l'énonciation [Aus62] ("le fait de produire un énoncé") et définit celle-ci comme un acte qui sert avant tout à produire des effets sur son destinataire. Les approches antérieures du langage se bornaient à l'étude de la véracité d'une proposition alors que par les actes de langages, Austin désigne l'ensemble des actions intentionnelles effectuées au cours d'une communication. La communication est donc considérée comme une action dans laquelle on trouve un *locuteur* qui émet un message et un *allocutaire* qui le reçoit. Cet acte de communication se situe dans un contexte de communication.

Austin et ses successeurs ([Sea72], [Van88]) définissent un acte de langage comme une structure complexe composée de trois composantes que nous allons à présent détailler.

Composantes d'un acte Un acte de langage est défini comme la réunion de trois composantes qui sont :

1. La *composante locutoire* qui correspond au mode de production (oral, écrit, etc). Dans le cadre de notre travail, nous ne nous étendrons pas plus sur cet aspect considérant que tout message est transmis au travers d'un réseau de communication.
2. La *composante illocutoire* représente l'intention de l'acte effectué par le locuteur sur le destinataire de l'énoncé. Elle est généralement exprimée sous la forme **F(p)** avec **F** symbolisant la *force illocutoire* et **p** représentant le *contenu propositionnel.* Le type d'acte est souvent marqué par un verbe appelé performatif. Par exemple, la phrase "Ferme la porte!" peut être exprimée par "Commander(Fermer la porte)" dans laquelle "commander" est le performatif et "fermer la porte" correspond au contenu propositionnel. De même, les phrases "Il fait beau." et "Fait-il beau ?" ont le même contenu propositionnel mais pas le même performatif : "affirmer" dans le premier cas et "questionner" dans le deuxième.

   Cette composante a été particulièrement étudiée et définie de manière beaucoup plus pointue par d'autres chercheurs ([Sea72], [SV85], [Van88], [BB81]) qui ont repris les travaux d'Austin. Nous présenterons une partie de ces travaux dans le paragraphe suivant.
3. La *composante perlocutoire* porte sur les effets que les actes illocutoires peuvent avoir sur l'état du destinataire, ses actions, ses croyances et conduire à une modification de l'environnement dans lequel le locuteur et l'allocutaire sont immergés.

   Ainsi le performatif "persuader" vise à modifier un des états mentaux de

l'allocutaire. Dans l'exemple précédent "Commander(Fermer la porte)", le locuteur s'attend à ce que l'allocutaire ferme la porte.

Composante illocutoire et composante perlocutoire Un acte de langage est une intention d'action impliquant que les mots véhiculés par le message décrivent un état qui doit être mis en relation avec le monde. Cinq directions d'ajustement ont été définies afin de représenter l'intention de faire une correspondance entre les mots et le monde. Un classement des verbes correspondant à chaque direction d'ajustement, a été proposé ([Sea79], [SV85] puis [Van88]). Nous trouvons donc 5 catégories (assertifs, directifs, promissifs - ou engageants, expressifs et déclaratifs). Afin de mieux comprendre, nous allons nous arrêter quelques instants sur ces 5 classes :

Une proposition contenant un verbe ayant le but :
- **Assertif** décrit un état du monde (ajustement des mots avec le monde). Ex : Dire que la porte est fermée.

- **Directif** consiste à faire une tentative linguistique pour amener l'allocutaire à faire une action future (ajustement du monde aux mots). Ex : Demander de fermer la porte.

- **Promissif** vise à engager le locuteur à faire une action future (ajustement du monde aux mots à la différence que c'est le locuteur qui va réaliser l'ajustement). Ex : Promettre de fermer la porte.

- **Déclaratif** permet d'accomplir une action par le seul fait de l'énonciation ; c'est la raison pour laquelle ce but a la double direction d'ajustement. Ex : Déclarer la séance ouverte.

- **Expressif** décrit des états mentaux du locuteur et a la direction d'ajustement vide. Ex : Aimer la couleur bleue.

Cependant cette liste suscite beaucoup de critiques (certains verbes allemands ne peuvent être classés facilement [BB81]) et n'est donc pas considérée comme définitive. A titre d'exemple, on trouve parfois les verbes directifs divisés en 2 parties : interrogatifs et exercitifs [Fer95].
Dans notre étude, nous n'avons pas jugé utile de faire cette distinction et nous nous sommes limités au classement en 5 catégories citées précédemment. Par ailleurs, dans ces 5 catégories nous nous sommes uniquement intéressés aux trois premières.

La composante perlocutoire prend deux aspects selon que l'on se place du côté du locuteur ou de l'allocutaire :
- du côté du locuteur, une attente sur l'état du monde et sur l'allocutaire est exprimée : on attend la réalisation d'une action future pour un acte directif

ou un nouvel état mental de l'allocutaire pour les autres types d'actes : assertif, engageant, déclaratif ou expressif. Il faut donc être capable d'exprimer ces attentes afin d'obtenir le résultat souhaité.
- du côté de l'allocutaire, il doit être capable de reconnaître l'intention du locuteur pour produire l'effet escompté. De plus, la reconnaissance de l'acte peut permettre à l'allocutaire de déduire certaines croyances du locuteur et donc d'enrichir ses états mentaux (cf. section 1.2.4).

La composante perlocutoire est souvent représentée par le biais de *conditions de succès* qui permettent à l'allocutaire de reconnaître sans ambiguïté l'intention du locuteur et de *conditions de satisfaction* qui vont permettre au locuteur de raisonner sur l'état du monde résultant de l'énonciation de son acte et de vérifier s'il a été satisfait [CdV98] [Van88].

Actes de langage en SMA

La théorie des actes de langage a été source d'inspiration pour le développement et l'implantation de différents langages d'interaction. Evidemment, cette théorie a suscité également nombre de critiques. L'une des critiques formulée à juste titre, est qu'elle ne considère que l'acte isolé et ne traite absolument pas de la séquence d'interactions entre les intervenants d'une discussion [Bra93]. Cohen et Levesque pensent que l'important est la reconnaissance de l'intention du locuteur et non la reconnaissance de la force illocutoire d'un acte de langage pour la communication [CL90b] [CL90c]. C'est la raison pour laquelle les méthodes définies à partir de la théorie des actes de langages pour les langages d'interaction permettent de *représenter de façon explicite l'intention dans la communication.*

L'interaction dans les systèmes multi-agents prend de multiples formes. Comme on l'a vu, il existe différents axes de recherche et diverses solutions ont été proposées. Nous reviendrons sur des exemples de langages d'interaction au chapitre 5. Dans le cadre de notre travail, nous nous sommes focalisés sur des communications sophistiquées avec notamment le développement d'un langage basé sur la théorie des actes de langages.

1.2.3 Dimension Organisation

La dimension O (Organisation) est liée aux moyens de structuration du SMA incluant les règles de fonctionnement et de comportement, les relations entre les agents.

En effet, beaucoup de recherches dans le domaine de l'organisation se sont inspirées des phénomènes observés dans les sociétés humaines pour définir et mettre en place une structure organisationnelle. Nous retrouvons ainsi à des degrés divers (analogie avec la société humaine) de nombreux termes bien connus. Nous allons donc préciser la (ou les) définition(s) que nous reconnaissons concernant le terme organisation :

*L'**Organisation** regroupe les représentations et les mécanismes nécessaires à la structuration des agents dans le système.* On peut définir également l'organisation comme l'ensemble des relations entre agents dans lesquelles nous distinguons les liens issus de la *structure organisationnelle* [HBSS00] [PCL87] et les dépendances calculées [CMC92b] [All98] [SD95] [LCd97] à partir des actions, ressources, plans et buts qu'ont chacun des agents par exemple.

Relations de dépendances

Ces dernières années, de nombreux travaux ont été menés sur la construction de mécanismes de raisonnement social dans les SMA et plus précisément sur la *théorie des dépendances* [CMC92b]. Une relation de dépendance existe entre deux agents si pour atteindre son but l'un des agents doit nécessairement avoir recours aux services offerts par le deuxième agent. En réalité, les relations peuvent être relatives aux *agents* mais aussi aux *tâches* et aux *ressources* définissant ainsi des dépendances *tâche-tâche, agent-tâche, ressource-agent, tâche-ressouce, etc* [All98] [Dec96] [LCd97]. L'étude des relations de dépendances a été complétée [CMC92a] en distinguant de nombreux qualificatifs : et-dépendance (plusieurs actions nécessaires pour atteindre un but ou plusieurs agents différents doivent intervenir par exemple), ou-dépendance (alternatives de partenaires ou d'actions), unilatérale , bilatérale (deux agents dépendent l'un de l'autre par rapport à un même but ou non), négative (entre tâches : la réalisation de l'une empêche la réalisation de l'autre), positive, etc. Ainsi pour entreprendre une action, les agents peuvent avoir d'abord besoin de déterminer les relations de dépendances existant entre eux afin de déterminer leur degré d'*autonomie* [Cas90]. Ces relations constituent un moyen de mettre en place un raisonnement social et sont donc importantes pour la coopération.
Un autre moyen de prendre en compte l'organisation est de s'appuyer sur une structure organisationnelle au moyen d'un modèle organisationnel.

Modèle organisationel MOISE

Nous allons présenter ici le modèle organisationnel MOISE sur lequel nous nous appuyons. Ce modèle [HBSS00] distingue trois niveaux pour représenter l'organisation :

1. le *niveau individuel* qui permet de spécifier *ce* qui doit être fait par un agent en introduisant la notion de *rôles*. Ces rôles permettent de spécifier les tâches et les responsabilités de l'agent. Les rôles sont définis par les caractéristiques des problèmes auxquels nous nous intéressons. Ces caractéristiques que nous retrouvons à plusieurs niveaux d'un SMA (agent, interaction, organisation, etc) sont les buts à réaliser (*goals*), les plans à suivre (*plans*) et les actions et ressources qui apparaissent dans ces plans. La notion de tâche (*task*) recouvre un ensemble composé d'un but, d'un ou plusieurs plans (avec les actions et ressources associées) qui permettent d'accomplir ce but. Le concept

de mission est associé à une tâche qui doit être accomplie ; ainsi un rôle est un ensemble de missions.

2. le *niveau social* qui spécifie *comment* et *avec qui* interagir. Cela est réalisé en définissant des liens organisationnels qui correspondent à des relations de coopération entre les agents. Un lien organisationnel est une relation orientée entre deux rôles (rôle source et rôle cible). Il existe trois types de liens : les liens d'autorité, de communication et d'accointances qui permettent notamment de contraindre et limiter les échanges entre les membres de l'organisation.
3. le *niveau collectif* qui permet de regrouper des agents dans la structure en définissant des groupes et ainsi spécifier *qui* regrouper ensemble.

Ainsi la *Structure Organisationnelle* peut être définie comme un graphe dont les liens et les rôles sont respectivement les arcs et les nœuds. Cette structure instanciée sur un SMA réel avec des agents associés aux rôles permet de définir l'*Entité Organisationnelle* (*ensemble des agents sur lequel une* structure organisationnelle *est imposée*). Découlant des différents types de liens, on définit un graphe de contrôle avec les liens d'autorité, un graphe de communication avec ceux de communication et un graphe d'accointances à partir des liens d'accointances. Ces notions permettent de spécifier de manière explicite ce qui doit être fait, comment et avec qui. La sémantique du modèle MOISE peut-être trouvée dans [Han02].

Nous présenterons au chapitre 6.3, nos travaux qui s'appuient sur ce modèle ; c'est pourquoi nous introduisons brièvement ici la syntaxe utilisée pour définir une structure organisationnelle dans ce modèle :

⟨os⟩ : := (os :name ⟨osname⟩ :roles (⟨rname⟩*) :links (⟨lname⟩*)
:groups (⟨gname⟩*))

Les rôles organisationnels sont un ensemble de missions obligatoires (O) ou permises (P).

⟨role⟩ : :=(role :name ⟨rname⟩ :missions ((O | P ⟨mname⟩)*)

Une mission est définie comme un ensemble de permissions sur les buts, plans, actions ou ressources correspondants :

⟨mission⟩ : :=(mission :name ⟨mname⟩ :goals (⟨goal⟩*) :plans (⟨plan⟩*)
:actions (⟨action⟩*) :resources (⟨resource⟩*))

Trois types de liens organisationnels sont utilisés afin de définir : la structure de contrôle (*authority* links (Aut)), la structure d'échange de données (*communication* links (Com)) et la structure d'accointances (*acquaintance* links (Acq)).

⟨link⟩ : :=(link :name ⟨lname⟩ :type ⟨type⟩ :source ⟨context⟩ :target ⟨context⟩
:constraint ⟨constraint⟩)
⟨type⟩ : := Aut | Com | Acq
⟨constraint⟩ : := Protocols | Acts
⟨context⟩ : :=(:role ⟨rname⟩ :missions (⟨mname⟩*))

En fonction du type de liens, il est possible de définir des contraintes en termes de protocoles ou d'actes sur la manière dont les agents jouant le rôle *source* et le rôle *target* (cible) interagiront. Un groupe est défini par :

1. un ensemble de *rôles*,
2. un ensemble de *missions* appartenant aux missions qui définissent ces rôles et qui peuvent être activées dans le contexte du groupe,
3. un ensemble de *liens* qui est un sous-ensemble de liens dont les rôles "source" et les rôles "cible" appartiennent à l'ensemble des rôles prédéfinis ci-dessus (dans la définition du groupe).

⟨group⟩ : :=(group :name ⟨gname⟩ :roles (⟨rname⟩*) :links (⟨lname⟩*)
:missions (⟨mname⟩*))

Un exemple concret de structure organisationnelle relatif à une application sur lequel nous travaillons, sera donné au chapitre 6.3. De plus, nous reviendrons plus précisément sur les aspects organisationnels qui nous intéressent en section 3.3.

1.2.4 Dimension Agent

La dimension A (Agent) désigne l'ensemble des fonctionnalités de raisonnement interne d'un agent. Elle peut concerner les méthodes de planification, d'introspection, ainsi que celles relatives au raisonnement sur ses propres compétences et connaissances.

Les méthodes, techniques et modèles existants sont nombreux et généralement adaptés en fonction du contexte. Il n'existe pas de modèle ou de concept uniformément accepté sur la dimension agent. Cependant le modèle le plus connu et le plus utilisé, servant de base à de nombreux travaux est sans aucun doute le modèle *BDI* [RG95b]. Pour clarifier les idées au sujet de ce qui peut être regroupé derrière la dimension Agent, nous allons décrire ce modèle dans ce qui suit. D'autres travaux plus spécifiques seront présentés au chapitre 3.

Le modèle BDI

Dans une architecture BDI, l'état d'un agent est représenté par trois structures appelées *attitudes mentales* : ses croyances (*Belief*), ses désirs (*Desire*) et ses intentions (*Intention*) :

1. les croyances correspondent à son modèle du domaine : les informations qu'il possède.
2. les désirs définissent ses motivations et sont aussi appelés *goals*.
3. les intentions regroupent les états délibératifs de l'agent : ce qu'il a décidé de faire.

Les intentions des agents peuvent être définies à différents niveaux d'abstraction et sont éventuellement raffinées jusqu'à aboutir à des actions primitives ; c'est à dire directement exécutables. Les attitudes mentales sont fondamentales car elles déterminent le fonctionnement de l'agent et donc du système. La particularité du modèle est de mettre ces trois notions au même niveau. Un agent *BDI* fonctionne de la manière suivante (cf. figure 1.2) : une liste de possibilités (*options*) est générée afin de satisfaire les intentions courantes et vient s'ajouter à celles préétablies à partir des croyances et des désirs de l'agent. Les intentions représentent le sous-ensemble de ces *options* qui ont été sélectionnées pour adoption ; toujours sur la base des croyances et des désirs de l'agent. Enfin lorsqu'une action atomique au niveau de cette structure intentionnelle peut être réalisée, elle est exécutée. Si une intention est satisfaite ou ne peut plus l'être, elle est abandonnée et les croyances sont mises à jour pour recommencer le cycle.

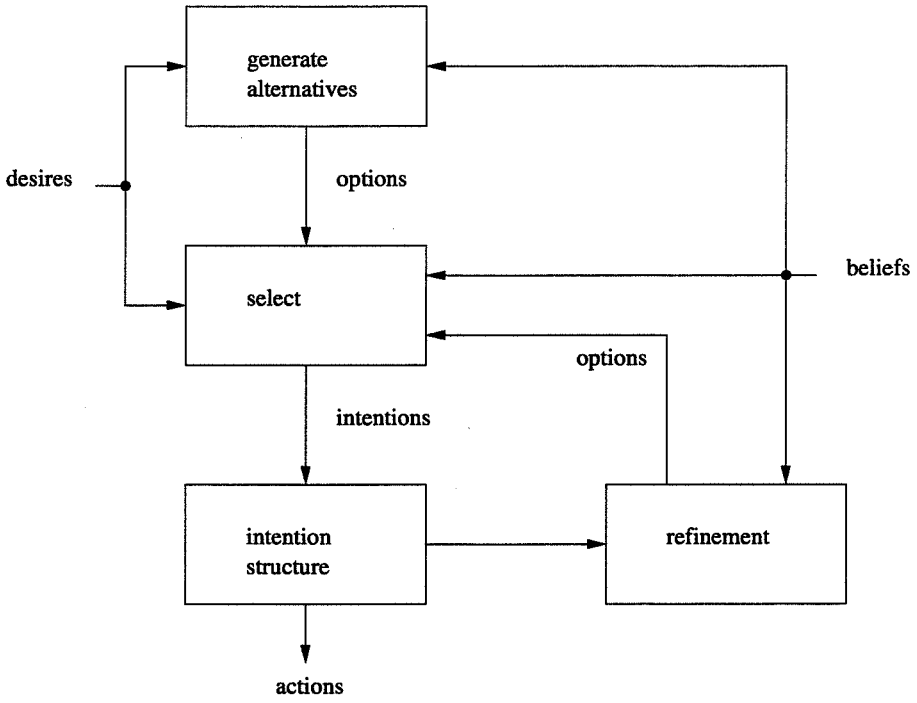


FIG. 1.2 – Architecture BDI selon [NL96a].

Un schéma de l'architecture BDI comparée à l'*architecture d'agent motivé*, est proposé dans [NL96a]. Dans cet exemple, le modèle est un peu plus complexe car il fait apparaître les différentes étapes de raffinement successifs qui permettent d'aboutir à des intentions simples et des actions directement exécutables.

Cette architecture a connu un certain nombre de critiques auxquelles les auteurs ont répondu dans [RG95a]. Ces critiques portaient sur l'intérêt en pratique de distinguer ces trois niveaux ou au contraire sur l'insuffisance de trois niveaux pour formaliser l'état d'un agent ou encore sur l'utilité de logiques BDI multi-modales qui ne sont pas directement exécutable ou suffisamment efficaces en pratique. Il existe d'autres théories et formalisations sur la notion d'intentions parmi

lesquelles nous pouvons citer [Bra87] [BIP88] [CL90a]. Nous compléterons cette vue de la dimension Agent en précisant que cette architecture est générale et que de nombreux travaux sont une spécialisation de cette architecture BDI. Comme nous l'avons dit précédemment, nous verrons au chapitre 3 quelques exemples illustrant ces propos.

Il existe une grande diversité d'architectures d'agents proposées. S'appuyant sur l'approche *Voyelles*, trois types d'architectures sont différenciées dans [Boi01] :

1. l'*architecture d'agent autonome* qui correspond à un agent possédant des capacités d'action et de perception sur son environnement mais aucune capacité explicite de coopération avec d'autres agents.
2. l'*architecture d'agent interagissant* qui fournit en outre, à l'agent des capacités d'interaction avec les autres agents du système.
3. l'*architecture d'agent social* qui complète l'architecture d'agent interagissant avec des capacités de gestion des relations entretenues avec les autres agents du système.

Des architectures existantes ont ainsi été classées en s'intéressant d'une part aux capacités offertes aux agents comme décrit ci-dessus et d'autre part à la dimension de raisonnement qui sépare agent réactif, délibératif ou hybride (cf section 1.1.1). Le classement de ces architectures ainsi que de plus amples informations sur les architectures d'agents peuvent être trouvés dans [Boi01].

Dualité SMA/Agent

Comme nous le voyons, l'approche *Voyelles*, en plus de posséder d'indéniables qualités méthodologiques (voire même pédagogiques) permet d'avoir une vue globale du système multi-agent en mettant en lumière successivement chacun des aspects importants du SMA. Nous complétons cette vision en précisant qu'il est possible de se placer à deux niveaux pour étudier un SMA ; au niveau du système lui-même : le SMA ou bien au niveau de l'entité de base qui agit : l'agent. Nous allons ainsi voir qu'il est possible d'affiner cette approche en spécifiant ces deux niveaux.
Lorsque l'on s'intéresse à un système multi-agent suivant l'approche *AEIO*, chaque dimension est considérée au niveau du SMA. Cependant, nous pouvons voir derrière le terme Agent : la dimension A ou bien l'entité autonome et ses spécificités propres (capacités de raisonnement, langages et protocoles d'interaction utilisés, connaissances organisationnelles, capacités de perception, etc). Nous constatons donc qu'il est possible de retrouver les composantes *AEIO* à l'intérieur même de l'agent (entité autonome) [Boi01]. Afin de lever toute ambiguïté, nous appelerons *facettes* les composantes *AEIO* que nous pouvons distinguer *au sein d'un agent*, et réservons *dimensions* pour les composantes AEIO que nous situons *au niveau du SMA*[5]. Nous noterons les facettes par les lettres minuscules a, e, i et o pour

[5]Cette vision facettes est une vision d'analyse et de conception du niveau agent. Elle peut déboucher sur de multiples implémentations comme montré dans [Boi01]. Actuellement, rares sont

bien les distinguer des dimensions d'analyse du niveau SMA.

Facettes Agent, Interaction, Organisation et Environnement

- La facette *a* (agent) représente la partie Agent au sein de l'agent (entité logicielle). Pour des raisons évidentes, nous l'appellerons dans la suite *raisonnement interne*. Sa définition est la même que pour la dimension (cf. Dimension Agent) mais concernera cette fois un agent ou un modèle d'agent particulier. Cette nuance est particulièrement utile dans le cas d'un système multi-agent avec des agents *hétérogènes* et donc possédant des mécanismes de raisonnement interne différents les uns des autres. Cette dernière précision est évidemment valable pour les connaissances et mécanismes appartenant à chacune des autres facettes.
- La facette *e* (environnement) concerne tous les mécanismes de perception et d'action que l'agent est capable d'utiliser pour interagir avec l'environnement exclusivement.
- La facette *i* (interaction) regroupe tout ce que l'agent est à même de mettre en œuvre pour interagir avec les autres agents. Cela rassemble les langages d'interaction et les protocoles dont il dispose, ainsi que les mécanismes de gestion de conversations.
- La facette *o* (organisation) rassemble les mécanismes et les connaissances de fonctionnement et de comportement vis à vis des autres agents.

Nous pouvons donc voir un agent comme un ensemble de facettes qui coopèrent et se coordonnent soulignant ainsi le problème du *contrôle des facettes*.

1.3 Coordination dans les SMA

L'aspect *coordination* tel qu'il est évoqué généralement en SMA s'intéresse au système global (relatif aux problématiques spécifiques d'un SMA : décentralisation, distribution, cohérence, etc). Comme nous l'avons déjà signalé en introduction, la *coordination* est un problème important du domaine SMA [DL95] [DL87] [Jen93]. Nous allons, pour commencer, essayer de définir ce terme.

1.3.1 Définitions de la coordination

Dans le langage courant, le terme *coordination* correspond à *l'agencement des éléments pour obtenir un ensemble cohérent* selon le dictionnaire *Larousse*, ce qui, rapporté à un SMA peut se décliner sous la définition suivante :
"Spécification du comportement et définition de la structure et des interactions des

celles qui correspondent directement à ces facettes. Nous pouvons citer celle proposée récemment dans [Ric01] avec les notions de briques et inter-briques. Dans le chapitre 8, nous proposerons également une implémentation de ces facettes dans l'architecture d'agent TAG.

agents afin que le système reste toujours dans un état cohérent, c'est à dire qu'ils concourent de manière générale à la réalisation d'un ou plusieurs buts communs". Dans les applications des SMA, des dépendances de différentes natures existent entre les modules qui composent le système : accès simultané à une imprimante ou à une entité dans une base de données (partage de ressources), capacité de réaliser un ordonnancement dans un module, capacité de lancer des ordres de fabrication dans un autre module (partage de compétences), objectifs complémentaires ou incompatibles (partage d'objectifs), par exemple. Parmi différentes définitions existantes, nous retiendrons la suivante : la coordination correspond à l'activité de gestion de ces différentes dépendances [MC91].
Selon la manière dont elle est assurée, la coordination peut conduire à différents modes de fonctionnement comprenant, entre autres, la *coopération*[6], la *compétition* ou la *collaboration.* L'expression notamment de cette coordination est un problème crucial et de nombreux moyens ont été développés permettant de mettre en place cette coordination.

1.3.2 Moyens de mise en œuvre

Les moyens de mise en œuvre de la coordination peuvent être aussi divers que le respect d'une structure hiérarchique, la mise en place de réglementation (règles de comportement des entités du système) ou même la spécification d'une entité spécialisée appelée généralement *coordinateur* qui se chargera de gérer cette coordination. Ferber distingue quatre formes de coordination d'actions [Fer95] qui sont la synchronisation, la planification, la réglementation et la coordination réactive qui est mise en pratique justement par des agents de type réactifs capables de réagir rapidement. Plus récemment, on peut trouver dans [LCd97] un état de l'art sur la coordination aboutissant à la description d'une approche basée sur la gestion des dépendances entre activités. Avec ces différentes formes de coordination, de nombreuses techniques (et à divers niveaux d'un SMA) ont été développées (BDI [RG95b], communication explicite, stigmergie[7] [BCBK99], blackboard [HR85], [GHJ+86], [Dec91], etc).

La diversité des aspects recouvrant la notion de coordination a fait apparaître la nécessité de mieux la caractériser. Ainsi par exemple, dans [Fer94] deux perceptions de la *coordination* sont distinguées :

- au *niveau externe* si elle est indépendante de l'intention des membres de la société d'agents et ainsi perçue par un observateur extérieur, de manière *émergente.*

[6]Selon les auteurs, les termes *coordination* et *coopération* sont parfois utilisés l'un pour l'autre ; dans ce mémoire, nous considérons que nous utiliserons *coordination* comme terme générique.

[7]Le nom *stigmergie* est utilisé en biologie pour "décrire l'influence sur le comportement d'effets environnementaux persistants provenant de comportements antérieurs" [Gra59].

- au *niveau interne* quand la coopération est une attitude intentionnelle des agents qui cherchent à résoudre un problème de manière distribuée.

Dans ce mémoire, nous nous limiterons à présenter une perception de la **coordination au niveau interne** (définie précédemment) en considérant une couche de contrôle propre à chacun des agents : **ce sont les agents qui se coordonnent.**
Lors de la résolution d'un problème global, les agents accomplissent un certain nombre de tâches qui les concernent, constituant une résolution partielle du problème global. Il peut arriver que l'accomplissement de ces tâches individuelles sans concertation avec les autres agents ait un effet négatif sur la résolution du problème global. Les agents doivent donc d'une part *interagir* afin de coordonner leurs tâches et d'autre part *respecter une organisation.* Cette remarque souligne l'intégration de la coordination *au sein de plusieurs dimensions AEIO* et donc *la nécessité de différents outils dédiés à chacune des dimensions.*
La définition de ces outils dépend des caractéristiques des systèmes qui nous intéressent. Comme nous l'avons dit, ces systèmes sont (1) *distribués* : chaque entité doit donc pouvoir prendre connaissance des actions des autres pour se coordonner avec ces dernières; (2) *décentralisés* : chacune des entités agit de manière autonome quitte éventuellement à devoir négocier certaines de ces actions pour maintenir cette coordination; (3) *ouverts* : la coordination consiste également à pouvoir prendre en compte le fait que des entités peuvent s'insérer ou disparaître du système.

1.4 Discussion

Dans ce chapitre, nous avons décrit brièvement notre vision du domaine SMA, vision particulière puisqu'elle s'appuie fortement sur l'approche *Voyelles* : nous distinguons quatre dimensions dans un SMA qui sont **l'agent, l'interaction, l'organisation** et **l'environnement**. Les dimensions *Agent, Organisation* et surtout *Interaction* ont été décrites plus précisément car elles présentent des théories et des techniques qui sont utilisées en deuxième partie. Nous avons complété cette approche en y distinguant les niveaux *SMA* et *agent.* Pour synthétiser notre contexte de travail, nous rappelons que nous nous intéressons à des systèmes multi-agents distribués, décentralisés, ouverts et dynamiques dont les agents sont de type délibératifs et hétérogènes[8].

L'accent a été également mis sur la problématique de la coordination. Nous avons vu qu'elle peut être considérée comme transversale à l'approche *Voyelles* puisqu'elle peut être exprimée à l'aide de différentes dimensions.
Nous n'avons pas encore évoqué l'aspect dynamique des systèmes que nous étudions mais dès à présent, son lien avec la problématique de la coordination est facile à comprendre : le choix du moment pour effectuer une action n'est pas ano-

[8]Les agents ne sont pas obligatoirement semblables : ils possèdent chacun des capacités ou des fonctionnalités propres, par exemple.

din par exemple. Au contraire, la place et l'influence de la composante temporelle dans un SMA est moins évidente à appréhender mais elle sera analysée en détails dans le chapitre 3 grâce à l'approche *Voyelles*. Pour le moment, l'omniprésence des aspects temporels dans les SMA dits *évolutifs* ou *dynamiques*, nous incite à nous intéresser plus précisément à la dimension temporelle en informatique.

2

Raisonnement temporel

Sommaire

2.1 A propos de la dimension temporelle

Il est une dimension incontournable qui nous domine et même nous emprisonne, omniprésente dans notre vie, nous servant tour à tour de référence globale, locale ou commune à la société : **le temps**. Depuis l'antiquité, la philosophie s'est attachée à essayer de définir, représenter et comprendre le concept temporel par l'intermédiaire de Platon qui voyait le temps comme *image mobile de l'éternité* [Pla60], Saint-Augustin qui constatait : *Qu'est-ce que le temps, ? Si personne ne me le demande, je le sais ; mais si on me le demande et que je veuille l'expliquer, je ne le sais plus* [SA97] ou même Héraclite avec le concept de *mobilisme universel* [Her50]. Plus contemporains, nous pouvons également citer Bergson [Ber07], Heidegger [Hei27] et Kant [Kan81] parmi les nombreux philosophes qui se sont

penchés sur cet épineux problème. En fait, les scientifiques de toute discipline sont confrontés au temps. Le mathématicien Pythagore n'hésitait d'ailleurs pas à conseiller : *Avec de l'ordre et du temps, on trouve le secret de tout faire et de tout bien faire...* Le temps est également une des quantités fondamentales du monde physique, qui pour sa part, propose un milieu indéfini, homogène, un temps unique et absolu dans ces différentes disciplines. Que ce soit en astronomie où les cosmologistes sont en quête perpétuelle du début -voire de la fin- du temps, en mécanique avec la définition de Newton : *Le temps absolu, vrai et mathématique, en lui-même et de sa propre nature, coule uniformément sans relation à rien d'extérieur* [New87], ou bien en thermodynamique, en cinétique, en physique quantique ou nucléaire [Ein21], la Physique a dû, dans ses différentes spécialités, apprendre à composer avec cette dimension particulière. Cet état de fait se retrouve dans la plupart des domaines sujets à étude et la Littérature, elle-même, (au sens propre du terme), témoigne de la fascination et du pouvoir qu'exerce cette dimension sur le monde. Pour ne prendre qu'un exemple : le thème de la fuite du temps est à l'origine de nombreuses œuvres poétiques (*Le lac* de Lamartine [Lam49], *Le goût du néant* de Beaudelaire [Bea57], etc.) ou littéraires avec La Bruyère [LB89] ou Proust [Pro13].

De la même manière, dans le domaine particulier qui nous intéresse, l'étude du temps est bien plus récente. En effet, si, comme dans certains domaines cités précédemment, le temps a d'abord été nié, l'*étude du temps en informatique* se révèle très riche et toujours d'actualité. Diverses raisons peuvent être données pour justifier cette négation première :

1. besoin inexistant dû à l'étude de certaines classes de problèmes uniquement que l'on pourrait qualifier de "statiques" ou "intemporels".

2. difficulté de modélisation liée à la limitation des machines.

Dans ce chapitre, nous verrons ainsi les problématiques relatives à l'étude du temps en informatique, en nous intéressant aux définitions et aux aspects théoriques. Nous ferons la distinction pour cela entre *Représentation du temps* et *Raisonnement Temporel.*

2.2 Etude du temps en informatique

Malgré un certain nombre de limites (cf. section 2.1 ci-dessus), la nécessité de prendre en compte des aspects de plus en plus complexes (et notamment les aspects dynamiques), a permis de faire émerger des travaux sur le temps. En l'occurrence, ces travaux sont issus de l'informatique dite *temps réel* [AVSHR90], des systèmes répartis [Lam78], du parallélisme [Pat96] [EAdC89], du multimédia (dans un domaine très récent) [LSI96] [Lay62] [DK95] et plus particulièrement de l'**Intelligence Artificielle** (IA) [Say90], [Hay95] .

2.2.1 Introduction à l'étude du temps en Intelligence Artificielle

Dans ce chapitre, nous nous limiterons à présenter différents moyens de caractériser le temps en IA. Il convient de préciser que certains aspects du temps en informatique nous éloignent de la notion de temps irréversible du monde réel comme certains travaux considérant plusieurs passés ou plusieurs futurs que nous évoquerons dans ce qui suit ; cependant il faut garder à l'esprit que les éventuelles entorses, conventions ou simplifications qui sont utilisées pour le représenter n'ont pour but que de nous permettre de manipuler plus aisément et même tirer profit (à des fins de : diagnostic, prédiction, simulation, etc., comme nous le verrons plus tard) de cette composante temporelle.

2.2.2 Représentation et manipulation du temps

L'étude de la composante temporelle est caractérisée par la diversité des problèmes à traiter et leur complexité. Omniprésent dans le langage naturel, de manière explicite ou implicite, le temps peut souvent être source d'ambiguïtés lorsque le contexte est peu ou mal défini. Nous commencerons donc par présenter et définir les termes plus ou moins couramment employés pour décrire les notions temporelles. Dans ce rapide tour d'horizon, nous essaierons de faire référence aux travaux clés dans le domaine. Le lecteur intéressé pourra également se reporter aux articles cités dans [Say90] et [Hay95].
Cette partie sépare les deux aspects distingués dans l'étude du temps : nous commencerons par nous intéresser à sa *représentation* d'une part puis nous verrons sa *manipulation* appelée généralement *Raisonnement Temporel* (RT) d'autre part. Par abus de langage, il semble que le terme *raisonnement temporel* soit parfois utilisé de manière générale, c'est à dire sous une acception englobant les deux aspects. Nous essaierons autant que possible de maintenir la distinction dans ce chapitre. Nous allons donc commencer par nous intéresser à la représentation du temps en IA.

2.3 Représentation du temps

Comme dans le langage naturel, il existe plusieurs possibilités de représenter la composante temporelle. En effet, celle-ci peut être *implicite* ou *explicite*. Nous verrons ensuite sa *topologie* et sa *structure*, les différentes propriétés qui peuvent être représentées et nous terminerons ce parcours rapide de différentes représentations possibles du temps par quelques *approches formelles*.

2.3.1 Représentation implicite/explicite

La représentation *implicite* met en exergue les relations de causalité et les mécanismes d'évolution entre les événements du système que l'on souhaite étudier. Ainsi dans ce type de représentation, nous expliciterons les événements par une relation les liant entre eux ; avec l'utilisation par exemple, de relations de précédence temporelle se traduisant dans le langage par les prépositions : "avant", "après", "jusqu'à" : l'arrivée de la nuit *après* le coucher du soleil. Nous pouvons également utiliser une *relation de causalité* : le fait que "le soleil se couche" sera à l'origine de la tombée de la nuit.

La représentation *explicite* permet d'appréhender le temps dans sa globalité comme une variable particulière, l'attribut d'un événement ou d'une action. La caractéristique générale de cette représentation est de considérer un temps mesurable (quelle que soit l'échelle choisie ou la "granularité"). La définition des liens entre le temps et les événements considérés permet de maintenir une vision cohérente d'un ensemble de phénomènes étudiés. Pour reprendre l'exemple cité précédemment, nous dirions à présent que : "le 01/01/01, le soleil se couche à 18h00 et la nuit tombe de 18h00 à 19h00". En toute rigueur, il conviendrait de préciser le lieu, ce qui nous permettrait d'évoquer l'incontournable dualité espace-temps mais ce n'est pas notre propos ici. Apparemment, dans la littérature ces différentes représentations (implicite/explicite) semblent souvent liées à l'approche choisie pour représenter le temps :

1. *descriptive* en s'attachant à représenter les différents états du monde et les modifications entre eux (correspondant plus à une représentation explicite).
2. *relationnelle* mettant en exergue les liens de causalité entre les actions ou les événements et leurs conséquences sur le monde (correspondant généralement à une approche implicite).

2.3.2 Topologie du temps

La richesse du domaine temporel mène à s'interroger sur la manière et l'objet de la représentation du temps appelés aussi topologie. A ce niveau, il convient de distinguer la structure générale du temps pris dans sa globalité (la ligne de temps) et la structure de l'entité de base ou *primitive temporelle* qui sera manipulée. Sur le plan général, le temps peut être considéré comme :

1. *Linéaire* : une seule ligne de temps, un seul passé et un seul futur séparés par le présent. C'est la nature du temps des processus et des événements réels, ainsi que celui des univers déterministes.
2. *Ramifié* : soit vers le passé, si nous connaissons le présent (unique) et que l'on cherche à déterminer des déroulements passés permettant de l'expliquer ; soit vers le futur, si nous imaginons plusieurs déroulements possibles comme dans les problèmes de simulation ou de planification par exemple. Il est possible

éventuellement, de combiner un temps ramifié à la fois dans le passé et le futur.

3. *Parallèle* : qui permet de représenter le temps linéaire ou ramifié de différents agents et ainsi d'établir des corrélations entre ces différents temps.
4. *Circulaire* : permet de représenter des phénomènes périodiques.
5. *Infini* ou *borné* : un temps borné permet d'ajouter les notions de commencement et/ou de fin offrant par exemple, des bornes de référence.

Comme nous l'avons évoqué ci-dessus en fonction du problème et du domaine à représenter, il faut choisir une structure de temps adéquate. Ainsi un temps ramifié dans le passé sera bien adapté pour faire du *diagnostic* comme par exemple pour certaines applications médicales ; à l'inverse pour faire de la *prédiction* ou de la *simulation*, le choix d'un temps ramifié dans le futur pourra s'avérer pratique. En histoire ou en géologie, il est courant et pratique d'utiliser un temps borné permettant de se positionner par rapport à des événements clés ou de référence. D'autre part, comme nous l'avons dit, il faut préciser la structure de l'entité temporelle de base qui constituera les éléments de la ligne de temps précédemment spécifiée (la structure de l'entité de temps correspond à la nature de l'unité temporelle choisie) : la primitive choisie peut être l'*instant* ou bien l'*intervalle*. Cependant la représentation obtenue possède des propriétés différentes et il est parfois possible de passer d'une représentation à l'autre [VB83] [Tsa87].
A présent que nous avons une représentation globale avec ces deux aspects (ligne de temps et primitive temporelle), cette représentation temporelle peut également être complétée par certaines propriétés. Nous pouvons ainsi nous demander si le temps doit être : mesurable ? dense ou discret ? continu, homogène, isotrope (propriétés identiques dans le futur, le présent et le passé) ? Là aussi évidemment, les choix dépendront du domaine et du type de problème abordé.

Cette définition du concept temporel montre la difficulté et la richesse du problème abordé. Pour parvenir à maîtriser un concept complexe, la formalisation est un des moyens couramment utilisés. Intéressons nous à présent aux approches formelles du temps.

2.3.3 Théories du temps et approches formelles

Afin de lever l'ambiguïté du temps, différentes approches formelles ont été proposées mais elles ne représentent souvent seulement qu'une formalisation d'une certaine notion de temps parmi celles que nous avons pu présenter ci-dessus. Parmi les théories les plus représentatives, qui ont été développées pour représenter le temps nous pouvons citer les travaux suivants :

- Mac Carthy et Hayes [MH69] ont développé leur *calcul de situation* qui est un formalisme logique permettant de représenter sous le terme de *situation*, un état de l'univers à un instant de temps. Ce fut une des premières tentatives pour prendre en compte les aspects dynamiques d'un système en

Intelligence Artificielle. Dans ce travail, une situation n'a pas de durée. La transition d'un état à un autre est causée par les *événements* et les *actions* qui se produisent. Le calcul de situations s'avère inadapté dans un contexte multi-agent car il est impossible de représenter des actions simultanées ainsi que des changements continus [LS92].

- Mc Dermott [McD82] propose une logique temporelle du premier ordre avec une structure à base d'*instants* et un temps infini, continu, ramifié dans le futur et discret. Chaque *état* est caractérisé par sa date d'occurrence. Un ensemble totalement ordonné d'états allant jusqu'à l'infini (ou historique possible du monde) constitue une *chronique*. Une chronique admet toujours des branchements vers des futurs possibles. Un intervalle est défini comme un ensemble d'états totalement ordonné et convexe. Enfin les états et les chroniques représentent des périodes pendant lesquelles des *propositions* peuvent changer de valeur de vérité et des *événements* peuvent se produire. Une proposition est définie par l'ensemble des états pendant lesquels la proposition admet la valeur de vérité *vraie*. Un événement est identifié par l'ensemble d'intervalles durant lesquels il se produit. Le plus petit intervalle dans cet ensemble correspond à l'occurrence de l'événement. La logique qu'il définit est une logique temporelle du premier ordre, le temps intervient grâce aux prédicats spécialisés *OCCURS* et *HOLDS*. Ainsi, $OCCURS(e, t)$ et $HOLDS(p, i)$ signifient respectivement "*e survient à t*" et "*p est vrai pendant i*". Cette logique a été ensuite critiquée pour son inadaptation à la représentation des changements dans le monde par Allen. Il soulignait qu'un modèle à base d'instants n'est pas intuitif pour représenter un événement (qui, dans la pratique, peut lui-même toujours être décomposé en une suite d'événements).

- Allen [All83] [All84] a défini la théorie de l'action et du temps basée sur la notion d'intervalle avec un temps linéaire, continu et infini. Le choix de l'intervalle comme unité de temps est plus intuitif et mieux adapté à la représentation des états du monde. Allen propose un ensemble de 13 relations sur les intervalles (cf. figure 2.1).
  Ces 13 opérateurs sont disjonctifs exclusivement. Ils représentent l'ensemble des positions relatives possibles de deux intervalles. Il est intéressant de noter que ces 13 relations sont exprimables à partir de la seule relation *meets* ("touche"). L'ensemble des opérateurs de cette théorie souvent utilisée est parfois complété par d'autres afin de simplifier certaines expressions [Mou92] [CPB99]. Par exemple, Allen lui-même définit de nouvelles relations comme *IN* qui correspond à la relation *DURING* en ajoutant la possibilité pour les deux intervalles $t1$ et $t2$ d'avoir la même borne origine ou la même borne finale :

  $$IN(t1, t2) \Leftrightarrow (DURING(t1, t2) \vee STARTS(t1, t2) \vee FINISHES(t1, t2)).$$

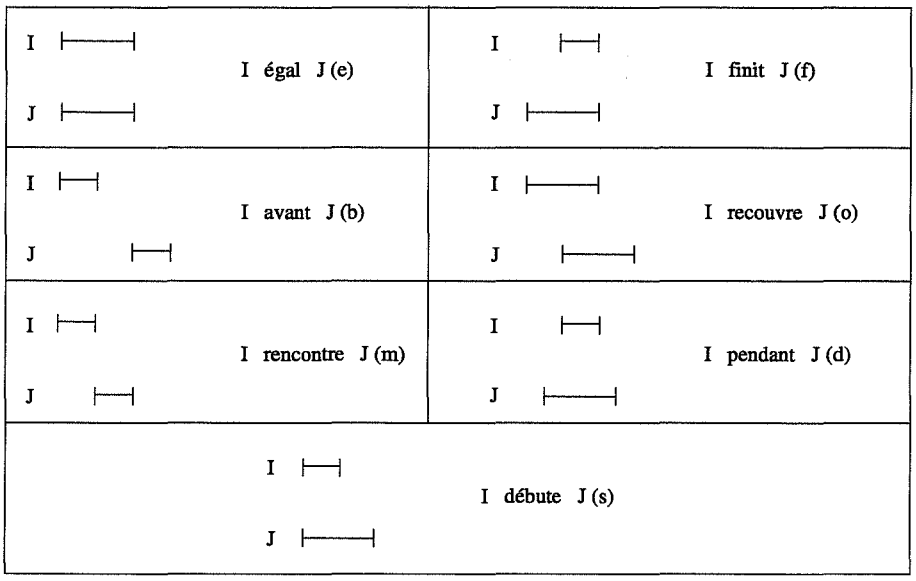


FIG. 2.1 – Les treize relations entre deux intervalles I et J : e, b, m, s, f, o, d et leurs inverses : bi, mi, si, fi, oi, di.

Nous pouvons noter également que l'ensemble des relations entre intervalles muni de l'intersection et de la composition forme une algèbre [Dra98].

- Kowalski et Sergot [KS86] proposent le *calcul des événements* basé sur une architecture de programmation logique permettant de raisonner sur les événements et le temps. La notion d'événement est différenciée du temps, ce qui permet de représenter des événements ayant des temps inconnus ainsi que des événements qui sont partiellement ordonnés et concurrents. Un événement est formalisé dans le sous-ensemble de clauses de Horn de la logique classique complété par la négation. La formalisation qui en résulte est exécutable comme un programme logique. La différence entre le calcul des situations présenté plus tôt et le calcul des événements est surtout conceptuelle : Le calcul de situations considère des états globaux alors que le calcul des événements s'intéresse aux événements locaux et aux intervalles de temps. Sur ce point de vue, l'approche est très semblable à celle d'Allen.

Il existe beaucoup d'autres travaux [Dou94] [PGS90], notamment différentes logiques formelles spécialisées sur un aspect particulier du temps, sortant ici de notre problématique. Le lecteur intéressé pourra trouver dans [GGT00] des informations sur les différentes logiques modales liées au temps comme la logique temporelle des propositions, logique temporelle linéaire, non transitive, à plusieurs dimensions, etc. Une revue de différentes théories temporelles peut être consulté dans [Lon89] ainsi que dans [Hay95].

A présent que nous possédons une idée globale sur les possibilités et les problématiques liées à la représentation du temps, nous pouvons à présent, nous

intéresser à sa manipulation : le raisonnement temporel (RT) en lui-même.

2.4 Raisonnement temporel

L'autre difficulté lorsque l'on s'intéresse à la composante temporelle est liée à la manipulation de ces représentations afin de pouvoir faire des calculs, conduire des raisonnements à partir de celles-ci. Cet aspect est généralement appelé le *raisonnement temporel* : il recouvre un aspect très informatique de l'étude du temps. En Intelligence Artificielle, le raisonnement temporel peut être défini par *raisonner sur ce qui change au cours du temps* ce qui correspond au **raisonnement sur le temps** mais il convient également de souligner l'existence dans des domaines spécifiques et spécialisés (systèmes temps réel, parallélisme, etc.) d'un **raisonnement dans le temps** qui permet de s'intéresser au temps nécessaire pour le calcul [Dzi90]. Nous rappelons que nous ne nous intéressons pas au raisonnement dans le temps dans ce travail puisque le temps de calcul n'est pas la préoccupation primordiale des systèmes auxquels nous nous intéressons. Différents problèmes très souvent issus du milieu industriel sont associés au raisonnement temporel : la planification, l'ordonnancement de tâches, le suivi de processus, le diagnostic, la prédiction. Certains nécessitant éventuellement une représentation du temps particulière comme nous l'avons vu précédemment.

2.4.1 Dimensions de raisonnement

Enfin de la même manière que pour le choix d'une représentation, en fonction du problème et du domaine à représenter, il faut choisir une dimension de raisonnement. Ce choix devra en outre être adapté à la représentation choisie précédemment. Ainsi, il est possible de s'orienter vers un raisonnement à visée pragmatique (*résultats utilisés pour engendrer des objectifs d'actions, produire des plans* [Ric90])ou à visée épistémique (*résultats utilisés pour construire des interprétations* [Ric90]), un raisonnement inductif ou déductif; il faudra choisir entre un raisonnement *sur le temps* ou bien *dans le temps* comme évoqué précédemment ou enfin être capable de traiter le problème du cadre ("frame problem") [Sho86] qui consiste à déterminer ce qui persiste d'un moment à un autre lorsque ceci n'est pas explicitement spécifié [SM88].

2.4.2 Outils pour manipuler le temps

Quelques outils ont été développés en IA permettant de conduire un raisonnement temporel. Nous distinguerons les approches dites *classiques* basées sur différentes logiques et les *approches par graphes* avec une méthode généralement de propagation de contraintes. Dans le cas des approches classiques, les logiques

temporelles exécutables offrent la possibilité de mettre en œuvre un raisonnement temporel (faire des déductions et raisonner sur le temps).

Approches dites "classiques"

Nous proposons deux exemples principaux (Allen et Ferguson puis Morgenstern et Stein) qui sont à l'origine de nombreux travaux dans le domaine.

- Allen et Ferguson [AS95] ont proposé un formalisme qui intègre à la fois le temps, l'action et les événements. Ces travaux ont été utilisés au sein du projet TRAINS [TA91] que nous présenterons par la suite (cf. section 2.4.3). Ces travaux s'orientent plutôt vers un raisonnement à visée *pragmatique* et de type *déductif*. Le langage formel développé par Allen et Ferguson est un langage du premier ordre dans lequel le temps n'est plus représenté comme une modalité mais comme un argument. Pour la mise en place d'un raisonnement temporel, les auteurs ont envisagé plusieurs niveaux de raisonnement :
  - La *logique* des intervalles qui assure la cohérence des relations entre intervalles (cf. section 2.3.3)
  - La *structure* du temps dans les liens qui unissent intervalles et prédicats.
  - La résolution du problème du cadre (*frame problem*) dont la solution appelée fermeture par les explications ("explanation closure") consiste à affirmer que les changements en cours ont une explication, une origine. Cette solution nécessite une ontologie temporelle suffisamment riche pour exprimer la causalité ; en l'occurrence ils utilisent l'intervalle, les événements et l'action.

- Morgenstern et Stein ont développé une théorie du raisonnement causal [MS88] appelées *théorie de l'action motivée* ("Motivated Action Theory", *MAT*) permettant également d'élaborer des solutions au problème du cadre. Les auteurs distinguent pour leur part, deux parties dans le raisonnement temporel :

  1. Une description particulière de l'état du monde et des événements, qui sera appelée *chronique* ("chronicle description").

  2. Un ensemble de connaissance causale de base qui sera appelée *théorie*.

  Ces deux parties réunies forment une théorie instanciée. Pour résumer, nous dirons que la notion de motivation est définie de la façon suivante : une proposition est dite *motivée si et seulement si elle est une conséquence logique de la théorie instanciée*. Le principe de raisonnement temporel proposé est alors le suivant : *Préférer les états futurs qui minimisent le nombre de propositions non motivées.*

Les approches classiques (Allen, Ferguson, Morgenstern, Stein) permettent généralement d'utiliser différents mécanismes déductifs comme les méthodes à base de tableaux sémantiques [PGS90], la résolution ou bien l'algorithme de Davis-Putnam. Cependant les preuves logiques et la complétude de ces travaux sont excessivement difficiles à mettre en œuvre dans une solution opérationnelle. En effet l'implémentation nécessite de faire des choix afin de satisfaire des critères d'efficacité comme une terminaison dans des temps acceptables.

Approches par graphes

Les systèmes mettant en œuvre une telle approche sont usuellement appelés gestionnaires de graphes temporels ("Time Map Managers", *TMM*) et s'intéressent à la gestion d'informations temporelles, c'est à dire la façon d'organiser la mise à jour et la recherche dans une mémoire temporelle. Dans cet objectif par exemple, Gerevini, Schubert et Schaeffer ont développé un ensemble d'algorithmes pour la gestion efficace de bases de relations temporelles [GSS94]. Ces gestionnaires de graphes temporels peuvent être classés en deux catégories :

1. les *approches symboliques* où les relations entre objets temporels sont symboliques. Certains gestionnaires ayant choisi cette approche reposent sur une structure du temps à base d'intervalles (MATS [KL91]), d'autres sur la notion d'instants (IxTet [MG89] [DGG93], TimeGraphI-II [GSS93]).
2. les *approches numériques* dans lesquelles les relations entre objets sont numériques. Dans ce cas également, la structure temporelle choisie peut être à base d'intervalle (S.A.Vere [Ver80]) ou basé sur l'instant (MTMM [Gro93]).

Sur le plan technique, un gestionnaire de graphes temporels ne s'intéresse qu'aux relations existant entre les objets temporels (succession, simultanéité) et aux moments où les événements se produisent [Het91]. Il gère également la cohérence et la complétude du graphe en vérifiant que les relations temporelles sont consistantes. Il infère également les relations déductibles de celles qui sont déjà connues. Cela signifie qu'il est nécessaire de posséder un langage support du raisonnement capable d'exprimer les relations entre objets temporels et des méthodes de propagation de contraintes pour conduire le raisonnement (satisfaisabilité, clotûre).

2.4.3 Travaux issus du domaine multi-agents

D'autres travaux sont issus plus précisément du domaine multi-agents avec une approche plus proche de la spécification formelle de logiciel en exprimant les propriétés dynamiques de systèmes multi-agents.

- Ainsi le langage concurrent MetateM [BFG+89] [Fis94] est un langage de programmation orienté agents. Il est accompagné d'une logique propositionnelle et multi-modale permettant d'exprimer à la fois le temps et les croyances pour raisonner sur les programmes. L'objectif de ces travaux est également de servir de support à un raisonnement (vérification de certaines propriétés) sur les propriétés dynamiques d'un SMA. Deux procédures de décision ont été développées s'appuyant sur des modèles (Konolige [Kon84] et sémantique des mondes possibles) et des techniques (méthode à base de tableaux et méthode à base de résolution) issues de l'IA. La topologie du temps choisie est linéaire.

- Le modèle formel de SMA développé par Singh [Sin94] a également pour but de spécifier des agents, des sociétés d'agents et d'en vérifier les propriétés. Ce formalisme incorpore le temps au moyen d'un langage formel noté **L** qui est une extension de la logique modale propositionnelle temporelle **CTL** [Eme90] enrichie de quelques opérateurs temporels. Le langage **L** adopte un temps ramifié dans le futur et les connaissances temporelles sont formulées par rapport à l'instant présent. Cependant aucune méthode de raisonnement n'est donnée si ce n'est la proposition d'utiliser éventuellement des techniques de "Model Checking" afin de vérifier certaines propriétés.

- Dans [LC00], est proposée une approche permettant de combiner une programmation à la fois impérative et déclarative pour spécifier et simuler des SMA. L'approche sépare les aspects temporels et non temporels notamment dans les contraintes (conflits et priorités). Nous laisserons les aspects temporels d'implémentation pour nous concentrer sur la spécification. Le temps est pris en compte dès le niveau des actions avec la mise en œuvre de contraintes temporelles sur les actions réalisées par les agents engagés dans une collaboration. L'utilisation d'une logique temporelle pour la conception est justifiée par d'une part, l'intérêt d'un langage de haut-niveau pour spécifier le comportement des agents et les mécanismes de collaboration et d'autre part, la possibilité de vérification de cohérence des différentes contraintes induites par les actions des agents. Nous reviendrons plus précisément sur ces travaux par la suite.

D'autres voies ont été explorées dans les SMA, en incorporant le temps à différents niveaux comme nous allons le voir dans la partie suivante. Le domaine du raisonnement temporel est très riche et les possibilités nombreuses, surtout en mettant l'accent sur une ou plusieurs propriétés particulières du temps ; les travaux menés dans le domaine le sont tout autant. Ils sont évidemment difficiles à comparer ; cependant l'approche *Voyelles* que nous avons vue (cf. section 1.2) peut nous aider à atteindre cet objectif.

2.5 Discussion

Ce chapitre nous a permis de faire un tour d'horizon des travaux existants sur le temps en IA. Il apparaît clairement que la prise en compte explicite du temps est un objectif complexe car le domaine est très riche. Les travaux à ce sujet sont d'ailleurs nombreux. Il ressort de cette étude qu'il est nécessaire de faire des choix aussi bien pour la **représentation du temps** que pour le **raisonnement temporel**. Ces choix sont évidemment liés. Les travaux sur les logiques notamment nous mettent en garde sur les difficultés pour passer du théorique à l'opérationnel. Ainsi, de nombreux écueils se présentent aux personnes souhaitant travailler sur la composante temporelle. La complexité et la richesse du domaine se révélant très vaste, nous avons limité les recherches au détriment de certains aspects (événements récurrents, algèbre de processus concurrents comme *Temporal LOTOS* [Reg93] par exemple). Ces aspects restent en marge du cadre de ce travail qui concerne à la fois le domaine temporel et le domaine multi-agent et pourront éventuellement s'inscrire dans les perspectives. Nous avons ensuite cherché à relier ces deux aspects ; ainsi, dans le chapitre suivant, nous nous intéressons à la prise en compte de la composante temporelle dans les SMA.

3

Temps dans les systèmes multi-agents

Sommaire

3.1 Introduction

Les travaux dans le domaine des systèmes multi-agents qui s'intéressent à la composante temporelle s'appuient naturellement sur les techniques décrites dans le chapitre précédent. Il aurait été possible de reprendre certaines caractéristiques énoncées plus haut liées aux choix de représentation, au contexte voire aux méthodes de raisonnement pour les comparer. Mais il nous a semblé plus intéressant de les comparer à partir d'une problématique transversale propre aux SMA : **la coopération d'entités autonomes organisées interagissant au sein d'un**

même environnement. Nous allons donc essayer de classer la prise en compte du temps de différents travaux en fonction des dimensions proposées par l'approche *Voyelles*. Les choix de représentation et de manipulation du temps restant un axe secondaire d'analyse.

En utilisant cette grille, nous allons présenter quelques travaux qui nous paraissent significatifs. Nous nous intéresserons précisément à la manière dont le temps est représenté et est utilisé au sein de chacune de ces dimensions. Nous ne prétendons pas être exhaustifs et certains travaux sont évidemment à l'origine de nombreuses études et extensions ultérieures n'apparaissant pas ici. Nous commencerons donc par voir comment le temps est pris en compte au sein de la dimension **Interaction**, puis nous nous intéresserons aux travaux mettant en œuvre le temps dans l'**Organisation**. Enfin, nous terminerons par la dimension **Agent** en étudiant la prise en compte du temps au sein même du raisonnement interne de l'agent. Un grand absent de ce parcours est le temps dans l'**Environnement** ; nous donnerons cependant quelques mots à ce sujet et les raisons pour lesquelles nous n'avons pas présenté cette étude. Nous proposerons dans la section 3.6, un tableau récapitulatif des travaux présentés permettant de les classer suivant notre grille d'analyse mais aussi en fonction des choix effectués sur la représentation du temps et le raisonnement temporel.

Nous commençons par sélectionner des travaux qui prennent en compte la composante temporelle au niveau de l'interaction.

3.2 Interactions temporelles

Dans les SMA, la distribution géographique et fonctionnelle de certains systèmes impose la mise en place d'interactions par le biais de langages, de protocoles et, même dans certains cas, de mécanismes de gestion de conversations (cf. chapitre 1). Ces contraintes associées à une forte dynamique du système lui-même ou de l'environnement dans lequel il se trouve plongé (places de marché, Internet, etc.) nécessitent en outre la prise en compte explicite des caractéristiques temporelles qui y sont associées.
Dans les travaux sur l'interaction dans les SMA, deux composantes sont souvent distinguées au sein d'un message : *force illocutoire* et *contenu propositionnel* (cf. la théorie des actes de langage [Sea69a], chapitre 1.2). De ce fait, les informations temporelles peuvent apparaître au sein de chacune d'elles :

- l'explicitation des informations temporelles au sein du *contenu propositionnel* rejoint les problématiques de définitions d'ontologies. Nombreuses sont celles qui proposent des représentations explicites d'informations temporelles (TOVE [Lab93], ontolingua [FFR96], KADS (méthodologie pour construire des ontologies), etc.). Ces informations ont peu d'incidences sur la gestion de l'interaction elle-même si ce n'est la définition de traducteurs de langages

d'expression de contraintes temporelles d'un langage commun aux agents à celui propre au raisonnement interne de l'agent ou à un logiciel externe qu'encapsule l'agent, comme celui décrit dans [BDSF97] par exemple.

- d'autres travaux, en revanche, explicitent la composante temporelle au niveau de la *force illocutoire* et la gèrent explicitement dans les interactions entre agents. C'est le cas, par exemple, de l'architecture d'agents proposée par [BF96] au travers d'une couche interaction, indépendante de l'application, gérant les informations temporelles imposées par le protocole de conversation. Nous allons illustrer rapidement ce besoin au travers de l'exemple suivant. Imaginons deux questions classiques :
  - Pouvez-vous me donner l'heure ?
  - Pouvez-vous me donner votre emploi du temps de la semaine prochaine ?

  En dehors du contenu propositionnel (ici "donner l'heure" et "donner l'emploi du temps de la semaine prochaine"), certaines informations portant sur le séquencement des messages, importantes pour le processus de coopération, sont implicites : la première question attend une réponse rapide alors que pour la suivante, le locuteur est moins impatient. Si ces notions sont implicites en langage naturel, elles nécessitent d'être explicitées dans un langage artificiel.

Les travaux que nous avons retenus dans cette section ont utilisé cette dernière démarche. Comme nous allons le voir, certains se sont intéressés essentiellement aux langages de communication agent [FIP97], [SJNP98]. D'autres ont proposé des protocoles temporels [AGJ+94], [NJFM97] (ADEPT), [Ca99] ou encore ont envisagé la mise en œuvre du temps dans les conversations entre agents permettant ainsi de contraindre, par exemple, les temps de conversation [Tra96]. Parmi ces travaux, certains ont été également retenus parce qu'ils mettent en œuvre des techniques ou formalismes spécifiques comme les logiques ([LC00] [PAC96]) ou les réseaux de Pétri pour représenter le temps [Ca99].

3.2.1 Langages de communication temporels

Comme nous allons le voir, les informations temporelles représentées au sein des langages de communication ont différentes finalités. Nous commencerons par le langage *ACL FIPA* qui permet d'expliciter une contrainte temporelle sur la réponse, puis nous verrons un langage orienté vers la négociation dans lequel le temps est utilisé comme information pour la gestion de l'historique de la conversation. Et nous terminerons par divers langages proposés par les plates-formes SMA.

ACL FIPA

Le langage de communication agent de la FIPA (ACL) [FIP00] est basé sur la théorie des actes de langages. Les spécifications proposent un ensemble de types de messages et la description de leurs champs. Chaque acte de communication est décrit sous forme narrative et présente une sémantique formelle basée sur une logique modale. Ces spécifications fournissent également une description normative d'un ensemble de protocoles d'interaction. Parmi les paramètres prédéfinis des messages, le champ *reply-by* dénote un temps ou une expression temporelle qui spécifie l'échéance maximum pour la réponse. Le format temporel spécifié est basé sur le format ISO 8601 [WW98], avec des extensions pour pouvoir exprimer des durées en millisecondes et un temps relatif. A notre connaissance, aucune autre information particulière n'est spécifiée pour le choix de la représentation du temps.
Les contraintes temporelles explicitées sur le délai de réponse ne nous paraissent pas suffisantes et doivent être complétées. En effet, nous pensons que la prise en compte du temps dans l'interaction concerne l'acte de langage lui-même et peut éventuellement être liée au contenu propositionnel.

Argumentation temporelle pour la négociation

Un domaine d'étude relativement récent en SMA, étroitement lié à l'interaction est la négociation. Bien qu'il soit difficile de concevoir une négociation sans date limite, les aspects temporels, indéniables, sur le sujet ne sont pas toujours pris en compte et encore moins explicités. Une étude sur la négociation basée sur l'argumentation a été proposé dans [SJNP98]. Elle présente un langage de communication entre agents comportant un argument temporel. Cet argument porte sur l'aspect illocutoire (cf. théorie des Speech Acts [Sea69b]) et est considéré comme une estampille temporelle. Il est utilisé notamment pour définir un historique du dialogue de négociation. Le temps est considéré comme discret et comme un ensemble entièrement ordonné d'instants.

Ce travail fournit un support pour la négociation incluant une prise en compte explicite des aspects temporels mais ne proposant pas une gestion de contraintes temporelles complexes. Cependant l'argument temporel est utilisé pour conduire la négociation. Le raisonnement peut également être étendu avec des conditions qui changent en fonction du temps. En d'autres termes, la mise en place de règles (temporaires) telles que *"ligne de conduite BT (British Telecom)=24h"* $\Rightarrow$ *"prévoir une équipe complète"* permet de modifier le comportement dépendant de la négociation en fonction de contraintes temporelles : dans cet exemple, le fait de poursuivre la ligne de conduite *BT* même la nuit (24h) impose de maintenir une équipe complète même en service nocturne et donc d'être "intransigeant" dans le cadre de la négociation afin de respecter le niveau de qualité exigé par la conduite "BT".

Autres travaux

Pour terminer avec les aspects temporels dans les langages de communication agent, nous évoquerons les différentes plates-formes de développement SMA qui offrent souvent une prise en compte de la composante temporelle comme c'est le cas, par exemple, dans la plate-forme *ZEUS* [NNLC99] où une date sur les messages (émission et réception) permet de définir des "timeouts" au niveau des communications pour traiter un message. Cette estampille temporelle ne concerne donc pas l'interprétation du message (au sens contenu propositionnel). Le langage utilisé est du type ACL décrit précédemment. Dans *TÆMS* [DL94] [Dec96] [WSXL99], des messages de type temporel comme "*requires delay*", "*causes*" ou "*inhibits*" permettent de coordonner des agents ayant des tâches en commun en introduisant des types de messages spécifiquement temporels.
La plupart de ces langages sont basés sur la théorie des actes de langages, possèdent des arguments temporels mais aucun n'est explicitement dédié à la représentation du temps.

3.2.2 Protocoles et gestion de conversations

Au delà des capacités d'expression d'informations temporelles au sein des messages échangés entre les agents, certains travaux se sont intéressés à l'expression des contraintes temporelles dans les protocoles d'interaction conduisant ainsi à une gestion temporelle de conversations. Les travaux qui suivent, s'intéressent principalement à ces aspects temporels liés aux protocoles et à la gestion des conversations. Nous présenterons dans un premier temps, le système "GOAL" [PAC96] pour lequel la conversation et les contraintes temporelles sont uniquement contrôlées par un protocole. Nous décrirons ensuite le système ADEPT avec l'ensemble des travaux sur la négociation décrit dans [Far00] où la gestion des conversations fait l'objet de raisonnement temporel dédié via l'utilisation de stratégies ainsi que le système TRAINS [ASF+95] qui propose un agent spécialement dédié à la conversation.

GOAL

Le Project GOAL [PAC96] vise à développer des outils génériques pour supporter un nouveau paradigme de gestion par projet. Les auteurs se sont intéressés à la normalisation d'interactions entre agents autonomes en spécifiant des messages et des protocoles pour la coopération et la communication inter-agent. Le langage de spécification proposé dont beaucoup de caractéristiques sont héritées d'un langage de programmation logique, permet de spécifier différents mécanismes de communication. On retrouve ce type d'approche dans [LC00] (dont nous avons parlé en section 2.3).
Ce travail propose un champ "timeout" mesuré sur l'horloge de l'agent émetteur qui permet de s'assurer de la terminaison de l'instruction liée à l'envoi du message

si certaines réponses n'arrivent pas (comme par exemple sur un appel d'offres). De même, certaines contraintes temporelles peuvent être explicitées dans le cadre d'une négociation (date de début, timeout, date de réponse, durée de réalisation de la proposition, etc.). Les auteurs soulignent que les spécifications proposées établissent ce que les agents doivent faire et non comment cela devrait être fait. Pendant le déroulement de conversations, l'agent connaît à la fois la spécification du protocole et l'historique des messages échangés lors d'une conversation. Cela permet de valider les messages arrivant et respectant les spécifications du protocole et de pouvoir faire une analyse des services proposés, des besoins et des offres à partir des messages reçus. Seuls les messages considérés comme valides (respect du protocole dont notamment les aspects temporels) sont enregistrés dans l'historique de la conversation.
Cette étude lie la notion de temps au niveau spécification (définition des protocoles) avec son instanciation (gestion des conversations) où est vérifiée la validité des messages et l'historique.

Nous allons voir à présent l'utilisation de protocoles dans la négociation avec le projet ADEPT.

ADEPT

Dans ADEPT [AGJ+94], les auteurs se sont intéressés à la gestion de négociation dans le cadre d'offre de services. De manière générale, deux niveaux sont considérés au sujet du temps dans la négociation :

- le temps nécessaire pour obtenir un accord. Ce temps doit être raisonnable.
- le temps pendant lequel les services négociés doivent être exécutés. Ce temps est important dans la plupart des cas et crucial dans d'autres.

Nous retrouvons ainsi les deux niveaux évoqués en introduction dans le cadre de la gestion de conversation : le temps relatif à l'interaction et le temps véhiculé par les messages ayant une répercussion sur le fonctionnement de l'agent.

Dans le cadre d'ADEPT, nous distinguons :

- le *temps relatif au contenu des messages* échangés et qui fait l'objet de la conversation est le temps auquel les services négociés doivent être exécutés. Les protocoles de négociation comportent des propriétés temporelles relatives à la gestion des contrats : la durée représente le temps maximum que l'offreur peut prendre pour terminer le service, les dates de début et de fin représentent le temps pendant lequel l'accord est valide. Les agents doivent ainsi se mettre d'accord sur une fenêtre temporelle n'entrant pas en conflit avec les plannings propres de chacun des agents participant à la négociation.
- le *temps relatif à l'interaction* elle-même est lié à la gestion des conversations. Des stratégies sont mises au point pour spécifier un comportement particulier de l'agent au cours de la négociation. Ces stratégies se basent sur

trois critères principaux qui sont : le *temps* (échéance de la négociation), les *ressources* (de manière à ce que l'agent ne dépense pas plus de ressources que la valeur du contrat) et le *comportement de l'adversaire* (pour que l'agent ne soit pas exploité pendant la négociation). En fonction de la stratégie mise en place, différentes tactiques peuvent être utilisées. Par exemple, si le temps est important, deux types de tactiques existent : *boulware* qui consiste à maintenir son offre le plus longtemps possible et à faire des concessions juste au dernier moment, et *conceder* qui se déplace rapidement vers la valeur limite correspondant par exemple, au prix maximum que peut proposer l'agent.

La répercussion des informations temporelles sur les plans de l'agent et cette gestion des stratégies permet de mettre en évidence la relation entre les aspects temporels dans le raisonnement interne (stratégies choisies par exemple) et dans la partie interaction (comportement pendant la négociation, etc.). Aucune information n'est donnée sur la représentation du temps choisie. Les exemples présentent un temps mesurable à base d'instants.

TRAINS

Le système TRAINS [Tra96] [ASF+95] propose un SMA pour la gestion de conversations avec des êtres humains. Nous évoquerons uniquement les parties liées à la conversation du système TRAINS dont le gestionnaire de dialogue est implémenté par un agent hybride. En effet, l'auteur justifie l'utilisation d'une partie réactive de l'agent par l'aspect critique que peut révéler un retard ou un délai dans le comportement relatif à une réponse : la même réponse peut avoir une connotation différente si elle est retardée. Sans rentrer dans les détails, il nous a semblé intéressant de relever que l'auteur définit la notion de plans de conversations comportant trois types de contraintes : les contraintes sur les agents, sur les objets manipulés dans le cadre de la conversation et les contraintes temporelles. De plus, les ensembles de contraintes formés sont estampillés en fonction de leur date de validité permettant ainsi de limiter les informations manipulées à un moment donné. Cela revient à définir une fenêtre temporelle de validité pour la plupart des contraintes évoquées.

Les travaux décrits ici soulignent également l'existence de contraintes temporelles relatives aux protocoles et à la gestion de conversations et l'importance d'expliciter ces contraintes. A présent, intéressons-nous à des aspects plus techniques concernant la mise en œuvre de raisonnement temporel au sein de l'interaction.

3.2.3 Mise en œuvre du Raisonnement temporel

Dans les travaux suivants, nous nous intéressons à la mise en œuvre du raisonnement temporel pour prendre en compte le temps au sein de l'interaction.

Travaux de P. De Loor et al.

Il serait incomplet de présenter des travaux sur le temps sans citer de travaux mettant en œuvre une logique temporelle. Comme nous l'avons décrit en 2.3, dans [LC00], une approche est proposée permettant de combiner une programmation à la fois impérative et déclarative pour spécifier et simuler des SMA. Le langage déclaratif est ainsi basé sur la logique temporelle d'Allen qui a l'avantage de fournir ce langage de haut-niveau pour décrire l'interaction. Comme nous l'avons souligné en section 2.4.2, les logiques temporelles exécutables offrent en outre, la possibilité de mettre en œuvre un raisonnement temporel (faire des déductions et raisonner sur le temps). La partie qui nous intéresse particulièrement dans ce travail, correspond à la définition des messages (adresses et moments d'émission) assurant le respect des contraintes temporelles. Ainsi des messages spécifiques (*start()*, *stop()*, *mustStart()* et *mustStop()*) sont envoyés quand une contrainte temporelle impose de donner la priorité à l'action courante sur les autres actions. Cela permet de forcer ces actions à démarrer ou s'arrêter.
Les contraintes temporelles sont définies à la spécification et semblablement à [SJNP98], des messages spécifiquement temporels se chargent d'empêcher un blocage à l'exécution. D'autres méthodes peuvent être utilisées pour gérer le déroulement de l'interaction et les contraintes temporelles. Parmi elles, les réseaux de Pétri comme nous allons le voir dans ce qui suit.

Travaux de Chen et al.

Dans le cadre de la gestion de chaîne de production, un SMA à base de négociation par Réseau de Pétri Coloré (référencé par la suite par NRPC) a été défini dans [Ca99]. Les performatifs utilisés pour définir la négociation sont issus d'un sous-ensemble d'ACL FIPA mais cet ensemble peut être étendu. Le processus de négociation est modélisé par le biais d'un réseau de Pétri coloré. Ce dernier est utilisé comme un outil de modélisation pour des systèmes dans lesquels la communication, la synchronisation et le partage des ressources jouent un rôle important. En effet, le réseau de Pétri (et ses extensions) fournit les primitives pour traiter le processus d'interaction : il offre des possibilités relativement simples pour représenter et contrôler des communications (parcours d'un jeton spécifique), pour synchroniser différentes actions (spécification d'états, de places et de transitions pour y accéder) et enfin partager des ressources (accès, quantification contrôlés par des jetons spécifiques). Dans ce SMA, le déroulement des interactions et notamment du protocole est contrôlé par un "timer" générant un jeton (token) spécial. Il existe trois sortes de places : *Inactive*, *Waiting*, and *Thinking* qui correspondent aux états possibles d'un agent durant le processus de négociation. Par exemple, quand l'acheteur décide de démarrer un processus de négociation, il passe de l'état *Inactive* à *Waiting* en prenant l'action *Send Messages (CFP)*. Le type de réponse est codé par une couleur spécifique (*proposal*, *accept-proposal*, *reject-proposal* ou *terminate*). A la réception du message, l'acheteur passe à l'état *Thinking*. En

cas de non-réponse, le Timer génère un jeton spécial (non-coloré) qui contrôle la continuité du déroulement.

Ces quelques travaux donnent un bon exemple de la prise en compte actuelle des aspects temporels au sein des différentes composantes de l'interaction comme le langage de communication, les protocoles et les conversations. Cependant, nous pouvons remarquer qu'il n'existe pas de langage d'interaction ni les autres aspects relatifs à l'interaction (protocoles, gestion de conversations) spécifiquement temporels ; c'est à dire adapté à l'explicitation de ces contraintes temporelles.
Les travaux suivants concernent la prise en compte du temps dans l'organisation.

3.3 Organisations temporelles

Avec l'accroissement de la complexité des applications construites, la dimension organisationnelle dans les systèmes multi-agents est devenue de plus en plus nécessaire. Quelques travaux se sont intéressés aux aspects temporels dans cette dimension. Nous considérons dans un premier temps, des travaux ayant explicité la composante temporelle dans les relations sociales entre agents, calculées par les agents eux-mêmes (réseaux de dépendance notamment) puis nous verrons la prise en compte du temps dans une structure organisationnelle. Certains ont même introduit le temps dans la formalisation d'obligations et de permissions [DK97] qui sont deux notions fondamentales pour formaliser et spécifier des structures organisationnelles.

3.3.1 Relations sociales temporelles

Nous présenterons tout d'abord le système STARS [All98] [ABS00] qui met en place un raisonnement social temporel entre les agents à partir de calcul de dépendances. Nous nous arrêterons ensuite sur la description des travaux de Moulin [BBM97] et Rousseau [Rou95].

STARS

Dans STARS [All98], l'accent est mis sur un raisonnement temporel social au travers de la définition de réseaux de dépendances temporelles et de mécanismes adaptés. Devant exécuter leurs tâches tout en restant cohérents avec les exécutions des tâches mises en œuvre au sein d'autres agents du système, chaque agent essaie de tirer partie des différentes relations existant entre les tâches (complémentarité, opposition). Celles-ci sont utilisées pour calculer des réseaux de dépendance [SD95] entre les agents. Exploitant les contraintes temporelles existant entre tâches, les aspects dynamiques sont explicitement introduits dans ces réseaux. Ainsi, trois types de relations de dépendance temporelle sont calculés : concurrence, besoin,

et aide. Les contraintes temporelles sont exprimées en utilisant le calcul des intervalles d'Allen [All84]. Le réseau de dépendances complet est représenté par un graphe statique qui est mis à jour dynamiquement et partiellement (seules les relations actives sont représentées) pendant le fonctionnement, tout au long de l'exécution des tâches. Une fenêtre temporelle est définie pour chaque agent lui permettant de savoir quelles sont les relations de dépendances actives à un moment donné. Cela permet à un agent de connaître, à n'importe quel moment, quels sont les agents qu'il doit contacter pour que ses tâches et éventuellement celles des autres agents soient réalisées correctement.

Les règles de raisonnement temporel social (sur les différents types de dépendance) génèrent ainsi les interactions des agents (besoin de contacter un autre agent pour réaliser une tâche ou une partie de la tâche, par exemple). Nous voyons ainsi apparaître des relations entre les différentes facettes au sein même de l'agent. Ces règles s'appuient simultanément sur des relations temporelles entre agents mises à jour via une exécution locale ou via les interactions qui se déroulent entre les agents.

Travaux de MOULIN et ROUSSEAU

Dans le modèle d'agent développé par Anne-Claire Boury-Brisset et Bernard Moulin (référencé par la suite par MAT), des informations temporelles sont explicitées dans les relations sociales entre agents.
Ce modèle intégre des mécanismes de raisonnement multiples [BBM97] et certains termes relevant de l'organisation comme le *rôle* ou les *relations entre agents* sont complétés par une durée représentant une période de validité. La prise en compte du temps peut être à la fois explicite (intervalles temporels de validité) ou implicite (prédicats logiques) mais vue de manière globale : il n'y a pas de distinction entre ce qui relève de l'organisation, de l'environnement ou d'une autre dimension. En effet, les faits et les croyances manipulés par les agents sont des prédicats marqués temporellement et cette prise en compte concerne donc tout type de fait ou de prédicat. La notion d'organisation n'est ainsi pas explicite mais se retrouve aisément au sein des prédicats. La représentation du temps choisie est uniforme et utilise les intervalles de temps pour exprimer les différentes situations. Une version étendue des relations d'Allen [Mou92] est utilisée pour représenter les relations temporelles entre les états mentaux. Un historique est généré par chaque agent pour garder une trace des connaissances périmées. Les connaissances relatives aux relations de l'agent avec les autres sont appelées relations interpersonnelles.

Dans la continuité de ces travaux, nous trouvons la thèse de D. Rousseau qui s'intéresse à la modélisation et la simulation de conversations dans un univers multi-agent [Rou95]. L'approche multi-agent est basée sur la négociation et la coopération pour modéliser et simuler des conversations entre des planificateurs

intelligents. L'aspect organisationnel est représenté par la capacité des agents à raisonner à la fois sur le contenu de leur modèle mental et sur celui des autres agents. Les agents manipulent ainsi des informations temporelles et des contraintes. Le temps est géré dans leur approche à l'aide d'une horloge [Liz88] pour déterminer le temps présent. Les états mentaux sont séparés en deux catégories : les états mentaux de base et les états mentaux de nature sociale. Nous nous intéresserons particulièrement à cette dernière catégorie qui regroupe les aspects organisationnels de l'agent. A l'intérieur de cette catégorie, nous retrouvons la notion de relations interpersonnelles qui permettent aux agents de jouer un rôle social. L'intervalle de temps associé à chacune des relations correspond à la période pendant laquelle l'état mental est effectif. Par exemple, *Relation-INT(rel-int1, Dartagnan, [1,100], Collègues de travail(Dartagnan, Atos))* est une relation inter-personnelle simplifiée, spécifiant à Dartagnan qu'Atos est "collègue de travail" pendant l'intervalle $[1, 100]$. D'autres contraintes ou relations temporelles peuvent être ajoutées et spécifiées par les opérateurs d'Allen [All84] [All83]. Là encore, une structure organisationnelle n'est pas définie de manière explicite mais elle peut éventuellement être retrouvée (extraite) des connaissances sociales de l'agent.

Ces dernières approches (Moulin et Rousseau) ont le mérite de choisir une représentation uniforme du temps permettant d'expliciter les aspects temporels relevant de l'organisation et de les rendre "manipulables" par le raisonnement interne de l'agent.

3.3.2 Structures organisationnelles

Une autre manière de voir l'organisation est de s'intéresser à la structure organisationnelle qui spécifie la partie sociale du comportement des agents les uns avec les autres. Nous nous intéresserons à Team Planned Activity [KLR+92] puis nous verrons le travail de Jonker, Treur et de Vries [JTd01].

Team Planned Activity

Dans Team planned activity [KLR+92], les auteurs s'intéressent à la représentation de plans communs pour des équipes d'agents situés dans des environnements dynamiques, et plus précisément à la manière dont les agents s'organisent pour former des équipes adaptées à une activité complexe et synchronisée. Les auteurs visent à la définition d'un langage pour décrire l'activité d'une équipe.
Du point de vue organisationnel, les notions de rôles et d'équipes sont positionnées grâce à des opérateurs temporels particuliers (séquence, choix non déterministe et parallélisme) mais une dimension précise et explicite du temps n'est pas exprimée. On ne trouve donc qu'un positionnement relatif de notions entre elles mais aucune notion numérique ou de positionnement par rapport au temps global n'est permise.

Organisation comme outil de spécification des besoins

Jonker, Treur et de Vries [JTd01] (référencé par la suite JTV) s'intéressent à la manière dont les spécifications de la dynamique d'une organisation en termes de groupes, de rôles à l'intérieur d'un groupe et d'interactions entre rôles peuvent constituer un lien entre les spécifications des besoins d'un SMA d'une part et celles des agents individuels d'autre part. L'objectif est de permettre aux agents d'anticiper le comportement futur des agents à partir des comportements prévus pour un rôle ou à partir des motivations personnelles des autres agents. Ils s'appuient sur le fait que d'un point de vue dynamique, une structure organisationnelle apporte des spécifications de contraintes sur les interactions et le comportement des rôles. Le modèle de structure organisationnelle est similaire à celui proposé dans MadKit [FGJ+00].

Les contraintes temporelles qu'ils définissent sont principalement basées sur les relations temporelles entre les "entrées" et les "sorties" des instances de rôles, que ce soit pour les spécifications de comportement de rôle, d'interactions au sein de groupes ou d'interactions inter-groupes. La référence au temps est faite de manière formelle : le temps est discret et borné et la primitive temporelle est l'instant. Le langage temporel défini se rapproche du calcul de situation [MH69] par son formalisme logique. Le raisonnement temporel s'appuie également sur l'approche *BDI* [RG95b] pour représenter l'état des agents.
L'objectif du travail mené dans JTV est d'améliorer l'étendue de la coordination par la prise en compte de la composante temporelle au sein de la dimension Organisation.

Ces travaux soulignent l'existence d'aspects temporels dans l'organisation qu'elle soit basée sur des relations sociales ou sur une représentation à partir de structures organisationnelles. Il existe donc peu de travaux permettant de prendre en compte de manière explicite les aspects temporels dans l'organisation or dans le cadre de systèmes dynamiques, le respect des aspects temporels liés à l'organisation peut se révéler crucial. La définition et l'utilisation d'outils adaptés pour expliciter et tenir compte de la dimension temporelle permettrait de résoudre une partie du problème qui nous intéresse. Pour compléter cette analyse, nous allons nous arrêter à présent sur la dimension Environnement bien qu'elle ait été peu abordée dans cette étude. Le paragraphe suivant justifie ce choix et donne cependant quelques pistes à ce propos.

3.4 Temps et Environnement

Pour que la grille d'analyse corresponde intégralement à l'approche *AEIO*, il conviendrait de s'attacher à l'étude de la dimension environnement (qui regroupe les fonctionnalités liées aux capacités de perception et d'action de l'agent sur son

environnement). Cependant, cette dimension est encore assez peu étudiée principalement à cause de sa forte dépendance à la fois avec l'application et avec l'objectif du SMA (cf. section 1.1.3. Nous estimons que la diversité des rares travaux dans ce domaine ne serait pas représentative d'une prise en compte du temps globalement au niveau de l'environnement. Il est possible cependant de trouver des travaux où le temps de l'environnement prend une place particulière comme dans *DVMT* [LC83] [LC87] [DHL89] ou plus généralement dans des travaux relevant plus de la robotique où la modélisation de l'environnement s'avère incontournable. Nous intégrerons au sein du raisonnement interne (cf. section suivante 3.5) des aspects temporels pouvant éventuellement relever de l'environnement comme par exemple, les lois d'évolution du monde que l'on trouve dans (*ActeM*) ou le sous-système de perception (*Guardian*) que nous verrons dans la section suivante. Pour terminer notre analyse, nous allons maintenant voir comment la composante temporelle a été prise en compte au sein du raisonnement interne de l'agent.

3.5 Agents temporels

Nous allons présenter ici des travaux dont la composante temporelle apparaît au sein de la dimension Agent.
De nombreux travaux existent. Nous complétons cette vision de l'agent temporel en revenant sur l'approche AEIO et la prise en compte du temps à ce niveau : nous avons choisi de nous intéresser plus particulièrement à des architectures d'agents apparaissant dans des systèmes où le temps était pris en compte. Nous allons donc voir quelles facettes peuvent être considérées comme temporelles et essayer de faire apparaître la manière dont le temps est géré non seulement au sein du raisonnement interne de l'Agent avec l'agent *Guardian* mais aussi au niveau de l'interaction avec *ADEPT* [AGJ+94] et de l'organisation avec l'agent *STAx* [ABS00].

Nous nommerons dans la suite, une facette raisonnement interne capable de gérer le temps : $a(t)$ et. $e(t)$, $i(t)$ et $o(t)$ respectivement les autres facettes que nous avons mises en exergue à partir de l'approche AEIO.

3.5.1 Temps au sein de la dimension Agent

De nombreux travaux proposent des agents comportant une gestion du temps uniquement au sein du raisonnement. Dans ces systèmes, cette gestion du temps mise en place au sein du raisonnement se révèle généralement propre à l'utilisation des compétences et des stratégies de l'agent. Nous présenterons à ce sujet *Guardian* [HR90] [HRWA+92] qui présente l'avantage d'expliciter clairement la prise en compte des aspects temporels au sein de la structure d'un agent. Nous pouvons également citer le système *Alarms* [NL96b] architecture d'agent abstraite appelée "motivated agency" [NL95] qui a pour but de compléter et d'étendre les architectures *BDI* (cf. section 1.2.4 et qui propose une spécification temporelle

très complète de la notion de but ou le modèle *ActeM* [LM90] qui propose un planificateur temporel spécifique et manipule le temps de manière explicite.

L'architecture d'agent intelligent *Guardian* proposée dans [HR90] [HRWA+92] aborde les problèmes d'exécution temps-réel dans le cadre de systèmes à base de connaissances en interaction avec un environnement dynamique. Il n'existe pas de communication entre agents mais seulement des perceptions d'événements et des actions au niveau de l'environnement.

Mise en œuvre du raisonnement Le raisonnement de l'agent se fait sur ses connaissances (buts, contraintes, tâches, ressources et environnement). Du point de vue de l'aspect temporel, les informations perçues s'inscrivent dans une chronologie et sont donc périssables. L'*utilité* des traitements se dégrade en fonction du temps et l'*efficacité* des actions dépend de leur synchronisation avec les événements externes. Ces aspects, particulièrement sensibles dans le cadre temps-réel peuvent également apparaître et avoir leur importance dans une prise en compte de la composante temporelle plus générale ; c'est pourquoi nous les soulignons ici.

L'agent *Guardian* est constitué principalement de trois sous-systèmes appelés *cognition*, *perception* et *action* (cf. 3.1). En se replaçant dans l'approche *voyelles*,

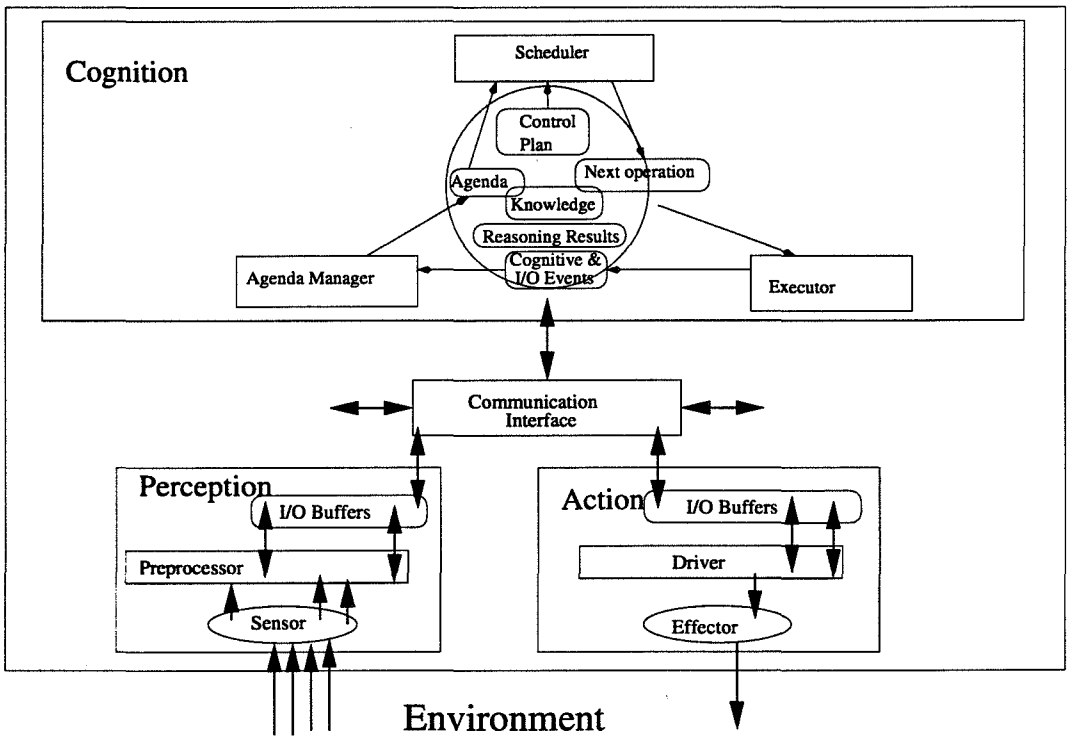


FIG. 3.1 – Architecture d'agent Guardian

les sous-systèmes *perception* et *action* correspondent à la facette Environnement puisqu'ils permettent de capter les informations provenant de l'environnement et

agir dessus. Nous considérons que le temps est géré au sein de la facette environnement dans cette architecture d'agent à deux niveaux :

1. le temps est géré au sein du sous-système de *Perception* par l'intermédiaire du *preprocessor*. Ce sous-système capte et interprète les événements provenant de l'environnement. Les *événements* sont estampillés au moment où ils sont perçus.
2. le sous-système d'*Action* utilise le *Driver* pour envoyer aux effecteurs appropriés les commandes successives aux moments opportuns et joue ainsi le rôle de *Timer*.

Selon notre point de vue, le sous-système *Cognition* correspond à la facette Agent qui gère le raisonnement, les connaissances et les décisions de l'agent. Cette architecture est une extension de l'architecture très connue "BB1" [Hr84]. L'architecture comporte un sous-système cognitif qui gère le raisonnement temporel au moyen d'un *ordonnanceur* (scheduler). Celui-ci manipule des échéances ("deadlines") pour la génération des plans de l'agent. Une présélection des actions réalisables est effectuée par le "gestionnaire d'agenda". La composante temporelle est ainsi utilisée pour sélectionner les différentes actions à effectuer directement au cœur de la partie cognitive de l'agent, le moteur de raisonnement temporel est l'ordonnanceur. L'*agenda* et surtout le *scheduler* permettent de prendre en compte le temps au niveau du raisonnement interne de l'agent.
Nous pouvons donc synthétiser de la manière représentée sur la figure 3.2.
Nous exprimons ainsi une prise en compte explicite du temps dans une facette

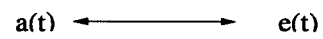


FIG. 3.2 – Prise en compte du temps dans l'architecture d'agent Guardian

"environnement" ($e(t)$) qui est ensuite transmis vers (l'équivalent d') une facette "agent" ($a(t)$). Le trajet réciproque concerne les informations temporelles relatives aux actions planifiées par la facette "agent" qui sont ensuite envoyées vers la facette "environnement" pour être exécutées au bon moment. L'expression du temps dans la facette "environnement" correspond à deux aspects particuliers :

1. une estampille temporelle pour dater les événements captés dans l'environnement,
2. une date pour exécuter les actions planifiées.

Bien qu'aucun exemple de prise en compte des aspects temporels ne soit explicitement présenté dans l'article étudié ; l'application médicale implémentée laisse penser que le temps semble être manipulé à la fois sous forme d'instants et d'intervalles. Des informations sont données sur la granularité en présentant des applications implémentées nécessitant des contraintes de l'ordre de la minute ou de l'heure et d'autres de l'ordre de la seconde.

Ce sont les deux types d'aspects temporels (cf. ci-dessus) qui seront respectivement utilisés et produits par la facette "agent". Il paraît ainsi évident que ces deux facettes sont dépendantes l'une de l'autre. Ceci offre un faible degré de flexibilité quant à la prise en compte du temps puisque les aspects temporels provenant des différentes facettes ne peuvent être traités indépendamment.

Nous allons voir à présent des modèles où le temps est pris également en compte au niveau de l'interaction.

3.5.2 Temps exprimé également au sein de l'interaction

Certaines architectures apparaissent dans des systèmes capables en outre de gérer des interactions. Cependant, même si des informations temporelles sont véhiculées entre les agents, via les messages, celles-ci ne sont généralement pas prises en compte dans la gestion même de l'interaction. Par exemple, *AOP* [Sho90] qui s'appuie sur une logique épistémique temporelle, présente également une prise en compte explicite du temps non seulement au sein de la dimension Agent mais également au niveau de l'Interaction. Pour mettre en exergue une véritable prise en compte du temps dans l'interaction, nous allons revenir sur le modèle *ADEPT* (cf. section 3.2.2.0).

Le modèle *ADEPT* [AGJ+94] présenté précédemment, propose également une prise en compte du temps au niveau de l'interaction. Nous ne reviendrons pas ici sur la description des aspects temporels mais nous allons plutôt mettre en exergue les particularités de l'architecture pour cette prise en compte.

L'architecture d'agent *ADEPT* est composée de cinq modules (cf.figure 3.3). A la différence de l'architecture précédente (Guardian), l'agent est capable de prendre en compte le temps dans l'Interaction : en effet, le module *IMM* de gestion de l'interaction est capable de gérer les contraintes temporelles liées à la négociation par exemple ou les problèmes provenant du module de communication (*CM*) ; ces modules représentent ce qu'on appelle la facette "Interaction" ($i(t)$). Le temps est pris en compte à deux niveaux dans l'interaction :

1. dans la négociation avec les contraintes temporelles associées,
2. dans la communication puisque les contraintes liées à cette négociation doivent être transmises.

Pour la facette "Raisonnement Interne", un module spécifique est dédié à l'ordonnancement des activités de l'agent : le *Situation Assessment Module* (SAM).

L'architecture d'agent ADEPT permet donc une prise en compte du temps à la fois au niveau du raisonnement interne ($a(t)$) et de la facette interaction ($i(t)$) (cf. figure 3.4).

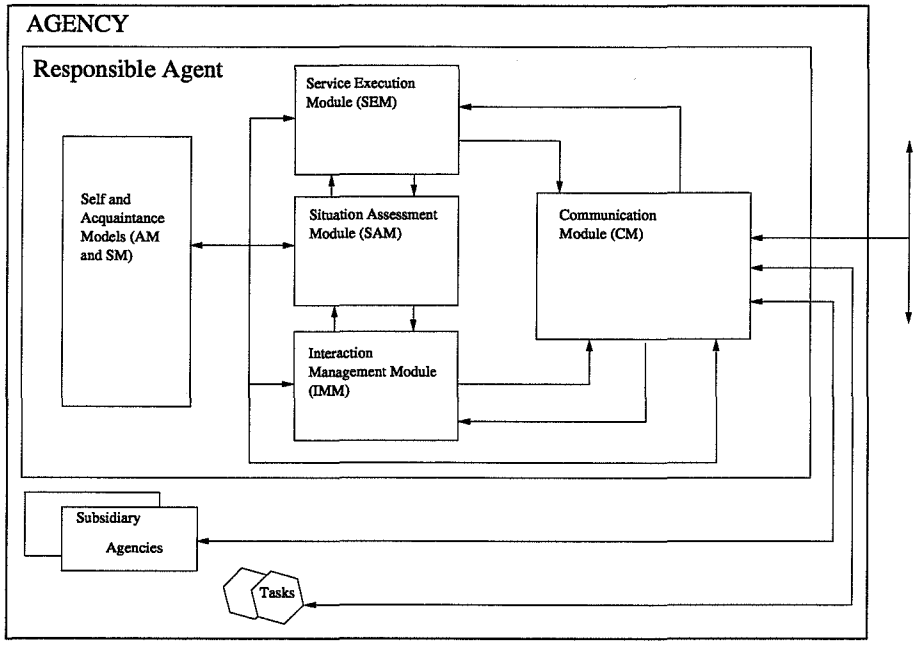


FIG. 3.3 – Architecture d'agent ADEPT

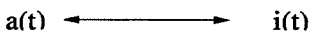


FIG. 3.4 – Prise en compte du temps dans l'architecture d'agent ADEPT

Ces informations sont échangées avec le raisonnement interne et plus particulièrement utilisées ou spécifiées lors de l'ordonnancement des activités. Encore une fois, peu d'informations sont données quant à la prise en compte du temps (représentation) au sein de l'interaction.
A présent, nous allons voir une architecture d'agent tenant compte des aspects temporels également au sein de l'organisation.

3.5.3 Temps pris en compte dans les facettes "aio"

Peu de travaux s'intéressent en outre au temps dans l'organisation. Parmi ceux que nous avons vu, nous pouvons tout de même citer le modèle d'agent proposé par Moulin et Boury-Brisset. Cependant, l'aspect organisationnel n'est pas clairement défini ou séparé des autres aspects. L'architecture d'agent *STAx* que nous présentons ici, propose, elle, chacune des dimensions clairement explicitée.

Le système STARS propose un modèle d'agent temporel spécifique appelé *STAx*. Dans ce système, les agents STAx [ABS00] ont été développés pour la supervision de systèmes industriels et évoluent dans un environnement dynamique, le système supervisé. Ce dernier émet au fil de son exécution différents événements que les agents intéressés captent et interprètent. Selon les déductions réalisées, ils coopèrent entre eux pour construire un état cohérent de l'exécution du système.

De manière similaire à ce que nous avons vu dans Rousseau [Rou95], deux niveaux sont distingués au sein du raisonnement interne de l'agent : le *niveau individuel* et le *niveau social.* Au sein du raisonnement interne, les contraintes temporelles sont prises en compte à la fois au niveau de l'exécution des différentes tâches et aussi plus précisément au niveau de la gestion des buts. Les deux aspects de la composante temporelle (*absolu* : durée d'une tâche, ancrage dans le temps global ; *relatif* : positionnement par rapport aux autres) sont pris en compte.

L'architecture de l'agent temporel comporte une structure d'état mental qui désigne toute la partie dynamique des différentes représentations et connaissances de l'agent. A chaque type de structure mentale (croyance, but, engagement social, plan, tâche, etc.) est associé un (ou plusieurs) champ temporel spécifique (estampille, durée, délai, date, etc.). La granularité dépend du domaine d'application. En l'occurrence, l'exemple d'application réalisée porte sur la reconnaissance de scénarios pour la supervision d'une flotte de bus. Ces scénarios sont exprimés sous forme de graphes de contraintes temporelles entre des événements qui devraient être émis par le système supervisé dans un contexte de fonctionnement correct. L'architecture de l'agent est basée sur un gestionnaire de tâches qui génère des contraintes temporelles (deadlines) et vérifie leur satisfaction pour des scénarios activés. Un planificateur construit les plans pour la reconnaissance des scénarios dans le raisonnement interne et la structure du temps est linéaire à base d'instants.
Sur le plan organisationnel, les agents doivent exécuter leurs tâches tout en restant cohérents avec les autres agents du système. Comme nous l'avons vu avec le système STARS (cf. section 3.3.1), l'approche propose une représentation explicite des aspects dynamiques dans un réseau de dépendance et offre des moyens pour utiliser ces aspects dans un cadre de raisonnement temporel social. Les règles de raisonnement temporel social (sur les différents types de dépendances) génèrent ainsi les interactions des agents. Du point de vue de l'interaction, un protocole de coopération particulier est associé à chaque type de dépendance. Le langage d'interaction permet d'exprimer des contraintes temporelles (le délai à ne pas dépasser pour la réponse par exemple).

Comme nous l'avons souligné précédemment, l'agent *STAx* prend en compte la dimension temporelle au sein des facettes a, e, i et o. La facette Environnement correspond au module *perception* (cf. figure 3.5) qui gère la détection des changements de l'état des ressources dans l'environnement et génère en sortie le type de l'événement qui s'est produit ainsi que sa date d'observation. Ce module est donc capable de prendre en compte de manière explicite le temps. La facette interaction est représentée par le module de *Réception* et celui de *Communication.* Sur le plan du raisonnement interne, le temps est pris en compte au sein d'un *Gestionnaire de Tâches* et du système de *Planification* qui est notamment chargé de l'exécution des plans en utilisant les délais générés par le gestionnaire de tâches. Enfin pour terminer, le temps est pris en compte au niveau de la facette organisation, par le

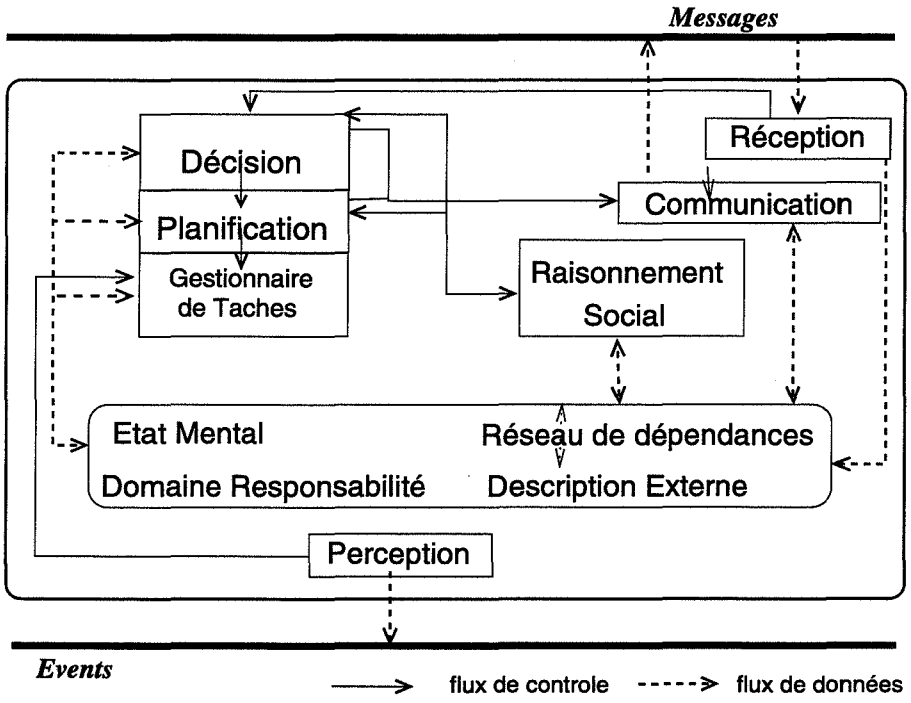


FIG. 3.5 – Architecture d'agent STAx

biais du système de raisonnement social avec la gestion du temps dans les relations de dépendances.

Dans le cadre de cette architecture, nous pourrions représenter la prise en compte du temps de la manière représentée sur la figure[9] 3.6.

la facette agent ($a(t)$) s'appuie sur les informations temporelles des facettes organisation ($o(t)$) et interaction ($i(t)$) pour gérer les aspects temporels relatifs au raisonnement interne.

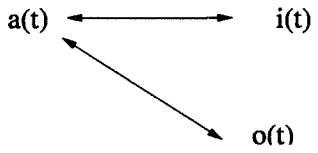


FIG. 3.6 – Prise en compte du temps dans l'architecture d'agent STAx

Ce modèle d'agent temporel présente donc une prise en compte assez complète du temps dans l'agent. Nous pouvons cependant remarquer que le langage temporel est limité (des dates ou des délais), que la dimension organisationnelle correspond à la gestion des dépendances temporelles mais ne propose pas de structure organisationnelle temporelle.

[9]Nous ne reprenons pas la facette $e(t)$ sur cette figure car son fonctionnement est peu décrit.

L'étude des architectures d'agents temporels que nous avons esquissée dans cette section nous permet de montrer qu'il est possible de reprendre la classification des différentes architectures d'agents proposée dans [Boi01] (cf. fin de la section 1.2.4) pour la compléter par la prise en compte de la dimension temporelle et distinguer ainsi trois types d'agents temporels :

1. le premier type d'agent pouvant posséder une composante temporelle serait un *agent autonome*, c'est à dire un agent comportant uniquement les facettes "a" et "e". Un *agent autonome temporel* est donc un agent susceptible de gérer le temps dans deux facettes "$a(t)$" et/ou "$e(t)$". Dans les systèmes décrits ci-dessus, *Guardian* ou *ActeM* relèveraient de cette appellation,
2. les *agents interagissants* sont des agents possédant la facette "i". Ainsi un *agent interagissant temporel* serait un agent dans lequel l'une des facettes au moins gère le temps "$a(t)$", "$e(t)$" et/ou "$i(t)$". Concernant la facette "$i(t)$", cela peut se concrétiser par la prise en compte d'un langage d'interaction temporel, la mise en place et le suivi de protocoles d'interactions temporels et/ou à un niveau supérieur par la gestion de conversations en tenant compte des contraintes temporelles qui en émanent. L'architecture d'agent d'*ADEPT* ou d'*AOP* est un exemple de ce type d'agent (même si, pour ce dernier, il n'y a pas de raisonnement lié à l'interaction),
3. un *agent social* correspond à un agent interagissant présentant en outre des capacités de gestion des relations entretenues avec les autres agents du système et des capacités de structuration entre eux. Ainsi un *agent social temporel* présente des capacités de gérer et représenter le temps dans l'une au moins des facettes "$a(t)$", "$e(t)$", "$i(t)$" et/ou "$o(t)$". Comme nous l'avons vu, l'architecture d'agent *STAx* relève de cette dernière catégorie.

En outre, chacune des architectures décrites propose une façon particulière de prendre en compte le temps dans l'agent et conforte ainsi les différents aspects explicités par le biais de l'approche *AEIO*.

3.6 Synthèse-Discussion

Dans ce chapitre, grâce à la grille *AEIO* que nous nous étions définie, nous avons pu faire apparaître la diversité des travaux temporels au sein d'un SMA. Comme nous l'avons décrit au chapitre 1, en projetant les dimensions A, E, I et O en *facettes* a, e, i, o au sein de l'agent, nous avons pu compléter cette approche, en nous arrêtant sur quelques architectures d'agents. Nous avons ainsi pu en faire ressortir la prise en compte du temps par facettes selon notre point de vue ainsi que la manière dont est géré le temps. Chacun de ces travaux présente une manière particulière de représenter et raisonner sur le temps ou un aspect particulier de la composante temporelle.
L'approche *AEIO* se révèle donc très utile et il devient ainsi beaucoup plus clair

de voir comment le temps a été pris en compte jusqu'à présent dans les SMA. Forts de cette étude, nous pouvons à présent en faire une synthèse et élaborer une proposition de prise en compte de la composante temporelle dans les SMA adaptée à notre cadre de travail et notre contexte d'étude.
Dans ce paragraphe, nous rassemblons les différents travaux cités dans le tableau[10] ci-dessous. Nous en aurons une vue plus générale mais évidemment réductrice de la prise en compte des aspects temporels dans les SMA. Les systèmes sont classés suivant la prise en compte du temps au sein de chaque dimension (AIO). Nous avons également fait apparaître en gras, les systèmes dans lesquels le temps était pris en compte dans d'autres dimensions que celle que nous avons choisie de mettre en exergue.

Place du temps	**Systèmes étudiés**
Temps dans A	Guardian, Alarms, Actem
Temps dans I	ACL FIPA, **AOP**, **ADEPT**, DeL00, NRPC, GOAL, TRAINS
Temps dans O	TeamPlan, **STAx**, **MAT**, **JTV**

L'état de l'art que nous venons de réaliser met en lumière plusieurs points. Certains concernent les SMA de manière générale :

- le temps dans les SMA est une composante complexe. Nous pensons que les dimensions A, E, I et O sont un outil d'analyse intéressant pour l'appréhender plus facilement.

d'autres, les dimensions AEIO :

- différentes catégories d'agent existent selon leur capacité à prendre en compte la composante temporelle issue de chacune des dimensions I, O ou E. A notre connaissance, il n'existe pas d'agent temporel capable de gérer le temps explicitement exprimé dans ces quatre dimensions au sein du SMA.
- de plus, nous n'avons pas trouvé d'outils pour prendre en compte le temps explicitement dans l'interaction (comme par exemple un langage d'interaction temporel) ni dans l'organisation,

enfin, les autres points concernent la théorie et le raisonnement :

- dans la plupart des travaux dans les SMA où le temps est présent, ce dernier est considéré comme un paramètre quelconque au même titre que pourrait l'être d'autres paramètres comme le coût, la qualité, la fiabilité, etc. Nous pensons cependant qu'il est possible de tirer profit de l'ajout de la composante temporelle à des fins de coordination comme nous le verrons au chapitre 7.7,
- à la lumière de ces travaux, la théorie du temps utilisée semble avoir peu d'importance puisque ces derniers s'appuient sur diverses théories apparem-

[10]Une description plus complète de certains systèmes (seulement évoqués ici) peut être trouvée dans [CBS02].

ment indifféremment. En outre, lorsque le temps est abordé dans les différentes dimensions, généralement une représentation uniforme est utilisée. Nous pouvons en déduire qu'il est possible d'utiliser la même théorie dans les différentes dimensions afin d'avoir une représentation uniforme du temps dans l'agent,

- enfin, les descriptions que nous avons faites, soulignent que, même si la composante temporelle n'est pas prise en compte au sein de toutes les dimensions, elle n'est pas non plus indépendante, au sein de la dimension où elle existe, des autres dimensions. Elle apparaît souvent tributaire d'une autre dimension (agent généralement).

Ces remarques nous ont incités à nous demander ce que pourrait être un *SMA temporel* dans lequel le temps serait présent dans chacune de ces dimensions. Il apparaît qu'afin de pouvoir expliciter le temps au sein de chacune des dimensions, des outils spécifiques (utilisant si possible une même représentation du temps) doivent être développés et créés en s'appuyant sur des techniques ou des théories existantes. Dans la partie suivante, nous proposons donc un modèle de SMA temporel où le facteur dynamique est pris en compte explicitement au sein de l'interaction, de l'organisation et offrant la possibilité de mettre en place différents types d'agent temporel.

Deuxième partie

Modèle de système multi-agents temporel

Introduction

Dans ce qui suit, nous allons présenter à partir des conclusions des travaux précédents quelques propositions permettant de mettre en œuvre un SMA temporel. Cette partie est divisée en 4 chapitres qui proposent chacun un aspect particulier de la composante temporelle dans un SMA :

- nous proposons dans le chapitre qui suit, un langage d'expressions temporelles qui sera l'outil de base pour expliciter le temps dans le SMA ainsi qu'un gestionnaire de relations entre intervalles qui offrira un premier moyen pour la mise en place d'un raisonnement temporel.
- le second chapitre définira un modèle d'interaction temporel en présentant un langage d'interaction temporel. La composante temporelle sera également prise en compte explicitement au niveau des protocoles d'interaction et des conversations qu'ils contrôlent.
- le troisième chapitre développera un modèle d'organisation temporelle basé principalement sur la spécification d'un langage de structures organisationnelles temporelles s'appuyant sur le modèle organisationnel *MOISE*.
- le quatrième et dernier chapitre de cette partie présentera un modèle d'agent temporel permettant d'intégrer et de mettre en œuvre les modèles précédemment spécifiés. Ce modèle d'agent s'appuie sur une architecture *a, e, i, o* pour intégrer le temps dans toutes les facettes que nous avions décrites en première partie.

Nous avons vu précédemment qu'en fonction des besoins (domaines d'applications, objectifs, etc.), cette composante temporelle peut intervenir de manières très diverses et que les modèles et outils à ce sujet, manquaient dans les SMA. L'objectif de cette partie est de proposer des modèles temporels qui permettent de définir un système multi-agent temporel où la composante temporelle peut être explicitée quelle que soit les dimensions dans laquelle elle va intervenir.

4

Outils pour représenter et manipuler le temps

Sommaire

4.1 Introduction et Hypothèses

Comme nous l'avons vu précédemment au chapitre 2, la prise en compte de la dimension temporelle nécessite auparavant une définition précise des aspects temporels que l'on souhaite pouvoir représenter et éventuellement manipuler. Nous décrirons donc dans ce chapitre, les différents outils que nous avons été amenés à élaborer dans notre contexte de travail. En l'occurrence, nous présenterons un langage d'expressions temporelles et un gestionnaire de relations entre intervalles. Afin de bien comprendre les choix qui ont été faits, nous allons pour commencer décrire un certain nombre d'hypothèses que nous avons posées.
Tout d'abord, certaines particularités des systèmes multi-agents par rapport aux systèmes "classiques" imposent de nouvelles contraintes. Lorsque l'on s'intéresse à la composante temporelle, il est fondamental que les différents agents manipulent le même temps. En d'autres termes, nous faisons l'hypothèse que **les horloges des différents agents du système sont synchronisées**. En effet, suivant la *granularité* du temps considérée (millisecondes, minutes, jours, mois, etc.), un écart plus ou moins important entre les horloges de deux agents peut se révéler catastrophique. Ce problème est très complexe en lui-même puisque nous sommes actuellement incapables de prédire absolument le temps du parcours d'un mes-

sage sur le réseau du fait de l'architecture même et des protocoles bas-niveau d'Internet. Heureusement, il existe un certain nombre de logiciels permettant de synchroniser des machines avec une précision satisfaisante. Dans le cadre de notre travail, nous ferons l'hypothèse que nous obtenons cette prècision (cf. chapitre 8 sur l'implémentation). A titre informatif, de tels logiciels permettent d'obtenir des précisions de l'ordre du millième voire du millionième de seconde sur les horloges de référence (Los alamos, Paris, etc) mais au prix d'une consommation réseau et mémoire très lourde. Dans le cadre de notre étude, le centième de seconde sera déjà amplement suffisant.
Une autre hypothèse que nous faisons, concerne les fuseaux horaires puisque nous nous intéressons à des systèmes industriels dont les diverses unités peuvent éventuellement se trouver dans des pays différents. Cette fois-ci encore, bien qu'il soit bien moins difficile à résoudre, nous ne prenons pas en compte ce problème mais nous précisons juste que des outils logiciels permettent de s'en affranchir ou du moins de le gérer de manière transparente pour l'utilisateur.
Pour terminer avec les hypothèses que nous posons, nous compléterons cette introduction en précisant que nous considérons des agents "bienveillants" (*benevolent*) ; c'est à dire qu'ils ne cherchent pas à tromper délibérément les autres agents que ce soit d'ailleurs, au niveau du temps ou à tout autre sujet.

4.2 Langage d'expressions temporelles

Dans les systèmes fortement dynamiques, le temps peut prendre différents aspects. Comme nous l'avons vu précédemment, cette composante peut se retrouver au sein des Agents, de l'Environnement, des Interactions et de l'Organisation. Pour être en mesure d'exprimer les contraintes temporelles susceptibles d'apparaître dans le cadre des systèmes que nous étudions, il est nécessaire de définir un langage temporel. A ce sujet, il est important de bien spécifier notre contexte de travail au sujet de la composante temporelle pour définir nos besoins. Ainsi le langage temporel que nous avons défini, s'appuie sur les éléments [11] suivants :

- prise en compte explicite de la composante temporelle afin de pouvoir ensuite manipuler facilement les données temporelles et tirer profit de l'ajout de cette composante.
- uniformisation des éléments d'un SMA susceptibles d'être complétées par une composante temporelle afin que des informations temporelles puissent être échangées et manipulées quelle que soit leur dimension (*AEIO*) de provenance. Par exemple, les contraintes temporelles issues du raisonnement interne seront au même format que celles qui seront utilisées par l'interaction.
- approche plutôt descriptive que relationnelle (cf. section 2.3.1) permettant

[11] Certains de ces besoins sont directement déduits du domaine d'application (les systèmes industriels) auxquels nous nous intéressons.

une manipulation plus intuitive des données mais en contrepartie évidemment moins concise.
- utilisation d'un temps linéaire, continu, infini (cf. section 2.3.2) qui nous permet de simplifier sa manipulation et qui convient pour l'instant, aux systèmes auxquels nous nous intéressons.
- raisonnement sur le temps car le temps nécessaire au calcul n'est pas crucial dans le cadre des systèmes auxquels nous nous intéressons.
- raisonnement à visée pragmatique car les résultats seront utilisés pour engendrer des objectifs d'actions, produire des plans.

Sur la base de ces hypothèses, nous avons donc choisi, afin d'expliciter et représenter les dimensions temporelles dans un SMA, de nous appuyer sur le calcul des intervalles d'Allen (cf. chapitre 2) [All84]. La primitive temporelle est donc l'*intervalle*. Ce choix est justifié par le fait que l'intervalle est plus expressif et mieux adapté à la représentation des états du monde. De plus, la théorie d'Allen fournit une algèbre des intervalles qui permet de manipuler plus aisément et de manière plus intuitive des expressions temporelles complexes.
Nous rappelons ou précisons que l'objectif n'est pas d'élaborer une nouvelle technique de raisonnement temporel puisque celle-ci peut différer en fonction de l'objectif de l'application définie mais de proposer un langage sur lequel différentes techniques de raisonnement existantes pourraient être adaptées. Nous présenterons en section 4.3, un outil pour manipuler les expressions temporelles construites avec ce langage.

4.2.1 Expressions temporelles

Ce langage étant susceptible d'être utilisé pour exprimer le temps dans différentes dimensions d'un SMA, nous faisons apparaître dans sa définition, différents termes issus d'autres langages dédiés à :

- l'expression d'actions (nommé L_α) : actions dans l'environnement, actions internes à l'agent (cf. chapitre 7), actions de communication (speech act) telles que nous pourrons les définir plus loin (cf. chapitre 5), actions de réorganisation.
- l'expression de formules de description du monde et des états mentaux des agents (appelé L_ϕ). Ce langage s'inspire de la théorie *BDI* (cf. section 1.2). Une définition précise de ce langage est donnée ci-après.
- l'expression de composantes organisationnelles (nommé L_ω) telles que rôle, lien, groupe que nous introduirons au chapitre 6.

Nous allons donc commencer par décrire les langages L_α et L_ϕ.

Actions

Le langage d'actions, L_α, permet d'exprimer les actions qu'un agent est capable de réaliser. Le langage d'actions est défini par les formules BNF suivantes :

$\alpha ::= \underline{a} \mid \underline{sa} \mid \alpha_1;\alpha_2 \mid \alpha_1 \& \alpha_2 \mid \alpha_1 \vee \alpha_2 \mid \neg\alpha \mid \mathsf{any} \mid any(\phi)$

$\underline{a} \in L_{act}$ et $\underline{sa} \in L_{speech}$.

L_{act} est l'ensemble de toutes les actions atomiques que les agents peuvent échanger et exécuter localement. L_{speech} sera décrit dans le chapitre suivant.

La signification de $\alpha_1;\alpha_2$ est l'exécution séquentielle des actions α_1 et α_2 ; de $\alpha_1 \& \alpha_2$ est l'exécution en parallèle des deux actions ; $\alpha_1 \vee \alpha_2$ correspond au choix entre les actions α_1 et α_2. L'expression $\neg\alpha$ signifie la non-exécution de l'action α afin de pouvoir interdire une action. L'action any est une action universelle. Enfin, l'action $any(\phi)$ correspond à l'utilisation de n'importe quelle action qui conduit à l'état du monde décrit par ϕ. A ce niveau, nos travaux s'inspirent de ceux menés par [DW95], bien que cet aspect ne soit pas aussi explicite dans ces-derniers.

Formules

Afin de décrire le monde et les états mentaux des agents, nous définissons le langage L_ϕ. Ce langage utilise des opérateurs semblables à ceux que l'on peut trouver dans la théorie *BDI* ("Belief-Desire-Intention") (cf. section 1.2) [GL87]. Cependant certaines modalités diffèrent puisque nos travaux s'appuient également sur le plan temporel sur les travaux de Allen et de Shoham entre autres. Ils nous permettent de compléter la description du monde au moyen d'un opérateur d'engagement social et de capacité [Sho88]. Une formule de L_ϕ est définie de la manière suivante :

$\phi ::= \underline{\phi} \mid G(x, exp, \phi) \mid B(x, exp, \phi) \mid I(x, exp, \alpha) \mid S\_C(x, y, exp, \alpha)$

avec $exp \in L_T$ (cf. section Temps ci-dessous).

$\underline{\phi}$ est une formule du premier ordre, $\alpha \in L_\alpha$.

- La signification de $G(x, exp, \phi)$ est que l'agent x possède le but ϕ à n'importe quel moment t tel que exp est vérifié à t. La sémantique est semblable à la notion de désir (desire) définie par la théorie *BDI* [RG95b]. Selon cette dernière, un agent possède un Goal G à l'instant t si et seulement si G est vrai dans tous les mondes dans lesquels l'agent désire se trouver (*Goal-accessible Worlds*) à l'instant t. S'étant inspirés de ce modèle, nous considérons que le but s'étendant sur un intervalle de temps i représente l'état du monde qu'il souhaite atteindre, tout au long de cet intervalle i.
- La signification de $B(x, exp, \phi)$ est que l'agent x croit ϕ à n'importe quel moment t tel que exp est vérifié à t. Selon la théorie *BDI*[RG95b], un agent possède une croyance ϕ, dénotée $Bel(\phi)$, à l'instant t si et seulement si ϕ est vraie dans tous les mondes "Belief-accessible" de l'agent à l'instant t.
- $I(x, exp, \alpha)$ signifie que l'agent x a l'intention de réaliser α afin que l'expression temporelle exp soit satisfaite.

- $S\_C(x, y, exp, \alpha)$ signifie que l'agent x est engagé auprès de l'agent y à exécuter l'action α de manière à ce que exp soit vérifiée.

Nous définissons la modalité Can afin d'exprimer la capacité d'un agent à exécuter une action comme suit : $Can(x, exp, \alpha)$.

Temps

Le langage L_T définit des expressions temporelles exp. La sémantique de ce langage est semblable à celle définie par Allen (cf. section 2.3.3) puisqu'il s'appuie sur son algèbre. Pour cela, nous utilisons un raccourci implicite : une expression temporelle est considérée comme un intervalle complexe et non pas comme une valeur booléenne. Le calcul de cet intervalle est réalisé de telle sorte que l'interprétation de cette expression temporelle soit égale à vraie. Les expressions temporelles peuvent être simples $\underline{exp}$ ou composées avec des opérateurs de conjonction $\wedge$ ou disjonction $\vee$:

$exp ::= \underline{exp} \mid exp_1 \wedge exp_2 \mid exp_1 \vee exp_2$
$\underline{exp} ::= i \mid [\alpha] \mid [\phi] \mid [\omega] \mid \underline{exp}_1\ \mathcal{R}\ \underline{exp}_2 \mid (\alpha \mid \phi \mid \omega)$ periodic $\underline{exp}_2\ \underline{exp}_3$

Une expression temporelle simple $\underline{exp}$ correspond à une expression temporelle de base $(i, [\alpha], [\phi], [\omega])$ ou au positionnement relatif de deux expressions temporelles simples à l'aide d'opérateurs temporels.

Les expressions temporelles de base sont :

- un intervalle temporel i qui peut être une durée d ou un intervalle avec un instant de début t_s et un instant de fin t_e :
  $i ::= [d] \mid [t_s, t_e]$
  L'expression $[d]$ est équivalente à l'intervalle suivant $[t_?, t_? + d]$ où $t_?$ est un instant indéterminé. Nous utiliserons t_{now} pour désigner le temps courant et i_+ pour dénoter l'intervalle $[t_{now}, +\infty[$. Le format d'implémentation du temps correspond au format iso8601 dans les exemples qui suivront. Cependant, afin de simplifer ces derniers, nous présenterons le temps dans un format restreint (c'est à dire ne reprenant pas systématiquement l'année, le mois, etc) ou bien de manière explicite afin de lever des ambiguïtés en fonction des besoins.
- les intervalles $[\alpha]$, $[\phi]$ ou $[\omega]$ dont les limites sont le point de début et de fin de, respectivement, l'exécution de l'action α ($\alpha \in L_\alpha$), de l'état vrai de la proposition ϕ ($\phi \in L_\phi$), ou de la validité du terme organisationnel ω ($\omega \in L_\omega$) concerné (validité d'un rôle, d'un lien, etc).

Les relations temporelles sont celles définies dans [All84] :

$\mathcal{R} ::= <$ | $>$ | $=$ | m | mi | o | oi | d | di | s | si | f | fi | beg | end | in

La signification de $<$, $>$, $=$, m, mi, o, oi, d, di, s, si, f, fi est la même que celle définie par Allen, (avant, après, égal, rencontre (meet), rencontre-inverse, recouvre, recouvre-inverse, pendant, pendant-inverse, débute, débute-inverse, termine, termine-inverse).
Pour des raisons de commodités, nous avons également défini l'opérateur triadique periodic qui requiert une action α, une formule ϕ ou un terme organisationnel ω comme premier terme. Cet opérateur est utilisé pour spécifier une exécution périodique visant à satisfaire l'expression du premier terme avec une fréquence exprimée dans le second terme et pendant un intervalle spécifié dans le troisième terme.

Nous illustrons l'utilisation de ce langage dans la section suivante.

4.2.2 Exemples

Les exemples qui suivent, permettent d'avoir une idée de l'utilisation de ce langage. Ces exemples sont issus du domaine de l'imprimerie car nous travaillons sur une application portant sur des alliances d'ateliers d'impression que nous présenterons dans la troisième partie.

- "[Print] in $[10, 20]$" signifie que l'action Print doit être exécutée à l'intérieur de l'intervalle temporel $[10, 20]$,
- "[ink filled up to 75%] = $[10, 20]$" signifie que l'état "ink filled up to 75%" doit être maintenu pendant un intervalle égal à $[10, 20]$.

Des expressions plus complexes peuvent être écrites. Imaginons que l'action FillIn pour obtenir l'état "ink filled up to 75%", doive être exécutée pendant l'intervalle $[10, 20]$, et qu'aussitôt que cette action se termine, l'état "ink filled up to 75%" doit être maintenu pendant 20 (cf. figure 4.1). Nous écrirons :

([FillIn(ink filled up to 75%)] in [10,20]) $\wedge$ ([ink filled up to 75%] = [20])$\wedge$
([FillIn(ink filled up to 75%)] m [ink filled up to 75%])

Afin d'être plus clairs, nous avons spécifié dans l'exemple ci-dessus, à la fois l'action et l'état qui en découle.

D'autres exemples apparaîtront par la suite illustrant des notions plus spécifiques à chacune des dimensions prises en compte (c'est à dire relatives à l'interaction, à l'organisation, à l'agent).

Les expressions temporelles pouvant rapidement devenir complexes, nous avons développé [Gau99] un gestionnaire de relations entre intervalles nous permettant de réduire et simplifier ces expressions.

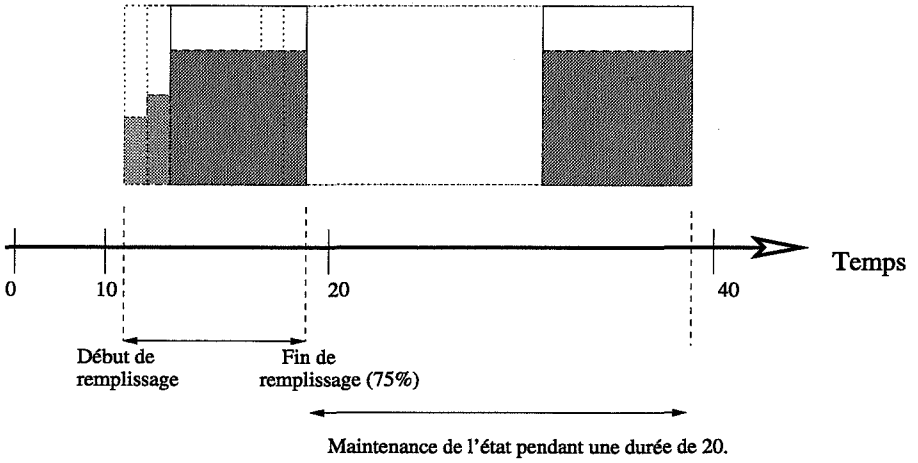


FIG. 4.1 – Exemple de solution illustrant les contraintes définies ci-dessus

4.3 Gestionnaire de relations entre intervalles : GRIS

Comme nous l'avons vu en première partie (cf. section 2.4), la manipulation de données temporelles ou raisonnement temporel nécessite la définition d'outils adaptés. Nous avons défini un outil permettant de manipuler les expressions temporelles spécifiques écrites dans le langage L_T décrit précédemment. Nous ne donnerons ici que les grandes lignes relatives à la définition de cet outil mais ces caractéristiques précises sont disponibles dans [Gau99]. Pour conduire le raisonnement temporel à partir des choix de représentation du temps décrits dans ce qui précède, nous distinguons trois niveaux :

1. **La gestion des relations entre intervalles** qui consiste à assurer la cohérence et la complétude d'un ensemble de relations qui sont les éléments de base menant à un raisonnement plus évolué. Ces relations sont contraintes par la logique des intervalles d'Allen.
   ex : La relation "L'intervalle I est avant l'intervalle J" permettra de déduire "L'intervalle J est après l'intervalle I" en utilisant la connaissance "les relations *avant* et *après* sont inverses dans l'algèbre d'Allen".
2. **La gestion des objets temporels**. Ces objets temporels correspondent aux actions, aux propositions, aux propriétés et de manière générale aux objets définis par les langages L_α, L_ϕ et L_ω. Dans le formalisme d'Allen et Ferguson, ces objets sont représentés dans un langage du premier ordre avec des prédicats. Ce niveau correspond donc à la vérification des contraintes associées à chaque type d'objet temporel.
   ex : "L'imprimante est indisponible ce week-end" permettra de déduire que "L'imprimante est indisponible dimanche" grâce à la connaissance "Un fait vrai sur un intervalle l'est également sur tous ses sous-intervalles".
3. **Le raisonnement causal**. L'objet temporel "retard inattendu dans les impressions, ce matin" signifie "panne de machine avant ce matin" à partir

de connaissance causale du type : "lorsque retard inattendu alors panne de machine avant".

Le raisonnement temporel est complété par la prise en compte de la *granularité* qui se fait en amont du premier niveau puisqu'elle modifie les relations entre les intervalles.
Par exemple, un intervalle I de 2 minutes 30 secondes et un intervalle J de 2 minutes et 01 seconde seront égaux à une granularité égale à la minute mais I sera supérieur si l'on descend à une granularité de l'ordre de la seconde.

Ces trois niveaux ont constitué les trois étapes de la mise en œuvre de chacun des modules de l'outil développé. L'architecture pour le raisonnement temporel proposée par [Gau99] est décrite sur la figure 4.2.

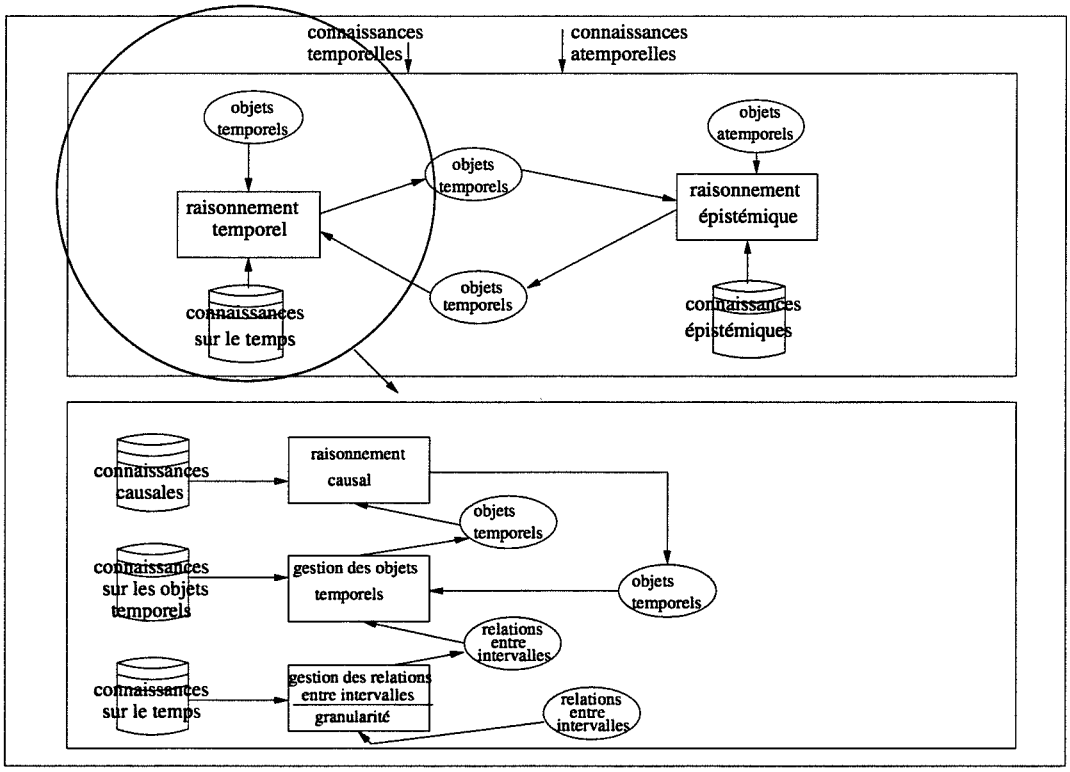


FIG. 4.2 - Une architecture pour le raisonnement temporel

La fonction de gestion de relations entre intervalles est réalisé par un gestionnaire de graphes temporels inspiré des travaux de [GSS94] sur *TimeGraph*. Ce gestionnaire est capable de tenir compte d'une granularité et d'exprimer des informations qualitatives et quantitatives. En effet, l'approche est symbolique mais offre la possibilité de traiter des données métriques et de gérer des relations entre intervalles. Le fonctionnement du gestionnaire de relations entre intervalles (appelé *GRIS*) comporte trois étapes :

1. *La traduction intervalle-instant et instant-intervalle* puisque le gestionnaire adopte pour la représentation en interne des informations temporelles, l'instant et les relations usuelles entre instants ($<$,$\leq$,$=$).
2. *La construction du graphe temporel* qui constitue la représentation structurée de l'ensemble des relations données en entrée.
3. *L'interrogation du graphe* qui est nécessaire car la structure de graphe choisie est non-entièrement connectée (la totalité des informations n'est pas explicitement représentée) permettant ainsi de gagner en temps de calcul et en espace mémoire.

Pour la mise en œuvre de la gestion des objets temporels et du raisonnement causal, un démonstrateur est utilisé. Le démonstrateur automatique choisi est *SPASS* (**S**ynergetic **P**rover **A**ugmenting **S**uperposition with **S**ort). Ce démonstrateur de théorèmes du premier ordre a été développé au *Max-Planck-Institute für Informatik* [Wei98]. Des informations complémentaires notamment sur son fonctionnement peuvent être trouvées dans [Gau99] mais un schéma de l'architecture d'implémentation du gestionnaire de relations entre intervalles est présenté (cf. figure 4.3).

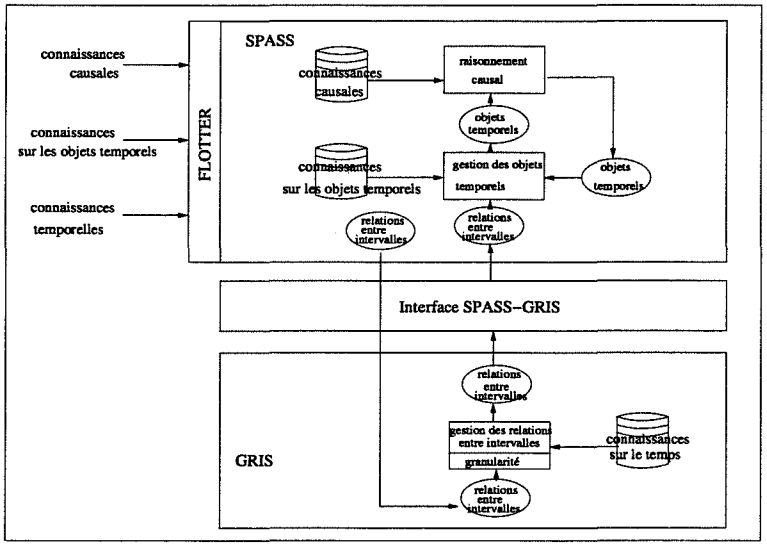


FIG. 4.3 – L'architecture de l'implémentation superposée avec l'architecture conceptuelle

De plus amples informations ainsi que des exemples de fonctionnement et l'interface développée afin de pouvoir l'utiliser indépendamment du SMA sont présentés en annexe A.

A présent, que nous avons défini des outils pour représenter et manipuler les expressions temporelles qui nous intéressent, nous présentons dans le chapitre qui suit notre proposition pour la prise en compte du temps au sein de l'interaction.

5

Interaction temporelle dans les SMA

Sommaire

5.1 Introduction

Le modèle d'interaction temporelle que nous présentons ici reprend successivement les principales composantes que nous avons évoquées précédemment : langage principalement, mais aussi protocole et conversation (cf. section 1.2.3). L'objectif de ce chapitre est de décrire les moyens de communication permettant

l'expression explicite de la composante temporelle dont disposent les agents pour exprimer leurs besoins d'une manière compréhensible par l'ensemble des agents.

La principale contribution de ce chapitre concerne la définition du langage d'interaction temporel *TACL* avec une sémantique basée sur des actes de langage temporels. La première ébauche de ce langage a été défini en DEA [Car98]. Il a ensuite été spécifié de manière formelle, clarifié et complété en thèse. En étudiant les langages existants, nous avons pu mettre en exergue différentes expressions de la sémantique de l'acte de langage : la *spécification opérationnelle* [BD94] qui correspond à une implémentation directement dans le code, la *spécification logique* [LF94] qui propose une formalisation logique des actes en laissant l'interprétation libre et l'*interprétation logique* [Sad91] qui utilise une spécification logique ainsi qu'un moteur d'inférence pour les interpréter. La syntaxe quant à elle, est souvent directement dépendante de la sémantique associée mais il est intéressant de remarquer que certains choix permettent de séparer différents niveaux d'informations afin de distinguer les différentes couches relatives à l'interaction. La sémantique du langage *TACL* s'appuie sur une interprétation logique et la syntaxe associée au langage est semblable à celle du langage qu'utilise *ASIC* [BD94]. Ils seront présentés dans la section suivante.
Afin de contrôler l'échange de messages, nous complétons le modèle d'interaction temporel en décrivant et en spécifiant des protocoles d'interaction temporels ainsi qu'en présentant des aspects temporels liés à la gestion de conversations qui en découlent.

5.2 Langage d'interaction temporel : TACL

Afin que les agents puissent communiquer de manière explicite, ils doivent utiliser un langage commun. Dans le contexte de nos recherches, ce langage d'interaction doit en outre permettre d'expliciter les aspects temporels. Nous considérons que des agents évoluant dans un environnement dynamique, doivent communiquer à la fois *dans le temps* et *au sujet du temps.* Concrètement, les contraintes temporelles échangées par les agents doivent être structurées selon trois niveaux différents : le premier est relatif au routage des messages, le second au processus de coopération entre les agents que mettent en place les échanges et le dernier au contenu proprement dit des messages. Nous avons montré (cf. section 3.2) qu'en dehors du contenu propositionnel, certaines informations portant sur le séquencement des messages, importantes pour le processus de coopération, sont implicites. Si ces notions sont implicites en langage naturel, elles nécessitent d'être explicitées dans un langage artificiel. D'autre part, l'exécution d'une action peut être liée à l'acte de communication : exécuter cette action 10 minutes après la réception de ce message.

1. Le premier niveau (*communication* noté :com, cf. Figure 5.1) correspondant au routage des messages, comporte les dates d'envoi ou de réception des

messages (estampilles), d'éventuelles contraintes temporelles de validité ou de synchronisation dépendant du middleware (infrastructure de communication) utilisé.

2. Le niveau suivant (*multi-agent* noté :mas, cf. Figure 5.1) contraint le processus de coopération entre les agents proprement dit. Dans le contexte d'un langage s'appuyant sur la théorie des actes de langage, cela correspond entre autres choses, à s'intéresser aux aspects temporels relevant de la force illocutoire qui exprime l'intention du message. Nous reviendrons plus en détails sur ce niveau dans ce qui suit.
3. Enfin le troisième et dernier niveau relève de l'*application* et correspond au contenu (noté :content, cf. Figure 5.1) du message lui-même. Les informations qu'il contient seront interprétées par le raisonnement interne du receveur du message.

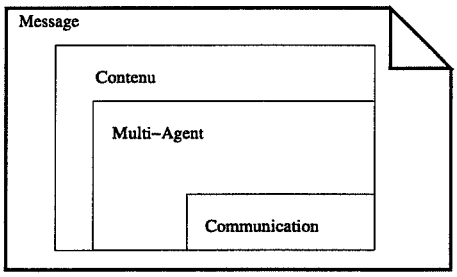


FIG. 5.1 – Structure des contraintes temporelles dans l'interaction

Nous commencerons donc par présenter le langage L_α permettant de décrire les actions ainsi que celui dédié à la description du monde et des états mentaux des agents L_ϕ (cf. section 4.2).

Le langage TACL (Temporal Agent Communication Language) est basé sur les actes de langage qui sont considérés comme des actions de communication paramétrables par une expression temporelle. Notre approche sépare les aspects temporels et atemporels dans le langage. TACL s'appuie sur le langage d'expressions temporelles L_T que nous avons décrit dans le chapitre précédent. Nous allons à présent, décrire un sous-ensemble du langage dédié L_α décrit lors de la présentation de L_T qui permet de définir des actions de communication. Nous poursuivrons par la définition de la sémantique associée aux actes de langage puis nous présenterons la syntaxe du langage *TACL*.

5.2.1 Actions de communication

Nous distinguons au sein du langage d'actions, L_α décrit précédemment (cf. section 4.2.1), deux types d'actions qu'un agent est capable de réaliser : celles

que l'agent peut exécuter ($\underline{a}$) et celles qui sont des *actions de communication* ($\underline{sa}$: speech act). L_{speech} est l'ensemble de toutes les actions de communication (performatifs) qui peuvent être utilisées par les agents pour communiquer.

L_{speech}=$inform, command, propose, refuse, accept, etc.$

Dans le cadre de notre travail, nous avons été amenés à définir les cinq actes de langage ci-dessus mais cette liste n'est pas exhaustive, le langage L_{speech} peut être enrichi facilement puisque d'autres types d'actes peuvent être définis et utilisés en fonction des besoins liées aux applications.

5.2.2 Sémantique des actes de langage

Une sémantique a été définie pour ces actes de langage temporels avec des pré-conditions, des post-conditions et un effet perlocutoire (effet sur le receveur). Dans le cadre de nos travaux, les aspects illocutoire et perlocutoire (cf. section 1.2.3) sont importants car la composante illocutoire permet de représenter de façon explicite l'intention dans la communication, limitant ainsi toute inférence pour sa reconnaissance alors que la composante perlocutoire permet de s'assurer que l'allocutaire répond bien aux attentes du locuteur.

Au niveau sémantique, nous nous sommes orientés vers une approche similaire à celle du langage ARCOL [Sad91] repris dans les spécifications de la FIPA et correspondant à une interprétation logique (cf. introduction de ce chapitre). Une telle formulation est intéressante dans notre cas pour exprimer le temps dans l'interaction comme nous allons le voir par la suite. Notre travail se situe dans le même courant que [Sad91] et [FIP00]. Dans cette section, nous allons décrire la sémantique des différentes actions de communication qui sont définies dans L_{speech}. Cette sémantique n'a pas d'autre objectif que de nous permettre d'exprimer les dimensions temporelles dans la force illocutoire. Elle ne prétend pas notamment définir une nouvelle sémantique pour les actes de langage.

Un acte de langage est une action de communication qui est paramétrée par une expression temporelle ($exp \in L_T$) et une action ($\alpha \in L_\alpha$) ou une formule du premier ordre ($\phi \in L_\phi$) :

$\underline{sa} \in L_{speech}$

$\underline{sa}$: := ⟨name⟩(⟨sender⟩,⟨receiver⟩,⟨exp⟩,ϕ) | ⟨name⟩(⟨sender⟩,⟨receiver⟩,⟨exp⟩,α)

⟨sender⟩ et ⟨receiver⟩ correspondent aux identifiants des agents qui, respectivement envoient et reçoivent l'action de communication. Le receveur (*receiver*) peut éventuellement être un groupe dans le cas d'un *broadcast*.

Une action de communication $\underline{sa}$ doit correspondre à un état mental de l'agent afin de savoir si elle est possible. Il faut également définir explicitement les modifications à faire sur cet état mental lorsque l'acte de langage est réussi. C'est

pourquoi nous définissons une action de communication comme un triplet ⟨ Prec, Post, Pe ⟩ où Prec, Post et Pe sont respectivement les pré-conditions, les post-conditions et l'effet perlocutoire. Chacun d'entre eux est exprimé à partir des définitions ci-dessous.

- Prec : ensemble des conditions nécessaires pour déclencher l'action. Elles correspondent aux *conditions sur le contenu propositionnel*, *aux conditions préparatoires* et *aux conditions de sincérité* décrites dans [Van88].
- Post : ensemble des *postconditions*. Il est composé s'une liste d'ajouts (+) et d'une liste de retraits (-), qui doivent être effectués sur l'état mental de l'émetteur.
- Pe : description de l'état mental souhaité au niveau du receveur.

Actes de langage temporels

Dans la suite, afin de simplifier la présentation, nous utiliserons i, j, k, variables existentielles, pour exprimer les expressions temporelles telles que celles définies précédemment (cf. section 4.2). Un "prime" est ajouté à une expression i, afin d'exprimer que i' est calculée à partir de i, de t_{now} et de la durée d'envoi du message. En effet, différentes stratégies peuvent être définies pour intégrer un état mental comportant une composante temporelle. Par exemple, si un agent envoie un message spécifiant qu'il croit qu'il fait beau à t=10 unités de temps ; si le temps de transport sur le réseau prend 2 unités de temps, l'agent recevra l'information à t=12, il pourra donc en fonction de sa stratégie d'intégration soit considérer qu'il fait beau depuis l'envoi du message (10) ou depuis sa réception (12). Des exemples et des commentaires sur ces stratégies ou algorithmes de mises à jour des états mentaux correspondants sont donnés en annexe B. Présentons maintenant, quelques exemples d'actes de langages. Cet ensemble correspond aux actes dont nous avons besoin pour les applications que nous présenterons en troisième partie. Il n'a pas été nécessaire de spécifier des actes de type déclaratif et expressif plus spécifiques au traitement du langage naturel (cf. section 1.2). Cependant la formalisation d'autres actes peut être trouvée dans [Car98].

Actes Assertifs

L'action de communication inform est un acte assertif qui est utilisé par l'agent x pour informer l'agent y de la vérité de ϕ.

inform(x, y, i, ϕ)
Prec : $B(x, i, \phi) \wedge B(x, j, \neg B(y, k, \phi)) \wedge t_{now} \in i \cap j \cap k$
Post : (+) $B(x, i', B(y, i'', \phi))$ (-) $B(x, j, \neg B(y, k, \phi))$
Pe : $B(y, i'', \phi)$

Pour déclencher un acte *inform*, l'agent doit croire un fait et croire que ce fait

n'est pas connu de l'autre agent. Cet état du monde correspond aux pré-conditions (*Prec*).
Aprés l'envoi du message de type *inform*, l'état mental de l'agent est modifié de la manière suivante : la croyance sur le fait inconnu par l'autre agent est retiré pour être remplacé par la croyance, qu'à présent, l'autre agent connaît le fait. Les postconditions (*Post*) définissent cette mise à jour.
Enfin l'agent receveur possédera un nouvel état mental correspondant à l'effet perlocutoire (*PE*) : la croyance sur le fait pendant l'instant spécifié.

Actes Directifs

- L'action communicative command est un acte directif qui engage le receveur à exécuter l'action exprimée dans les paramètres.

  command(x, y, i, α)
  Prec : $I(x, i, \alpha) \wedge B(x, j, Can(y, k, \alpha)) \wedge (k \text{ in } i) \wedge t_{now} \in i \cap j \cap k$
  Post : (+) $B(x, i', S\_C(y, x, i'', \alpha))$ (-) $I(x, i, \alpha)$
  Pe : $S\_C(y, x, i'', \alpha)$

  Les préconditions spécifient que pour envoyer un acte de type directif, l'agent doit avoir l'intention qu'une action soit faite et croire que cette action peut être réalisée par l'autre agent. Les postconditions et l'effet perlocutoire concernent ensuite l'engagement par l'autre agent à effectuer l'action.

- L'acte propose est un acte directif permettant à l'émetteur de demander au receveur d'exécuter l'action α dans le futur avec la possibilité de refuser. Les post-conditions et l'effet perlocutoire sont modifiés par rapport à l'acte "command" puisque l'allocutaire y ne devra possèder que l'intention d'accomplir la tâche mais sans y être formellement engagé vis à vis de x.

  propose(x, y, i, α)
  Prec : $I(x, i, \alpha) \wedge B(x, j, Can(y, k, \alpha)) \wedge (k \text{ in } i) \wedge t_{now} \in i \cap j \cap k$
  Post : (+) $B(x, i', I(y, i'', S\_C(y, x, i'', \alpha)))$ (-) $I(x, i, \alpha)$
  Pe : $I(y, i'', S\_C(y, x, i'', \alpha))$

Actes Commissifs

- L'acte accept est un acte commissif permettant à l'émetteur de s'engager vis à vis du receveur à exécuter une action dans le futur.

  accept$(x. y, i, \alpha)$
  Prec : $B(x, k, I(y, j, S\_C(x, y, i, \alpha))) \wedge B(x, l, S\_C(x, y, i, \alpha)) \wedge t_{now} \in k \cap l$
  Post : (-) $B(x, k, I(y, j, S\_C(x, y, i, \alpha)))$
  Pe : $B(y, l', S\_C(x, y, i, \alpha)))$

Les préconditions spécifient que, pour envoyer un acte de type commissif, l'agent x (émetteur) doit savoir que l'agent y (allocutaire) a l'intention que l'agent x s'engage auprès de l'agent y pour réaliser une action α et que l'agent x (émetteur) doit croire qu'il s'engage à réaliser l'action α. Les post-conditions et l'effet perlocutoire concernent ensuite le retrait de l'intention d'engagement chez l'agent x (émetteur) et l'ajout de la connaissance d'engagement chez l'agent y (allocutaire).

- L'acte refuse est un acte commissif permettant à l'émetteur de refuser de s'engager vis à vis du receveur à exécuter une action dans le futur.

refuse(x, y, i, α) :
Prec : $B(x, k, I(y, j, S\_C(x, y, i, \alpha))) \wedge B(x, l, \neg S\_C(x, y, i, \alpha)) \wedge t_{now} \in k \cap l$
Post : (-) $B(x, k, I(y, j, S\_C(x, y, i, \alpha)))$
Pe : $B(y, l', \neg S\_C(x, y, i, \alpha))$

La définition de l'acte refuse est semblable à la précédente (acte accept) mais l'agent x (émetteur) possède la connaissance qu'il ne s'engage pas à exécuter l'action α.

Exemples

En revenant sur les exemples d'interactions temporelles donnés dans la section 3.2, nous allons montrer comment nous pouvons en exprimer certains avec les définitions ci-dessus.

Supposons que l'agent A utilise un acte de type command pour demander à l'agent B l'heure actuelle ou ce qu'il fera la semaine prochaine. Pour simplifier, nous restreignons le format temporel à heure/minutes/secondes (hh :mm :ss).

1. Le premier cas s'écrira de la manière suivante :

   command(A, B, [inform] in [t_{now},t_{now}+0 :0 :10], inform(current_time))

   A demande à B de l'informer dans les 10 secondes au sujet de l'heure actuelle : L'action communicative inform doit se situer à l'intérieur de l'intervalle [t_{now},t_{now}+0 :0 :10].

2. Le second cas sera écrit :

   command(A, B, [inform] in [t_{now},t_{now}+48 :0 :0], inform(about next_week))

   A demande à B de l'informer dans les deux jours au sujet de ce qu'il fera la semaine prochaine : l'action communicative inform (correspondant à la réponse demandée) doit se situer dans l'intervalle [t_{now},t_{now}+48 :0 :0].

Nous donnons, maintenant, un exemple, où la contrainte temporelle ne porte plus sur une action ou un prédicat donné dans le quatrième paramètre de l'acte de langage mais sur l'acte de langage lui-même. Imaginons que l'agent A veuille demander à l'agent receveur B de réaliser l'action FillIn dans le futur, en lui laissant la possibilité de refuser (utilisation de l'acte propose). Il veut le prévenir, également, qu'il enverra le même message périodiquement toutes les 5 minutes et pendant un jour. De plus, il veut que l'action démarre aussitôt que le message est reçu. Ces contraintes temporelles sont spécifiées de la manière suivante :

propose(A, B, ([propose] periodic [0 :5 :0] [24 :0 :0]) $\wedge$ [FillIn] mi [propose], FillIn)

A la réception d'un tel acte, le receveur B saura que A lui enverra périodiquement le même acte. Ainsi le receveur peut utiliser cette information pour anticiper cette demande et être prêt à exécuter l'action en fonction des contraintes spécifiées dans le message.

5.2.3 Syntaxe du langage

Sur le plan syntaxique, nous nous sommes inspirés du langage du système ASIC [BD94] dont la séparation des différentes parties indépendantes du message est très intéressante puisqu'elle nous permet de séparer et d'expliciter chaque aspect de la composante temporelle apparaissant dans l'interaction.

Le langage TACL est composé de trois champs, qui rendent explicites les trois sortes d'information liées aux couches qui ont été présentées au début de cette section :

1. :comm pour la couche communication,
2. :mas pour la couche système multi-agent,
3. :content pour la couche application.

Dans le cadre de notre travail, nous ne décrirons en détails que le champ :mas puisque les autres champs ne sont pas spécifiques aux systèmes multi-agents.

⟨message⟩ : :=(:comm (⟨communication⟩)
:mas (⟨mas⟩)
:content (⟨application⟩))

Le champ ⟨communication⟩ comprend la totalité des informations relatives au routage du message. Il contient entre autres, l'identité de l'émetteur et du receveur, la date d'émission.

Le champ ⟨mas⟩ comprend la totalité des informations nécessaires pour la gestion d'une conversation entre les agents, comme les informations nécessaires à l'interprétation du message. Nous reviendrons plus précisément sur les aspects protocoles et gestion de conversations par la suite. Pour commencer ici, nous le décrirons simplement avec les sous-champs qui suivent :

⟨mas⟩ : := :act ⟨act⟩
:time ⟨time⟩
:nature ⟨descriptor⟩
:protocol ⟨protocol⟩
:conversation ⟨conversation⟩

- ⟨act⟩ définit l'ensemble des actes de langages qui seront utilisés. La sémantique de ces actes correspond à celle qui a été décrite dans la section précédente.

  ⟨act⟩ : := inform | command | propose | accept | refuse

- ⟨descriptor⟩ permet d'interpréter correctement le contenu du champ :content comme une ressource, un plan, une action ou un but[12] (goal).

  ⟨descriptor⟩ : := resource | plan | action | goal

- le champ ⟨time⟩ est directement défini à partir du langage temporel L_T :

  ⟨time⟩ : := (⟨expr⟩)
  ⟨expr⟩ : := ⟨texp⟩ | ⟨expr⟩ AND ⟨expr⟩ | ⟨expr⟩ OR ⟨expr⟩
  ⟨texp⟩ : := A | P | Ss | Sr | ⟨inte⟩ | ⟨texp⟩ ⟨rel⟩ ⟨texp⟩ |
  ⟨texp⟩ periodic ⟨texp⟩ ⟨texp⟩
  ⟨inte⟩ : := [*duration*] | [⟨t⟩, ⟨t⟩]
  ⟨t⟩ : := now | ? | infty | *real*

  où :
  - ⟨*rel*⟩ est défini en fonction des relations temporelles $\mathcal{R}$ (cf. section 4.2).
  - *duration* correspond à une valeur exprimée dans un format temporel,
  - now correspond à t_{now}, ? à $t_?$ et infty à $+\infty$,
  - A, P, respectivement, dénotent l'action et la formule contenue dans le

[12] une définition ainsi qu'une description de ces notions seront décrites dans le chapitre 7.5

champ :content. Ils sont utilisés pour construire les expressions temporelles de base telles que $[\alpha]$ et $[\phi]$.
- Ss dénote l'acte de langage qui est exprimé dans :act (pour l'émission). Sr dénote l'acte de langage qui est spécifié dans l'étape suivante (l'état suivant) du protocole (il est équivalent au *reply-by* de la FIPA [FIP00]). Les deux sont utilisés pour construire des expressions temporelles de base telles que $[\underline{sa}]$.

- Le champ ⟨protocol⟩ et ⟨conversation⟩ sont utilisés pour la gestion, respectivement, du protocole et de la conversation. Ils seront décrits plus tard.

Le champ ⟨application⟩ concerne l'intégralité des informations utilisables par l'application. Comme nous l'avons dit en introduction, la dimension temporelle est explicitée à l'intérieur du champ ⟨mas⟩. Nous pouvons cependant noter que rien n'empêche ce champ ⟨application⟩ de contenir un composant temporel interprétable par l'application.

Comme nous l'avons souligné, le langage comporte un champ "protocole" dans lequel la composante temporelle apparaît également. Ces protocoles temporels sont décrits dans la section suivante. Il faut également préciser que le langage *TACL* n'est pas "FIPA compliant" (c'est à dire qu'il ne respecte pas la totalité des spécifications proposées par la FIPA) puisqu'il propose des spécifications temporelles qui pourraient compléter les propositions de la FIPA. Il est cependant relativement proche dans sa conception de FIPA ACL et pourrait être adapté pour reprendre notamment tous les champs spécifiés par la FIPA.

5.3 Protocoles temporels : TIP

Certaines parties du travail décrit précédemment sont directement réutilisables dans les autres composantes relatives à l'interaction (protocole, conversation). Sans approfondir aussi complètement l'étude, nous nous sommes penchés sur le cas des protocoles pour montrer que le langage d'expression temporel peut également servir à contraindre des protocoles.
Pour illustrer ces propos, nous décrivons ici un langage de description pour exprimer des protocoles dans lequel la composante temporelle est explicitée. Ces protocoles sont basés sur les actes de langage décrits précédemment. Nous rappelons que les protocoles sont utilisés afin de contrôler et structurer les échanges d'informations entre les agents. Comme nous l'avons évoqué au chapitre 1, de nombreuses études concernent les protocoles. Nos travaux s'inspirent de ceux menés par la FIPA [FIP00] et le formalisme est basé sur des recherches menées en parallèle au sein de notre laboratoire. L'objectif dans cette section n'est pas de redéfinir de nouveaux protocoles mais d'expliciter les aspects temporels suscep-

tibles d'apparaître dans la description d'un protocole qui sera alors appelé *TIP* pour **T**emporal **I**nteraction **P**rotocol.

Dans ce mémoire, nous considérons qu'une structure de base du protocole met en jeu exclusivement **deux** agents : l'initiateur et le participant. Les protocoles d'interaction temporels sont des graphes état-transition dans lesquels les transitions expriment des contraintes sur le processus de conversation lui-même (cf. champ :condition) et sur le contenu du message lui-même (cf. champ :value). Les expressions utilisées dans ces deux champs sont exprimées sur le langage L_T.

5.3.1 Langage de description de protocole temporel

Pour commencer nous allons donner une définition des termes utilisés dans la description du langage que nous présenterons ensuite.

Définitions des termes

Cette section distingue 2 parties :

1. la première correspond à la description du format permettant l'explicitation d'un protocole.
2. la seconde partie présente deux exemples de protocoles simples instanciés : le protocole *inform* (I) et le protocole *request_ refuse-accept* (R-RA).

Un protocole est défini par :
- un nom de protocole,
- un initiateur correspondant à l'agent commençant le protocole,
- un participant : agent recevant le message de l'initiateur,
- un but initiateur (qui permet de déclencher l'utilisation du protocole),
- un champ "body" (permettant de spécifier l'arbre du protocole "état-transition", avec des opérateurs du type séquence, parallélisme, choix exclusif),
- un champ "transition" (définissant chaque transition : nom, état cible, condition d'activation, etc.).

Nous proposons ci-dessous une esquisse de langage de description (grammaire) de protocole temporel afin d'illustrer les aspects qui nous intéressent :

⟨protocol⟩ : :='(' protocol :name ⟨prname⟩
:initiator ⟨varId⟩
:participator ⟨varId⟩
:goal ⟨goal⟩
:body ⟨body⟩
:transitions '(' ⟨transition⟩* ')' ')'

⟨prname⟩ représente le nom du protocole.

Le champ ⟨body⟩ décrit les différentes séquences de messages constituant le protocole. Il permet de décrire l'arbre complet du protocole ; c'est à dire l'enchaînement de chaque étape ("transitions") possible du protocole.

⟨body⟩ : :=⟨transID⟩ |'('⟨body⟩⟨operator⟩⟨body⟩')'
⟨operator⟩ : := ';'| '|' | '⊕'

Les différents opérateurs permettant de spécifier la structure du protocole sont la séquence (;), le parallélisme (||) et le choix exclusif (⊕).
Par exemple : $(tr1\,;\,(tr2|tr3|tr4)\,;\,(tr5|tr6)\,;\,(tr7|tr8))$ correspond à l'arbre du protocole FIPA ContractNet que nous avons décrit au chapitre 1. Il est représenté avec les transitions étiquettées par les actes de langage correspondant par la figure 5.2. Nous considérons que le nom d'une transition correspond à l'acte de langage utilisé.

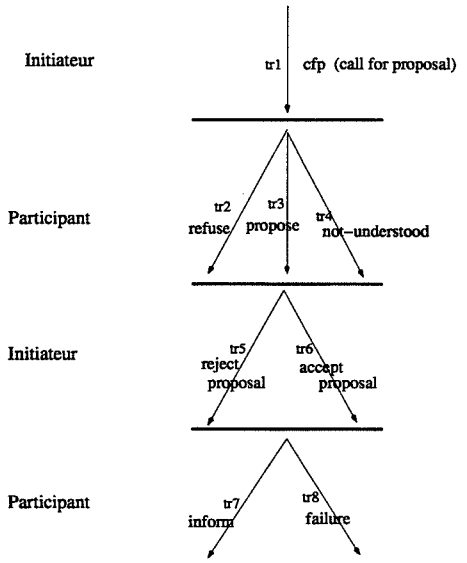


FIG. 5.2 – Arbre de description du protocole ContractNet de la FIPA

⟨transition⟩ : := :name ⟨transID⟩
:target ⟨stname⟩
:condition ⟨transition-condition⟩*
:value ⟨msg-constraints⟩

Une transition est décrite par un nom (⟨transID⟩), un état cible (target), des conditions sur la transition ainsi qu'éventuellement des contraintes sur le message

lié à l'utilisation de cette étape du protocole.

⟨stname⟩ : := 'init' | 'success' |'fail' | string

Le champ "state" permet de décrire l'état courant du protocole.
Les différents états du protocole que nous utilisons, sont "init" pour la phase d'initialisation, "successful" s'il a réussi et "failed" s'il a échoué. D'autres possibilités d'états peuvent être spécifiées en fonction des besoins du protocole.

⟨prname⟩ : := string
⟨transID⟩ : := string
Les noms représentant les protocoles et les transitions et les différents identifiants sont de simples chaînes de caractères.

⟨transition-condition⟩ : := 'true' | ⟨condlang⟩
Dans la définition du langage de conditions (⟨condlang⟩), nous incluons un champ *time* afin d'exprimer des conditions sur le déclenchement de la transition relative à des expressions temporelles. Ces transitions doivent être vérifiées pour déclencher la transition et faire évoluer la conversation représentée par l'exécution du protocole. La mention "true" permet de ne préciser aucune condition particulière. Des conditions sur les transitions peuvent être éventuellement directement spécifiées dans le langage de programmation.

⟨msg-constraints⟩ : := ⟨TACLmessage⟩
Dans la définition des contraintes sur le message (⟨msg-constraints⟩), nous trouvons à nouveau la définition d'un message *TACL*. Les contraintes sur le message définissent un "gabarit" (template) dans lequel le message envoyé (ou reçu) doit correspondre. La différence, ici, est que les champs sont optionels (si un champ n'est pas spécifié, l'agent a toute liberté pour le remplir). De cette manière, les contraintes temporelles qui sont données, définissent le cadre pour spécifier les autres contraintes.

Afin d'illustrer ce langage, nous avons spécifié quelques protocoles que nous utilisons dans les applications présentées en troisième partie. A présent, nous allons en décrire deux exemples.

5.3.2 Exemples pour deux protocoles simples

Le premier protocole décrit correspond à l'émission d'un message d'information tout simple (sans attente de réponse ou de confirmation). Cet exemple permet de voir et comprendre la spécification d'un protocole dans le langage de description précédemment décrit. Les aspects temporels spécifiques aux proto-

coles apparaîtront dans l'exemple suivant. Tous les champs précédemment décrits n'apparaissent pas systématiquement dans ces exemples puisqu'ils sont issus de l'implémentation.

5.3.3 Protocole I

Le protocole I (pour "Inform") est décrit de la façon suivante :

```
(protocol :name "I" :initiator "X" :participator "Y"
          :goal "BxnotBy(Z)"
          :body (t_inform)
          :transitions ( :transition :name t_inform :target successful
                                     :condition TRUE
                                     :value "" )
          )
```

Le second exemple décrit un protocole un peu plus complexe : un agent fait une proposition (acte de langage *propose*) et le receveur doit répondre soit par une acceptation (acte de langage *accept*) soit par un refus (acte de langage *refuse*). Cet exemple permet de mettre en exergue des contraintes temporelles dans la définition du protocole.

5.3.4 Protocole P-RA

Le protocole P-RA (pour "Propose-Refuse ou Accept") est décrit de la façon suivante :

```
(protocol :name "P_RA" :initiator "X" :participator "Y"
          :goal "IxSCyDone(Z)"
          :body (propose) ; (refuse ⊕ accept)
          :transitions (( :transition :name propose :target P
                                      :condition TRUE
                                      :value "time = SRin[t_now,t_now+00 : 01 : 00]")
                        ( :transition :name refuse :target successful
                                      :condition TRUE
                                      :value "")
                        ( :transition :name accept :target successful
                                      :condition "A <= 08 : 00 : 00"
                                      :value "")
          )
```

Cet exemple présente deux contraintes temporelles :

1. l'activation de la transition "propose" impose une *contrainte sur le message* TACL qui sera envoyé : le champ "time" de ce message spécifiera que l'agent attend une réponse (SR : Speech Reply) dans la minute.

2. une *condition sur la transition* "accept" qui vérifiera que la durée de la tâche acceptée (A) liée à la proposition reçue, n'excède pas 8 heures.

Nous avons vu que nous pouvons spécifier une contrainte sur la réponse au moyen du langage *TACL*. La prise en compte du niveau supérieur ("protocole") permet de contraindre temporellement tout l'enchaînement des messages liés à un protocoles en en spécifiant explicitement les contraintes temporelles. Cette section montre que ce qui a été défini dans le cadre du langage d'interaction temporel est applicables aux protocoles. Nous rappelons que ce langage de description de protocole est une ébauche et nécessite d'être complété. Cet état de fait est le même pour la section qui suit : nous préciserons ces perspectives en conclusion.

L'instanciation d'un protocole donne lieu à une conversation. La gestion des conversations est un problème que nous avons eu l'occasion d'évoquer précédemment (cf. chapitre 3). La prise en compte de la composante temporelle au niveau de l'interaction peut également concerner la gestion des conversations comme nous allons le voir à présent.

5.4 Gestion de conversations

Nous définissons une conversation comme l'instanciation d'un protocole. Cette instanciation comme nous allons le souligner, soulève ses propres problèmes. Bien souvent, dans de nombreux travaux traitant des protocoles, les aspects relatifs à la gestion des conversations sont négligés ou non-évoqués. Cependant, la réalisation concrète d'un SMA ou tout simplement la validation de ces recherches (la mise en œuvre des protocoles) impose de prendre en compte ces notions. Dans le cadre de notre travail, nous avons défini une structure de conversation correspondant à l'instanciation d'un protocole temporel décrit précédemment et permettant de suivre une filiation entre conversations. Nous avons choisi de mettre en exergue deux problématiques liées à la gestion de conversation :

1. identifier de manière unique chacune des conversations mises en place. Pour cela, l'identifiant est composé d'une estampille temporelle permettant de dater chaque conversation à sa création. Cet identifiant peut également être utilisé afin de gérer temporellement une conversation ou éventuellement un historique des conversations.

2. suivre la filiation entre les conversations afin de pouvoir supprimer en cas de fin (échec, succès, annulation, etc.) d'une conversation toutes ses descendantes. Cela permet notamment de simplifier et réduire les messages qui n'aboutiront pas en mettant en place un mécanisme de ce type. La prise en compte de l'aspect temporel et notamment la définition de contraintes telles que des échéances ("deadlines") offre un moyen simple pour constater une "fin" (cf. plus haut) de conversation. C'est pourquoi nous avons choisi de décrire cette problématique.

Avant de se pencher sur ces problèmes, nous introduisons auparavant la structure de conversation que manipule un agent. Nous définissons une conversation avec les champs suivants :

- conv-ID : qui correspond à l'identifiant de conversation. Nous reviendrons sur son élaboration dans ce qui suit,
- initiator : initiateur de la conversation,
- participant : allocutaire de la conversation,
- current state : étape courante de la conversation,
- current protocol : protocole actuellement utilisé
- time : date limite d'existence de la conversation,
- status : waiting, successful, failed, ? : état de la conversation permettant de spécifier son activité. Les significations sont respectivement "en attente", "réussie", "échouée" et "inconnu" pour un état indéterminé.

Une conversation est définie par l'ensemble de ces paramètres. Nous nous attarderons ici sur l'identifiant de conversation (*conv-ID*) qui doit être unique. Cet identifiant contient donc une estampille temporelle qui est générée au moment de la création de la conversation.

5.4.1 Identifiant de conversation

Un identifiant de conversation (conv-ID) est composé de trois champs et est décrit sous la forme suivante :

⟨conv-ID⟩ : := ⟨date⟩ '|' ⟨AgtX⟩'-'⟨AgtY⟩ '|' '('⟨conv-ID-up⟩ | 'MAIN")'

1. ⟨date⟩ : une estampille temporelle qui date la conversation à sa création.
2. ⟨AgtX⟩-⟨AgtY⟩ : les interlocuteurs qui correspondent aux identifiants des agents impliqués : l'agent initiateur et l'allocutaire.
3. ⟨conv-ID-up⟩ : un identifiant de "conversation-mère" permettant de spécifier la conversation dont elle dépend ou que la conversation est la première et ne dépend d'aucune autre. Dans ce dernier cas, ce champ est rempli avec la valeur "MAIN".

La conjonction de l'estampille temporelle et de l'information concernant les interlocuteurs rend cet identifiant unique[13].

Une conversation possède le format suivant : Date|AgtX-AgtY|Conv-IDup

[13]Nous faisons l'hypothèse que les agents ne sont pas multiprocesseurs et donc ne peuvent créer chacun qu'une conversation à la fois.

5.4.2 Filiation entre conversations

Une conversation entre deux agents peut engendrer d'autres conversations dépendantes de la précédente. Par exemple, lorsqu'un agent reçoit une proposition, il peut demander à un autre agent de réaliser une partie de la proposition et ainsi créer une nouvelle conversation dépendante de la précédente.

Concrètement si un agent A envoie à un agent B un message au moment t=12h30m12s (format simplifié), il initie une conversation dont l'identifiant en interne sera :
Conv-ID : "12 :30 :12|A-B|MAIN"

L'agent B choisissant d'initier une nouvelle conversation avec l'agent B dépendante à t=12h32m10s de la conversation "12 :30 :12|A-B|MAIN" créera une conversation dont l'identifiant sera :
Conv-ID :"12 :32 :10|B-C|12 :30 :12.A-B.MAIN"

Pour des soucis de confidentialité, dans un message *TACL*, le champ ⟨conversation⟩ (cf. section 5.2) ne reprend que les deux premiers champs et ainsi ne divulgue pas certains aspects comme l'existence, la date et les interlocuteurs d'une "conversation-mère". Comme expliqué plus haut, cette filiation permet d'éviter en cas d'échec que de nombreuses sous-conversations restent en attente ou tout simplement soient poursuivies inutilement. L'annulation d'une conversation impose toutefois l'envoi d'un message. L'aspect temporel permet également de contraindre l'existence d'une conversation et ainsi supprimer une conversation dont la limite temporelle est dépassée. Cette valeur est calculée à partir des informations temporelles relatives au protocole utilisé mais pourrait éventuellement être transmise par l'initiateur au moyen d'un champ supplémentaire dans l'identifiant de conversation (ou en complétant le champ ⟨date⟩ avec une date de fin, par exemple).

Cette structure reste basique et est uniquement décrite ici à titre indicatif des possibilités offertes par la composante temporelle sur la gestion de conversation. Cette partie des travaux reste exploratoire et nécessite de plus amples recherches afin d'appréhender la dimension temporelle au sein de conversations complexes ; nous y reviendrons dans les perspectives.

Tous les aspects de l'interaction ayant été définis, nous illustrons l'échange de message TACL dans ce qui suit.

5.5 Exemples récapitulatifs d'échanges de messages TACL

Nous allons maintenant présenter un exemple complet mettant en œuvre chacun des aspects précédemment décrits. Il concerne la proposition d'une tâche d'im-

pression par un agent que l'on appellera *manager* (M) à deux agents imprimeurs ($I1$ et $I2$). Nous illustrerons plusieurs scénarios différents permettant de souligner chacune des particularités temporelles que ce modèle d'interaction offre. Nous ne décrirons pas tous les mécanismes aboutissant à la construction du message mais uniquement ceux liés aux aspects temporels.

5.5.1 Premier message : proposition

Nous considérons que le *manager* souhaite voir effectuée une tâche d'impression avant 1 mois et connaît deux imprimeurs $I1$ et $I2$ capables de réaliser ce type de tâche ; c'est à dire qu'il possède les états mentaux correspondants. L'agent *manager* choisira donc un protocole dont le but initiateur correspond à l'un des états mentaux, en l'occurrence le protocole "P-RA", et initiera une conversation avec ce protocole en envoyant le message suivant spécifié en langage *TACL* :

(message :com (:sender *manager* :receiver *imprimeur_I1*
:priority 5
:date 2000 : 08 : 16
:sndDate 2000 : 08 : 16T18 : 17 : 01)
:mas (:act propose
:time (SR *in* $[t_{now}, t_{now}$+00 : 00 : 30]
$\wedge$ *done*(A) b $[t_{now}$,2000 : 09 : 16])
:nature *Action*
:protocol $P - RA$
:conversation "18 : 17 : 06|*manager* $- I1$"
:content (*realizecontract*(*printdocument*X, 100p, *color*))

La définition du protocole utilisé (P-RA) correspond à celle définie précédemment. Le message spécifie que l'action doit être réalisée avant un mois et que la réponse est attendue dans les 30 secondes spécifiant ainsi une contrainte temporelle plus stricte que celle spécifiée par le protocole (P-RA) mais n'enfreignant pas celui-ci. En cas de conflit entre les contraintes spécifiées dans le message et celles du protocole, l'agent recevant le message peut refuser d'accomplir l'action parce que l'agent émetteur ne respecte pas le protocole prévu.

Pour simplifier, nous ne décrirons que le suivi de protocole géré par le manager. Ces informations liées au protocole en cours que nous appelons "conversation" auront le format suivant :

- conv-ID = "18 :17 :06|manager-I1|MAIN"
- initiator = "manager"
- participator = "$I1$"
- current state = "init"
- current protocol = "P-RA"
- status = "waiting"

La conversation se met en état d'attente ("waiting") de la réponse qui correspondra au prochain état du protocole utilisé.

5.5.2 Second message : réponse

A la réception du message, l'agent imprimeur $I1$ interprète le message, évalue sa faisabilité et envoie sa réponse dans les trente secondes par rapport à la date de réception ($18h17m04s$ par exemple). Le suivi du protocole offre deux possibilités de réponse "accept" ou "refuse".

Acceptation

En cas d'acceptation la réponse aura la forme suivante :

(message :com (:sender *imprimeur_I1* :receiver *manager*
:priority 5
:date 2000 : 08 : 16
:sndDate 2000 : 08 : 16T18 : 17 : 25)
:mas (:act accept
:time $(done(A)b[t_{now}$,2000 : 08 : 31])
:nature *action*
:protocol $P - RA$
:conversation "18 : 17 : 06|$manager - I1$"
:content $contract(printdocumentX, 100pages, color)$)

L'agent imprimeur $I1$ accepte en précisant que la tâche sera effectuée d'ici le 31/08/2000 ; cette réponse comporte le même identifiant de conversation pour que l'agent manager identifie à quoi correspond cette réponse.

Refus

Imaginons à présent que la réponse soit négative. Ce fait peut être lié à différents cas de figure :

- son plan de charge ne lui offre pas la possibilité de réaliser la tâche.
- l'agent est incapable de répondre dans le temps imparti car il a d'autres tâches à traiter pour le moment.

Les raisons ci-dessus sont relatives à deux aspects temporels particuliers, le premier lié à la réalisation de la tâche et le second au déroulement du protocole. En cas de refus, la réponse aura la forme suivante :

(message :com (:sender *imprimeur_I1* :receiver *manager*
:priority 5
:date 2000 : 08 : 16
:sndDate 2000 : 08 : 16T18 : 17 : 12)
:mas (:act refuse

:time (SR *in* $[t_{now}, t_{now}+00:00:30]$
$\wedge$ $done(A)$ b $[t_{now}, 2000:09:16]$)
:nature *action*
:protocol $P-RA$
:conversation "$18:17:06|manager-I1$"
:content $contract(printdocumentX, 100pages, color)$)

A la réception du message, la conversation du manager passera dans l'état "successful". Si aucune réponse n'arrive dans le temps imposé, le gestionnaire de conversations la fera passer dans l'état "failed" et le comportement ensuite de l'agent dépendra des stratégies mises en place dans ce cas de figure. Nous y reviendrons lorsque nous décrirons le modèle d'agent par la suite (cf. section 7.9.

Les exemples proposés ci-dessus mettent en exergue la richesse de la composante temporelle au niveau de l'interaction et soulignent différents moyens de la prendre en compte.

5.6 Discussion

Dans ce chapitre, nous avons proposé un modèle permettant de prendre en compte les aspects temporels relatifs à l'interaction. Notre objectif était de pouvoir exprimer les aspects temporels au sein de la force illocutoire au moyen d'un langage d'interaction dont la sémantique et la syntaxe sépare les aspects temporels et atemporels. Ces objectifs sont réalisés au moyen du langage *TACL* qui nous permet de faire porter le temps à la fois sur l'acte de communication et sur le contenu propositionnel. Nous utilisons également *TACL* pour définir des protocoles temporels (*TIP*) ainsi que pour mettre en place une structure de conversation pouvant servir de base au développement ultérieur d'un raisonnement temporel plus évolué concernant l'interaction.
Tous les aspects temporels peuvent être explicités au moyen du langage L_T permettant une représentation uniforme de la composante temporelle au sein de l'interaction. Ce langage d'expressions temporelles est basé sur la théorie et l'algèbre d'Allen offrant un important pouvoir expressif et une représentation du temps intuitive.

A présent, tournons-nous vers la dimension sociale et intéressons-nous plus particulièrement aux aspects temporels dans l'organisation.

6

Organisation temporelle dans les SMA

Sommaire

6.1 Introduction

Comme nous l'avons précisé dans le premier chapitre, différentes modélisations de l'organisation sont possibles. L'une d'entre elles consiste à prendre en compte les dépendances entre agents. Un modèle d'organisation temporelle basé sur les relations de dépendances a été proposé par [All98] (cf. section 3.2). Dans ce modèle, le facteur temporel est pris en compte dans le raisonnement social des agents pour être appliqué à la supervision de systèmes industriels. Dans le cadre de ce travail, nous nous sommes intéressés à une modélisation à base de structure organisationnelle (plus adaptée à notre domaine d'application). Notre travail se situe dans la même veine que [JTd01] que nous avons évoqué en section 3.3.2.
Ce chapitre concerne l'ajout de la dimension temporelle dans l'organisation, et plus particulièrement la définition d'un langage de structures organisationnelles temporelles. Cette présence se retrouve dans l'organisation sous de nombreux aspects tels que la durée d'un rôle, les relations entre les entités, l'expression de

contraintes temporelles...

6.2 Dimensions temporelles d'une structure organisationnelle

En réalité dans notre société, de nombreuses organisations sont définies et restreintes avec des contraintes temporelles : l'organisation, ses rôles, ses missions ou ses liens possèdent une durée spécifique et commencent ou finissent à des instants particuliers.
Par exemple, le rôle de président de la république a une durée de 5 ans en France, le lien d'autorité d'un directeur sur un de ses employés est limité à un intervalle bien défini (cependant généralement implicite) et à des tâches spécifiques, une mission peut en contraindre une autre, etc. Dans le cadre de notre travail par exemple, l'objectif est de pouvoir également spécifier qu'une tâche de reliure suit toujours la tâche d'impression du *job* concerné. Autre exemple, il peut être nécessaire d'exprimer que l'autorité d'un manager (gestionnaire d'atelier d'impression) sur un imprimeur augmente à mesure que le temps restant pour réaliser l'impression diminue.
Cependant, comme nous l'avons vu (cf. section 3.3.2), ces aspects ne sont que rarement pris en compte explicitement dans la représentation de l'organisation dans les SMA.

Nous rappelons que les termes spécifiques à l'organisation (comme par exemple les notions de rôles, de groupes et de liens) sont regroupés dans un langage organisationnel que nous appelons L_ω[14]. Nous le décrirons plus précisément dans la section suivante. Afin de pouvoir exprimer la composante temporelle au sein d'une structure organisationnelle, nous avons choisi de compléter chacun des termes de L_ω par deux attributs :

1. *les propriétés temporelles* (tp) qui regroupent tous les aspects externes à la structure organisationnelle; c'est à dire l'ancrage dans le temps global des différents termes organisationnels. Cet ancrage peut être relatif ou réel (absolu). Les propriétés temporelles ne sont en relation qu'avec des entités **externes** à la structure organisationnelle temporelle (*tos* pour "temporal organizational structure").
   Par exemple, chaque terme organisationnel possède une durée prévue qui peut être exprimée comme une contrainte au niveau de la structure organisationnelle (*os*) : quand un agent joue un rôle, ce dernier a généralement une date de début et une date de fin. Comme nous l'avons précisé, cette date peut être relative : par exemple, un agent "manager" ne peut pas délivrer les impressions reliées pour vérification de qualité la nuit quand l'entreprise responsable du contrôle de qualité est fermée. Ainsi, la configuration tempo-

[14]Les éléments de ce langage dépendent évidemment du modèle organisationnel utilisé.

relle du rôle de manager est contrainte par les heures classiques et habituelles de travail de société et cela s'applique également dans le cadre d'agents informatiques. Les propriétés temporelles peuvent être appliquées sur tous les termes d'une *os*. Il peut s'agir également de caractéristiques temporelles spéciales comme la répétition ou la périodicité permettant d'ancrer différents termes dans le temps global commun.

2. *les contraintes temporelles* (tc) qui permettent d'exprimer les aspects temporels internes à la structure comme le positionnement d'une mission par rapport à une autre, par rapport à un rôle, etc. Ces contraintes sont exclusivement **internes** à une structure organisationnelle. A l'intérieur d'une structure organisationnelle, il est généralement possible de spécifier la position d'un rôle par rapport à un autre.
   Pour illustrer simplement ces propos, dans le cadre d'un contrat d'impression, nous pouvons imaginer que le premier rôle joué est le rôle "Client" puis temporellement parlant, le suivant est le rôle "Manager" du contrat et enfin le rôle "Printer" qui correspond à l'exécuteur du contrat. Ce champ permet par exemple, d'exprimer les contraintes temporelles entre les missions (tâches) d'impression et de reliure dont nous avons parlé précédemment.

Pour prendre en compte ces aspects temporels dans l'organisation, nous avons choisi de définir un langage de structures organisationnelles temporelles mettant en œuvre les attributs temporels proposés ci-dessus. Ce langage s'appuie sur le modèle de structure organisationnelle *MOISE* [HBSS00] décrit au chapitre 1 ; il utilise également le langage d'expressions temporelles L_T présenté au chapitre 4 en le combinant avec le langage dédié : L_ω. Nous allons pour commencer revenir sur ces deux aspects.

6.3 Langage organisationnel

Le langage d'organisation, L_ω, évoqué au chapitre 4, est l'ensemble des termes organisationnels qui sont nécessaires pour définir une organisation. Ce langage organisationnel est défini par :

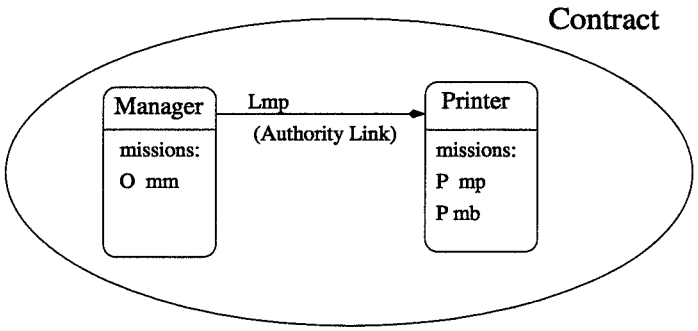
$\omega ::= os \mid role \mid link \mid mission \mid group$

avec *os* signifiant "organizational structure" ; c'est à dire la définition complète d'une organisation spécifique. La syntaxe d'une structure organisationnelle a été donnée au chapitre 1 (cf. section 1.2.3).
Afin de comprendre plus concrètement ce qu'est une structure organisationnelle (*os*), l'exemple suivant propose une *os* représentant un contrat d'une alliance d'ateliers d'impression. Cet exemple se base sur un projet particulier sur lequel nous reviendrons en troisième partie. A ce stade du mémoire, nous dirons simplement que plusieurs ateliers d'impression doivent se regrouper dans une alliance afin de traiter différentes tâches d'impression. Nous spécifions les règles de mise en place de cette alliance au moyen d'une structure organisationnelle.

Exemple de structure organisationnelle

Cette os "contract" (cf. Figure 6.1) possède deux rôles : un Manager et un Printer (Imprimeur). Un lien nommé l_{MP} est défini entre ces deux rôles. Pour simplifier, aucun groupe n'est spécifié.

(os :name contract :roles (Manager Printer) :links (l_{MP}) :groups ())



mm: mission de management
mp: mission d'impression (printing)
mb: mission de reliure (binding)

FIG. 6.1 – Exemple de structure organisationnelle : *contract*

Le rôle "Manager" possède une mission obligatoire "mm" qui satisfait un but ("gm") et un plan unique "pm". Ce plan consiste à réaliser les buts d'impression (printing) "gp" et de reliure (binding) "gb". Aucune action ni ressource n'est permise. Ainsi, lorsqu'un agent jouera le rôle "Manager", il aura l'obligation d'activer la mission "mm" et ensuite de réaliser le but "gm" en déléguant l'exécution des différentes composantes du plan "pm" (car les deux buts apparaissant dans le corps du plan n'appartiennent pas à l'ensemble des buts autorisés pour cette mission).

(role :name Manager :missions ((O mm)))
(mission :name mm :goals (gm) :plans (pm) :actions () :resources ())

Le rôle "Printer" possède deux missions permises "mp" et "mb", consistant respectivement à la réalisation de la tâche d'impression "gp" ainsi que celle de reliure "gb".

(role :name Printer :missions ((P mp) (P mb)))
(mission :name mp :goals (gp) :plans (pp)
:actions (print) :resources ())
(mission :name mb :goals (gb) :plans (pb)
:actions (bind) :resources ())

Le lien "l_{MP}" correspond à un lien d'autorité du rôle "Manager" sur le rôle "Printer" dans le contexte des missions : "mm", "mp", "mb".

(link :name l_{MP} :type Aut
:source (:role Manager :missions (mm))
:target (:role Printer :missions (mp mb))
...)

Cependant, comme nous l'avons dit, dans le contexte de systèmes dynamiques, il est crucial pour une organisation d'être flexible ; c'est à dire capable de s'adapter à des circonstances en perpétuelle évolution et donc, comme nous allons le voir maintenant, une des voies est d'avoir une composante temporelle clairement explicitée au sein de la structure organisationnelle.

Ce constat montre la nécessité de définir un langage de structures organisationnelles temporelles. Un langage a donc été défini afin de spécifier les contraintes temporelles de la structure organisationnelle que les agents doivent respecter.

6.4 Langage de structures organisationnelles temporelles : TOSL

La syntaxe du langage de structures organisationnelles temporelles (*TOSL*) est déduite du modèle *MOISE* décrit précédemment (cf. section 1.2).

Comme nous avons vu précédemment (cf. section 6.2), nous avons choisi de compléter chacun des termes de L_ω par deux attributs : les propriétés temporelles (tp) et les contraintes temporelles (tc).

La définition sous forme BNF de ces deux champs temporels est la suivante :

:tp (⟨exp⟩)
:tc (⟨exp⟩)

avec *exp* correspondant à une expression temporelle définie dans le langage L_T présenté précédemment (cf. section 4.2).

A l'intérieur de chaque champ, les termes organisationnels (cf. $[\omega]$ dans la définition de L_T) qui peuvent apparaître sont ceux qui sont accessibles depuis la définition du champ lui-même. La référence au champ dans lequel les propriétés ou les contraintes temporelles sont définies est faite en utilisant *self*.

Un niveau supérieur ⟨otos⟩ a été ajouté pour organiser différentes structures organisationnelles temporelles entre elles. Ce niveau pourrait être utile lorsqu'un système utilise plusieurs *tos* changeant pendant le fonctionnement (voire susceptibles de se succéder, se superposer, etc.).

⟨otos⟩ : :=(otos :name global :os (⟨osname⟩*)
:tp ⟨exp⟩ :tc (⟨otostempconst⟩*))

6.4.1 Validité temporelle

Une structure organisationnelle peut se révéler très complexe en fonction du système qu'elle représente.

Dans MOISE, la cohérence structurelle peut être vérifiée lors de la spécification grâce à un outil [Han02] basé sur le calcul des dépendances ajoutées sur l'organisation. Il permet de vérifier que les rôles intervenant dans la définition d'un lien existent bien, qu'un lien d'autorité n'existe que dans un unique sens entre deux rôles, qu'il existe bien un rôle permettant de réaliser une mission spécifique, etc.

Dans le contexte de ce mémoire, nous allons décrire les aspects liés à la validité temporelle par suite des propriétés temporelles que nous avons explicitées.

Une *tos* présente une hiérarchie qui permet de délimiter ces propriétés inhérentes à la structure et ainsi ne nécessitant pas d'être redéfinies chaque fois.. Ainsi, une *tos* est définie par un ensemble de rôles, de liens et de groupes. Par conséquent, la durée de chacun des "sous-termes" doit être contenue dans celle de ceux de plus haut niveau et évidemment dans la durée de la structure organisationnelle temporelle elle-même. Ces contraintes conditionnent l'existence de ces termes. Ces propriétés sont considérées comme existantes par défaut afin de maintenir des contraintes de validité temporelle sur la structure ; c'est à dire par exemple, qu'une mission ou un rôle ne doit pas être activé si leur structure organisationnelle temporelle (*tos*) associée n'existe plus ou du moins n'est plus active. Par exemple, l'arbre hiérarchique d'une *tos* (cf. figure 6.2) permet de déterminer ainsi que la durée d'une mission est associée à un rôle dont elle ne peut dépasser la durée.

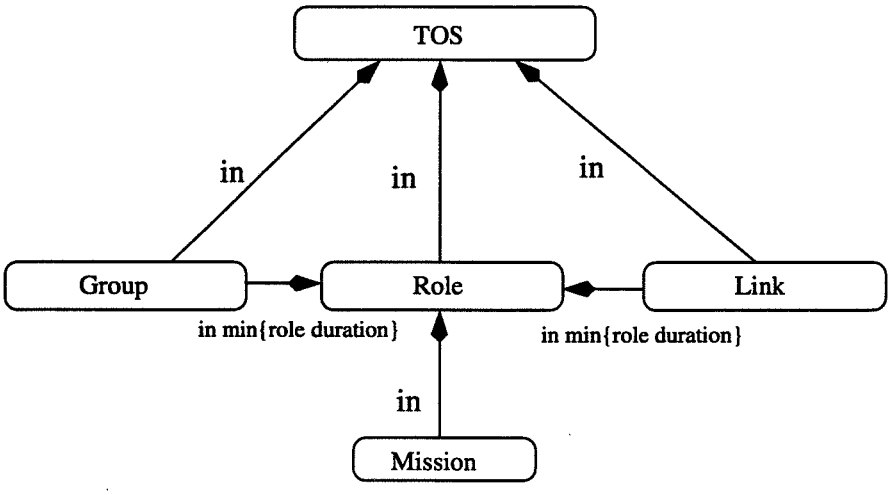


FIG. 6.2 – Arbre hiérarchique d'une structure organisationnelle temporelle

Nous utilisons une notation pointée si nécessaire pour parcourir la struc-

ture. Ainsi pour exprimer une "mission1" appartenant au "role1", nous écrirons role1.mission1 afin de supprimer toute ambiguïté. Ces propriétés peuvent s'exprimer au moyen de la relation in, pour chacun des termes de la structure organisationnelle temporelle :

- [role] in [tos] , [link] in [tos] ,[group] in [tos]
- [role.mission] in [role]

Nous précisons que ces règles ne fournissent qu'une vérification "faible" de la validité temporelle par rapport à ce que serait une vérification complète de la validité. Cette dernière vérification nécessiterait une prise en compte exhaustive de toutes les entités (au sens large) du processus étudié : c'est à dire les buts, plans, actions, ressources définissant chaque mission entre autres. Cette vérification ne peut être effectuée ici puisqu'elle dépend notamment d'informations liées en particulier aux applications concernées ainsi qu'au modèle organisationnel utilisé.

D'autres contraintes implicites sont moins évidentes à mettre en exergue. Ainsi, comme nous l'avons dit dans la section précédente, la durée d'un lien est considérée par défaut correspondre à la durée du plus court des deux rôles associés à ce lien : en effet, si l'un des rôles se termine, les liens correspondants n'existent plus non plus.
De la même manière, nous considérons qu'un groupe est conditionné par l'existence de l'ensemble complet des rôles qui définissent ce groupe et c'est pourquoi la durée d'un groupe ne peut dépasser la durée du plus court de ses rôles. Par exemple, un groupe "classe" peut être composé d'un rôle "professeur" et d'un rôle "étudiant". Dans l'entité organisationnelle, cela n'aura pas de sens d'avoir une classe sans étudiant et vice-versa. Cette contrainte permet d'éviter ce genre de problème mais l'hypothèse peut être différente en fonction de chaque groupe. Ces propriétés par défaut peuvent être exprimées de la façon suivante :

[Link] in min([Link.SourceRole],[Link.TargetRole])
[Group] in min([Group.Roles])

Néanmoins, afin d'offrir plus de flexibilité, d'autres propriétés peuvent être précisées à l'intérieur de la définition de la *tos* dans la mesure où elles ne remettent pas en cause la cohérence. Cela signifie que les propriétés précédentes expriment des contraintes temporelles sur la durée maximale d'un terme mais il est possible de réduire ces contraintes au moyen de prérequis plus spécifiques et plus restrictifs si l'application l'impose. Par exemple, la durée d'un lien peut être restreinte (comme nous l'avons montré dans le deuxième exemple de la section précédente) à la durée d'une mission particulière ou même d'un rôle externe.

Comme nous l'avons évoqué au chapitre 3, l'organisation peut également être modélisée par des opérateurs déontiques tels que ceux que nous retrouvons dans la définition des missions (permises ou obligatoires). Ceux-ci peuvent être également complétés par des contraintes temporelles.

A présent que le langage *TOSL* a été défini, reprenons l'exemple de structure organisationnelle "contract" complétée par ses composantes temporelles.

6.4.2 Exemples de spécification TOSL

Nous rappelons que l'exemple qui suit propose une structure organisationnelle représentant un contrat pouvant se mettre en place dans un groupement d'entreprises comme évoqué en section 6.3. Cette fois-ci la structure est temporelle (tos) et s'avère plus complexe pour les besoins de l'exemple. Le contrat concerne des ateliers d'impression et de reliure. Dans cet exemple, cette tos "contrat" possède quatre rôles un "Manager", deux imprimeurs respectivement noir&blanc et couleur "bwPrinter" et "cPrinter", et un relieur "Binder". Un lien d'autorité relie le "Manager" avec chacun des autres rôles nommé ("l_{MbP}", "l_{McP}", "l_{MB}"). Le "Binder" a également de l'autorité sur les imprimeurs avec les liens ("l_{BbP}", "l_{BcP}"). Pour simplifier, là encore, aucun groupe n'est défini mais la spécification est similaire.

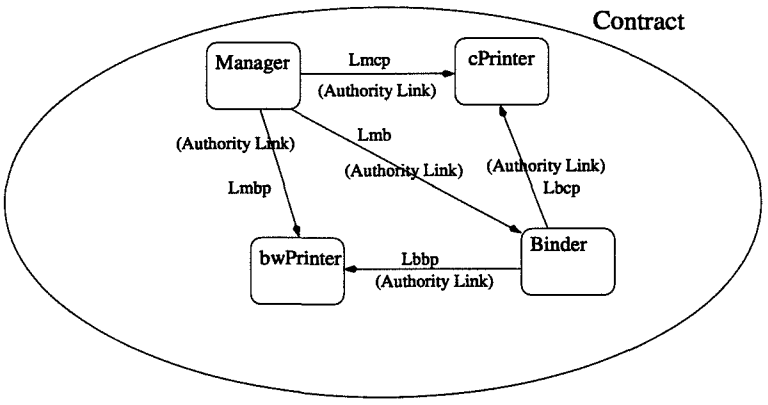


FIG. 6.3 – Exemple 2 de structure organisationnelle : *contract.*

Premier exemple

Imaginons pour commencer, que l'une des règles relatives au fonctionnement de l'alliance impose un contrôle-qualité pour chacune des tâches réalisées par le Manager du contrat. Si la qualité n'est pas satisfaisante, certaines tâches peuvent nécessiter d'être refaites. Cette règle va contraindre la durée de chacun des rôles ainsi que la durée des liens d'autorité :

- la durée de chaque rôle "(bwPrinter", "cPrinter" et "Binder)" sera la même que celle du "Manager",

- la durée des liens d'autorité associés du "Manager" sur les autres agents sont maintenus pendant la totalité du temps de réalisation du contrat.

Les propriétés temporelles spécifiées permettent de situer l'exécution de ce "contrat" entre le 23 et le 30 juin (cf. figure 6.4) alors que les contraintes temporelles permettent d'imposer que les rôles Imprimeur et Manager se finissent en même temps (cf. figure 6.5) :

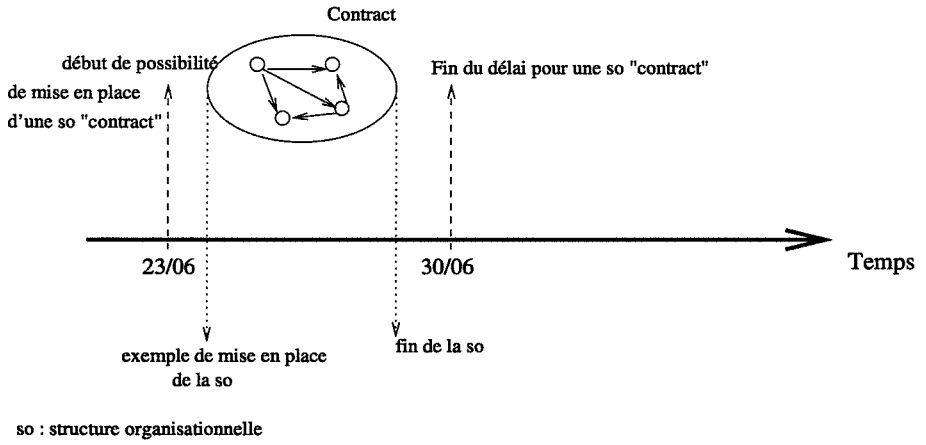


FIG. 6.4 – Propriétés temporelles concernant de la structure organisationnelle : *contract.*

(tos :name contract :roles (Manager bwPrinter cPrinter Binder)
:links (l_{MbP} l_{McP} l_{MB} l_{BbP} l_{BcP}) :groups ()
:tp (self d [06/23 06/30])
:tc ((bwPrinter f Manager) $\wedge$ (cPrinter f Manager) $\wedge$ (Binder f Manager))
)

Les contraintes temporelles sur les rôles ont été exprimées de manière à ce qu'ils ne se terminent qu'aussitôt que le manager a terminé, afin de permettre une réactivation de l'exécution au cas où le contrat ne se serait pas terminé avec succès : même si par exemple, l'échec se situe au niveau du relieur lors du processus de reliure. Le manager peut ainsi décider de commander à nouveau la réalisation du travail aux imprimeurs. Comme nous le voyons ci-après, les rôles "bwPrinter" et "cPrinter" ne possèdent pas de propriétés ni de contraintes temporelles. Par défaut, ils ont la même durée que celle de la *tos*. C'est la raison pour laquelle le lien d'autorité entre le "Manager" et ces deux rôles existe encore. Nous avons vu que par défaut (cf. section 6.4.1 : validité temporelle), la durée des liens correspond à la durée du plus court des rôles qui les définissent.

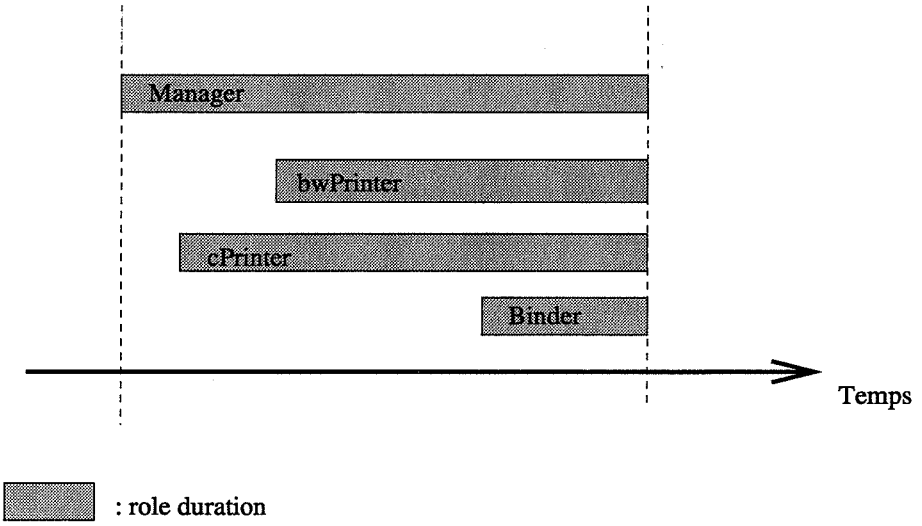


FIG. 6.5 – Contraintes temporelles entre les différents rôles de la "tos" : *contract.*

Le rôle "Manager" possède deux missions obligatoires. La première definit ce qui est permis dans le contexte de la gestion du contrat (mission "mm"). La seconde est dédiée au contrôle-qualité (mission "mq"). Elles seront précisées ci-dessous. Les contraintes temporelles de ce rôle (cf. figure 6.6) précisent que :

- la mission "mq" suit immédiatement (m=meets) la mission "mm",
- l'instant de début de ce rôle est égal à l'instant de départ de "mm" et enfin,
- son instant de fin correspond à la fin de "mq".

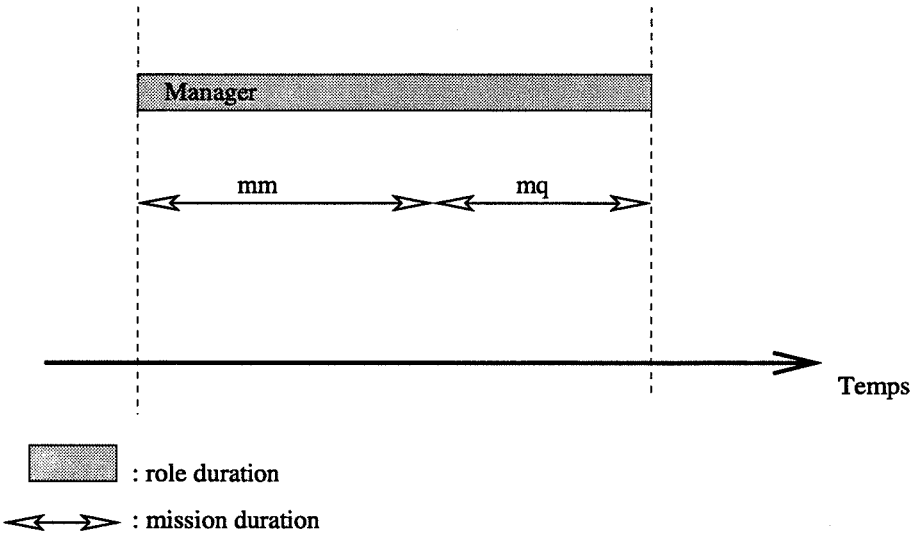


FIG. 6.6 – Contraintes temporelles dans la définition du rôle Manager et de ses missions associées.

(role :name Manager :missions ((O mm) (O mq)) :tp ()
:tc ((mm m mq) $\wedge$ (self f mq) $\wedge$ (self s mm)))
where

```
(mission :name mm :goals (g0) :plans (p0)
         :actions () :resources () :tp () :tc ())
(mission :name mq :goals (g4) :plans (p4)
         :actions (qc) :resources () :tp () :tc ())
```

La mission "mm" est composée d'un but permis "g0" et d'un plan permis "p0" qui réalise "g0". Aucune action ni ressource n'est permise dans cette mission. Le but "g0" correspond à l'exécution complète du job d'impression. Le plan "p0" définit une manière d'exécuter le job, en autorisant par exemple, une exécution parallèle de la réalisation des quatre sous-buts suivants :

1. bwPrinting 30 before 03/25 ("g1"),
2. cPrinting 15 before 03/28 ("g2"),
3. Binding 45 before 03/30 ("g3"),
4. quality control 45 before 03/30 ("g4").

Ainsi, l'agent qui jouera ce rôle, devra déléguer l'exécution du plan "p0". La mission "mq" est composée du but permis "g4", ainsi que du plan permis "p4" permettant de réaliser "g4" avec l'action permise "qc" (quality control).

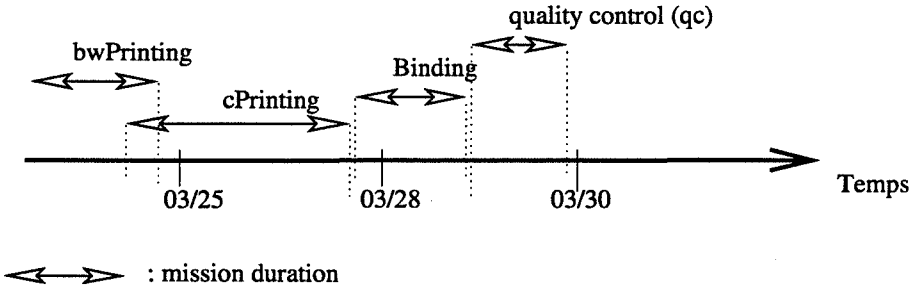


FIG. 6.7 – Illustration d'un exemple d'ordonnancement en accord avec les contraintes spécifiées.

De la même manière, dans ce qui suit, nous définissons les rôles et les missions relatifs au "bwPrinter", "cPrinter", "Binder" (cf. figure 6.8). Nous pouvons noter que la mission "mq1" pour le "Binder" est uniquement permise mais pas obligatoire. Aucune propriété temporelle n'est spécifiée. Un exemple de propriétés temporelles est donné avec les définitions des missions "mm1", "mm2" et "mm3". Nous pouvons observer que ces propriétés imposent que la réalisation de la mission "mm1" pendant les heures de nuit alors que la mission "mm3" doit être réalisée pendant les heures du jour.

```
(role  :name bwPrinter :missions ((O mm1)) :tp () :tc ())
(role :name cPrinter :missions ((O mm2) :tp () :tc ())
(role :name Binder :missions ((O mm3) (P mq1)) :tp ()
      :tc ((self s mm3)∧ ((self f mq1) ∨ (self f mm3)))
(mission :name mm1  :goals (g1) :plans (p1) :actions () :resources ()
```

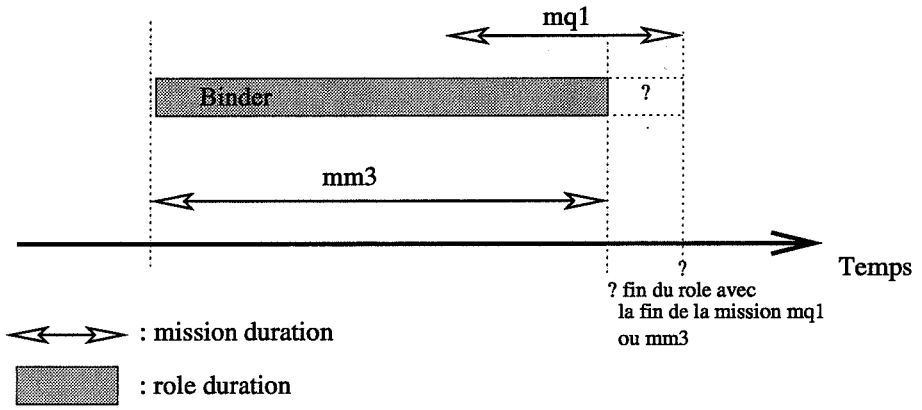


FIG. 6.8 – Contraintes temporelles du relieur (Binder).

```
                    :tp (self d [8 :00pm 8 :00am]) :tc ())
(mission :name mm2  :goals (g2) :plans (p2) :actions () :resources ()
                    :tp (self d [12 :00am 8 :00am]) :tc ())
(mission :name mm3  :goals (g3) :plans (p3) :actions () :resources ()
                    :tp (self d [8 :00am 8 :00pm] :tc ())
(mission :name mq1  :goals (g4) :plans (p4) :actions (qc) :resources ()
                    :tp () :tc ())
```

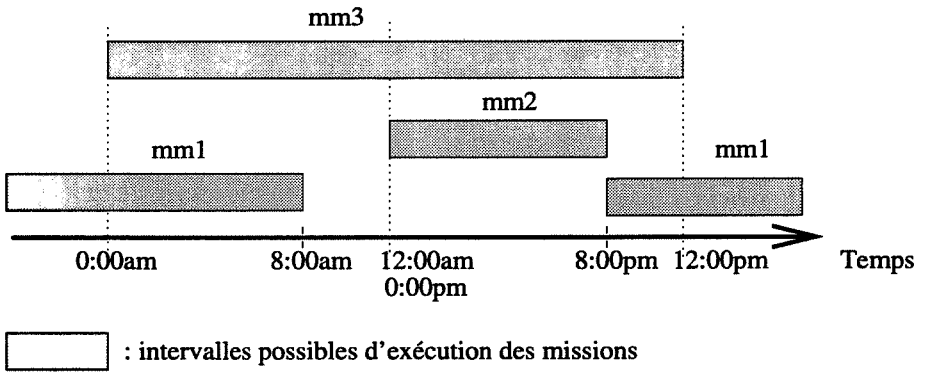


FIG. 6.9 - Exemples de propriétés temporelles explicitant les intervalles autorisés pour réaliser les missions.

Dans ce premier exemple, aucune expression temporelle ne positionne temporellement les liens. Comme nous l'avons dit, nous verrons dans la section suivante (cf. 6.4.2) que leur durée est limitée par défaut par celle du plus court des rôles associés.

(link :name l_{MbP} :type Aut

:source (:role Manager :missions (mm mq))
:target (:role bwPrinter :missions (mm1)) ...
:tp () :tc ())
(link :name l_{McP} :type Aut
:source (:role Manager :missions (mm mq))
:target (:role cPrinter :missions (mm2)) ...
:tp () :tc ())
(link :name l_{MB} :type Aut
:source (:role Manager :missions (mm mq))
:target (:role Binder :missions (mm3)) ...
:tp () :tc ())
(link :name l_{BbP} :type Aut
:source (:role Binder :missions (mm3))
:target (:role bwPrinter :missions (mm1)) ...
:tp () :tc ())
(link :name l_{BcP} :type Aut
:source (:role Binder :missions (mm3))
:target (:role cPrinter :missions (mm2)) ...
:tp () :tc ())

Le second exemple que nous allons présenter maintenant, permet de montrer que la modification d'une règle temporelle a un impact sur la définition de la structure organisationnelle.

Second exemple

Supposons, à présent, que les règles aient changé, et que le contrôle de qualité doit être délégué par le manager au relieur (binder), par exemple. Cette modification de la règle a des conséquences sur la spécification temporelle de la structure organisationnelle, elle-même. Les liens d'autorité du "Manager" sur les imprimeurs existent toujours mais leur durée est contrainte uniquement au traitement de la sous-tâche (bwPrinting, cPrinting) elle-même. Un lien d'autorité du "Binder" sur les imprimeurs ("bwPrinter" "cPrinter") existe durant le traitement et durant le contrôle-qualité. Les rôles imprimeurs peuvent être arrêtés après le contrôle de qualité et avant la reliure.

Ces aspects mènent à la définition d'une nouvelle structure organisationnelle temporelle "contract02", en grande partie semblable à la précédente définition. Les changements sont exprimés dans le champ :tc où la mission "mq1" apparaît :

(tos :name contract02 :roles (Manager bwPrinter cPrinter Binder)
:links (l_{MbP} l_{McP} l_{MB} l_{BbP} l_{BcP}) :groups ()
:tp (self d [03/23 03/30])
:tc ((bwPrinter f Binder.mq1) $\wedge$ (cPrinter f Binder.mq1) $\wedge$ (Binder f Manager))

)

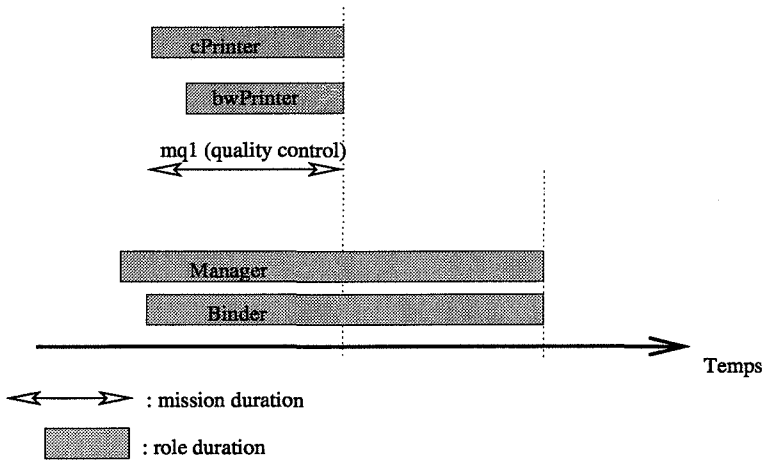


FIG. 6.10 – Nouvelles contraintes temporelles sur la définition de la tos *contract2*

Les nouvelles contraintes sont exprimées sur les rôles d'imprimeurs qui doivent se terminer aussitôt que le contrôle-qualité réalisé par le rôle relieur ("Binder") est achevé. Comme nous pouvons le voir, la définition de la mission "mq" a changé : l'action de contrôle-qualité n'est plus autorisée. L'agent concerné doit par conséquent déléguer l'exécution du contrôle de la qualité.

```
(role :name Manager :missions ((O mm) (O mq)) :tp ()
    :tc ( (mm m mq) ∧ (self f mq) ∧ (self s mm) ) )
(mission :name mm :goals (g0) :plans (p0)
        :actions () :resources () :tp () :tc ())
(mission :name mq :goals (g4) :plans ()
        :actions () :resources () :tp () :tc ())
```

La modification de la règle n'impose aucune modification sur la définition des autres rôles.

En cas d'échec lors du processus de reliure par l'agent jouant le rôle "Binder", l'agent jouant le rôle "Manager" peut lui ordonner de renégocier les jobs d'impressions, puisqu'à partir du moment où les imprimeurs ont achevé leur contrat : le

manager n'a plus d'autorité sur eux. L'agent jouant le rôle "Manager" peut imposer une telle pénalité sur l'agent jouant le rôle "Binder" tant qu'aucun des deux rôles n'est terminé. Ceci sera exprimé au niveau des liens "l_{MbP}" et "l_{McP}" au moyen de nouvelles contraintes spécifiant que le lien doit se terminer aussitôt que les missions d'impressions sont achevées.

(link :name l_{MbP} :type Aut
:source (:role Manager :missions (mm mq))
:target (:role bwPrinter :missions (mm1)) ...
:tp () :tc (self f mm1))
(link :name l_{McP} :type Aut
:source (:role Manager :missions (mm mq))
:target (:role cPrinter :missions (mm2)) ...
:tp () :tc (self f mm2))

Le contexte d'utilisation des liens d'autorité entre le relieur ("Binder") et les imprimeurs a été étendu à la mission "mq1".

(link :name lBbP :type Aut
:source (:role Binder :missions (mm3 mq1))
:target (:role bwPrinter :missions (mm1)) ...
:tp () :tc ())
(link :name lBcP :type Aut
:source (:role Binder :missions (mm3 mq1))
:target (:role cPrinter :missions (mm2)) ...
:tp () :tc ())

Ces contraintes sont plus restrictives pour la définition des liens ainsi elles ne risquent pas d'imposer une durée excessive ou inacceptable à la structure. En d'autres termes, elles ne rompent pas la validité temporelle globale de la structure organisationnelle temporelle que nous avons présentée dans la section précédente.

6.5 Discussion

Les modèles et techniques que nous avons présentés dans ce chapitre, permettent de prendre en compte de manière explicite la composante temporelle au sein de l'organisation. En fonction des applications, certains sont mieux adaptés que d'autres. Le modèle proposé est appliqué à un modèle organisationnel

spécifique mais devrait aisément pouvoir être transposé à tout autre modèle organisationnel. Notre choix sur un modèle s'appuyant sur la notion de structure organisationnelle a été influencé par les applications concrètes sur lesquelles nous avons travaillé. En effet, ces applications que nous verrons en troisième partie, présentent un aspect organisationnel (dynamique) se modélisant facilement à l'aide de structures organisationnelles. Le langage TOSL [CB01] permet d'expliciter les aspects temporels internes et externes d'une structure organisationnelle et présente l'avantage de s'appuyer sur le langage d'expressions temporelles L_T défini au chapitre 4 complété par le langage L_ω. Par rapport aux différentes approches présentées au chapitre 3, nous avons fait le choix de séparer explicitement les aspects temporels de l'organisation par rapport aux autres dimensions d'un système multi-agents. Ces aspects temporels sont représentés eux-mêmes de manière explicite mais dans un format commun aux autres dimensions. Le résultat de ces choix nous a permis de développer un langage de spécification de structures organisationnelles temporelles directement interprétable au sein de l'agent.

Par conséquent, pour utiliser et prendre en compte ces techniques et modèles relatifs à l'organisation (présentés dans ce chapitre) et à l'interaction (cf. chapitre 5) selon les besoins, il est nécessaire de définir une structure d'agent permettant de gérer l'une ou l'autre indépendamment voire les deux simultanément. Nous proposons un modèle d'agent adapté à ces besoins dans le chapitre suivant.

7

Modèle d'agent temporel : TAG

Sommaire

7.1 Différents niveaux de prise en compte du temps dans l'agent

Nous allons voir à présent où et comment différents aspects temporels décrits précédemment peuvent s'intégrer dans une architecture d'agent. Pour commencer, mettons l'accent principalement sur les types de temps qui peuvent intervenir dans un agent du fait de son fonctionnement au sein d'un SMA temporel.

En fonction de la dimension concernée et de sa provenance, nous nous sommes penchés sur la nature du temps manipulé au sein de l'agent et nous distinguons quatre "types de temps" différents. Si nous considérons un agent constitué des quatre facettes *a, e, i, o*, ces différents temps de l'agent interviennent principalement à différents endroits de l'architecture de l'agent (cf. figure 7.1).

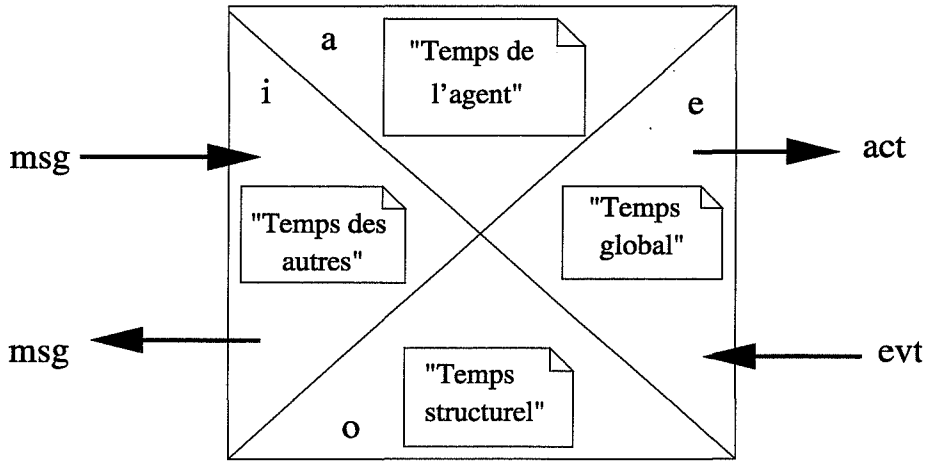


FIG. 7.1 – Les temps de l'agent TAG

Dans le contexte de travail où nous nous situons, les agents évoluent au sein d'un environnement très dynamique et doivent être à même de prendre en compte cette composante temporelle à la fois dans :

- l'*interaction* pour pouvoir échanger des messages temporels. Si l'on souhaite caractériser ce type de Temps, cela correspond au *temps des autres* qui regroupe les aspects temporels provenant des autres agents. Il recouvre principalement les aspects du temps liés aux messages reçus par l'agent et plus généralement relatifs à l'interaction,
- l'*organisation* pour mettre en place et suivre une structure organisationnelle temporelle. Nous appelons donc ce temps le *temps structurel* : nous considérons que ce temps concerne tous les aspects temporels qui permettent de structurer le système,
- le *raisonnement interne* pour gérer l'ordonnancement de ses propres tâches. Nous appelons ce type de Temps le *temps de l'agent* puisqu'il concerne le temps qui est pris en compte uniquement au sein du raisonnement interne de l'agent. Il constitue la partie de la composante temporelle qui est utilisée par l'agent pour gérer son comportement en générant et déclenchant ses actions,
- l'*environnement* en agissant aux moments opportuns et en percevant des informations utiles provenant de l'extérieur. C'est à dire être capable de les dater et d'effectuer un "filtrage temporel" pour vérifier et retenir ce qui a un sens pour l'agent. Nous distinguons donc dans cette dimension, le *temps global* lié à l'environnement, c'est à dire par exemple, les aspects de la composante temporelle relatifs aux événements qui adviennent dans l'environnement, à l'évolution des ressources, etc.

Nous pensons que cette approche du temps peut aider à avoir une vision plus aisée de la composante temporelle dans un agent de la même manière que l'approche *AEIO* offre une vision synthétique d'un SMA.

Les systèmes que nous avons décrits, mettent chacun l'accent sur un aspect particulier. Cette explicitation de quatre types de temps peut offrir un angle pour aborder le problème temporel et s'affranchir d'une partie de la complexité et de la richesse de ce problème en délimitant plus clairement les aspects que nous souhaitons traiter.

Le modèle d'agent a été construit et motivé à partir de nos observations issues de la première partie de ce mémoire. Il est également adapté aux modèles que nous avons définis précédemment. Enfin, afin d'être plus précis et de ne pas rester à un niveau trop abstrait, certains aspects sont relatifs aux applications que nous présenterons en troisième partie. Cela nous permet en outre, de définir plus concrètement les aspects temporels.

7.2 Architecture de l'agent : TAG

L'architecture d'agent temporelle (*TAG* : **T**emporal **Ag**ent) que nous proposons, comporte quatre modules périphériques s'articulant autour d'un cinquième central (cf. figure 7.2). Les quatre modules représentent les différentes facettes a, e, i, o et le cinquième rassemble les mécanismes relatifs au **contrôle** et à la coordination des facettes ainsi que les états mentaux (MS) propres à chacune des facettes de l'agent. Ces facettes fonctionnant de manière autonomes et indépendantes, il est nécessaire de mettre en place un contrôleur permettant de s'assurer qu'elles n'interfèrent pas entre elles en prenant des décisions irrationnelles ou ne provoquent pas d'interblocages.

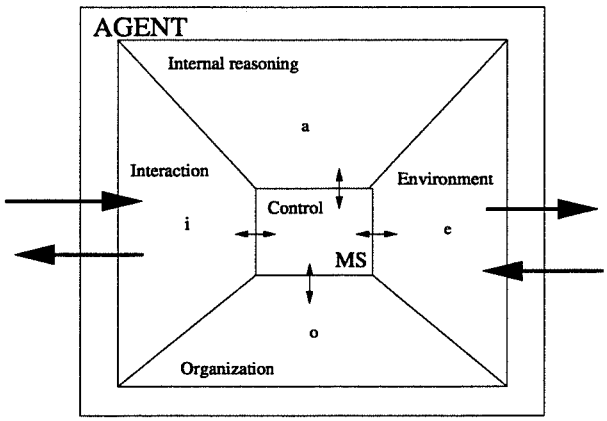


FIG. 7.2 – Architecture d'agent temporel : TAG

Nous avons orienté notre architecture vers un fonctionnement des facettes autonome et indépendant des autres facettes. Chacun des modules[15] sera défini plus

[15]Nous utilisons le terme "module" plutôt que "composant" dans le sens où cette architecture n'est pas une architecture d'agent à base de composants comme on peut le trouver dans [MB01] [Ric01].

précisément dans ce qui suit mais pour commencer, nous décrirons les aspects généraux. Ainsi, la totalité de l'agent (chacun des modules présentés ci-dessus) utilise une base de connaissance propre à l'agent.

Etats mentaux de l'agent

Comme nous l'avons dit, nous nous intéressons à des agents de type délibératifs. Notre modèle d'agent s'inspire du modèle BDI (cf. chapitre 1.2) et utilise une base de connaissance qui gère des états mentaux. Cette base de connaissance est structurée également selon les facettes : les états mentaux goal, belief, intention et s-commit pour la notion d'engagement (cf. section 4.2.1) sont classés en fonction de la facette (a,e,i ou o) à laquelle ils appartiennent. Ce classement permet de simplifier les traitements ainsi que les recherches au sein de la base de connaissance de l'agent. Les états mentaux possèdent une estampille temporelle spécifiant leur validité. La maintenance de la base de connaissance se trouve plus aisée car chacune des facettes peut retirer un état mental devenu désuet[16]. Tous les aspects temporels sont spécifiés avec le langage d'expressions temporelles L_T

Ces choix nous permettent d'avoir une représentation uniforme et donc un modèle unifié d'expression et manipulation du temps au sein de l'agent.

A présent, nous allons compléter la description de notre modèle d'agent en présentant chacun des modules de l'architecture.

7.3 Facette Interaction

La facette interaction de l'agent fonctionne de manière indépendante et autonome en suivant un cycle spécifique. Pour décrire le fonctionnement, nous adopterons une approche descendante. Ce cycle comporte trois phases principales qui sont :

1. la *réception de messages* consistant à vérifier si des messages ont été reçus et restent en attente d'être traités.
2. le *traitement des messages* qui concerne l'interprétation des nouveaux messages, la vérification et la mise à jour des états mentaux liés à l'interaction et la préparation des messages à émettre.
3. l'*émission des messages* en attente d'être envoyés.

La phase intermédiaire *traitement des messages* est évidemment la plus complexe et se décompose en différentes sous-phases. Prenons le cas de réception de message et de message à émettre :

[16]Il est possible d'intégrer à ce niveau un gestionnaire d'historique permettant de garder une trace au besoin des états mentaux qui sont retirés au lieu de les supprimer directement. Notre contexte d'application ne nous a pas amené à spécifier un tel gestionnaire.

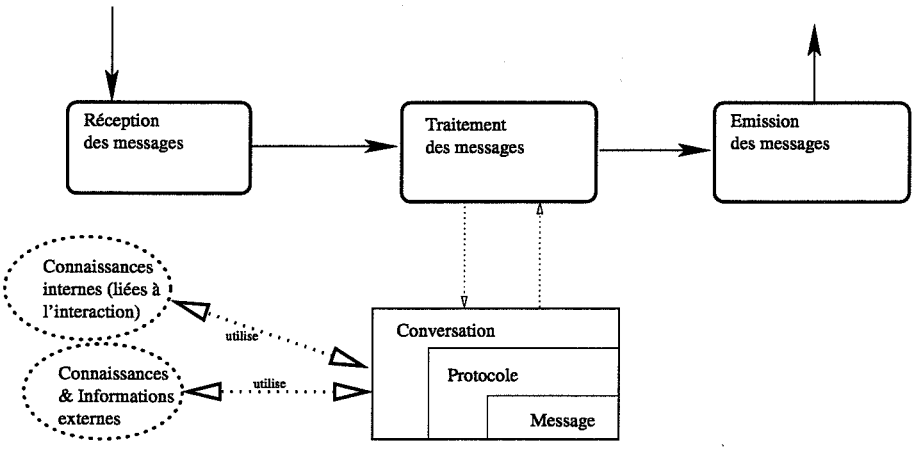


FIG. 7.3 – Fonctions de traitement de l'interaction

Message reçu :
Le message est interprété en traitant chacune des parties relatives à l'interaction :

1. L'identifiant de conversation est confié à la partie gestion de conversation (cf. fig 7.4) qui contient et active les mécanismes de gestion de conversations. En l'occurrence, une structure de conversation est mise à jour ou est créée si on se trouve en début de conversation.

2. Les informations liées au protocole sont interprétées comme par exemple la mise en place d'une garde temporelle pour envoyer la réponse, la vérification du respect du protocole et la proposition de transitions possibles. Les informations au sujet du protocole sont situées dans une base de protocoles (cf. fig 7.4).

3. Le message en lui-même est interprété en reconstruisant l'état mental correspondant à l'intention du locuteur à partir des champs mas (sma) et content (contenu) du message.

Message à émettre :
Le message est construit en réutilisant un identifiant de conversation à partir d'une conversation existante ou en en créant une nouvelle si nécessaire (cf. chapitre 5.4). Ensuite le message est construit en sélectionnant l'état suivant du protocole en cours (cf. chapitre 5.3) ou un protocole satisfaisant dans le cas d'une nouvelle conversation. Enfin chacun des champs du message en lui-même sont complétés (cf. chapitre 5.2). Le temps est géré par l'intermédiaire d'états mentaux de type interaction et cette gestion s'appuie sur le gestionnaire de relations entre intervalles *GRIS* pour vérifier le respect des contraintes temporelles comme une date limite de réponse.

Cette description brève permet d'avoir une idée du fonctionnement mais des exemples concrets seront donnés en troisième partie.

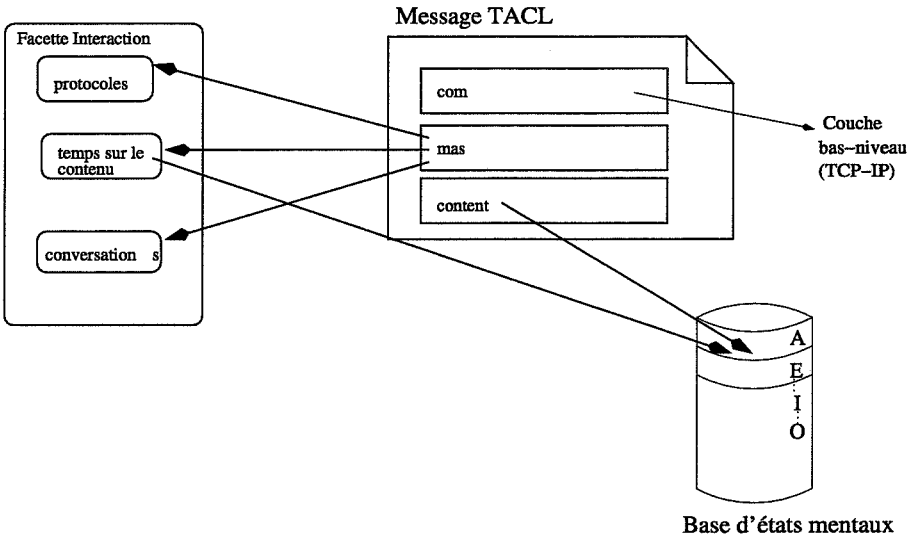


FIG. 7.4 – Interprétation des trois niveaux du message

7.4 Facette Organisation

Le fonctionnement de cette facette consiste à maintenir valide une entité organisationnelle instanciée à partir d'une structure organisationnelle. Chronologiquement, l'agent dispose d'un ensemble de structures organisationnelles spécifiées en langage *TOSL* (chargées à l'initialisation de l'agent) qui lui permettent de mettre en place une entité organisationnelle temporelle. Cette entité est mise à jour à partir des informations parvenant à la facette organisation (vérification des états mentaux concernant l'organisation) et maintenue à jour sur le plan temporel par le gestionnaire organisationnel.

Par exemple, si une structure organisationnelle de type "contract" (cf. exemples du chapitre 6) est mise en place dans le SMA, les agents chargent la structure correspondante et l'instancient (créant ainsi une entité organisationnelle) à mesure que les informations organisationnelles leur arrivent. Ces informations peuvent être par exemple : "l'agent X joue le rôle Manager pendant l'intervalle de temps T", "l'agent X cherche un agent Imprimeur", etc.

Le gestionnaire d'organisation s'appuie sur le gestionnaire de relations entre intervalles *GRIS* pour traiter et maintenir les aspects temporels de la structure organisationnelle active (ou entité organisationnelle).

Cette facette fonctionne de manière autonome et indépendante dans le sens où elle gère uniquement les informations organisationnelles sous forme d'états mentaux temporels, la disparition ou l'apparition d'une croyance se fait de concert avec la mise à jour de l'entité organisationnelle correspondante. Cette facette s'appuie uniquement sur ses connaissances organisationnelles pour vérifier la cohérence des informations provenant de l'extérieur.

Cycle du gestionnaire d'organisation Le cycle lié au gestionnaire d'organisation est également simple (cf. figure 7.5). Deux types de tâches sont distinguées :

1. l'*instanciation de la structure organisationnelle*, qui permet de spécifier quel agent joue quel rôle et éventuellement quelle mission. Cette instanciation se fait à partir de l'arrivée d'états mentaux liés à l'organisation. Lorsque l'agent lui-même décide de prendre part à la structure organisationnelle, il en informe les autres agents (pour l'instant, un message de type "broadcast" est envoyé) ce qui entraînera une *instanciation de la structure organisationnelle* par ces agents.
2. la *maintenance temporelle de la structure*[17] qui concerne la mise à jour de la structure à partir des informations temporelles disponibles : un rôle qui est terminé est "retiré" de la structure, par exemple.

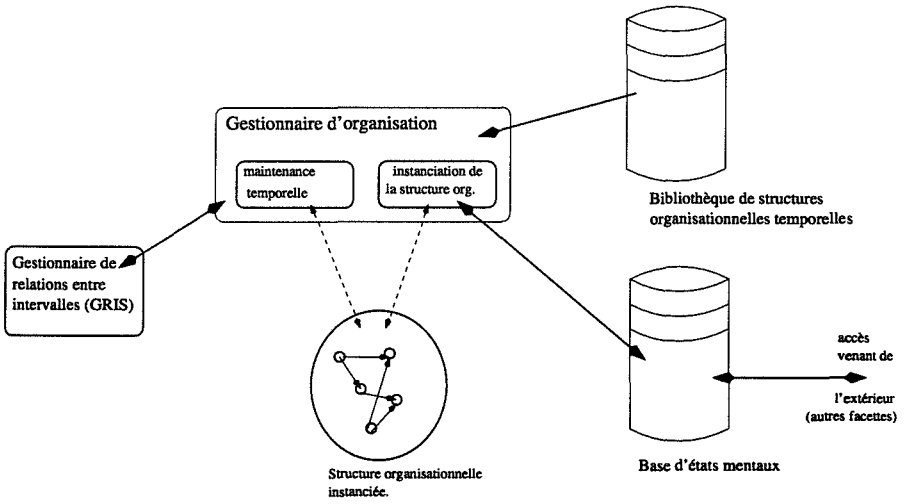


FIG. 7.5 - Fonctionnement du gestionnaire de structure organisationnelles temporelles

7.5 Facette agent ou raisonnement interne

La facette agent (ou raisonnement interne) contient les mécanismes de l'agent pour gérer et ordonner ses actions. Notre architecture reprend un modèle relativement classique (s'appuyant sur la hiérarchie Goal, Plan, Action) mais qui a donné lieu à de nombreuses variantes. Ce modèle est issu du monde industriel qui distingue les trois niveaux :

1. *stratégique* correspondant à la notion de but (Goal)
2. *tactique* qui concerne les moyens concrets, les "recettes" de réalisation (Plan) permettant d'atteindre les buts du niveau *stratégique*.

[17]Nous utilisons ici également le terme de "structure" organisationnelle pour uniformiser la dénomination des traitements mais cette structure étant instanciée, le terme adéquat est "entité organisationnelle".

3. *opérationnel* est relatif aux actions primitives qui sont directement exécutables et qui constituent les plans du niveau *tactique.*

Notre modèle manipule donc des buts, satisfaits au moyen de plans qui correspondent à une séquence d'actions directement exécutables par les agents. La forte dépendance avec l'application explique pourquoi nous ne pouvons donner beaucoup de précisions ici. Ces aspects seront précisés et complétés au chapitre 8.

Cycle de traitement interne

Le cycle de traitement du raisonnement interne comporte quatre opérations différentes qui sont l'adoption, le raisonnement, la décision et l'engagement (cf. figure 7.6).

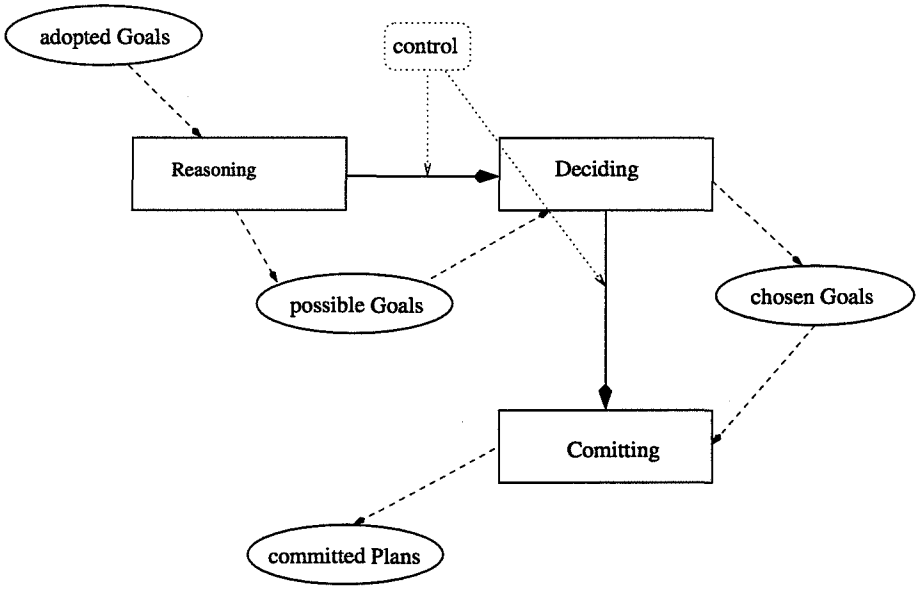


FIG. 7.6 – Les phases du raisonnement de l'agent TAG

Le raisonnement interne d'un agent sélectionne parmi les buts qu'il peut poursuivre (qui ont été adoptés : phase d'adoption), certains buts réalisables (phase de raisonnement). Il sélectionne ensuite un plan (phase d'engagement) permettant d'atteindre ce but parmi ceux disponibles (phase de décision). Le plan est ensuite suivi et réalisé action par action. De nombreuses études et méthodes (GANTT, PERT, etc.) et même des logiciels (Ilog Solver, Ilog Planner, Ilog Scheduler, modules spécialisés de SAP, etc.) permettent de s'affranchir des problèmes relatifs aux notions d'ordonnancement et de planification, c'est pourquoi nous ne revenons pas dessus dans le cadre de ce mémoire. Sur le plan temporel, les "savoir-faire" de l'agent sont spécifiés avec une durée de réalisation maximale permettant de s'assurer en cas de réalisation qu'il est au moins réalisable dans le temps demandé. La figure 7.6 souligne également l'intervention possible de la facette contrôle que nous présenterons en section 7.7.

7.6 Au sujet du traitement de l'environnement

Comme nous l'avons vu, la composante temporelle est peu exploitée au niveau de l'environnement dans les SMA. Dans le cadre de notre travail, cet aspect est également peu développé car les applications sur lesquelles nous avons travaillé ne nécessitaient pas un modèle d'environnement temporel explicite.

Cependant, nous proposons une description sommaire de la facette environnement qui est avec la facette interaction, la seule facette de l'agent possédant des entrées/sorties avec l'extérieur de l'agent (cf. figure 7.2).

Le fonctionnement de la facette environnement peut être similaire à ce qui est proposé dans l'architecture présentée par Barbara Hayes-Roth pour Guardian (cf. section 3) avec deux fonctions principales :

1. une fonction d'"entrée" (*perception*) permettant de percevoir et sélectionner des événements provenant de l'environnement, de les estampiller temporellement afin d'en faire des connaissances accessibles et utilisables par les autres facettes.
2. une fonction de "sortie" (*action*) qui se charge d'interpréter "l'agenda" (actions ordonnancées) prévu par la facette *raisonnement interne* et d'exécuter au bon moment chacune des actions demandées.

7.7 Contrôle des facettes

Nous avons vu au chapitre 3 que la composante temporelle peut concerner de nombreux aspects; certains relevant directement de l'application (ordonnancement des tâches), d'autres plutôt relatifs à l'aspect "multi-agent" (gestion temporelles des communications, respect des protocoles et d'une structure organisationnelle, etc.). Tous ces aspects doivent pouvoir être pris en compte au sein du modèle d'agent. Cependant, selon les applications ou les objectifs du SMA (cf. chapitre 1), tous ces aspects ne doivent pas obligatoirement être pris en compte simultanément. Pourtant bien qu'étant très complexe à gérer, une vision complète des aspects temporels peut éventuellement imposer la combinaison de tous ces aspects. Ce chapitre propose ainsi une possibilité d'intégrer et de généraliser les modèles d'interaction et d'organisation temporels présentés dans les chapitres qui précèdent. L'agent servant alors de "creuset" récoltant les outils nécessaires pour pouvoir prendre en compte tel ou tel aspect de la composante temporelle. Nous précisons que le modèle d'agent proposé ici n'est qu'"un" creuset possible permettant de d'intégrer et éventuellement rassembler les outils définis dans les autres chapitres (4, 5 et 6) de cette partie.

Le modèle en facettes décrit précédemment permet de pouvoir prendre en compte chacune des dimensions du temps mises en exergue au début de ce cha-

pitre :

- le temps dans l'agent avec la facette raisonnement interne ($a(t)$),
- le temps dans l'environnement avec ($e(t)$),
- le temps dans l'interaction avec ($i(t)$),
- le temps dans l'organisation avec ($o(t)$).

Cependant, comme nous l'avons souligné en conclusion du chapitre 3, les architectures d'agent existantes (cf. section 3.5) imposent une gestion particulière du temps au sein de l'agent : généralement, une des facettes, souvent le raisonnement interne de l'agent ($a(t)$) est prépondérante sur les autres. Cette gestion particulière est souvent accompagnée d'une propagation unidirectionnelle au sein des autres facettes. Notre modèle propose un fonctionnement "indépendant" des facettes ou du moins, permet à chacune des facettes de fonctionner même en l'absence de l'une ou l'autre des autres facettes. Par exemple, nous pouvons définir un *agent social temporel* et passer à un *agent temporel interagissant* en inhibant $o(t)$ ou encore à un *agent autonome temporel* en désactivant également $i(t)$.

Nous avons également pu remarquer que, lorsque le temps était pris en compte dans différentes facettes, il n'était pas possible de modifier dynamiquement l'ordre dans lequel il était pris en compte. Ces deux limitations sont un frein à un véritable raisonnement temporel intégrant l'ensemble des dimensions qu'apportent les modèles manipulés au sein des SMA. En effet, selon les applications, il peut être intéressant de pouvoir privilégier la contrainte temporelle issue d'une relation avec un autre agent ou liée au rôle dont l'agent est responsable, par rapport aux contraintes temporelles relatives au plan qu'il est en train d'exécuter (possibilités de "procrastination"). Dans d'autres cas, l'inverse pourra être plus intéressant. De même, pour interaction/raisonnement interne ou même interaction/organisation.

De manière générale, il est donc nécessaire de définir un mécanisme offrant la possibilité de privilégier une facette par rapport aux autres et pouvoir ainsi spécifier des comportements d'agents dirigé par l'organisation (comportement que nous appelons *o-driven*) ou par l'interaction (*i-driven*), par exemple. Le contrôleur que nous présentons dans cette section, peut jouer ce rôle.

7.7.1 Fonctionnalités du contrôleur

La composante *contrôle* du modèle *TAG* a trois fonctions :

1. une fonction de *vérification* pour répondre à des questions du type "ai-je le droit" de jouer tel rôle, d'effectuer telle action, etc.

2. une fonction de *contrôle* (au sens "maîtrise") qui s'appuiera sur des stratégies pour valider des choix dans la sélection des buts, des plans, etc. De manière pragmatique, cette fonction permettra de répondre à une question de type : "quelles sont les autres possibilités disponibles ?".
3. une fonction de *comportement "pathologique"* qui spécifiera le comportement de l'agent en cas de problème. Ces problèmes peuvent être le non respect de règles organisationnelles, de contraintes temporelles spécifiées ou n'importe quel type d'infraction susceptible d'entraîner un comportement incohérent de l'agent (et donc du SMA). Cette dernière fonction consiste à offrir une réponse à la question : "que dois-je faire en cas d'infraction ?".

La composante *contrôle* peut se révéler très riche et très complexe et dans le cadre de ce travail, nous illustrons uniquement des aspects relatifs au respect des aspects temporels. Dans ce mémoire, nous considérons que la *fonction de vérification* s'appuie sur la structure organisationnelle pour vérifier le respect des règles. La fonction de contrôle s'appuie sur des règles stratégiques simples comme "priorité à la rapidité d'exécution" qui permettra de choisir le plan le plus rapide à exécuter ou bien "fiabilité maximale" qui consistera à choisir le plan qui imposera à l'agent de déléguer le moins de tâches possibles par exemple donc de choisir un plan dont il est capable de réaliser un maximum d'actions sans devoir faire appel à d'autres agents. Ces aspects relèvent du contrôleur puisqu'ils peuvent être relatifs à différentes facettes : la stratégie "fiabilité maximale" par exemple peut être imposée par l'organisation : un agent échouant à réaliser les tâches pour lesquelles il s'est engagé, peut ensuite être évincé d'une alliance interorganisationnelle.

7.7.2 Spécification de comportement

La fonction de contrôle décrite ci-dessus se prête bien à la mise en œuvre d'un comportement comme évoqué précédemment. En effet, il suffit de mettre en place (éventuellement temporairement) une stratégie privilégiant une facette particulière : si l'on souhaite mettre en place un comportement "*o-driven*", le contrôleur appliquera une stratégie permettant de respecter les aspects organisationnels avant tout autre.

Les différents comportement que l'on peut spécifier sont les suivants :

1. *o-driven* qui privilégie les aspects organisationnels. Ce comportement peut être schématisé de la manière présentée sur la figure 7.7.

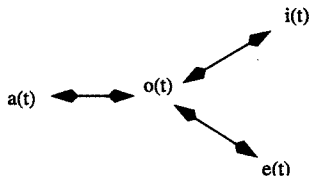


FIG. 7.7 – Le comportement *o-driven*

2. *i-driven* qui satisfait avant tout les contraintes liées à l'organisation. Le comportement *i-driven* se symbolise de la manière présentée sur la figure 7.8.

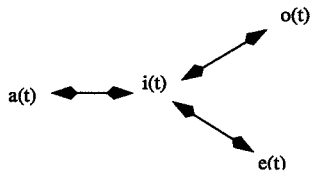


FIG. 7.8 – Le comportement *i-driven*

3. *a-driven* qui permet de s'assurer que l'agent respecte son plan de charges en ignorant les informations provenant des autres facettes. Nous symbolisons ce fonctionnement avec la figure 7.9.

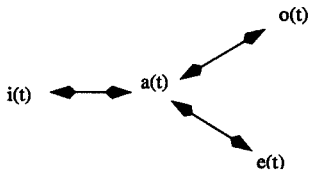


FIG. 7.9 – Le comportement *a-driven*

4. *e-driven* qui privilégie les aspects relatifs à l'environnement et se peut se schématiser de la manière présentée sur la figure 7.10.

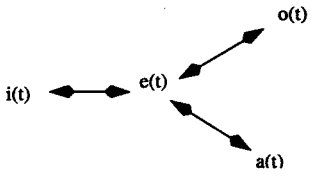


FIG. 7.10 – Le comportement *e-driven*

Il est ainsi possible de particulariser le fonctionnement de l'agent en fonction du déroulement des échanges entre facettes. Nous allons à présent illustrer ces différents échanges.

7.7.3 Echanges de messages entre les facettes

Nous avons évoqué à plusieurs reprises précédemment, que différentes informations transitent entre les facettes. Afin d'illustrer nos propos, nous présentons quelques exemples concrets de messages susceptibles d'être échangés :

- de la facette *interaction* vers la facette *organisation* : lors de la réception d'un message, la facette interaction peut interroger la facette organisation afin de vérifier si un lien d'autorité impose une réponse positive au message.
- de la facette *organisation* vers la facette *interaction* : lorsque l'agent décide de jouer un rôle ; il en informe les autres en envoyant un message par le biais de la facette interaction.
- de la facette *raisonnement interne* vers la facette *organisation* : lorsqu'un agent souhaite exécuter un plan avec une action qu'il n'est pas capable de réaliser lui-même, il interroge la facette organisation pour obtenir un agent capable de réaliser l'action à qui il pourra déléguer la tâche.
- de la facette *agent* vers la facette *interaction* : pour déléguer une tâche, le raisonnement interne passe les informations à la facette interaction pour que cette dernière envoie un message à l'agent concerné.

Comme nous l'avons vu, le *contrôle* offre une possibilité de gérer ces "interactions" entre les facettes permettant ainsi mettre des priorités sur l'une ou l'autre des facettes.

L'intervention du contrôle dans le fonctionnement normal est dépendant des choix faits au niveau de l'application. En effet, il n'est pas toujours nécessaire d'avoir un contrôle omniprésent à chaque étape de raisonnement de l'agent. Nous présentons cependant dans la section suivante, un exemple de fonctionnement global de l'agent permettant de comprendre l'intervention de chacune des facettes. D'autres exemples seront proposés dans la troisième partie.

7.8 Exemples de fonctionnement

Pour terminer, nous allons voir différents exemples de fonctionnement afin d'illustrer les notions abordées dans ce chapitre.
Imaginons qu'un agent reçoive un message lui spécifiant de réaliser une impression dans l'heure avec une réponse dans la minute.

Le fonctionnement "classique" pourrait être le suivant (cf. figure 7.11) :

1. le message est traité par la facette $i(t)$, une garde est mise en place pour respecter le délai sur la réponse,
2. le contenu est décodé et mis à jour sous forme d'état mental concernant la facette $a(t)$ qui va évaluer en fonction de son plan de charge et de ses capacités si l'impression est réalisable dans le délai demandé,
3. la facette $o(t)$ vérifie le respect de la structure organisationnelle (que le rôle de l'agent, par exemple, autorise l'exécution d'une mission d'impression),
4. la facette $i(t)$ élabore ensuite la réponse et l'envoie avant que l'échéance soit passée.

5. la facette $e(t)$, enfin, génère l'action correspondant à l'impression demandée au moment prévu.

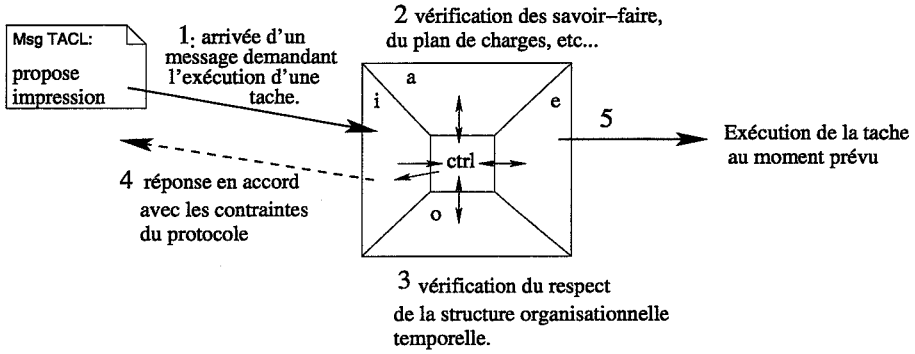


FIG. 7.11 – Exemple de fonctionnement de TAG

Ce déroulement ne précise pas l'intervention du module de *contrôle* qui comme nous l'avons dit, peut être configuré pour donner son accord à chaque étape. A présent, imaginons que le module de contrôle de l'agent spécifie un comportement spécifique :

- l'agent est *a-driven*, il possède un plan de charge très lourd, le message ne sera pas traité pour privilégier les traitements liés au raisonnement interne.
- l'agent est *o-driven*, le contrôleur empêche toute action enfreignant la structure organisationnelle par exemple ,
- l'agent est *i-driven*, tous les messages sont traités et si la réponse n'est pas prête dans les délais, le contrôle privilégie le respect des aspects relatifs à l'interaction en imposant l'envoi d'une réponse négative.
- l'agent est *e-driven*, le contrôleur privilégie les traitements liés à la facette $e(t)$ comme par exemple, les informations captées dans l'environnement provenant de l'imprimante (manque de papier, manque d'encre, etc.).

7.9 Discussion

Le modèle d'agent que nous avons décrit, peut intégrer les modèles liés à l'interaction et à l'organisation que nous avons proposés dans les chapitres précédents. Il permet une représentation uniforme des aspects temporels dans toutes les facettes en autorisant un fonctionnement indépendant de chaque facette et ainsi la définition de différents types d'agent temporel. Cela nous permet de définir un agent temporel $A(t)$ par le sextuplet suivant correspondant à quatre facettes temporelles, un mécanisme de contrôle ($\otimes(t)$) et une structure d'états mentaux (MS) :

$A(t)$= {a(t), e(t), i(t), o(t), $\otimes(t)$, MS}

Cette flexibilité quant à la définition du type d'agent est renforcée par la mise en place d'un module de contrôle permettant de spécifier un comportement

particulier et gérer une coordination temporelle au sein de l'agent. Cela nous permet de définir également le fonctionnement d'un agent temporel ($F_A(t)$) par une combinaison entre les différentes facettes temporelles et un module de contrôle temporel ($\otimes(t)$) gérant les interactions entre ces facettes :

$F_A(t)=\otimes(t)\ (a(t), e(t), i(t), o(t))$

Le fonctionnement d'un agent temporel correspond à la combinaison des quatre facettes temporelles possibles (a(t), i(t), e(t) et o(t)) contrôlées temporellement par le contrôleur ($\otimes(t)$) qui spécifie le comportement de l'agent. La focalisation de notre étude sur les aspects temporels dans les SMA nous a amené à envisager de se servir de la dimension temporelle pour aborder au niveau de l'agent, la problématique de la coordination sous l'angle temporel. En effet, nous avons vu que la prise en compte du temps peut faciliter la coordination des facettes puisqu'elle permet de s'assurer que les agents agissent au bon moment et en accord avec ce qui a été définis (règles d'interaction : protocoles, règles d'organisation : structures organisationnelles).

Ce modèle a été instancié sur deux applications qui sont décrites dans la partie suivante.

Troisième partie

Applications temporelles

Introduction

Dans cette partie, nous abordons des aspects plus concrets tels que l'implémentation des modèles proposés et leur application et mise en œuvre sur des cas d'application.
Cette partie commence avec un chapitre consacré à l'implémentation du modèle sur la plate-forme multi-agent *MAST* qui est développée au sein du laboratoire. Nous présentons ensuite deux applications permettant de valider notre modèle de prise en charge explicite du temps dans un SMA :

1. la première application est relative à la gestion d'alliances d'ateliers d'impression permettant de mettre en exergue un fonctionnement *i-driven* ainsi qu'un comportement *o-driven* selon les phases de l'application.
2. la seconde application concerne une entreprise fonctionnant sous un mode de gestion par projet. Nous verrons que l'on peut considérer que cet exemple nous permet de mettre en place une gestion orientée par l'organisation (*o-driven*).

Ces deux applications sont développées dans des objectifs différents : elles illustrent la *résolution distribuée de problèmes* pour la première et la *simulation décentralisée de problèmes* pour la deuxième.

8

Implémentation des modèles

Sommaire

Ce chapitre concerne l'implémentation des modèles et les techniques utilisées. Il complète notamment le chapitre précédent sur le modèle agent en décrivant les choix qui ont été faits pour le mettre en œuvre. Dans le cadre de notre travail, les stations de travail sont synchronisées grâce au logiciel *rdate* [rda94] et au protocole *ntp* [Mil01] mais d'autres possibilités sont disponibles ("*timed*" système client-serveur sous forme de package java [Sch98]). Nous commencerons par présenter la plate-forme sur laquelle nous avons implémenté les modèles proposés précédemment. Nous compléterons la description du gestionnaire de relations entre intervalles GRIS pour montrer son intégration au sein de la plate-forme. Nous décrirons ensuite les aspects liés à la mise en œuvre de TAG puis nous nous intéresserons à ceux concernant les langages TACL et TOSL.

8.1 Plate-forme multi-agents MAST

Pour commencer, nous précisons dès à présent que **cette plate-forme est actuellement en cours de développement**. La plate-forme *MAST* (Multi-Agent System Toolkit) est un environnement de développement et de programmation orientée multi-agents. Elle a deux objectifs :

1. fournir un environnement de développement pour construire des applications multi-agents (analyse, conception et programmation),
2. fournir une infrastructure d'exécution répartie et de test des applications multi-agents développées (validation, test, exploitation).

Aucun type d'application n'est a priori privilégié. Ainsi, le travail décrit dans ce mémoire a également pour objectif spécifique d'offrir et d'intégrer des moyens de prendre en compte des applications dans lesquelles la dimension temporelle doit être explicitée.

8.1.1 Services offerts

La plate-forme *MAST* a pour objectif de proposer les services suivants au programmeur :

1. un environnement d'exécution répartie des agents et de différents logiciels interopérant au travers de la plate-forme (module appelé "DeMAS"),
2. des outils d'observation, de mise au point et de test tant pour les agents que pour le système, ainsi que des outils d'administration d'une application multi-agents ("AdMAS")
3. des modèles réutilisables permettant d'organiser les traitements, de définir la structure d'échange commune, de définir les schémas de contrôle en termes de modèles d'Agent, d'Environnement, d'Interaction, d'Organisation ("GeMAS"),
4. des interfaces de développement d'applications multi-agents selon une méthodologie d'analyse et de conception ("MeMAS").

Cependant, au stade de développement actuel, par rapport aux plates-formes existantes (*Madkit*, *JADE*, *ZEUS*, *Volcano*), *MAST* est avant tout un environnement de développement avant d'être un environnement d'exécution.

8.1.2 Outils génériques

Au sein de la composante *GeMAS*, différents outils sont disponibles sur la plate-forme afin de développer un SMA. A propos des applications mises en place dans ce travail, deux outils sont utilisés :

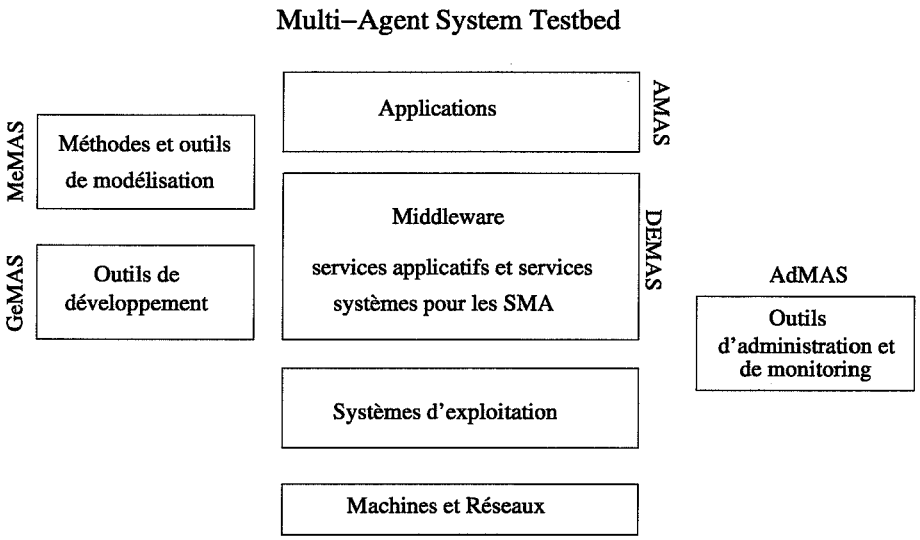


FIG. 8.1 – La plate-forme MAST

1. dans le cadre des études menées sur la dimension temporelle, nous avons ainsi développé un Gestionnaire de Relations en IntervalleS (GRIS) (cf. section 4.3) qui est basé sur le gestionnaire de graphes temporels *TimegraphI-II*. GRIS est un outil à part entière, adapté sur la plate-forme *MAST* et capable de fonctionner de manière indépendante. Il offre en outre un éditeur de graphes, une interface homme/machine permettant à l'utilisateur de rentrer des contraintes temporelles, de générer le graphe correspondant ainsi que toutes les relations temporelles possibles (ou souhaitées) qui peuvent en être déduites. Les relations entre objets temporels sont symbolique mais ce moteur peut traiter également des données numériques. Ce gestionnaire est codé en langage java et peut être utilisé par les agents pour déduire de nouvelles contraintes temporelles à partir d'un ensemble d'intervalles donné. La gestion du temps se fait de manière explicite avec le choix d'un temps linéaire, discret. La structure est basée sur la théorie d'Allen avec un temps manipulé sous forme d'intervalle traduit ensuite sous forme d'instants pour être utilisé dans les algorithmes de *TimeGraph*. Des copies d'écran illustrant ces fonctionnalités annexes sont présentées en annexe A.

2. un autre outil permettant de faire du raisonnement à base de règles a été adapté pour l'environnement MAST, il s'agit de *JESS* (*Java Expert System Shell*). Ce moteur d'inférence a été développé en Java et interprète les règles écrites en langage *CLIPS*. Des agents raisonnant à base de règles et utilisant ce moteur peuvent donc être développés sur MAST. Les règles peuvent être chargées à partir d'un fichier permettant de spécialiser le comportement d'un agent ou bien être ajoutées et modifiées en fonctionnement.

8.1.3 Ressources diverses liées à l'interaction et l'organisation

Actuellement, seul le langage d'interaction temporel *TACL* permet de prendre en compte la dimension temporelle dans les interactions. Quelques protocoles temporels (issus des besoins applicatifs) initient une bibliothèque de protocoles (cf. dans la suite section 8.4) qui s'étoffera avec le temps et étendra ainsi les possibilités d'interactions offertes aux agents. Quant à l'organisation, quelques exemples de structures organisationnelles temporelles (*tos*) ont été développées (issues également des besoins des applications sur lesquelles nous avons travaillé). Il faut reconnaître qu'à la différence des protocoles définis, ces différentes *tos* sont étroitement liées à l'application et donc difficilement réutilisables dans d'autres SMA. Cependant, le "canevas" sous-jacent peut être utilisé pour redéfinir ou adapter rapidement une nouvelle *tos* en attendant la finalisation d'outils s'appuyant sur les grammaires respectives définies (structure organisationnelle, protocole temporel). Après avoir décrit la plate-forme sur laquelle ce travail a été implanté. Intéressons nous plus particulièrement à sa mise en œuvre effective.

8.2 TAG : Description et schémas d'implémentation

Nous présentons dans cette section, la mise en œuvre du modèle d'agent temporel (cf. figure 8.2) et le schéma d'implémentation de l'agent à l'aide d'un diagramme de classes[18] (cf. figure 8.3). Afin de rendre la structure du code plus claire, nous avons également essayé de suivre et de faire apparaître pour l'implémentation l'approche *a, e, i, o.*

- La facette interaction utilise une base de connaissances sur l'interaction (KI), le langage TACL et la structure de protocoles temporels TIP.
  La facette organisation utilise des connaissances liées à l'organisation (KO), le langage TOSL pour charger une structure organisationnelle temporelle.
- La facette de raisonnement interne (RI) intègre des connaissances sur le raisonnement (KR) (connaissances sur ses propres compétences).
- La facette environnement n'apparaît pas dans notre implémentation car elle est peu développée dans notre cadre d'application. Ainsi les connaissances sur l'environnement (lois d'évolution par exemple) sont intégrées à la facette RI.

[18]Notation UML, les cardinalités, les noms de rôles, etc. sont omis

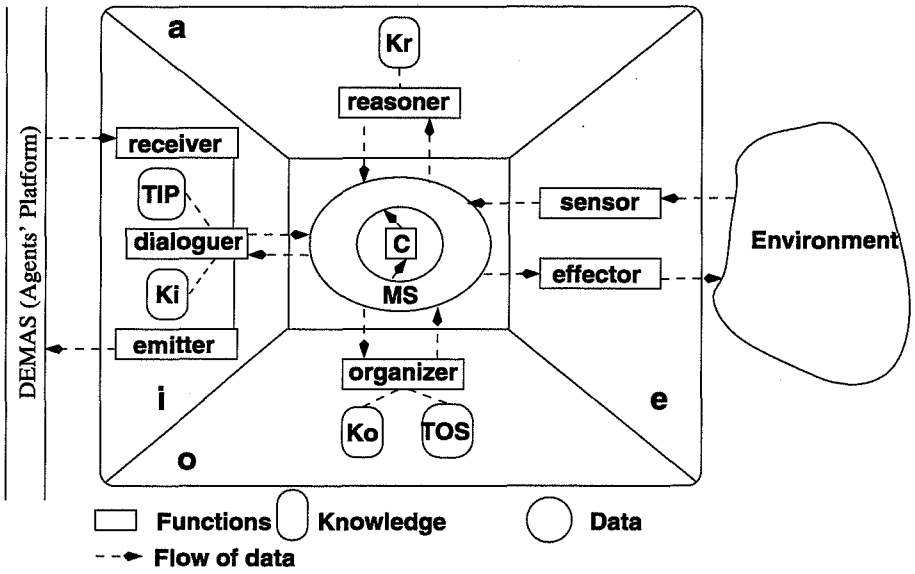


FIG. 8.2 – Architecture d'agent temporel social TAG

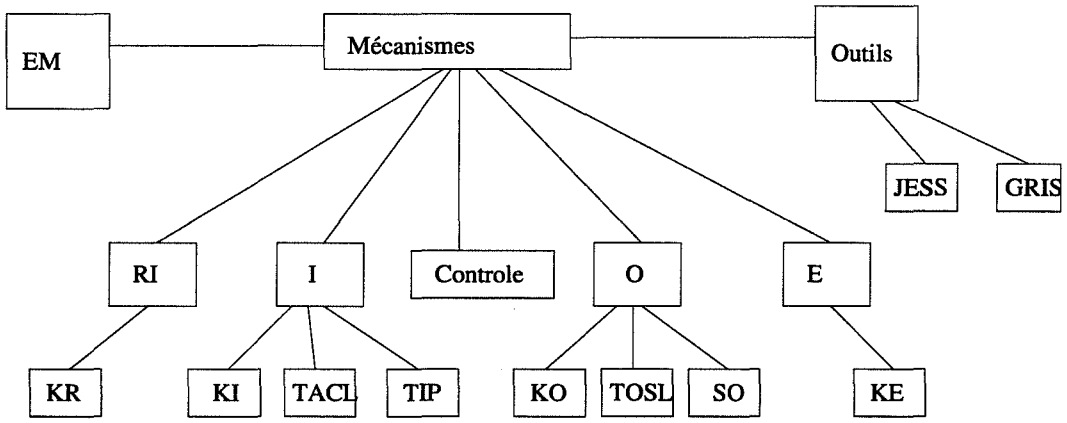


FIG. 8.3 – Diagramme de classes de TAG

8.2.1 Structure des différentes entités mentales manipulées

Nous allons passer en revue chacune des entités mentales utilisées par les facettes de l'agent.

But

Nous adoptons pour la description un point de vue "structure de donnée".

⟨goal⟩ : := :id ⟨goal-id⟩
:facet ⟨facet⟩
:time ⟨time⟩
:content ⟨content⟩

:status ⟨status⟩
:has-subgoal ⟨has-subgoal⟩
:has-plan ⟨has-plan⟩

- ⟨goal-id⟩ correspond à l'identifiant du but.
- ⟨facet⟩ spécifie l'ensemble des facettes auxquelles peut appartenir le but. La mention *undefined* permet de prendre en compte le cas où un but ne pourrait être attribué à une facette particulière comme par exemple un but imposant uniquement une sécurité maximale.
  ⟨facet⟩ : := a | e | i | o | *undefined*

- ⟨time⟩ permet de spécifier des contraintes temporelles sur un but. Il peut contenir les valeurs *min* et *max* qui correspondent respectivement à une date au plus tôt (respectivement tard) de début, et une évaluation de la durée nécessaire pour accomplir ce but. Les champs sont spécifiés sous forme d'expressions temporelles dans le langage L_T (cf. chapitre 4).
  ⟨time⟩ : := (min, max, duration)

- ⟨content⟩ correspond au contenu du but en spécifiant son *type* (champ relatif à l'application) comme le fait qu'il soit "dépendant du domaine" ou relatif à une phase spécifique de traitement comme nous le verrons au chapitre 10 par exemple et son expression dans le champ *value.*
  ⟨content⟩ : := (:type ⟨type⟩ :value ⟨value⟩)

- ⟨status⟩ spécifie l'ensemble des états qu'un but peut prendre pendant le fonctionnement de l'agent. Le cycle de vie d'un but est précisé dans la figure 8.4 ci-dessous.
  ⟨status⟩ : := new | chosen | activated | rejected | failed | done

  Nous considérons qu'un but est satisfait (*done* cf. figure 8.4) lorsque toutes les actions décrites dans l'un des plans du but, ont été accomplies avec succès.
- ⟨has-subgoal⟩ permet de préciser si le but se décompose en sous-buts en donnant la liste de ses sous-buts.
  ⟨has-subgoal⟩ : := '('⟨goal-id⟩*')'

- ⟨has-plan⟩ spécifie l'ensemble des plans permettant de réaliser ce but. Ce champ correspond à une liste d'identifiants de plans.
  ⟨has-plan⟩ : := '('⟨plan-id⟩*')'

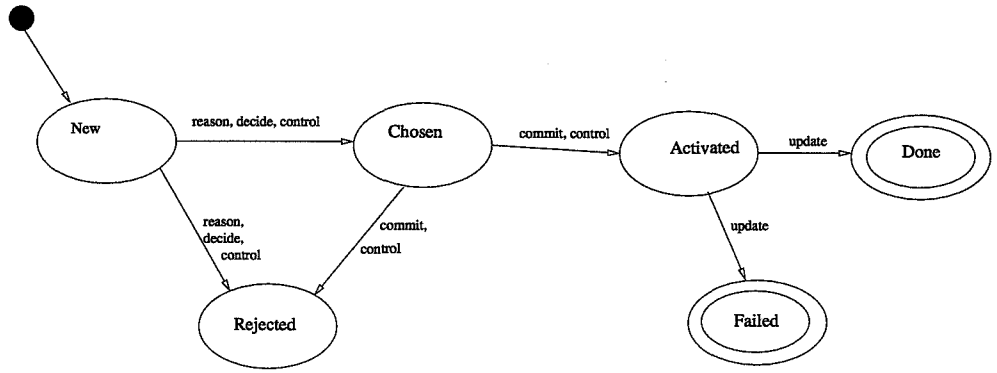


FIG. 8.4 – Le cycle de vie d'un but

Plan

Une structure de plan contient :
- l'identifiant du but qu'il permet de réaliser la liste des identifiants des actions qui doivent être réalisées,
- le langage utilisé est semblable à celui de description des arbres de protocoles (cf. chapitre 5.3) permettant de spécifier si besoin est la séquence, la concurrence ou l'alternative,
- le temps nécessaire à l'exécution complète du plan.
- l'état actuel du plan.

⟨plan⟩ : := :goal ⟨goal-id⟩
:content-list '(' ⟨action-id⟩* ')'
:time ⟨time⟩
:status ⟨status⟩

⟨status⟩ spécifie l'ensemble des états qu'un plan peut prendre pendant le fonctionnement de l'agent.

⟨status⟩ : := running | failed | done

L'échec (*failed*) ou la réussite (*done*) d'un plan génère une mise à jour de l'état du but associé (champ *status* décrit ci-dessus)

Action

Une action est décrite par un identifiant.

⟨action⟩ : := :id ⟨action-id⟩

Nous décrivons l'action de manière la plus simple possible car certaines caractéristiques de réalisation de ces actions ne peuvent être spécifiées de manière

abstraite. En effet, en fonction de capacités propres à chaque agent, la réalisation d'une action peut prendre des durées différentes, peut nécessiter des conditions particulières pour être exécutées, etc. C'est la raison pour laquelle l'agent possède en outre un ensemble de savoir-faire spécifiant justement l'ensemble des actions que l'agent est capable de réaliser. Nous utiliserons parfois dans la suite le terme de "tâche" à la place du terme *action* avec la même signification.

Savoir-faire

La description des savoir-faire de l'agent permet de spécifier la manière dont un agent est capable de réaliser ses actions. Un savoir-faire est constitué d'un identifiant, d'un champ temporel précisant la durée pour effectuer l'action ainsi que des conditions permettant de contraindre l'utilisation de cette action à certains contextes bien précis.

⟨savoir-faire⟩ : := :id ⟨id⟩
:time ⟨time⟩
:condition ⟨content⟩

La spécification des états mentaux tels que les croyances, les intentions, les engagements, etc. est très proche de leur définition formelle (cf. chapitre 4) et n'est pas redonnée ici.

8.2.2 Le choix du parallélisme par threads java

Afin d'avoir une implémentation en rapport avec le modèle proposé, lui-même adapté à l'approche choisie (cf. section 7.2), nous avons proposé de définir des modules spécifiques à chaque facettes, implémentés sous forme de "threads java" (processus autonomes). Ce choix d'implémentation permet d'obtenir le fonctionnement (d'agent ou de ses modules) le plus proche du "parallélisme". Ce choix se révèle éventuellement critiquable car difficile à mettre en œuvre d'une part et encore plus difficile à gérer ensuite d'autre part. Cependant comme nous l'avons évoqué au début de ce paragraphe, il est tout à fait dans l'"esprit" du modèle proposé. Pour pallier les difficultés relatives à ce "parallélisme", nous avons complété le modèle par un "contrôleur" implémenté par un thread "père" qui peut servir à synchroniser les différents threads et servir de garde-fou.
Sur un plan purement technique, la gestion de threads ou du moins du "temps-machine" alloué à l'exécution d'un thread est différente selon les systèmes d'exploitation et nécessite une bonne connaissance en système pour comprendre des comportements différents d'un même agent en fonction de la machine sur laquelle il est lancé.
D'autre part, des contraintes liées à l'implémentation, nous incitent à manipuler

un temps borné plutôt qu'infini mais ce ne sont que des artifices de programmation qui ne remettent pas en cause l'hypothèse plus réaliste et satisfaisante d'un concept de temps infini.

8.2.3 Extrait de code

Ces aspects peuvent éventuellement paraître complexes. Cependant, comme nous l'avons dit, l'approche AEIO nous permet de bien séparer les différents aspects. Ainsi, à chaque étape (gestion de l'interaction puis gestion des messages, des protocoles, etc.), la portion de code dédiée reste très simple. Pour illustrer nos propos et donner une idée de la simplicité du code lié à l'interaction au niveau de la classe Agent (écrit en JAVA), la boucle utilisée est donnée ci-dessous. Toutes les méthodes relatives à l'interaction sont regroupées dans une classe spécifique appelée "CommunicationHandler" (comm). Les aspects "traitement du message" sont appelés au sein de la méthode receiveMessage dans la classe "CommunicationHandler".

```
void treatInteraction()throwsGemasException {
    while (comm.hasMessagesToSend() || comm.hasNewMessages()){
        DisplayTools.step("Sendmessage");
        comm.sendMessage(this);
        DisplayTools.step("Receivemessage");
        comm.receiveMessage();
    }
    System.out.println("Finished sending and receiving messages");
}
```

Cet exemple donne une idée de l'architecture du code puisque l'on voit d'ores et déjà qu'en s'appuyant sur l'approche AEIO également pour l'implémentation, il est possible de rendre "transparent" (privé) l'appel de méthodes spécifiques dédiées au traitement d'une facette particulière pour le concepteur d'un agent temporel de type TAG.

8.3 TACL, TOSL : des fichiers de configurations et de structures

Au niveau de l'implémentation, un des objectifs est de fournir des méthodes et des techniques permettant de bâtir des SMA temporels. Nous avons choisi de spécifier les différents langages développés au moyen de fichiers au format *DTD*.

8.3.1 Définition de DTD

La *DTD* correspond à la définition type du document (**D**ocument **T**ype **D**efinition). Le standard *XML* dont nous parlerons à la section suivante (cf.section 8.3.3.0), fournit un moyen de vérifier la syntaxe d'un document grâce aux *DTD*. Ces fichiers permettent de décrire la structure des documents y faisant référence grâce à un langage adapté. Ces standards sont actuellement surtout utilisés dans le cadre des documents produits sur Internet. A titre indicatif, un document *XML* possédant une DTD et étant conforme à celle-ci est appelé *document valide.*
Cependant les agents devant manipuler des structures communes comme les protocoles ou les messages dans un langage particulier par exemple, nous allons voir que ces standards peuvent être applicables à l'implémentation d'agents logiciels.

8.3.2 Applications aux agents

Nous utilisons ces standards afin de s'assurer de la validité des structures reçus et émises par l'agent afin qu'ils soient également compréhensibles par les autres agents. Ce format permet de spécifier les grammaires correspondantes de manière "manipulable" par l'agent. Un ensemble de fichiers sont chargés au démarrage de l'agent permettant de s'assurer du respect des grammaires spécifiant les différentes entités (protocoles, messages, structures organisationnelles...) utilisées. Cela permet en outre de bénéficier pour les concepteurs de facilités pour la réalisation d'un SMA puisqu'ils disposent :

- d'outils spécifiques d'aide à la spécification ou de vérification de validité des "documents" produits,
- de la flexibilité offerte par ces normes pour définir leur propre format,
- d'un format très intuitif, lisible et reconnu,
- de possibilités de réutilisation d'anciennes spécifications avec un minimum de modifications,
- de mécanismes de validation et de vérification déjà implémentés sur la plateforme et réutilisables sans avoir à modifier le code en lui-même.

Dans le cadre de ce travail, pour les raisons expliquées plus haut, nous avons choisi de définir les DTD sous *forme externe* (en appelant un fichier contenant la grammaire)[19]. Les exemples de grammaires utilisées pour l'implémentation des applications sont données en annexe D.

8.3.3 Vers des bibliothèques de protocoles, d'organisations

Après avoir défini la grammaire pour représenter des protocoles temporels au format DTD (cf. ci-dessus), la norme *XML* nous permet de spécifier des protocoles spécifiques.

[19]Il est également possible d'inclure la grammaire au sein même du fichier *XML*, c'est la définition sous *forme interne.*

XML : Langage à balises extensibles

Le langage *XML* (*eXtensible Markup Language*) est un "méta-langage" permettant de définir d'autres langages. A la différence de *HTML*, il permet de décrire des données s'intéressant au contenu plutôt qu'à leur représentation. Depuis 98, *XML* est un langage reconnu puisque les spécifications *XML 1.0* sont recommandées par le *W3C*. Des outils existent pour extraire des données encodées au format *XML*, ces outils sont appelés "parseur" (*parser*). Certains *parseurs* sont dits "validants" c'est à dire qu'ils permettent en outre de vérifier la validité d'un document en s'appuyant sur la DTD. C'est le type de parseur[20] que nous utilisons dans les applications que nous avons décrites précédemment.

Bibliothèques chargeables par les agents

La définition de protocoles par le biais de fichiers *XML* nous permet ainsi de disposer d'un ensemble de protocoles que l'on peut charger à loisir à l'initialisation d'un agent. Ainsi de manière très simple, nous pouvons préciser les protocoles "connus" ou disponibles pour chacun des agents dans le système. L'objectif est de disposer à terme de bibliothèques de protocoles adaptables pour chacune des application développées. En plus des avantages cités dans la section DTD ci-dessus, le langage *XML* présente de nombreux atouts pour cela :

- autodescriptif et extensible (cf. ci-dessus),
- universalité et portabilité (texte uniquement sans caractères particuliers),
- intégrabilité (utilisable dans n'importe quel agent disposant d'un parser).

Nous avons réutilisé ce principe pour spécifier différents types de structures organisationnelles temporelles pouvant être mises en place au sein d'un système. Des exemples d'utilisation du format *XML* sont présentés en annexe D.

8.4 Discussion

La difficulté en systèmes multi-agents réside dans la séparation de ce qui dépend de l'*Agent* et de l'*Application*. C'est d'ailleurs la "méthodologie" proposée par les auteurs de la plate-forme *Zeus* [NNLC99], pour développer un SMA. Nous pouvons distinguer cet objectif dans l'implémentation. Certaines parties sont presque directement réutilisables pour élaborer un SMA temporel (les structures TACL, TOSL, TAG et l'outil GRIS) alors que d'autres nécessitent d'être adaptées à l'application (savoir-faire, protocoles disponibles, structures organisationnelles). Les choix qui ont été faits, permettent tout de même de simplifier le travail d'implémentation en limitant la grande partie de la conception à la définition de fichiers de configuration dans un format standard. Nous pensons qu'il manque pour l'instant au domaine SMA, les structures logicielles permettant de développer des

[20]Le parseur que nous utilisons est : OpenXML Public Licence 1.0 et est gratuit.

SMA en se consacrant uniquement à la partie analyse de la conception et non sur des parties récurrentes de l'implémentation comme il est encore nécessaire. Nous pouvons voir les propositions et les outils développés dans ce chapitre comme un élément de réponse à cet état de fait concernant la conception de SMA.

9

Gestion d'alliance d'ateliers d'impression

Sommaire

9.1 Introduction

Notre objectif avec cette application, est de valider l'approche présentée ainsi que les modèles dans le cadre d'une application fortement dynamique. Ceci sera illustré d'une part, par la mise en évidence des caractéristiques temporelles de l'application structurées selon les modèles proposés, à savoir A, E, I et O ; et d'autre part, par la réalisation d'un SMA temporel montrant deux manières de prendre en compte et de gérer le temps (i-driven et o-driven).

L'application que nous avons retenue, est celle d'alliances d'ateliers d'impression.

9.2 Description du projet E-Alliance

Nous nous sommes intéressés à la mise en place d'alliances inter-organisationnelles. Plus précisément, nous avons pris comme application un projet concernant la gestion d'ateliers d'impression.

Le cas d'application relève du domaine de l'impression numérique où des groupements d'ateliers d'impression peuvent coopérer via l'utilisation de l'infrastructure développée dans ce but. Notre travail se situe en marge du projet principal intitulé *E-Alliance.*

Ce dernier inclut deux laboratoires de recherche (LLP/CESALP et SMA/ SIMMO) et un partenaire industriel XRCE (Xerox Research Centre Europe, groupe CT). Le projet est soutenu par la région Rhône-Alpes et les résultats obtenus pourront aisément être appliqués à d'autres domaines de l'industrie de services, aux applications de commerce électronique et de e-business.
L'objectif d'*E-Alliance* concerne le développement d'une infrastructure logicielle pour la gestion d'alliances d'une part, et pour l'aide à la négociation et à l'exécution de contrats entre les différents partenaires du groupement d'autre part. Pour ce faire les techniques des SMA sont bien adaptées pour élaborer une infrastructure non intrusive, c'est à dire respectant l'autonomie des différents partenaires tout en les obligeant à respecter un ensemble de règles de fonctionnement définies, prenant en compte les contraintes d'hétérogénéité des partenaires du groupement. Pour notre part, nous nous sommes intéressés particulièrement à la prise en compte et la gestion des multiples aspects temporels qui sont naturellement présents dans de telles applications.

Nous allons, dans un premier temps, revenir sur le cadre d'application retenu en en précisant l'implantation SMA formulée et en en faisant ressortir les caractéristiques temporelles.
Dans un second temps, nous présenterons le déroulement d'un scénario au sein de ce système.
Ayant décrit les aspects génériques des modèles développés dans le chapitre précédent, nous ne soulignons ici que les aspects principaux mis en œuvre.

9.3 Définition d'une infrastructure logicielle multi-agents

Nous allons voir, à présent, que cette application est un système dynamique qui peut se modéliser par un SMA temporel. Pour cela, nous décrivons la manière dont le SMA constitue l'infrastructure logicielle support de cette application en

mettant plus particulièrement en évidence la gestion du temps et le raisonnement temporel mis en œuvre.

Si l'on reprend l'approche *AEIO*, les différents ateliers d'impression fonctionnent de manière autonome (dimension A), communiquent entre eux pour se déléguer des tâches (dimension I) et doivent respecter des règles lorsqu'ils agissent dans le cadre d'une alliance (dimension O). En précisant l'analyse multi-agent, nous pouvons mettre en exergue les aspects temporels au sein de chacune des dimensions :

- dimension A : L'organisme, les responsables des ateliers et chacun des ateliers d'impression sont assistés par autant d'agents logiciels qui coopèrent entre eux. Les compétences des agents *ateliers d'impression* correspondent à la capacité d'impression (vitesse, coût, nature du travail : couleur ou noir et blanc, qualité, etc.). En cas de proposition de nouvelle tâche par le responsable, l'atelier d'impression doit, pour accepter, être capable de la réaliser dans les temps demandés sans, pour autant, remettre en cause les tâches pour lesquelles il s'est déjà engagé.
- dimension I : Chacun des agents doit être capable de prendre en compte ses éventuelles dépendances avec d'autres agents pour réaliser une tâche. Par exemple, s'il partage une machine avec un autre agent, il doit s'assurer qu'elle sera disponible pendant le temps nécessaire à la réalisation de la totalité ou d'une partie de la tâche. Ainsi, en plus des contraintes locales, l'agent doit être à même de prendre en compte les *contraintes provenant des autres partenaires.* Pour cela, ils doivent communiquer sur le plan temporel et utiliser des protocoles particuliers tels que ceux définis dans le chapitre 5.
- dimension O : Les contrats à réaliser sont des structures organisationnelles temporelles (cf. exemples du chapitre 6). Nous constatons également, au niveau des aspects organisationnels, que l'agent doit connaître et prendre en compte ses dépendances temporelles avec d'autres agents.
- dimension E : Pour compléter il convient de s'intéresser aux contraintes fournies par l'environnement dans lequel les agents sont immergés : en l'occurrence, dans le cadre de tâches d'impression, les agents doivent répondre avec les délais imposés par les contraintes issues du marché. Nous n'avons pas pris en compte ces aspects dans le cadre de notre étude.

Dans le cadre de ce mémoire, nous distinguerons en particulier deux parties dans lesquelles l'aspect temporel nous concerne précisément :

1. la partie relative aux aspects temporels dans la négociation d'un contrat qui induit un comportement i-driven.

2. l'exécution du contrat en lui-même, avec le respect des règles de l'alliance, induisant un comportement o-driven.

9.3.1 Aspects temporels dans la négociation

Comme nous l'avons dit, la composante temporelle intervient à différents niveaux au sein d'un tel système. L'organisme propose à l'agent du responsable, un contrat qui comporte une échéance pour le réaliser. Cette échéance doit être connue et comprise des deux agents. La réponse (positive ou négative) attendue pour cette proposition comporte elle-même une limite temporelle. En effet, si le client ne reçoit pas la réponse souhaitée, il peut essayer de négocier ou choisir de s'adresser à un autre groupement. Nous voyons donc qu'au niveau des interactions, le langage (expression de l'échéance) et les protocoles (temps limite de réponse) doivent exprimer des caractéristiques temporelles. Dans cette première partie de l'application, nous voyons ainsi que le fonctionnement du SMA est orienté par l'interaction et notamment par les aspects temporels pris en compte dans cette dimension.

9.3.2 Aspects temporels dans l'exécution du contrat

Au moment de l'exécution du contrat, l'agent doit avoir planifié l'ordonnancement de ses tâches et doit gérer son plan de charges. A ce niveau, pour prendre ses décisions, l'agent s'appuie principalement sur ses connaissances sur la structure organisationnelle pour déterminer son comportement (rôle qu'il joue, missions qu'il exécute,...). Dans une alliance, le respect des règles définies dans le contrat est primordial, c'est pourquoi dans cette partie, nous considérons que le SMA est plutôt géré par l'organisation ou du moins les aspects temporels relatifs à l'organisation.

A partir de ces observations, nous avons plus précisément retenu un scénario particulier.

9.4 Description du scénario

Imaginons un organisme (type organisation gouvernementale comme l'ONU) ayant de grosses demandes en impression de documents. Cet organisme s'adresse à un groupement d'ateliers d'impression pour sous-traiter les différents travaux. Ceux-ci vont coopérer entre eux pour pouvoir répondre à cette demande en fonction de leurs plans de charge respectifs. A la suite de différentes négociations conduites dans un temps limité, entre le client et le responsable du groupement d'une part, entre le responsable du groupement et les ateliers d'impression d'autre part, un contrat est proposé. Ce contrat, pour des raisons évidentes, comporte, entre autres, une date limite et diverses contraintes temporelles d'exécution se répercutant sur les ateliers. Les négociations elles-mêmes ont dû intégrer en cours de route les évolutions du marché, les évolutions des plans de charge de chacun des

ateliers. En cours d'exécution, chaque atelier d'impression doit gérer ses propres contraintes temporelles pour respecter la partie du contrat sur laquelle il s'est engagé.

Les fichiers de configuration ont permis d'adapter simplement la structure d'agent temporel à cette application. Ces fichiers de configuration concernent l'ensemble des compétences propres à chaque agent du système et relatives au domaine de l'application, l'ensemble des protocoles disponibles pour chaque agent désirant communiquer avec les autres agents du système ainsi que les connaissances structurelles regroupées dans la définition d'une structure organisationnelle temporelle définissant l'alliance. Quelques exemples ont déjà été présentés pour illustrer la définition dans la deuxième partie de ce mémoire. Des informations plus détaillées sur ces fichiers peuvent être consultées en annexe C.

9.5 Exemples illustrant les priorités dans la gestion du temps

Au stade actuel du projet, les différents protocoles et les règles de l'alliance ne sont pas tous spécifiés, nous les avons donc définis dans le cadre du scénario de manière à illustrer les aspects précédemment décrits. Les exemples qui suivent, complètent ceux déjà présentés tout au long du mémoire.

La structure d'une alliance est composée de trois rôles possibles qui sont Manager, Imprimeur et Relieur complétée par un Client qui propose des contrats.

Exemple de contrat proposé

Le client souhaite voir réalisée d'ici la fin de l'année, la tâche d'impression suivante : imprimer puis relier 1 500 000 livrets d'information sur le passage à l'Euro avec 50 000 supplémentaires en langues étrangères dans chacun des départements limitrophes de pays dont la langue n'est pas le français. Ce type de contrat nécessitera l'appel de nombreux ateliers d'impression dont éventuellement certains situés en Italie, en Espagne ou en Allemagne. La possibilité de déléguer la tâche à une alliance permettra de s'affranchir d'un grand nombre de difficultés.

Pour simplifier la formalisation du contrat, nous ne précisons pas les contraintes de langues et de qualité, et considérons qu'il y a 2 000 000 de livrets à réaliser. Le livret est disponible sous forme numérique dans le format "EURO.liv".

Le client proposera donc le contrat suivant :

contract(ex : 2 000 000, fi : euro.liv)

Il s'accorde une semaine pour trouver un groupement capable de réaliser la tâche. Les contraintes temporelles sont spécifiées grâce au langage *TACL*. Le client

enverra donc vers les managers d'ateliers d'impression dont il a connaissance, le message suivant :

(message :com (:sender *client* :receiver *manager*
:priority 5
:date 2000 : 08 : 16
:sndDate 2000 : 08 : 16T18 : 17 : 01)
:mas (:act *propose*
:time ($SRin[t_{now}, 2001 : 08 : 25T00 : 00 : 00]$
$\wedge done(A)b[t_{now}, 2002 : 01 : 01]$)
:nature *Action*
:protocol $P - RA$
:conversation "$18 : 17 : 06|client - manager$"
:content $realizecontract(ex : 2000000, fi : euro.liv)$)

Sans décrire à nouveau tout le contenu d'un message *TACL* (voir pour cela le chapitre 5.2), nous pouvons remarquer que le message est envoyé le 16 août 2001, que la réponse est attendue entre l'instant présent et le 25 août 2001 minuit et enfin que l'action demandée doit être réalisée d'ici le 1er janvier 2002.

La phase de négociation

A la réception du message (18/07/2001 en fonctionnement classique), la facette interaction met en place une garde temporelle permettant d'imposer l'envoi d'une réponse dans les délais (25/08/2001 maximum), puis le manager analyse le contenu (content) pour évaluer les besoins. Ceci se fait à partir de ses compétences propres : il en déduit que la réalisation du livret se décompose en une phase d'impression couleur pour la couverture, une phase d'impression noir et blanc pour le contenu et une phase de reliure. Il contactera donc les agents concernés pour leur proposer respectivement la réalisation de la partie du contrat qui les concerne. Nous passons sur l'explicitation de ces messages qui relèvent des mêmes techniques que celles évoquées jusqu'à présent.

Suivant les stratégies mises en œuvre, le manager enverra sa réponse au client aussitôt qu'il a une réponse ou bien il attendra d'avoir toutes les possibilités pour proposer un client la meilleure (la moins chère, la plus rapide, etc.). La garde temporelle est relâchée pour cette conversation.

Par exemple, si le manager parvient à obtenir une réalisation pour le 1er décembre 2001, il enverra le message suivant :

(message :com (:sender *manager* :receiver *client*
:priority 5
:date 2000 : 08 : 19
:sndDate 2000 : 08 : 19T18 : 17 : 01)

:mas (:act *accept*
:time $(done(A)b[t_{now}, 2001:12:01])$
:nature *Action*
:protocol $P-RA$
:conversation "$18:17:06|client-manager$"
:content $realizecontract(ex:2000000, fi:euro.liv)$)

Au contraire, si l'échéance arrive avant d'avoir pu trouver des partenaires pour la réalisation complète du contrat, la facette interaction envoie une réponse avec un refus.

(message :com (:sender *manager* :receiver *client*
:priority 5
:date 2000 : 08 : 24
:sndDate $2000:08:24T18:17:01$)
:mas (:act $refuse$
:time $(done(A)b[t_{now}, 2001:01:01])$
:nature *Action*
:protocol $P-RA$
:conversation "$18:17:06|client-manager$"
:content $realizecontract(ex:2000000, fi:euro.liv)$)

D'autres phases de négociation peuvent apparaître entre les différents agents. Par exemple, les imprimeurs peuvent éventuellement négocier entre eux pour se déléguer des tâches ou partie de tâches : l'imprimeur acceptant (ou ayant accepté) une tâche peut ensuite essayer de négocier avec un autre imprimeur sa réalisation afin de soulager son plan de charge par exemple.

A la réception de la confirmation du contrat par le manager, la phase de réalisation (exécution du contrat) commence pour chacun des agents concernés.

Exécution du contrat

Chacun des agents réalise la tâche qui lui incombe en respectant les règles de l'alliance fixée dans la structure organisationnelle. Au un moment de la réalisation, le manager, par exemple, possède une vision particulière (cf. figure 9.1) de la structure organisationnelle (semblable à celle définie au chapitre 6.4) mise ne place pour la réalisation de ce contrat :

Comme on le voit sur le schéma (cf. figure 9.1), il connaît les agents jouant les rôles d'imprimeurs (*bwPrinter* et *cPrinter*), relieur (*Binder*) et évidemment manager, différents liens ainsi que les contraintes temporelles associées (les contraintes

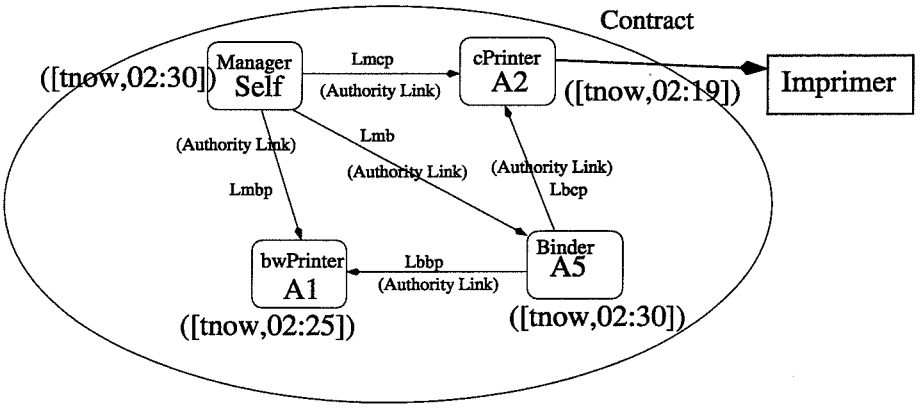


FIG. 9.1 – Une instanciation de la structure organisationnelle *contract* vue par le manager

temporelles sur les liens ne sont pas mentionnées sur la figure). En l'occurrence, l'agent jouant le rôle d'imprimeur couleur est actuellement en train d'effectuer la mission "imprimer". A ce même instant, la vision de la structure organisationnelle des autres agents est différente (et également partielle) puisque son instanciation dépend des informations communiquées à ces mêmes agents.

9.6 Discussion

Dans ce chapitre, nous avons montré une possibilité d'application de notre travail pour la définition d'alliances inter-organisationnelles. Le scénario décrit nous a permis de montrer l'aspect prédominant d'une facette par rapport à une autre et que cette prédominance peut varier pendant le fonctionnement. En l'occurrence, la première partie s'appuie surtout sur l'interaction liée à la négociation d'un contrat (comportement i-driven) alors que la seconde partie concernant l'exécution du contrat privilégie la facette organisation et le respect des règles de l'alliance (comportement o-driven).
L'implémentation a été faite de manière à instancier une structure d'agent temporel social TAG autorisant l'utilisation à la fois des modèles d'interaction et d'organisation temporels. Nous allons voir dans le chapitre suivant un autre type d'application mettant en œuvre un autre objectif de réalisation d'un SMA temporel.

10

Simulation d'entreprise fonctionnant par projet

Sommaire

10.1 Introduction

Dans le cadre de notre travail, nous avons également défini un partenariat avec le laboratoire LGPP (Laboratoire de Gestion et Procédés de Production) de l'**E**cole **P**olytechnique **F**édérale de **L**ausanne (EPFL)[21] nous offrant la possibilité de valider nos travaux en les appliquant à un cas concret. Nous avons donc instancié le modèle *TAG* précédemment décrit (cf. chapitre 7) à partir d'une étude d'entreprise fonctionnant par projet.

[21] Département de génie mécanique

Motivations

Ce mode de fonctionnement nécessite une forte flexibilité ainsi qu'une maîtrise importante de la dimension temporelle, c'est pourquoi notre travail est bien adapté à la simulation de telles entreprises. Nous pouvons distinguer deux objectifs dirigeant ces travaux :

1. industriel : du point de vue de l'entreprise, l'objectif de ce travail est l'explicitation du déroulement complet d'un projet notamment sur le plan des contraintes temporelles (jusqu'à présent plus ou moins implicites). Le but étant à terme, d'avoir une représentation précise des "compétences" et "savoir-faire" de chacun afin d'éviter une perte lors de la cessation d'activité d'une personne intervenant dans le déroulement d'un projet.
2. technologique : au sein du laboratoire, l'objectif étant de mettre en pratique et valider les outils définis pour prendre en compte le temps au sein d'un SMA.

10.2 Description de l'entreprise

Plus précisément, nous nous sommes intéressés à une entreprise orientée par projet, spécialisée dans la conception et la réalisation de wagons de chemin de fer. Les productions varient de wagons dédiés au transport de fret jusqu'aux voitures de transport de passagers ou même aux rames de tramway. Il est évident que la conception et la réalisation de chaque véhicule est spécifique aux besoins du client final et à ses objectifs propres (capacité, qualité, confort, etc.). Afin de s'adapter rapidement aux besoins des clients, l'entreprise fonctionne entièrement par projet. L'étude de cas a été réalisée sur une branche située en Suisse mais cette entreprise possède des secteurs ainsi que de nombreux sous-traitants localisés dans d'autres pays.

10.3 Fonctionnement d'une gestion par projet

On peut définir qu'une entreprise s'appuie sur une gestion par projet lorsque les commandes de ses clients nécessitent une adaptation de ses productions et de ses services par rapport à une commande précédente. La caractéristique principale de ces entreprises est donc d'être à même de répondre rapidement aux besoins des clients. Pour décrire l'application en question, nous nous appuierons sur un *scénario* correspondant à un projet spécifique qui a été l'objet d'une étude menée directement au sein de l'entreprise et nous illustrerons donc le déroulement du projet par différentes phases susceptibles d'apparaître pendant son exécution. L'analyse de cette étude nous a permis de décrire un scénario complet correspondant à la mise en place et au suivi d'un projet dans lequel les caractéristiques temporelles sont explicitées.

10.4 Caractéristiques temporelles et multi-agents

Nous allons, dans un premier temps décrire les aspects temporels globaux d'un projet (différentes phases) puis nous essaierons dans un deuxième temps, de les caractériser plus précisément dans un SMA selon A, E, I et O.

10.4.1 Description d'un projet

Chaque projet se décompose en différentes phases ; elles-mêmes décomposées en sous-phases. Chacune des phases peut se dérouler de manière différente en fonction de contraintes diverses inhérentes au contenu du projet (besoins du client, délais, coûts, etc). Ces projets bien que différents pour chaque cas s'appuient cependant sur un ensemble de "règles" prédéfinies qui définissent le fonctionnement d'une entreprise utilisant ce type de gestion. L'étude menée sur l'entreprise a permis de mettre en exergue les différentes étapes de la réalisation d'un projet. Ainsi, nous avons décrit un scénario comportant 3 phases dont la dernière est décomposée en 4 sous-phases :

1. Phase : *Initialisation*
2. Phase : *Evaluation de projet*
3. Phase : *Suivi de projet* :
   (a) Sous-phase : Enclenchement de projet
   (b) Sous-phase : Définitions des contrôles
   (c) Sous-phase : Réalisation
   (d) Sous-phase : Contrôle de qualité

Les diagrammes de séquences bâtis selon un point de vue temporel sont présentés en annexe C. Cette modélisation permet de représenter les aspects dynamiques qui nous intéressent particulièrement.

Simulation par scénario

Le scénario correspondant à la réalisation de ce projet est issu d'un contrat lié à la réalisation de rames de tramway mais dans le cadre de ce travail, nous ne rentrerons pas dans les détails liés à la conception d'une rame. Nous ne mettrons en exergue que les aspects relatifs à la gestion du projet en lui-même. Nous précisons cependant qu'inévitablement, certaines phases sont étroitement liées au contenu du contrat et peuvent ne pas être nécessaires dans un autre scénario ou au contraire de nouvelles peuvent éventuellement être ajoutées.

10.4.2 Définition du SMA

Nous pouvons ainsi voir une "*entreprise orientée projet*" comme un réseau d'entités interagissantes qui peuvent aussi bien être des individus, des groupes de personnes, des machines de toutes sortes, que des systèmes informatiques ou même d'autres entreprises. Toutes ces entités doivent donc coopérer d'une manière particulière et spécifique à chaque projet. Si l'on se penche sur les objectifs de ce type d'entreprises, nous retrouvons des caractéristiques proches des SMA :

- *Flexibilité* ou capacité de (ré)organisation.
- *Adaptativité* en fonction de la demande du marché.
- *Réactivité* (ou *Proactivité*) pour répondre rapidement à des changements de l'environnement.
- *Intégration* d'entités hétérogènes.
- *Interactions* entre ces entités pour échanger des informations.

Nous avons donc implémenté un SMA permettant de mettre en œuvre chacun des aspects qui nous intéressaient. Nous avons créé des agents selon le modèle *TAG* présenté dans la partie précédente. Chacun des agents est particularisé par la configuration de ses connaissances propres et ses compétences (savoir-faire dans le modèle TAG). Pour s'adapter aux besoins de l'application, de nouveaux champs peuvent apparaître afin d'être plus proches des termes utilisés en réalité : représenter les "savoir-faire" plus fidèlement par exemple (cf. annexe C).

Onze types d'agents différents ont été définis pour les besoins de l'application. **Les parties relatives aux connaissances internes propres à l'application ont été simulées** (compétences techniques comme l'émission d'un rapport de non conformité par exemple). L'objectif de notre travail était d'appliquer les modèles temporels (agent, interaction et organisation) pour expliciter et simuler les aspects temporels de la gestion d'un projet ; ainsi chacun des agents communique en langage *TACL*, s'appuie sur une structure organisationnelle temporelle et possède une structure en facettes de type *TAG*. Trois types d'agents sont "extérieurs" à l'entreprise, ce sont les types "client, fournisseur et contremaître-contrôleur". Les autres agents sont de type "décideur, responsable projet, responsable technique, responsable qualité, responsable achat, responsable méthode, responsable planning et contremaîtres-exécuteurs". En fait, ces types d'agents correspondent aux onze rôles définis dans l'application. Les agents possèdent un certain nombre de compétences propres qui leur permettront de jouer les rôles spécifiés dans la structure organisationnelle.

Aspects temporels

Les descriptions précédentes décrivent l'application et plus précisément la gestion d'un projet de manière globale mais l'objet de notre travail est de pouvoir

représenter de manière explicite tous les aspects temporels susceptibles d'intervenir dans le déroulement du projet. Comme nous l'avons vu, nous avons décrit le projet sous forme de diagrammes de séquence nous permettant de voir clairement l'enchaînement des actions et des messages tout au long de ce déroulement. Il nous reste à spécifier les contraintes temporelles associées à chacune des actions. En effet, une gestion de projet conventionnelle basée sur un processus de planification déterminé à l'avance est inenvisageable dans ce contexte car trop rigide et des modifications seraient trop lourdes à mettre en place. Pour un fonctionnement du type "gestion par projet", un modèle de SMA permettant de spécifier explicitement des contraintes temporelles dans chacun des "domaines" où elles peuvent intervenir, se révèle donc bien adapté.

10.5 Description de déroulement des phases

Les phases décrites en annexe C sont relativement complexes alors nous allons décrire ce déroulement de manière d'abord globale avant de rentrer ensuite plus précisément dans les détails avec la présentation des aspects temporels liées à ces phases. L'annexe C présente des schémas illustrant la mise en œuvre du SMA dans la cadre du scénario décrit ci-après. Nous nous concentrerons à ce niveau sur un fonctionnement normal, les comportements "pathologiques" (conflits divers) ont été évoqués précédemment dans la description du contrôleur (cf. chapitre 7).

10.5.1 Déroulement de la phase "Initialisation"

Le démarrage d'un projet est initialisé par la proposition d'un nouveau contrat par un client à un décideur. Ce dernier crée alors une équipe "superviseurs" constituée d'un responsable projet et d'un responsable technique. Un projet lié au contrat est défini en même temps. Ces actions correspondent à la phase appelée "Initialisation".

Explicitation des aspects temporels
L'agent Décideur reçoit une offre de contrat d'un agent Client à l'instant t=01 (unité de temps) dont le contenu est "realize contract". L'offre précise un délai pour réaliser la tâche t=1000 et un autre pour envoyer l'accord ou le refus de réalisation de ce contrat t=30. Le contenu correspond à une de ses compétences. L'exploitation de cette compétence (connaissance de la facette Agent) conduit à la création d'une équipe Superviseurs ("groupe" dans le modèle MOISE) constituée d'un responsable projet et d'un responsable technique, et à la création d'un projet lié au contrat, un nouveau but est généré et adopté (cf. chapitre 7.5) concernant l'évaluation du projet. C'est le début de la phase "Evaluation de projet".

10.5.2 Déroulement de la phase "Evaluation de projet"

Le décideur envoie donc au responsable projet, le projet pour évaluation. Une "équipe projet" est créée avec un responsable achat, un responsable qualité et un responsable technique pour évaluer la faisabilité respectivement en termes de coût, de qualité et technique. En fonction de leur réponse, le projet est soit accepté par le décideur qui en informe le client, soit refusé et les équipes correspondantes sont dissoutes.

Explicitation des aspects temporels
Imaginons que nous nous situons à l'instant t=02, à présent. L'agent Décideur ne possédant pas les compétences requises, confie alors l'évaluation de la faisabilité du projet au responsable projet avec un délai de réponse de t=15 et attend la réponse. Le nom de l'agent à contacter ainsi que ses compétences ont été trouvés à partir des connaissances organisationnelles (cf. structure organisationnelle temporelle 6.4). Le responsable projet utilise sa compétence pour créer l'équipe nécessaire à l'évaluation et délègue à chacun le soin d'évaluer la faisabilité du projet. Pour pouvoir répondre dans les temps demandés (t=15), il impose une réponse à ses interlocuteurs (responsable qualité, responsable méthode et responsable achat) avant t=12. Une fois qu'il a les différentes réponses (3), il envoie la réponse à l'agent décideur. Nous nous trouvons par exemple à l'instant t=10. Le décideur envoie sa réponse à l'agent client et confirme l'équipe projet.

10.5.3 Déroulement de la phase "Suivi de projet"

La troisième phase se produit en cas d'acceptation du projet dans la phase précédente. L'enclenchement du projet se fait en créant les autres équipes nécessaires ainsi qu'en réservant les ressources dont on va avoir besoin. Ensuite les pièces nécessaires sont commandées par l'intermédiaire du responsable achat aux différents fournisseurs et vérifiées par le responsable qualité avant la réalisation en elle-même. A ce niveau, le respect du planning est primordial avant un dernier contrôle qualité de chacune des parties avant la livraison au client.

Explicitation des aspects temporels
A présent, nous décrivons les aspects temporels de la phase "suivi de projet" qui est relativement complexe et qui a été décomposée en quatre sous-phases.

Sous-phase Enclenchement de projet

La phase "enclenchement de projet" commence par la négociation avec les autres responsables projet pour l'allocation des ressources. Chacun des interlocuteurs de cette négociation l'informe des ressources disponibles, des délais associés

et de la main d'œuvre nécessaire. Nous n'avons pas détaillé cette partie propre à chaque projet mais elle aboutit à la mise à jour du plan de charge de chaque agent. Chaque agent sait donc qu'il fera telle tâche utilisant telle machine à tel moment. Cette négociation utilise également les informations liées à la structure organisationnelle comme les contraintes et propriétés temporelles (cf. tc et tp chapitre 6.4) des missions concernées (comme par exemple envoyer dessins et plans de fabrication). Le responsable qualité crée ensuite l'équipe de "contremaîtres contrôleurs". L'étude de l'entreprise a montré que le responsable planning se charge de contrôler le déroulement des tâches au moment prévu, cet aspect peut éventuellement être shunté avec la mise en place du SMA temporel car les différentes prises en compte du temps (interaction, organisation et raisonnement interne) sont censées s'assurer du respect du planning prévu.

Sous-phase Définition des contrôles

Les résultats de l'analyse des dessins et des plans de fabrication donnent lieu à l'émission d'un rapport de non-conformité ("RNC") qui est envoyé au responsable technique et au responsable projet par le responsable qualité. Il envoie ensuite le plan de contrôle du projet. Ce plan de contrôle est également transmis au responsable méthode. Le pourcentage de tâches à sous-traiter est communiqué à chacun des agents par le responsable projet. La liste des achats qui en découlent est ensuite finalisée et évaluée pour être envoyée aux différents fournisseurs. A la réception, une vérification et un contrôle de certaines pièces sont réalisés donnant à nouveau lieu à l'émission d'un rapport de non-conformité afin d'autoriser le passage en phase de réalisation.

Sous-phase Réalisation

Cette sous-phase concerne la réalisation avec la création pour commencer d'une équipe de "contremaîtres exécuteurs" par le responsable méthode. Ce dernier leur demande alors la réalisation d'un ordre de fabrication concernant le projet ("OF projet"). La suite de la réalisation concerne les échanges entre les "contremaîtres-exécuteurs" et le "responsable planning" pour le contrôle des délais relatifs à la fabrication.

Sous-phase Contrôle de qualité

Les "contremaîtres exécuteurs" demandent ensuite au "responsable qualité" d'évaluer la conformité des travaux réalisés dans le cadre du projet qui renvoie en retour un rapport de non-conformité ("RNC"). Une partie de cette tâche de contrôle de conformité est déléguée à des contrôleurs spécialisés : les "contrôleurs qualité". En cas de non-conformité, les travaux incriminés doivent être refaits jusqu'à une conformité totale permettant alors au responsable qualité d'autoriser

la libération du produit fini pour livraison de la part des contremaîtres exécuteurs.

10.6 Discussion et perspectives

Cette application met en lumière différents points :

- l'omniprésence des aspects temporels dans une application concrète issue directement du milieu industriel et de la nécessité de les prendre en compte.
- l'intérêt indéniable des industriels pour la mise en place d'un outil logiciel permettant l'explicitation de ces aspects temporels
- l'adéquation de notre modèle de SMA temporel pour simuler de manière précise dans un premier temps la gestion d'un projet et ensuite un fonctionnement de gestion par projet.

Nous voyons que dans ce type d'application, tout le déroulement d'un projet est basé sur la mise en place et le respect d'une structure organisationnelle temporelle. Nous pouvons considérer que le SMA temporel mis en place est dirigé par l'organisation ("o-driven"). L'implémentation a été faite de manière similaire à l'application décrite dans le chapitre précédent. Les fichiers de configuration (cf. "XML" dans le chapitre 8) ont permis de l'adapter simplement à cette nouvelle application.

11

Conclusions et perspectives

Sommaire

Dans ce mémoire, nous nous sommes intéressés à la prise en compte de la dimension temporelle dans les SMA. Nous nous sommes pour cela, appuyés sur l'approche *AEIO*. Cette étude nous a amenés à proposer différents modèles permettant de prendre en compte le temps de manière explicite dans l'interaction et dans l'organisation. Nous avons également défini un modèle d'agent temporel offrant la possibilité d'intégrer les autres propositions.

11.1 Principales contributions de la thèse

Nous avons abordé le problème du temps sous différents aspects ce qui nous a permis d'avoir une vue relativement complète du temps dans un SMA et de proposer une prise en compte explicite du temps à la fois au sein d'un SMA et de l'agent. Nos travaux correspondent à une vision du temps à travers l'approche *AEIO* ce qui nous a amené à définir principalement :

- un **modèle d'interaction temporel** avec la spécification d'un langage d'interaction temporel (TACL). Ce langage est basé sur la théorie des actes de langage et est défini par une sémantique basée sur l'interprétation logique d'actes de communication temporels. La syntaxe permet de séparer différents niveaux de communication.
- un **modèle d'organisation temporel** qui permet de prendre en compte explicitement les aspects temporels relatifs à l'organisation. Il a été appliqué

au modèle organisationnel *MOISE* et a donné lieu à un langage de structures organisationnelles temporelles (TOSL).

Ces aspects ont été complétés par la proposition et la définition d'un **modèle d'agent temporel** (TAG) permettant éventuellement d'intégrer les modèles temporels précédents et de définir ainsi différents types d'agents temporels selon leur prise en compte ou non des aspects temporels dans l'organisation et/ou l'interaction.
De plus, un langage d'expressions temporelles basé sur la logique d'Allen a été spécifié et est utilisable dans chacun des modèles proposés, offrant ainsi une représentation uniforme du temps et un modèle unifié pour une représentation du temps explicite dans tous les niveaux de l'agent.

De manière plus théorique, nous pouvons maintenant donner une réponse à l'une des questions que nous nous sommes posée :
"Qu'est-ce qu'un SMA temporel ?"
Un SMA temporel correspond à la prise en compte du temps dans l'une ou l'autre, voire la totalité des dimensions du SMA :

$$\mathsf{SMA(t)} ::= < A(t), I(t), O(t), E(t) >$$

En effet, notre étude a montré que le temps intervient dans les différentes dimensions d'un SMA et dans les interelations (temporelles) entre ces différentes dimensions. Un "opérateur de combinaison" permet de gérer ces interdépendances entre les dimensions et de les coordonner temporellement. Nous pensons que chacune des combinaisons que l'on peut trouver, représente un fonctionnement de SMA temporel particulier.

Du point de vue de la mise en œuvre, nous nous sommes penchés sur la dimension Agent pour mettre en exergue de manière similaire différents types d'agents temporels en fonction de la prise en compte du temps et différents comportements relatifs à la mise en place d'une coordination temporelle ou "stratégies de coordination temporelle". L'"opérateur de combinaison" évoqué ci-dessus pour la définition d'un type de SMA est lié à la fois aux types d'agents temporels mis en œuvre et à leur comportement.

Ces aspects sont illustrés au moyen de deux applications qui sont représentatives de différents modèles de SMA temporel. La définition de ces modèles a conduit au développement d'une architecture d'agent temporelle offrant la possibilité de mettre en œuvre n'importe quel type d'agent temporel et donc de SMA

temporel.

Ces travaux suivent une approche transversale par rapport aux axes d'études habituels des SMA. Cette approche s'est révélée très riche et il était illusoire de vouloir aborder tous les aspects temporels d'un SMA en trois ans. Les choix qui ont été fait dépendent à la fois des recoupements avec les travaux des autres personnes du laboratoire, des pôles d'études de la communauté et des intérêts du moment.
Concernant la réalisation, le langage *TACL*, les protocoles *TIP* (sauf le langage de contraintes qui ne représente pas un aspect développé dans le manuscrit), le langage *TOSL* et l'agent *TAG* sont implantés. L'agent *TAG*, le contrôleur et le raisonnement interne n'ont pas une implémentation complète et générique. Ils sont spécialisés aux applications. En effet, la mise en œuvre complète générique étant un travail important qui s'appuie sur la spécification réalisée, nous avons préféré faire des implémentations particulières à chacune des applications pour valider notre approche.
Sur le plan temporel, les applications ne concernent pour l'instant que des exemples assez simples permettant toutefois de valider notre démarche et de montrer son intérêt. La prise en compte du temps devient très vite complexe lorsque l'on s'attaque à des problèmes importants. Nous avons donc préféré illustrer la diversité et la complémentarité des aspects temporels dans les exemples traités. En outre, l'extension de ces exemples à des cas plus complexes nécessite un temps de développement assez important.
De manière générale, ces travaux menés sur le temps dans les SMA nous ont ainsi ouvert un grand nombre de perspectives. Nous présentons donc dans la section suivante, les principales perspectives d'évolution de ces travaux qui nous paraissent intéressantes.

11.2 Perspectives

Comme nous l'avons souligné, les perspectives sont nombreuses. Nous allons donc passer en revue chacune des dimensions proposées par l'approche *Voyelles* pour présenter ces perspectives.

11.2.1 Perspectives liées à l'interaction

Comme nous l'avons dit, le langage de protocoles d'interaction temporels (TIP) est une ébauche, il conviendrait afin d'avoir une prise en compte plus complète du temps dans l'interaction de le finaliser en définissant notamment explicitement un langage de contraintes et en élaborant une bibliothèque de protocoles ainsi que des outils pour les configurer temporellement selon les besoins. De plus, nous nous limitons pour l'instant à des protocoles faisant intervenir deux agents, il pourrait

être intéressant d'étendre cette notion à un nombre supérieur à deux, notamment pour des phases de négociation comme celles qui pourraient apparaître dans des applications du type de E-Alliance.

La gestion de conversations, abordée très rapidement, mérite de plus amples travaux et notamment nous pensons que la composante temporelle pourrait aider à la gestion de conversations concurrentes. Comme les conversations sont des instanciations de protocoles, il paraît judicieux de travailler auparavant sur les protocoles, qui ne devraient pas manquer d'offrir eux-mêmes de nouvelles perspectives à la gestion des conversations.

11.2.2 Perspectives organisationnelles

Nous nous sommes intéressés à une des manières de représenter l'organisation : les structures organisationnelles. L'adaptation du moteur de relations de dépendances temporelles développé par Allouche [All98] permettrait de compléter le modèle organisationnel temporel que nous proposons. Ainsi, nous pourrions combiner notre modèle avec la théorie des dépendances [SD95] afin de compléter le raisonnement social des agents : conscients de l'organisation dans laquelle ils évoluent, les agents peuvent calculer eux-mêmes leurs relations ou dépendances avec les autres agents pour achever une tâche.
Une autre manière de représenter l'organisation consiste à utiliser des opérateurs déontiques (Obligation, Permission, etc.). Ces aspects existent dans notre modèle pour la spécification des missions mais ils pourraient être complétés par une composante temporelle permettant d'exprimer des contraintes telles que : "il est permis entre 12h et 14h d'effectuer une mission" qui, elle, possède d'autres contraintes (limitée à certains jours de la semaine par exemple). Enfin, différents outils de vérification de cohérence à la spécification d'une structure organisationnelle selon le modèle *MOISE* doivent être complétés pour intégrer et prendre en compte la validité temporelle que nous avons évoquée dans le cas de structures organisationnelles temporelles. A ce sujet, il serait intéressant également de définir des outils graphiques permettant de spécifier, modifier simplement une tos et constituer une véritable bibliothèque de structures organisationnelles temporelles types et moins spécifiques que celles définies par le biais des applications industrielles.

11.2.3 Perspectives relatives à l'environnement

Nous avons expliqué pourquoi nous avons limité l'étude du temps dans l'environnement. Cependant, il pourrait être tout de même intéressant de poursuivre ces travaux en s'orientant sur des SMA dans lequel l'environnement est très présent comme dans le domaine de la robotique. Bien que ces travaux s'appuient en majorité sur des agents de type réactif, l'étude de ces travaux pourraient éventuellement permettre d'aborder le problème de la modélisation de l'environnement

selon un angle nouveau et de mettre en lumière certaines caractéristiques relatives à la dimension temporelle.

11.2.4 Perspectives dans l'agent

L'une des perspectives les plus intéressantes réside dans l'étude de l'impact de la mise en place de différents types d'agents et même diverses stratégies de contrôle temporelles en fonction du type d'application choisi ainsi que l'inverse : pouvoir spécifier quelle stratégie utiliser en fonction de l'application concernée. Il pourrait être utile également d'intégrer des outils déjà développés pour le raisonnement interne (ordonnancement, planification) afin de faire le même type de tests.
Nous pouvons, en outre étudier l'impact de la mise en place de différentes stratégies de coordination temporelle simultanément. En d'autres termes, que se passe-t-il lorsque des agents temporels utilisent des stratégies différentes et quel est l'impact sur le comportement (la cohérence) du SMA lui-même ?
Enfin, il semble que les SMA sont arrivés à un niveau de maturité suffisant, pour commencer à sentir poindre le besoin d'outils de comparaison, de tests (benchmarks), il paraît probable que l'un des axes de comparaison intégrera la dimension temporelle, que ce soit au niveau des temps de réponses comme il est de mise généralement dans ce type d'outils ou tout simplement au sujet de la prise en compte des aspects temporels, de la réactivité.

Pour terminer, nous reviendrons sur nos objectifs initiaux : il apparaît clairement qu'en s'appuyant sur l'approche *Voyelles*, nous pouvons non seulement proposer différents modèles temporels complémentaires, concevoir et réaliser des SMA temporels bien adaptables et adaptés aux systèmes industriels dynamiques mais aussi, de manière plus générale, mettre en place un véritable "raisonnement temporel orienté multi-agents".

A

Exemples de fonctionnement du Gestionnaire de Relations entre IntervalleS

Cette annexe complète la présentation du gestionnaire de relations entre intervalles (GRIS) du chapitre 4.3.

A.1 Exemples de fonctionnement

Voici un exemple (cf. figure A.1) d'entrée type pour le gestionnaire de relations entre intervalles mettant en œuvre la granularité.

```
GRANULARITY
0000:00:00:00:02:00
THEORY
I (:E) [2001:05:24:16:01:00 , 2001:05:24:16:06:00]
I (:E) [2001:05:24:16:02:00 , 2001:05:24:16:09:00]
K (:B) I
K (:B) J

QUERY
I J
I K
I I
J J
K K
```

FIG. A.1 - Entrée type pour le gestionnaire de relations entre intervalles

Le résultat obtenu en sortie est présenté sur la figure A.2 et donne l'intervalle de fin de K (qui est en fait un instant) et précise que l'intervalle de début est inconnu.

```
I (:S) J
J (:S-) I

I (:A) K
K (:B) I

I = [2001:05:24:16:02:00 , 2001:05:24:16:06:00]

J= [2001:05:24:16:02:00 , 2001:05:24:16:08:00]

K = [? , ?] , [2001:05:24:16:02:00 , 2001:05:24:16:02:00]
```

FIG. A.2 – Résultat de l'analyse de l'entrée correspondant à la figure A.1

Autre exemple

Cet exemple permet de voir un résultat en mode symbolique permettant de placer un intervalle par rapport à un autre.

```
GRANULARITY
0000:00:00:00:05:00
THEORY
mission_PrintNB (:E) [2001:05:24:18:00:00 , 2001:05:24:18:47:00]
mission_Bind (:E) [2001:05:24:18:20:00 , 2001:05:24:18:44:00]
mission_PrintNB (:M) mission_PrintColor


QUERY
mission_Bind    mission_PrintColor
```

FIG. A.3 – Entrée type "spécification de problème"

Le gestionnaire de relations entre intervalles permet de préciser que les missions Bind et PrintColor "se touchent" (M : *meet*) comme le montre la figure A.4.

A.2 Utilisation de GRIS

Dans le cadre d'applications complexes, GRIS a été développé pour obtenir des résultats simplifiés à partir d'un ensemble de contraintes temporelles permettant ainsi d'interroger le gestionnaire sur le positionnement d'intervalles entre eux. Concrètement, si un agent joue un rôle "*Imprimeur*" pendant un intervalle

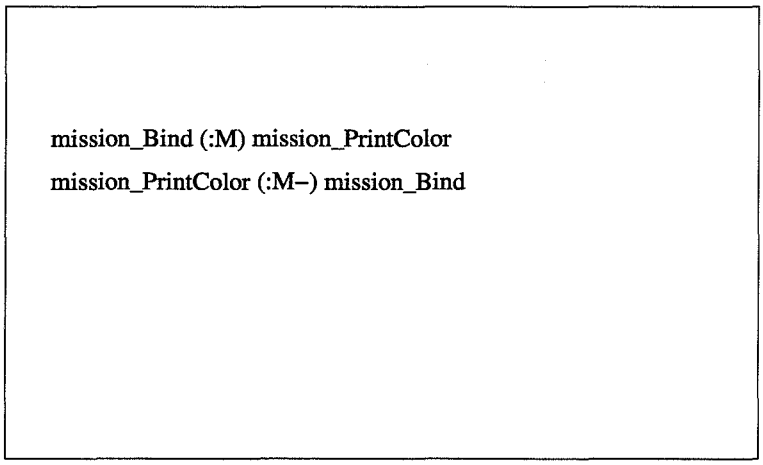


FIG. A.4 – Résultat de la requête de la figure A.3

[8 : 00, 12 : 00] (du jour actuel), et qu'on lui propose d'accomplir une mission de durée [00 : 30] dépendante du rôle entre [11 : 00, 12 : 00]. L'interrogation de GRIS permet de trouver l'intervalle possible de début de réalisation [11 : 00, 11 : 30] et de fin.

Autre exemple
Dans le cadre des actes de langages (sémantique), on vérifie que l'intervalle de validité I d'un état mental (par exemple une croyance) vérifie les préconditions de l'acte choisi (inform par exemple) en interrogeant GRIS : l'instant courant appartient-il à l'intervalle I ? ($[t_{now}, t_{now}]$ *in* I?). La vérification temporelle se fait après avoir sélectionné les états mentaux candidats et l'acte correspondant.

A.2.1 Raisonnement causal

Ce n'est pas spécifiquement GRIS mais SPASS (cf. schéma chapitre 4.3) qui se charge de capacité de raisonnement causal :
Afin d'expliciter ses possibilités, nous allons donner un exemple de problème :
Sur un intervalle $t0$ donné, l'agent C est en train d'exécuter une activité $activity1$. Un agent D qui observe l'agent C constate que sur l'intervalle suivant $t1$, l'agent C n'exécute plus $activity1$. Cet agent D possède de plus quelques connaissances causales concernant le fonctionnement de C. Il sait par exemple que :

- Si une activité sur un intervalle I n'est plus exécutée sur l'intervalle suivant (par suivant nous entendons que la relation "I *meets* J" est vérifiée), c'est qu'il y a eu changement d'activité sur un intervalle précédant J.
- S'il y a eu un changement d'activité sur I, c'est que l'agent A a effectué l'acte de langage "command" sur I.

L'agent D peut déduire de tout cela que l'agent A a effectué l'action command avant.

La transcription du problème pour automatiser le raisonnement est donnée ci-dessous) en DFG (syntaxe particulière au traducteur FLOTTER) :

A.2.2 Spécification du problème illustrant le raisonnement causal

$begin\_problem(Temp).$

$list\_of\_descriptions.$
$end\_of\_list.$

$list\_of\_symbols.$
$functions[(t0,0),(t1,0),(meets,0),(meeted\_by,0),(time,1),(command\_A\_C,0)].$
$predicates[(tempRel,3),(CHANGED\_C,2),(try,2),(activity1\_C,1)].$
$end\_of\_list.$

$list\_of\_formulae(axioms).$
$formula(TempRel(meeted\_by,t1,t0)).$
$formula(activity1\_C(t0)).$
$formula(not(activity1\_C(t1))).$
$formula(forall([T,U],implies(and(activity1\_C(T),not(activity1\_C(U)).$
$TempRel(meets,T,U)),exists([E,V],$
$and(CHANGED\_C(E,V),TempRel(meets,V,U)))))).$
$formula(forall([T,E],implies(CHANGED\_C(E,T),and(Try(command\_A\_C.T).$
$equal(t,time(E))))))).$
$end\_of\_list.$

$list\_of\_formulae(conjectures).$
$formula(exists([T],and(Try(command\_A\_C,T),TempRel(meets,T,t1)))).$
$end\_of\_list.$

$list\_of\_settings(SPASS).$
$set\_flag(FPModel,1).$
$set\_flag(DocProof,1).$
$set\_flag(PProblem,1).$
$set\_flag(PGiven,1).$
$set\_flag(PDer,1).$
$set\_flag(PEmptyClause,1).$
$set\_flag(PClRed,1).$

$end\_of\_list.$

$end\_problem.$

$CHANGED\_C$ représente l'événement changement d'activité de l'agent C, $command\_A\_C$ représente l'acte de langage command avec A pour locuteur et C pour récepteur. Les deux dernières formules sont les connaissances causales qui vont servir à raisonner. Ce que l'on demande au démonstrateur est de prouver que c'est une action $command\_A\_C$ qui a été effectuée avant $t1$.

La preuve trouvée par SPASS est donnée ci-dessous.

A.2.3 Preuve obtenue par SPASS en réponse au problème illustrant le raisonnement causal

Here is a proof with depth 5, length 18 :

$1[0 : Inp]|| - > activity1\_C(t0)*.$

$2[0 : Inp]||activity1\_C(t0) * - > .$

$3[0 : Inp]|| - > TempRel(meeted\_by, t1, t0)*.$

$4[0 : Inp]||CHANGED\_C(U, V) * + - > Try(command\_A\_C, V)*.$

$5[0 : Inp]||CHANGED\_C(U, V) - > equal(V, time(U))*.$

$6[0 : Inp]activity1\_C(U)||TempRel(meets, U, V) * + - >$
$CHANGED\_C(skf0(V), skf1(V)) * activity1\_C(V).$

$7[0 : Inp]activity1\_C(U)||TempRel(meets, U, V) * + - >$
$TempRel(meets, skf1(V), V) * activity1\_C(V).$

$8[0 : Inp]||Try(command\_A\_C, U)TempRel(meets, U, t1) * - >.$

$9[0 : TeI]|| - > TempRel(meets, t0, t1)*.$

$14[0 : GeR : 9.0, 6.1]activity1\_C(t0)|| - >$
$CHANGED\_C(skf0(t1), skf1(t1)) * activity1\_C(t1).$

$15[0 : SSi : 1.0, 14.0]|| - >$
$CHANGED\_C(skf0(t1), skf1(t1)) * activity\_C(t1).$

$16[0 : ClR : 15.1.2.0]|| - > CHANGED\_C(skf0(t1), skf1(t1))*.$

$20[0 : GeR : 16.0.4.0]|| - > Try(command\_A\_C, skf1(t1))*.$

$26[0 : GeR : 9.0, 7.1]activity1\_C(t0)|| - >$
$TempRel(meets, skf1(t1), t1 * activity\_c(t1).$

$27[0 : SSi : 1.0, 26.0]|| - > TempRel(meets, skf1(t1), t1) * activity\_C(t1).$

$28[0 : ClR : 27.1, 2.0]|| - > TempRel(meets, skf1(t1), t1)*.$

$30[0 : GeR : 28 : 0.8.1]||Try(command\_A\_C, skf1(t1)) * - >.$

$33[0 : ClR : 30.0, 20.0]|| - >.$

Les 8 premières clauses de la preuve représentent les formules spécifiées en entrée, les 10 suivantes sont les clauses déduites. Parmi ces 10 dernières, une est issue de la gestion des relations entre intervalles (GRIS), la clause 9. On le remarque à l'entête qui spécifie l'origine de la clause. "Tel" signifie ici "Temporal Inference".

Cependant, aucun raisonnement causal n'est actuellement implémenté dans le cadre de nos applications. Ceci fait partie des perspectives.

A.2.4 Rôle de l'interface GRIS/SPASS

Le but de l'interface GRIS/SPASS est d'assurer le dialogue entre les deux utilitaires :

- SPASS dispose de l'ensemble des informations temporelles. L'interface doit donc les extraire de SPASS et les envoyer à GRIS.
- GRIS produit de nouvelles informations temporelles (les relations entre intervalles déduites de celles données en entrée). Ces informations doivent alors être réinjectées dans le démonstrateur SPASS (nous avons fait le choix technique d'intégrer Gris à SPASS en tant que "library").

A ce niveau, se posent deux questions importantes :

- Quelles informations temporelles doivent être extraites et envoyées à GRIS ?
  Seules les relations entre intervalles présentées en tant que faits doivent être extraites. Ces relations sont soit issues de la spécification du problème, soit dérivées par le démonstrateur. Elles doivent toutes être extraites car ces informations sont, par nature, contextuelles (la relation qui lie deux intervalles I et J peut se trouver modifiée du fait des relations de ces deux intervalles avec un troisième K). Pour des raisons de simplicité, une hypothèse forte est faite : nous n'avons, ici, que des conjonctions de faits en se basant sur le constat que, dans de nombreux problèmes temporels, aucune disjonction entre prédicats temporels n'est utile.
- Quand ces informations temporelles doivent-elles être extraites ?
  En se basant sur les choix précédents, l'extraction est réalisée chaque fois qu'au moins une nouvelle relation entre intervalles est dérivée. Dans le fonctionnement de SPASS, un test est fait sur l'existence d'une nouvelle relation entre intervalles. Si celle-ci est avérée, le gestionnaire de relations entre intervalles est appelé et les nouvelles informations qu'il a éventuellement produites, sont insérées dans l'ensemble des clauses dérivées (cf. ensemble Dérivables dans le pseudo-code ci-dessous).

pseudo-code

ALGORITHME DE FONCTIONNEMENT DE SPASS

algorithm SPASS
begin
$readinputclauses from file$;
$Usables := reduced input clauses$;
$WorkedOffs := 0$;
$while(not(Proof found) and not(Model Found))$
$if\ (Empty Clause found) then$
$Derivables := SplitBacktracking(EmptyClause)$;
$else$
$GivenClause := Select(Usables)$;
$SelectAntecedentLiteral(GivenClause)$;
$MoveGivenClause from the Usables to the worked Offs$;
$Derivables := ComputeDerivables(GivenClause, WorkedOffs)$;
$forall(clauses Clause from the Derivables) do$
$insertReduced(Clause, Usables, WorkedOffs)$;
$keep Empty Clause, if found$;
end.

A.3 Interface utilisateur

La figure A.5 présente un autre exemple de requête et le résultat obtenu (en mode symbolique) au moyen de l'interface.

L'interface permet également d'afficher le graphe de contraintes temporelles afin d'en avoir une vue plus intuitive.

De plus amples informations peuvent être trouvées dans [Gau99].

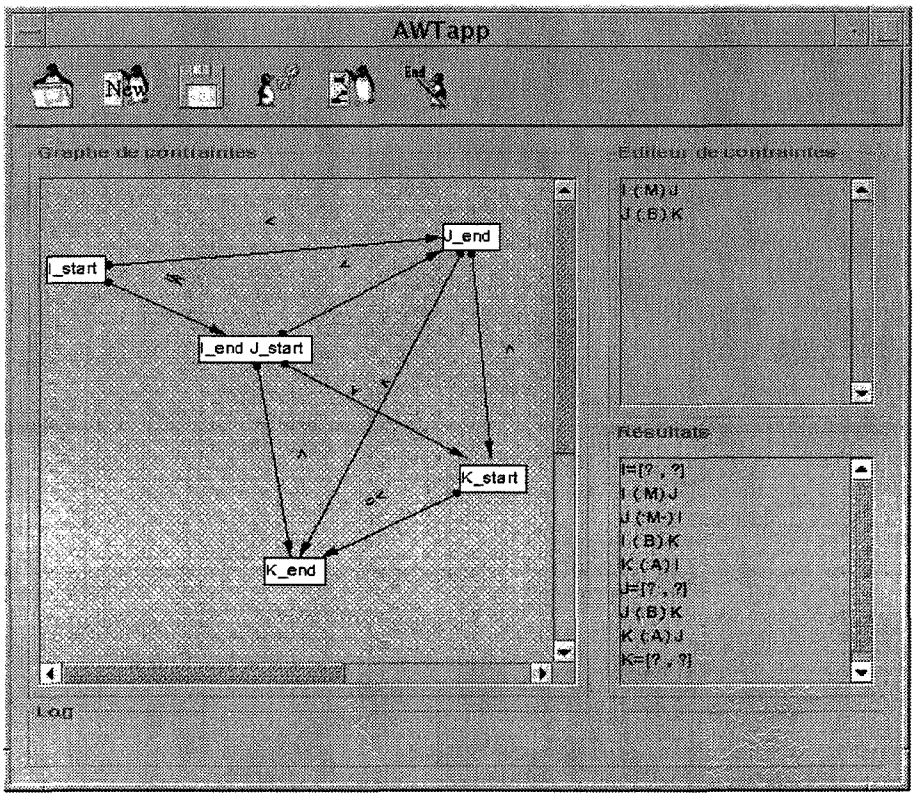


FIG. A.5 – Interface du gestionnaire de relations entre intervalles

B

Stratégies de mises à jour des états mentaux dans la sémantique des actes de langage

Dans cette annexe, nous évoquons différents exemples de stratégies permettant de mettre à jour les contraintes temporelles des états mentaux reçues à travers des messages *TACL* (cf. chapitre 5.2.2).

B.1 Commentaires sur les contraintes temporelles

La sémantique des actes de langages temporels que nous avons définie met en exergue l'évolution de contraintes temporelles (i, j et k) entre les *préconditions*, les *postconditions* et l'*effet perlocutoire*. Pour l'acte inform :

inform(x, y, i, ϕ)
Prec : $B(x, i, \phi) \wedge B(x, j, \neg B(y, k, \phi)) \wedge t_{now} \in i \cap j \cap k$
Post : (+) $B(x, i', B(y, i'', \phi))$ (-) $B(x, j, \neg B(y, k, \phi))$
Pe : $B(y, i'', \phi)$

Cette description met en exergue trois expressions temporelles i, j et k spécifiant les contraintes temporelles des préconditions et susceptibles d'évoluer pour devenir respectivement i' (et/ou i''), j' et k', etc. Pour être utilisables, les différents états mentaux doivent être actifs donc les intervalles i, j et k contiennent l'instant t_now. Enfin, nous avons différencié i, j et k car les intervalles correspondants sont néanmoins indépendants : ils concernent des croyances distinctes.
A présent, intéressons-nous à l'évolution de i en i' et en i''. Les *préconditions* nécessaires à l'envoi d'un acte de communication de type *inform* spécifient que l'agent x doit croire à l'instant i que la proposition ϕ est vraie et il doit croire qu'à l'instant j, l'agent y, lui ne croit pas que la proposition ϕ est vraie pendant k. Les *postconditions* spécifient une mise à jour des états mentaux relatives à l'envoi du message correspondant à l'acte ; c'est à dire à un moment ultérieur par rapport à

celui correspondant aux préconditions. Par conséquent, la croyance ajoutée selon les postconditions possède une date de validité qui peut être différente de celle relative aux préconditions. Elle prend sa validité à partir de l'envoi du message d'où la valeur i'. Dans notre cas de figure, nous avons également choisi d'illustrer une stratégie particulière pour i'' : la nouvelle croyance de y commencera à partir du moment de réception du message. La valeur i'' est mise à jour à partir de i. Concrètement, si $i = [18h30m00, 19h00m00]$ et que le message est envoyé à 18h30m00 et reçu deux minutes plus tard, $i'' = [18h32m00, 19h00m00]$. Ce cas de figure semble relativement particulier mais peut se généraliser de la manière suivante :
"Un agent ne garde pas de croyances passées."
Une autre stratégie plus simple consisterait à garder telles quelles les expressions temporelles liées aux croyances communiquées et ainsi garder la valeur de i tout au long du processus. Nous avons choisi d'expliciter une stratégie plus complexe afin de montrer une prise en compte concrète de l'utilisation d'acte de langages temporels dans la gestion des états mentaux bien que cela ne soit pas nécessaire dans le cadre des applications que nous avons développées. C'est pourquoi nous présentons ces informations en annexe.
Il est évidemment possible d'imaginer toutes sortes de combinaisons en fonction des besoins. La gestion d'un historique, par exemple, impose des informations précises sur les connaissances passées et par conséquent une stratégie de mise à jour adaptée.

Nous allons à présent, décrire les algorithmes relatifs à la mise à jour de ces états mentaux.

B.2 Exemples d'algorithmes

Ces exemples sont présentés en notation algorithmique. L'instant de début de l'intervalle X est noté DebX et l'instant de fin FinX. La durée de création d'un message est notée DCréation et la durée de transit du message sur le réseau DTransit.

Ainsi l'algorithme de mise à jour des croyances pour l'acte de langage inform est le suivant :
DebX' $\Leftarrow$ DebX + DCréation
FinX' $\Leftarrow$ FinX
DebX" $\Leftarrow$ DebX + DTransit
FinX" $\Leftarrow$ FinX

L'algorithme ci-dessus est valable pour chacun des actes de langage définis dans la section 5.2.2.

Cependant pour simplifier, nous avons introduit la notation t_now dénotant l'instant présent. Ainsi, nous nous intéressons uniquement à la fin de validité de l'état mental : l'instant de début est représenté par t_now pour éviter d'avoir à remettre à jour constamment la borne inférieure de chaque état mental.

C

Définition d'un scénario relatif à une entreprise fonctionnant par projet

Cette annexe correspond à l'étude de la modélisation de l'entreprise décrite dans le chapitre 10 par un SMA temporel.

C.1 Diagrammes de séquences

La représentation[22] se concentre sur la séquence des interactions selon un point de vue temporel. Cette modélisation sert à spécifier les aspects dynamiques avec une vision adaptée pour les systèmes multi-agents.

Ces diagrammes sont représentés au format *UML*.

Description des messages

Le tableau ci-dessous (cf. figures C.1, C.2, C.3, C.4 et C.5) correspond à l'ensemble et la chronologie des messages échangés au cours du déroulement du projet.

C.2 Langage de contenu

Un langage spécifique à l'application est défini. Ce langage apparaît dans le champ *content* du langage *TACL* (cf. chapitre 5).

Le tableau qui suit (cf. figures C.6, C.7, C.8), correspond à des exemples de messages envoyés pendant le déroulement d'un projet. Les numéros correspondent aux transitions des diagrammes de séquence présentés précédemment.

[22]Cette annexe est issue d'un document de travail appelé *Modélisation des agents PVS98* réalisé par O.Boissier (ENSMS-E), G. Pépiot (EPFL) et moi-même dans le cadre de ce projet.

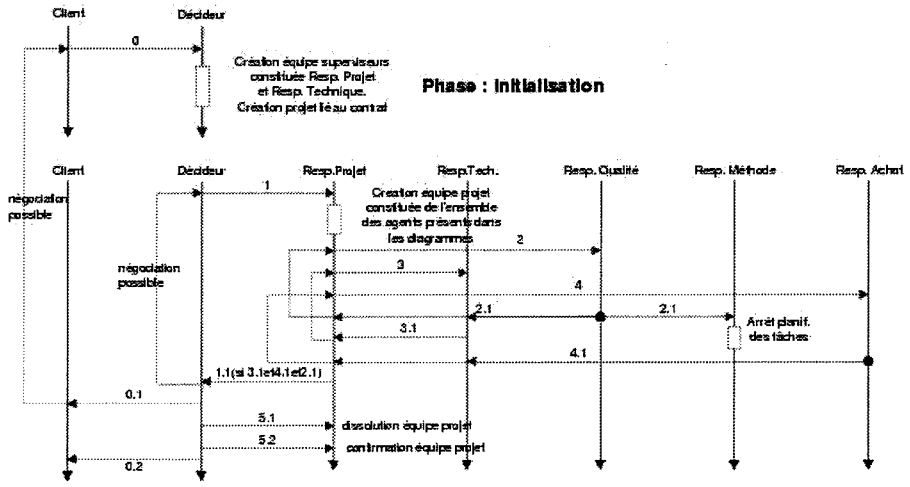


FIG. C.1 – Phase : Evaluation de projet

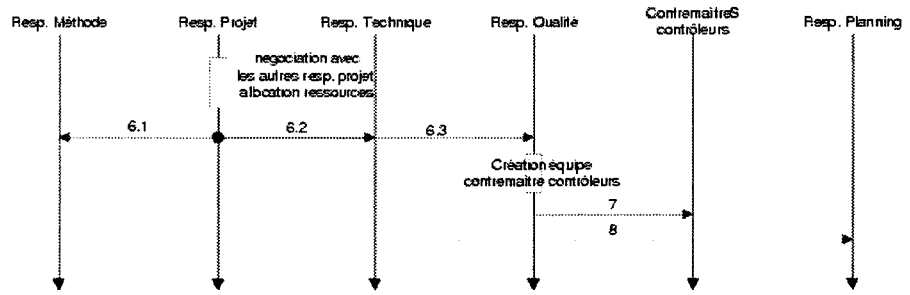


FIG. C.2 - Suivi de Projet / Enclenchement de Projet

Le tableau C.9 décrit quelques uns des termes utilisés dans les messages.

C.3 Description des compétences

Le tableau présenté ici (cf. figures C.10 et C.11) décrit une partie de l'ensemble des "compétences" au sens *savoir-faire* du chapitre 7. Cela permet de spécifier ce qu'est capable de réaliser une entité spécifique intervenant dans le déroulement d'un projet.

Compétences de l'agent "Decision-Maker"

Nous donnons ici à titre d'exemple, les "savoir-faire" de l'agent *Decision-Maker* (décideur) déduit de l'étude menée sur l'entreprise :

Name :Contract Dispatching ID :C300 Duration :00 :60 :00
Nature : ? Commentary :"no comment"
Name :Contract Evaluation ID :C400 Duration :00 :10 :00 ...

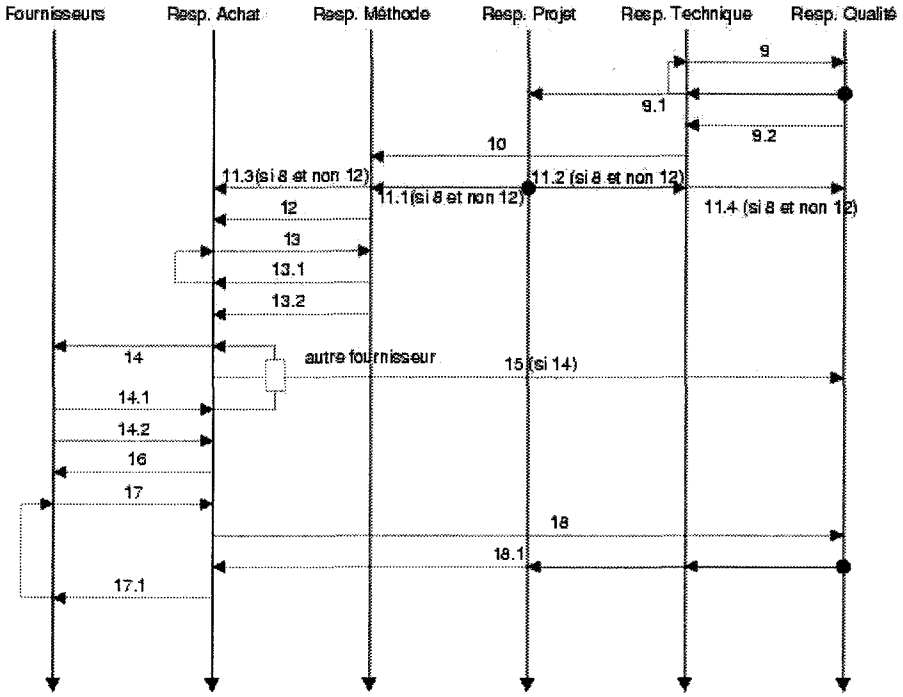


FIG. C.3 – Suivi de Projet / Définition des contrôles

C.4 Structure organisationnelle relative à un projet

Le schéma C.12 correspond à la structure organisationnelle chargée pour la gestion d'un projet (tous les liens ne sont pas représentés afin de ne pas avoir un schéma trop chargé).

C.5 Interface d'observation d'agent

Un exemple d'observation d'agent est donné sur la figure C.13.

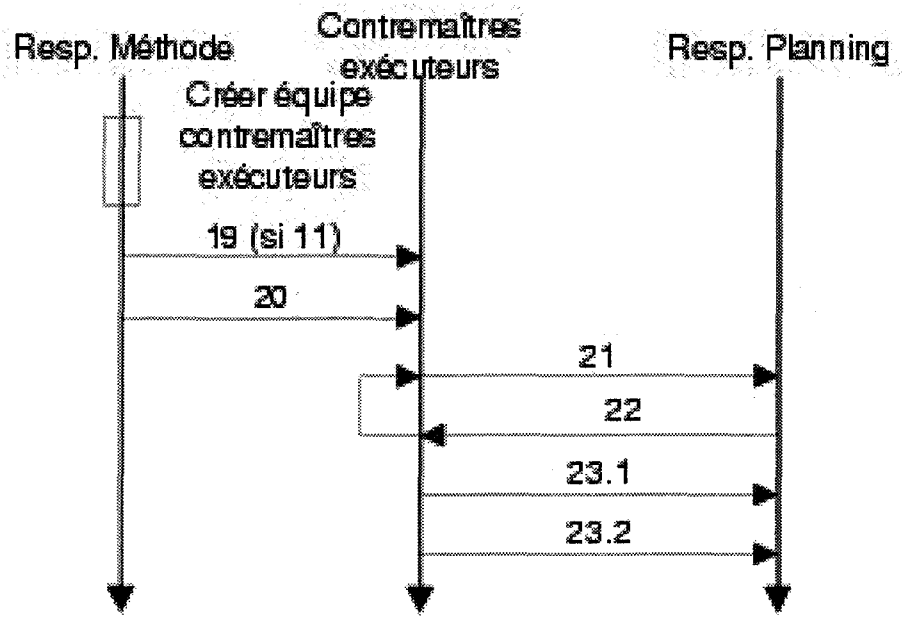


FIG. C.4 – Suivi de Projet / Réalisation

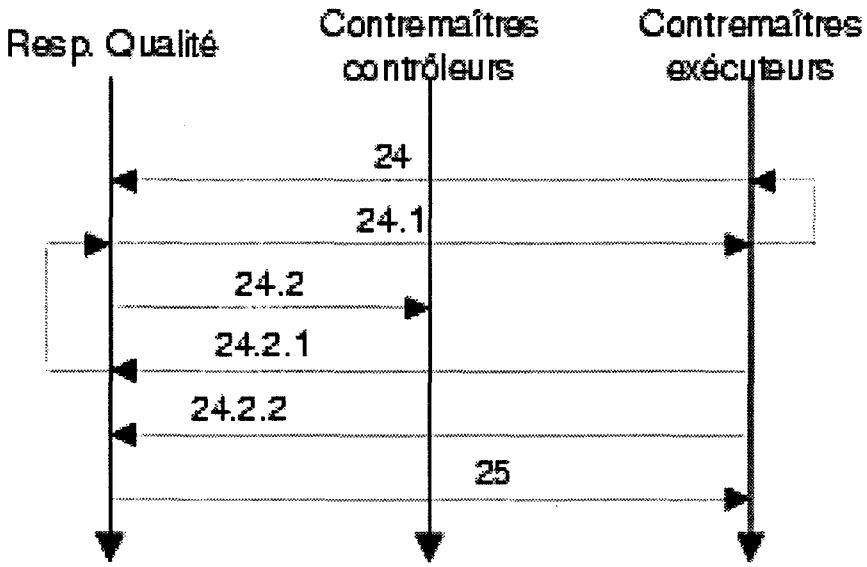


FIG. C.5 – Suivi de Projet / Contrôle de qualité

N°	From	To	Contenu	Commentaire
0	Client	Décideur	Propose realiser contrat	
0.1	Décideur	Client	Tell contrat_accepté=non	
0.2	Décideur	Client	Tell contrat_accepté=oui	
1	Décideur	Resp. Projet	Achieve évaluer projet	
1.1	Resp. Projet	Décideur	Tell faisabilité_projet=oui/non coût qualité technique projet	
2	Resp. Projet	Resp. Qualité	Achieve évaluer conditions_assemblage projet	
2.1	Resp. Qualité	Resp. Projet Resp. Méthode Resp. Technique	Tell faisabilité_projet=oui/non délai qualité projet	
3	Resp. Projet	Resp. Technique	Achieve évaluer concepts_techniques projet	
3.1	Resp. Technique	Resp. Projet	Tell faisabilité_projet=oui/non délai projet	
4	Resp. Projet	Resp. Achat	Achieve évaluer besoins_équipement projet	
4.1	Resp. Achat	Resp. Projet Resp. Technique	Tell faisabilité_projet=oui/non coût délai projet	
5.1	Décideur	Resp. Projet	Tell projet_accépté=non projet	
5.2	Décideur	Resp. Projet	Tell projet_accepté=oui projet	
6.1	Resp. Projet	Resp. Méthode	Tell ressources_disponibles délais main_œuvre projet	
6.2	Resp. Projet	Resp. Technique	Tell ressources_disponibles délais main_œuvre	
6.3	Resp. Projet	Resp. Qualité	Tell ressources_disponibles délais main_œuvre	
7	Resp. Qualité	Contremaître Contrôleur	Tell ressources_disponibles délais main_œuvre	

FIG. C.6 – Description des messages

8	Resp. Technique	Resp. Planning	Tell ressources_disponibles délais main_œuvre	
9	Resp. Technique	Resp. Qualité	Tell dessins_plans_fabrication projet	
9.1	Resp. Qualité	Resp. Technique Resp. Projet	Tell RNC projet	
9.2	Resp. Qualité	Resp. Technique	Tell plan_de_contrôle projet	
10	Resp. Technique	Resp. Méthode	Tell plan_de_contrôle projet	
11.1	Resp. Projet	Resp. Méthode	Tell %_tâches_à_soustraiter projet	
11.2	Resp. Projet	Resp. Technique	Tell %_tâches_à_soustraiter projet	
11.3	Resp. Projet	Resp. Achat	Tell %_tâches_à_soustraiter projet	
11.4	Resp. Projet	Resp. Qualité	Tell %_tâches_à_soustraiter projet	
12	Resp. Méthode	Resp. Achat	Achieve finaliser liste_achats projet	
13	Resp. Achat	Resp. Méthode	Achieve évaluer liste_achats projet	
13.1	Resp. Méthode	Resp. Achat	Tell conformité_liste_achats=non projet	
13.2	Resp. Méthode	Resp. Achat	Tell conformité_liste_achats=oui projet	
14	Resp. Achat	Fournisseur	Propose realiser contrat	Contrat inclus dans liste_achats, ide que 0
14.1	Fournisseur	Resp. Achat	Tell contrat_accepté=non	Idem que 0.1
14.2	Fournisseur	Resp. Achat	Tell contrat_accepté=oui	Idem que 0.2
15	Resp. Achat	Resp. Qualité	Tell liste_pièces_à_contrôler projet	
16	Resp. Achat	Fournisseur	Achieve contrat	
17	Fournisseur	Resp. Achat	Tell reception_matières	
17.1	Resp. Achat	Fournisseur	Achieve revoir réception	
18	Resp. Achat	Resp. Qualité	Tell certificat_matieres projet	
18.1	Resp. Qualité	Resp. Achat Resp. Technique Resp. Projet	Tell RNC projet	Pas de 18.2 car si pas de RNC, alors ok
19	Resp. Méthode	Contremaître Exécuteur	Achieve former groupe projet	
20	Resp. Méthode	Contremaître Exécuteur	Achieve réaliser OF projet	

FIG. C.7 – Description des messages (suite)

N°	From	To	Contenu	Commentaire
21	Resp. Planning	Contremaître Exécuteur	Tell heures_planif OF projet	
22	Contremaître Exécuteur	Resp. Planning	Tell heures_passées OF projet	
23.1	Resp. Planning	Contremaître Exécuteur	Achieve bloquer OF	
23.2	Resp. Planning	Contremaître Exécuteur	Achieve libérer OF	
24	Contremaître Exécuteur	Resp. Qualité	Achieve évaluer conformité_des_travaux_réalisés projet	
24.1	Resp. Qualité	Contremaître Exécuteur	Tell RNC	
24.2	Resp. Qualité	Contremaître Contrôleur	Achieve évaluer conformité_des_travaux_réalisés projet	
24.2.1	Contremaître Contrôleur	Resp. Qualité	Tell RNC	
24.2.2	Contremaître Contrôleur	Resp. Qualité	Tell conforme	
25	Resp. Qualité	Contremaître Exécuteur	Achieve libérer produit_fini_pour_livraison projet	

FIG. C.8 – Description des messages (suite)

Contenu	Type
tâches_à_soustraiter	Entier
besoins_équipement	
bloquer	Action
certificat_matières	
concepts_techniques	
conditions_assemblage	
conforme	booléen
conformité_des_travaux_réalisés	
conformité_liste_achats	booléen
contrat	
contrat_accepté	Booléen
coût	
délais	Entier
dessins_plans_fabrication	
évaluer	Action
faisabilité_projet	Booléen
finaliser	Action
former	Action
groupe	
heures_passées	

FIG. C.9 – Description des termes utilisés dans les messages

Agent	Message Entrant	Message Sortant	Compétences Activées	Commentaire
Resp. Qualité	2	2.1	C1: **Aider à la décision conseiller** - Evalue la faisabilité des tâches	
Resp. Technique	3	3.1	C2 : **Revoir le contrat** - Evalue la complexité des tâches en vue des ressources à prévoir	
Resp. Achat	4	4.1	C3 : **Identifier équipement** - Identifie avec les responsables techniques, les équipements et le matériel nécessaire à toutes les phases du projet C4 : **Aider conseiller** - Conseille les responsables techniques sur l'acquisition et le type de matériel C5 : **Estimer % valeur équipement** - Estime la proportion de l'équipement par rapport au coût global de l'affaire	
Resp. Projet	5.2	6.1, 6.2, 6.3	C6 : **Déterminer les ressources disponibles**	
Resp. Qualité	6.3	7	C7 : **Attribuer ressources** - Détermine les ressources à attribuer et à qui les attribuer (mise à jour de la liste des couples (ressources, destinataires des ressources)	
Resp. Technique	6.2	8	C8 : **Mise en plan de l'équipe** - Détermine les ressources à attribuer et à qui les attribuer C9 : **Mentionner structure générale du projet** - Détermine les spécifications techniques et contractuelles des tâches, mentionne la structure générale du projet C10 : **Mentionner points critiques** - Détermine critères de choix de l'équipe projet	

FIG. C.10 – Description des compétences

Agent	Message Entrant	Message Sortant	Compétences Activées	Commentaire
Resp. Qualité	9	9.1	C11 : **Lire les dessins** - Analyse qualité de dessins C12 : **Viser fiche de conformité** - Compare la réalisation ou la qualité par rapport aux exigences de départ C13 : **Bloque les plans** - Empêche l'utilisation de plans non conformes C14 : **Etablir rapport de non conformité** - Détermine les spécifications de la non conformité	
Resp. Qualité	9	9.2	C15 : **Définir contrôles** - Analyse des contrôles prévisibles C16 :**Etablir plan de contrôles** - Etablit les plans de contrôle nécessaires à partir des description techniques C17 : **Etablir liste de contrôles** - Détermine le matériel nécessaire aux contrôles sélectionnés C18 : **Evaluer compétences** - Détermine les compétences à assigner pour les contrôles	
Resp. Projet		11.1, 11.2, 11.3, 11.4	C20 : **Déterminer % de tâche à sous traiter**	
Resp. Achat	12	13	C21 : **Elaborer une liste d'achat**	
Resp. Méthodes	13	13.1, 13.2	C22 : **Evalue liste d'achat** - Aide les acheteurs à considérer les aspects techniques	
Fournisseur	14	14.1, 14.2	C23 : **Examine le contrat**	
Fournisseur	16	17	C24 : **Elabore une liste de matières premières qu'il doit fournir**	
Resp. Achat	17	17.1	C25 : ***	

FIG. C.11 – Description des compétences (suite)

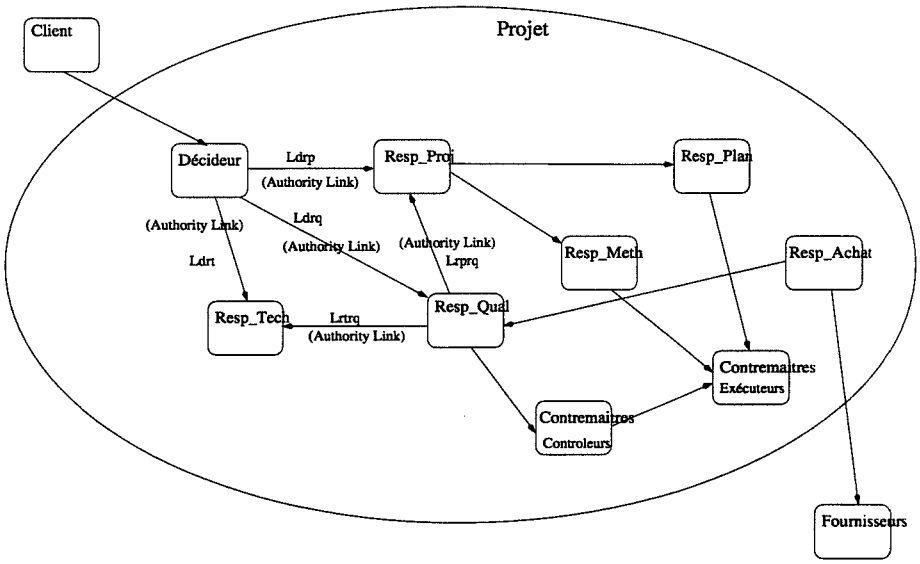


FIG. C.12 – Structure organisationnelle : Projet

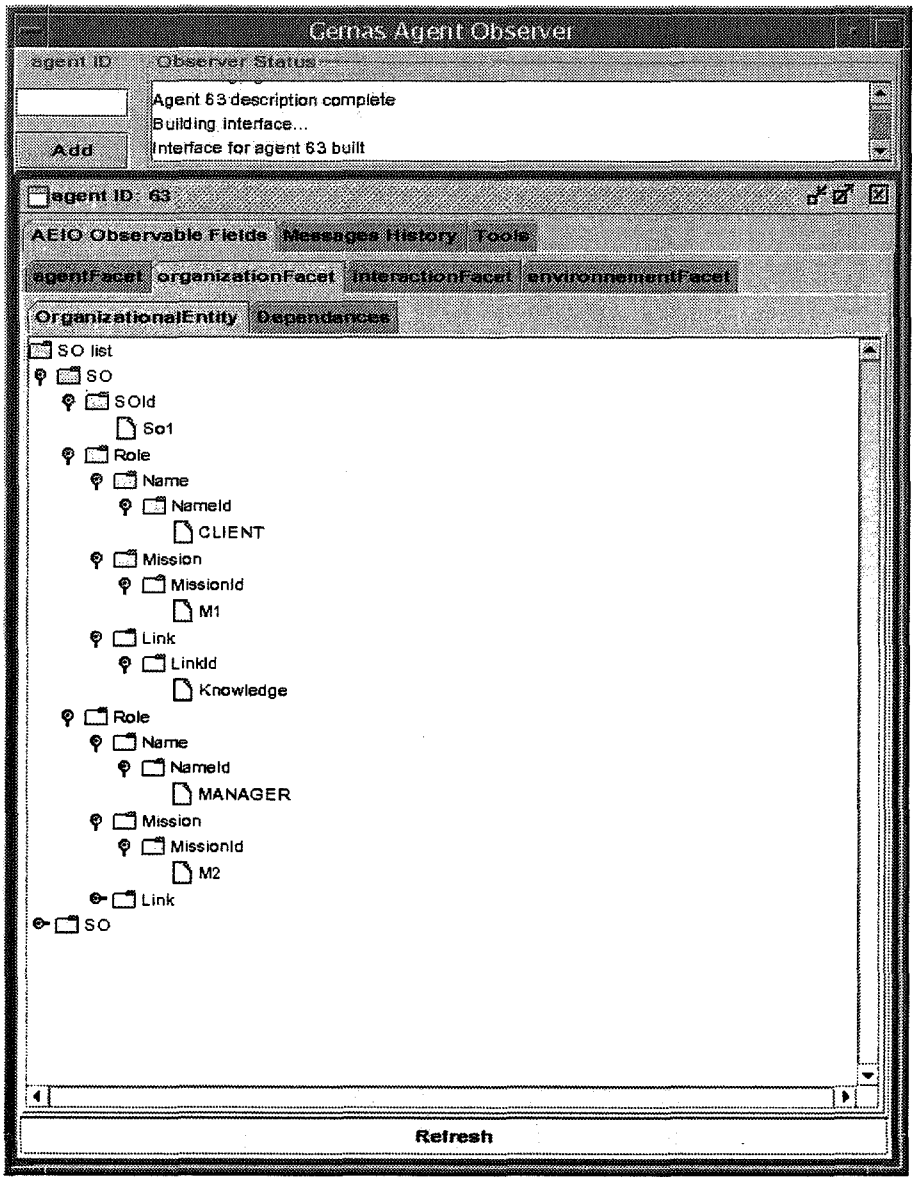


FIG. C.13 – Observation d'un agent TAG dans le cadre d'un projet

D

Exemples de fichiers de configuration relatifs à l'alliance d'ateliers d'impression

Cette annexe complète la présentation de l'application présentée dans le chapitre 9.

D.1 Fichiers DTD

D.1.1 Grammaire du langage TACL sous format DTD

⟨!ELEMENT message (mas,content)⟩
⟨!ELEMENT mas (act,time,nature,protocol,conversation)⟩
⟨!ELEMENT act (inform | command | propose | accept | refuse)⟩
⟨!ELEMENT inform EMPTY⟩
⟨!ELEMENT command EMPTY⟩
⟨!ELEMENT propose EMPTY⟩
⟨!ELEMENT accept EMPTY⟩
⟨!ELEMENT refuse EMPTY⟩
⟨!ELEMENT time (expr)⟩
⟨!ELEMENT expr (texp | *and_expr* | *or_expr*)⟩
⟨!ELEMENT *and_expr* (expr)+⟩
⟨!ELEMENT *or_expr* (expr)+⟩
⟨!ELEMENT texp (inte | A | P | Ss | Sr |
relation | *periodic_texp*)⟩
⟨!ELEMENT inte (duration | interval)⟩
⟨!ELEMENT duration (♯PCDATA)⟩
⟨!ELEMENT interval (t)+⟩
⟨!ELEMENT t (now | undefined | infty | instant)⟩

```
                                        ⟨!ELEMENT now EMPTY⟩
                                        ⟨!ELEMENT undefined EMPTY⟩
                                        ⟨!ELEMENT infty EMPTY⟩
                                        ⟨!ELEMENT instant (♯PCDATA)⟩
                                        ⟨!ELEMENT A EMPTY⟩
                                        ⟨!ELEMENT P EMPTY⟩
                                        ⟨!ELEMENT Ss EMPTY⟩
                                        ⟨!ELEMENT Sr EMPTY⟩
                                        ⟨!ELEMENT relation (rel,texp*)⟩
                                        ⟨!ELEMENT rel (♯PCDATA)⟩
                                        ⟨!ELEMENT periodic_texp (texp)+⟩
                    ⟨!ELEMENT nature (resource | plan | action | goal)⟩
                                ⟨!ELEMENT resource EMPTY⟩
                                ⟨!ELEMENT plan EMPTY⟩
                                ⟨!ELEMENT action EMPTY⟩
                                ⟨!ELEMENT goal EMPTY⟩
                    ⟨!ELEMENT protocol (♯PCDATA)⟩
                    ⟨!ELEMENT conversation (♯PCDATA)⟩
        ⟨!ELEMENT content (♯PCDATA)⟩
```

D.1.2 Grammaire des protocoles temporels TIP sous format DTD

Certaines parties sont modifiées ou simplifiées par rapport à la descrition du chapitre 5 en fonction des besoins de l'application.

```
⟨!ELEMENT protocol (body,transitions)⟩
⟨!ATTLIST protocol
                prname CDATA ♯REQUIRED
                initiator CDATA ♯REQUIRED
                participator CDATA ♯REQUIRED
                goal NMTOKENS ♯REQUIRED⟩
                ⟨!ELEMENT body (trans_name| compound_component)⟩
                                ⟨!ELEMENT trans_name (♯PCDATA)⟩
                                ⟨!ELEMENT compound_component (body,operator,body2)⟩
                                ⟨!ELEMENT operator (and | or | xor)⟩
                                ⟨!ELEMENT and EMPTY⟩
                                ⟨!ELEMENT or EMPTY⟩
                                ⟨!ELEMENT xor EMPTY⟩
                                ⟨!ELEMENT body2 (body)⟩
```

⟨!ELEMENT transitions (transition)+⟩
⟨!ELEMENT transition (message)⟩
⟨!ATTLIST transition
trans_name CDATA #REQUIRED
target_stname CDATA #REQUIRED
tcondition CDATA #REQUIRED⟩
⟨!ELEMENT message (mas,content)⟩
..."contraintes sur le message (format grammaire TACL)"...

D.1.3 Grammaire du langage TOSL sous format DTD

⟨!ELEMENT TOS (roles, links, groups)⟩
⟨!ATTLIST TOS
TOS_name CDATA #REQUIRED
tp CDATA #REQUIRED
tc CDATA #REQUIRED⟩
⟨!ELEMENT roles (role)+⟩
⟨!ELEMENT role (mission)+⟩
⟨!ATTLIST role
role_name CDATA #REQUIRED
tp CDATA #REQUIRED
tc CDATA #REQUIRED⟩
⟨!ELEMENT mission EMPTY⟩
⟨!ATTLIST mission
mission_name CDATA #REQUIRED
tp CDATA #REQUIRED
tc CDATA #REQUIRED⟩
⟨!ELEMENT links (link)+⟩
⟨!ELEMENT link EMPTY⟩
⟨!ATTLIST link
link_name CDATA #REQUIRED
link_type CDATA #REQUIRED
source_role CDATA #REQUIRED
target_role CDATA #REQUIRED
tp CDATA #REQUIRED
tc CDATA #REQUIRED⟩
⟨!ELEMENT groups (group)+⟩
⟨!ELEMENT group (*role_list*, *mission_list*)⟩
⟨!ATTLIST group
group_name CDATA #REQUIRED
tp CDATA #REQUIRED

tc CDATA ♯REQUIRED⟩
⟨!ELEMENT *role_list* (*role_name*)+⟩
⟨!ELEMENT *role_name* (♯PCDATA)⟩
⟨!ELEMENT *mission_list* (*mission_name*)+⟩
⟨!ELEMENT *mission_name* (♯PCDATA)⟩

D.2 Fichiers XML

D.2.1 Exemples de protocoles spécifiés au format XML

Le protocole I (pour "Inform") est chargé par les agents sous la forme suivante :

```
⟨protocol prname="I" initiator="X" participator="Y" goal="BxnotBy(Z)"⟩
        ⟨body⟩
                ⟨trans_name⟩t_inform⟨/trans_name⟩
        ⟨/body⟩
        ⟨transitions⟩
        ⟨transition trans_name="t_inform" target_stname="I" tcondition="TRUE"⟩
                                                                        ⟨message⟩
                                                                        ⟨/message⟩
        ⟨/transition⟩
        ⟨/transitions⟩
⟨/protocol⟩
```

Le protocole P-RA (pour "Propose-Refuse ou Accept") est disponible pour les agents sous le format suivant :

```
⟨protocol prname="P_RA" initiator="X" participator="Y" goal="IxSCyDone(Z)"⟩
        ⟨body⟩
                ⟨compound_component⟩
                                ⟨trans_name⟩propose⟨/trans_name⟩
                                ⟨operator⟩⟨and /⟩⟨/operator⟩
                                ⟨compound_component⟩
                                ⟨trans_name⟩refuse⟨/trans_name⟩
                                ⟨operator⟩⟨xor /⟩⟨/operator⟩
                                ⟨trans_name⟩accept⟨/trans_name⟩
                                ⟨/compound_component⟩
                ⟨/compound_component⟩
        ⟨/body⟩
        ⟨transitions⟩
```

```
                ⟨transition trans_name="propose" target_stname="P" tcondition="TRUE"⟩
                                        ⟨message⟩
                                        "time= SR in [t_now,t_now+00 :01 :00]"
                                        ⟨/message⟩
                ⟨/transition⟩
                ⟨transition trans_name="refuse" stname="P.R" tcondition="TRUE"⟩
                                        ⟨message⟩
                                        ⟨/message⟩
                ⟨/transition⟩
                ⟨transition trans_name="accept" stname="P.A"
                                        tcondition="task_duration<=08 :00 :00"⟩
                                        ⟨message⟩
                                        ⟨/message⟩
                ⟨/transition⟩
        ⟨/transitions⟩
⟨/skeleton⟩
```

D.2.2 Exemple de définition d'une structure organisationnelle temporelle au format XML

```
⟨TOS TOS_name="contract01" tp=" " tc=" "⟩
    ⟨roles⟩
        ⟨role role_name = "client" tp=" " tc=" "⟩
            ⟨mission mission_name="propose_contract" tp=" " tc=" "⟩
            ⟨/mission⟩
            ⟨mission mission_name="proposition evaluation" tp=" " tc=" "⟩
            ⟨/mission⟩
        ⟨/role⟩
        ⟨role role_name="decision_maker" tp=" " tc=" "⟩
            ⟨mission mission_name="evaluate contract" tp=" " tc=" "⟩
            ⟨/mission⟩
            ⟨mission mission_name="mission2" tp=" " tc=" "⟩
            ⟨/mission⟩
        ⟨/role⟩
        ⟨role role_name="planning_person_in_charge" tp=" " tc=" "⟩
            ⟨mission mission_name="schedule" tp=" " tc=" "⟩
            ⟨/mission⟩
            ⟨mission mission_name="elaborate planning" tp=" " tc=" "⟩
            ⟨/mission⟩
        ⟨/role⟩
        ⟨/roles⟩
    ⟨links⟩
        ⟨link link_name="Lcdm" link_type="authority" source_role="client"
            target_role="decision_maker" tp=" " tc=" "⟩
```

```
            ⟨/link⟩
            ⟨link link_name="Lcp" link_type="authority" source_role="client"
                target_role="planning_person_in_charge" tp=" " tc=" "⟩
            ⟨/link⟩
            ⟨link link_name="Ldmp"
                link_type="authority" source_role="decision_maker"
                target_role="planning_person_in_charge" tp=" " tc=" "⟩
            ⟨/link⟩
        ⟨/links⟩
        ⟨groups⟩
            ⟨group group_name="contract01group" tp=" " tc=" "⟩
                ⟨role_list⟩
                        ⟨role_name⟩client
                        ⟨/role_name⟩
                        ⟨role_name⟩decision_maker
                        ⟨/role_name⟩
                ⟨/role_list⟩
            ⟨/group⟩
            ⟨group group_name="contract02temporalgroup" tp=" " tc=" "⟩
                ⟨role_list⟩
                        ⟨role_name⟩planning_person_in_charge
                        ⟨/role_name⟩
                ⟨/role_list⟩
            ⟨/group⟩
        ⟨/groups⟩
⟨/TOS⟩
```

Bibliographie

[ABS00] M. Allouche, O. Boissier, and C. Sayettat. Temporal social reasoning in dynamic multi-agent systems. In *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, Boston, 2000.

[AGJ+94] J. A. Alty, D. Griffiths, N. R. Jennings, E. H. Mamdani, A. Struthers, and M. E. Wiegand. ADEPT : Advanced decision environment for process tasks : Overview & architecture. In *Proceedings of the 1994 BCS Expert Systems Conference (Applications Track, ISIP Theme)*, pages 359–371, Cambridge, UK, 1994.

[All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26, 1983.

[All84] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23 :123–154, 1984.

[All98] M. K. Allouche. Une société d'agents temporels pour la supervision de systèmes industriels. Thèse de Doctorat, ENS Mines Saint-Etienne/Université de Saint-Etienne, 1998.

[AS95] M.-K. Allouche and C. Sayettat. A multi-agent architecture for supervising dynamic systems. In *Proceedings of the first International Workshop on Decentralized Intelligent and Multi-Agent Systems (DIMAS)*, pages II/1–II/8, Krakow, Poland, 1995.

[ASF+95] Allen, J. F., Shubert, L. K., Ferguson, G., Heeman, P.and Hwang, H., Kato, T., Light, M., Martin, N., Miller, B., Poesio, M., and Traum, D. The trains project : a case study in building a conversational planning agent. *Journal of Experimental and Theoretical Artificial Intelligence*, (7) :7–48, 1995.

[Aus62] J.L. Austin. *How to do things with words.* 1962.

[AVSHR90] D. Ash, A. Vina, A. Seiver, and B. Hayes-Roth. Action-oriented diagnosis under real-time constraints. Technical Report Report No. KSL 90-39, Stanford University, Knowledge Systems Laboratory, Department of Computer Science, , Stanford, California 94305., June 1990.

[BB81] T. Ballmer and W. Brennenstuhl. *Speech act classification.* Ballmer, T., and Brennenstuhl, W. Speech act classification. SpringerVerlag, 1981. References Page 26, 1981.

[BBM97] A.C. Boury-Brisset and B. Moulin. Un modèle d'agent intégrant des mécanismes de raisonnement multiples. In *Revue d'Intelligence Artificielle*, volume 11, pages 73–107. Hermès France, 1997.

[BCBK99] C. Bourjot, V. Chevrier, A. Bernard, and B. Krafft. Coordination par le biais de l'environnement : une approche biologique. In M-P. Gleizes et P. Marcenac, editor, *JFIADSMA'99*, Ingénierie des systèmes multi-agents, pages 237–250, La Réunion, 1999.

[BD94] O. Boissier and Y. Demazeau. Asic : an architecture for social and individual control and its application to computer vision. In *Lectures Notes in Artificial Intelligence*, volume 1069, pages 135–149, Odense, Danmark, aug 1994. MAAMAW Workshop, Springer-Verlag.

[BDa01] Briot, J.-P., Demazeau, Y., and al. *Les systèmes multi-agents*. Hermes Science Publications, Paris, France, collection ic2 edition, 2001.

[BDSF97] J. Christopher Beck, Andrew J. Davenport, Edward M. Sitarski, and Mark S. Fox. Texture-based heuristics for scheduling revisited. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 241–248, Providence, Rhode Island, 1997. AAAI Press / MIT Press.

[Bea57] C. Beaudelaire. Le goût du néant, 1857. "(...) Et le Temps m'engloutit minute par minute, comme la neige immense un corps pris de roideur ; (...)".

[Ber07] Henri Bergson. L'évolution créatrice, 1907. La pensé et le mouvant, La perception du changement.

[BF95] Mihai Barbuceanu and Mark S. Fox. COOL : A language for describing coordination in multiagent systems. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, pages 17–24, San Francisco, CA, 1995. MIT Press.

[BF96] M. Barbuceanu and M. S. Fox. The architecture of an agent building shell. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI 1037)*, pages 235–251. Springer-Verlag : Heidelberg, Germany, 1996.

[BFG+89] H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens. METATEM : A framework for programming in temporal logic. In *REX Workshop on Stepwise Refinement of Distributed Systems : Models, Formalisms, Correctness (LNCS Volume 430)*, pages 94–129. Springer-Verlag : Heidelberg, Germany, June 1989.

[BG88] A. H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers : San Mateo, CA, 1988.

[BHS93] B. Burmeister, A. Haddadi, and K. Sundermeyer. Generic configurable cooperation protocols for multi-agents systems. In *4rth Modeling Autonomous Agent in Multi-Agent World Workshop*, Neuchatel, August 1993.

[BIP88] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4 :349–355, 1988.

[BMO01] B. Bauer, Müller, J. P., and J. Odell. Agent uml : A formalism for specifying multiagent interaction. In Paolo Ciancarini and Michael Wooldridge eds, editors, *Agent Oriented Software Engineering (International Conference on Software Engineering (ICSE))*, pages 91–103. Springer-Verlag : Heidelberg, Germany, 2001.

[Boi01] O. Boissier. *Systèmes Multi-Agents*, chapter 4 : Modèles et architectures d'agents. Hermès, 2001.

[Bra87] M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press : Cambridge, MA, 1987.

[Bra93] C. Brassac. Théorie des actes de langage et intelligence artificielle distribuée. In *Journées SMA du PRC-GDR-IA*, décembre 1993.

[Ca99] Ye Chen and al. A negotiation-based multi-agent system for supply chain management. In *Workshop on Agent based Decision-Support for Managing the Internet-Enabled Supply-Chain*, Seattle, May 1999. Third Conference on Autonomous Agents (Agents-99).

[Car98] Thibault Carron. Un langage d'interaction temporel pour un système multi-agent ouvert, dynamique et décentralisé. Rapport de DEA, Ecole Nationale Supérieure des Mines de Saint-Etienne/Université de Savoie, 1998.

[Cas90] C. Castelfranchi. Social power. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 49–62. Elsevier Science Publishers B.V. : Amsterdam, The Netherlands, 1990.

[CB01] T. Carron and O. Boissier. Towards a temporal organizational structure language for dynamics multi-agents systems. In Y. Demazeau and F. J. Garijo, editors, *Proceedings of the 10th European Workshop on Multi-Agent Systems, MAAMAW'01, LEIBNIZ, (electronic publishing)*, 2001.

[CBS02] Thibault Carron, Olivier Boissier, and Claudette Sayettat. Le temps dans les systèmes multi-agents. In A. Peninou R. Mandiau, E. Grislin-Le-Strugeon, editor, *Organisation et applications des SMA*, chapter 12, pages 133–166. Hermès, 2002.

[CD90] J. A. Campbell and M. P. D'Inverno. Knowledge interchange protocols. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 63–80. Elsevier Science Publishers B.V. : Amsterdam, The Netherlands, 1990.

[CdV98] B. Chaib-draa and D. Vanderveken. Agent communication language : A semantics based on the success, satisfaction and recursion. In *Proc. Agent Theories, Archit. and Lang.*, 1998.

[CL90a] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42 :213–261, 1990.

[CL90b] P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[CL90c] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3) :213–261, 1990.

[CMC92a] Castelfranchi C., M. Micelli, and A. Cesta. Dependence relations among autonomous agents. In éd. E. Werner et Y. Demazeau, editor, *Decentralized A. I.*, volume 3, pages pp. 215–227. Elseiver Science Publishers B. V., Amsterdam, NL, 1992.

[CMC92b] C. Castelfranchi, M. Miceli, and A. Cesta. Dependence relations among autonomous agents. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)*, pages 215–231. Elsevier Science Publishers B.V. : Amsterdam, The Netherlands, 1992.

[CPB99] T. Carron, H. Proton, and O. Boissier. A temporal agent communication language for dynamics multi-agents systems. In F. J. Garijo and M. Boman, editors, *Multi-Agent System Engineering (LNAI 1647)*, pages 219–234. Springer-Verlag : Heidelberg, Germany, 1999.

[Dec91] Keith S. Decker. Blackboard systems. *IEEE Expert*, 6(5) :71–72, October 1991.

[Dec96] Keith S. Decker. Tæms : A framework for environment centred analysis and design of coordination mechanisms. In Greg O'Hare and Nick Jennings, editors, *Foundations of Distributed Artificial Intelligence*, chapter 16. John Wiley and Sons, 1996.

[Dem95] Y. Demazeau. From interactions to collective behaviour in multi-agent systems. In *1st european conference on Cognitive Science, St Malo (France)*, mars 1995.

[Dem01] Y. Demazeau. Voyelles, 2001. HDR Spécialité informatique, Institut Nationale polytechnique de Grenoble.

[DF94] A. Drogoul and J. Ferber. Multi-agent simulation as a tool for modeling societies : Application to social differentiation in ant colonies. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems — Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-92 (LNAI Volume 830)*, pages 3–23. Springer-Verlag : Heidelberg, Germany, 1994.

[DGG93] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition. In *Proceedings of the 13th international joint conference on AI*, volume 1, pages 166–172, Chambéry-France, 1993.

[DHL89] Keith S. Decker, Marty A. Humphrey, and Victor R. Lesser. Experimenting with control in the DVMT. In *Proceedings of the Third Annual AAAI Workshop on Blackboard Systems*, Detroit, August 1989. Also COINS TR-89-85.

[DK95] A. Duda and C. Keramane. Structured Temporal Composition of Multimedia Data. In *Proc. International Workshop on Multi-media Database Management Systems*, Blue Mountain Lake, NY, August 1995.

[DK97] F. Dignum and R. Kuiper. Combining dynamic deontic logic and temporal logic for the specification of deadlines. In Jr. R. Sprague, editor, *Proceedings of thirtieth HICSS*, Wailea, Hawaii, 1997.

[DL87] E. H. Durfee and V. R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, Milan, Italy, 1987.

[DL91] Edmund H. Durfee and Victor R. Lesser. Partial global planning : A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5), September 1991. (Special Issue on Distributed Sensor Networks).

[DL93] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management*, 2(4) :215–234, December 1993. Special issue on "Mathematical and Computational Models of Organizations : Models and Characteristics of Agent Behavior".

[DL94] Keith S. Decker and Victor R. Lesser. Task environment centered design of organizations. In Ingemar Hulthage, editor, *Computational Organization Design*. AAAI Spring Symposium, 1994. Working Notes.

[DL95] K. Decker and V. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 73–80, San Francisco, CA, June 1995.

[DLC89] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1), March 1989.

[Doj94] M. Dojat. Contribution a la représentation d'expertises médicales dynamiques : application en réanimation médicale. Thèse de Doctorat, Université de Compiegne, 1994.

[Dou94] C. Dousson. Suivi d'évolutions et reconnaissance de chroniques, September 1994. Thèse de Doctorat.

[Dra98] T. Drakengren. Eight maximal tractable subclasses of allen's algebra with metric time. *journal of Artificial Intelligence Research*, 1998.

[Dro93] A. Drogoul. De la simulation multi-agents à la résolution collective de problèmes. Thèse de Doctorat, University Paris 6, Nov 1993.

[DW95] F. Dignum and H. Weiganddans. Modelling communication between cooperative systems. In J. Iivari and al.(eds), editors, *Proceedings of CAISE'95*, number LNCS-932, pages 140–153. Springer-Verlag, Berlin, 1995.

[Dzi90] D. Dzierzgowski. Quatre exemples de langages ou environnements pour le développement de programmes où le temps intervient. In *Techniques et Sciences Informatiques*, volume 9. 1990.

[EAdC89] P. Enjalbert E. Audureau and L. Fariñas del Cerro. Logique temporelle. In Masson, editor, *études et recherches en informatique.* Paris, 1989.

[Ein21] A. Einstein. Théorie de la relativité, 1921. "Un être humain est une partie du tout que nous appelons "Univers" (...) Une partie limité dans le Temps et dans l'Espace".

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier Science Publishers B.V. : Amsterdam, The Netherlands, 1990.

[Far00] P. Faratin. *Automated Service Negotiation Between Autonomous Computational Agents.* PhD thesis, University of London, Queen Mary College, 2000.

[Fer87] J. Ferber. Des objets aux agents : une architecture stratifi'ee. In Dunod, editor, *6ème Congrès Afcet de Reconnaissance des Formes et Intelligence Artificielle (RFIA'87)*, volume 1, pages 275–286, nov 1987.

[Fer94] Jacques Ferber. Coopération réactive et émergence. *Intellectica*, 19 :19–52, 1994.

[Fer95] J. Ferber. *Les systèmes multi-agents.* InterEditions, 1995.

[Fer97] N. Ferrand. Modèles multi-agents pour l'aide à la décision et la négociation en aménagement du territoire. Thèse de Doctorat, Université J. Fourier (Grenoble), 1997.

[FFR96] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server : a tool for collaborative ontology construction. Knowledge Systems Laboratory, Stanford University, 1996. http ://ontolingua.stanford.edu/.

[FG96] S. Franklin and A. Graesser. It is an agent, or just a program ? : A taxonomy for autonomous agents. In Wooldridge M. J. M¸ller J.P. and Jennings N. R., editors, *Intelligent Agents III : Theories, Architectures, and Languages (LNAI Volume 1193)*, pages 21–35. Springer-Verlag : Heidelberg, Germany, 1996.

[FGJ+00] Ferber, J., Gutknecht, O., Jonkeur, C. M., Müller, J. P., and Treur, J. Organizational models and behavioural requirements specifications for multi-agents systems. In *ECAI2000 Workshop on Modelling Artificial Societies and Hybrid Organizations*, unpublished, 2000.

[FIP97] FIPA. Agent communication language. Technical report, Foundation for Intelligence Physical Agent, http ://drogo.cselt.stet.it/fipa, 1997.

[FIP00] FIPA. Fipa communicative act library specification. Technical report, Foundation for Intelligence Physical Agent, 2000. http ://www.fipa.org.

[Fis94] M. Fisher. A survey of Concurrent METATEM — the language and its applications. In D. M. Gabbay and H. J. Ohlbach, editors, *Temporal Logic — Proceedings of the First International Conference (LNAI Volume 827)*, pages 480–505. Springer-Verlag : Heidelberg, Germany, July 1994.

[Gab96] P. Gaborit. Planification distribuée pour la coopération multi-agents. Thèse de Doctorat, Université Paul Sabatier (Toulouse), 1996.

[Gau99] Franck Gaultier. Raisonnement temporel dans les systèmes à agents. Rapport de DEA, Ecole Nationale Supérieure des Mines de Saint-Etienne/Université de Savoie, 1999.

[Geo83] Michael Georgeff. Communication and interaction in multi-agent planning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 125–129, Washington, D.C., August 1983. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 200–204, Morgan Kaufmann, 1988.).

[GGT00] P. Gochet, P. Gribonnot, and A. Thayse. *Logique 3 : méthodes pour l'intelligence artificielle.* Hermès Science Publications, 2000.

[GHJ+86] Alan Garvey, Michael Hewett, M. Vaughan Johnson, Robert Schulman, and Barbara Hayes-Roth. *BB1 User Manual.* Knowledge Systems Laboratory, Departments of Medical and Computer Science, Stanford, CA 94305, Common Lisp edition, October 1986. (Published as Working Paper KSL 86-61, Knowledge Systems Laboratory, Departments of Medical and Computer Science, Stanford University, Stanford, California 94305.).

[GK93] Barbara Grosz and Sarit Kraus. Collaborative plans for group activities. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambéry, France, August 1993.

[GL87] M.P. Georgeff and A.L. Lansky. Reactive reasoning and planning. In *AAAI Conference*, pages 677–682, 1987.

[Gra59] P. Grassé. *La reconstruction du nid et les coordinations interindividuelles chez Bellicostermes natalensis et Cubitermis sp. La théorie de la stigmergie : Essai d'interprétation du comportement des termites constructeurs.*, chapter 6, pages 41–81. Insectes Sociaux, 1959.

[Gro93] Eckehard Gross. Mtmm : Modified version of dean and mcdermott's tmm written in mcl. Available on diskette, 1993.

[GSS93] Alfonso Gerevini, Lenhart K. Schubert, and Stephanie Schaeffer. Temporal reasoning in timegraph i-II. *SIGART Bulletin*, 4(3) :21–25, 1993.

[GSS94] Alfonso Gerevini, Lenhart Schubert, and Stephanie Schaeffer. The temporal reasoning systems timegraph I-II. Technical Report TR494, 1994.

[Han02] M. Hannoun. Moise : un modèle organisationnelle pour les systèmes multi-agents. Thèse de Doctorat, ENS Mines de Saint-Etienne / Université de Saint-Etienne, 2002. à paraître.

[Hay95] P. J. Hayes. A catalog of temporal theories. Technical report, University of Illinois, 1995.

[HBSS00] Hannoun, M., Boissier, O., Sichman, J.S., and Sayettat, C. Moise : An organizational model for multi-agent systems. In *Proceedings*, Brazil, 2000. SBIA.

[Hei27] Martin Heidegger. Sein und zeit, 1927. Etre et temps : trad. E. Martineau, Authentica, 1985.

[Her50] Heraclite. De l'univers, 550. Fragments : "panta reï" fragment 59, 550-480 av.J-C.

[Het91] J.P. Heton. Le raisonnement en intelligence artificielle. Intereditions, 1991.

[Hew77] C. Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(3) :323–364, 1977.

[HJF95] J. Huang, N. R. Jennings, and J. Fox. An agent architecture for distributed medical care. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents : Theories, Architectures, and Languages (LNAI Volume 890)*, pages 219–232. Springer-Verlag : Heidelberg, Germany, January 1995.

[HLS96] Thomas Haynes, Kit Lau, and Sandip Sen. Learning cases to compliment rules for conflict resolution in multiagent systems. In Sandip Sen, editor, *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, pages 51–56, Stanford University, CA, March 1996.

[Hr84] B. Hayes-roth. BB1 : An architecture for blackboard systems that control, explain, and learn about their own behavior. HPP 84-16, Computer Science Departement, Stanford University, December 1984.

[HR85] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26, 1985.

[HR90] B. Hayes-Roth. Architectural foundations for real-time performance in intelligent agents. *The Journal of Real-Time Systems*, 2 :99–125, 1990.

[HRWA+92] B. Hayes-Roth, R. Washington, D. Ash, A. Collinot, A. Vina, and A. Seiver. Guardian : A prototype intensive-care monitoring agent, 1992.

[HRWHH89] Barbara Hayes-Roth, Richard Washington, Rattikorn Hewett, and Micheal Hewett. Intelligent monitoring and control. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 250–255, Detroit, August 1989.

[Jen93] N. R. Jennings. Commitments and conventions : The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3) :223–250, 1993.

[JSW98] Jennings, N.R., Sycara, K. P., and Wooldridge, M. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1 :7–36, 1998.

[JTd01] Jonker, C. M., Treur, J., and de Vries, W. Temporal requirements for anticipatory reasoning about intentional dynamics in social contexts. In Y. Demazeau and F. J. Garijo, editors, *Proceedings of the 10th European Workshop on Multi-Agent Systems, MAAMAW'01, LEIBNIZ, (electronic publishing)*, 2001.

[Kan81] Emmanuel Kant. Critique de la raison pure, 1781. Ed. PUF p.63-64, 1944.

[KL91] H. Krautz and P. Ladkin. Mats : Metric/allen time system, 1991.

[KLR+92] D. Kinny, M. Ljungberg, A. S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems — Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-92 (LNAI Volume 830)*, pages 226–256. Springer-Verlag : Heidelberg, Germany, 1992.

[Kon84] K. Konolige. A deductive model of belief. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 377–381, August 1984.

[KS86] R. Kowalski and M. Sergot. A logic-based calculus of events. In *New Generation Computing*, volume 4, pages 67–95. Ohmsha, ltd and springer-verlag edition, 1986.

[Lab93] Enterprise Integration Laboratory. Tove : Ontologies. Department of Industrial Engineering, 1993. (EIL), http ://www.eil.utoronto.ca/tove/toveont.html.

[Lam49] Alphonse (de) Lamartine. Le lac, 1849. "Ô temps ! Suspends ton vol, et vous, heures propices ! Suspendez votre cours : Laissez-nous savourer les rapides délices des plus beaux de nos jours ! (...)".

[Lam78] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communication of the ACM*, jul 1978.

[Lay62] N. Layaïda. Maintaining temporal consistency of multimedia documents. *RTMW'96*, 19962.

[LB89] Jean (de) La Bruyère. Les caractères, 1689. "Ceux qui emploient mal leur temps sont les premiers á se plaindre de sa brièveté (...), ceux au contraire qui en font un meilleur usage en ont de reste".

[LC83] Victor R. Lesser and Daniel D. Corkill. The Distributed Vehicle Monitoring Testbed : A tool for investigating distributed problem solving networks. *AI Magazine*, 4(3) :15–33, Fall 1983. (Also published in *Blackboard Systems*, Robert S. Engelmore and Anthony Morgan, editors, pages 353–386, Addison-Wesley, 1988 and in *Readings from AI Magazine : Volumes 1–5*, Robert Engelmore, editor, pages 69–85, AAAI, Menlo Park, California, 1988).

[LC87] V. R. Lesser and D. Corkill. DVMT A tool for investigation of distributed problem solving networks. In M. N. Huhns, editor, *Distributed Artificial Intelligence*. Morgan Kaufmann, 1987.

[LC00] P. De Loor and P. Chevaillier. Generation of agent interactions from temporal logic specifications. In M. Deville and R. Owens, editors, *16th IMACS World Congress*, August 2000.

[LCd97] S. Lizotte and B. Chaib-draa. *Concurrent Engineering : Research and Applications*, volume 4, chapter Coordination in CE Systems : An Approach Based on the Management of Dependencies Between Agents, pages 367–377. CERA, 1997.

[Ld95] M. Luck and M. d'Inverno. A formal framework for agency and autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 254–260, San Francisco, CA, June 1995.

[LF94] Y. Labrou and T. Finin. A semantic approach for KQML- a general purpose communication langage for software agents. In *CIKM'94*. ACM Press, 1994.

[Liz88] S. Lizotte. Système de planification et simulation de comportements. mémoire de maîtrise en intelligence artificielle, mars 1988. Université de Laval.

[LM90] M. Lizotte and B. Moulin. A temporal planner for modelling autonomous agents. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 121–136. Elsevier Science Publishers B.V. : Amsterdam, The Netherlands, 1990.

[Lon89] Derek Long. A review of temporal logics. *The Knowledge Engineering Review*, 4(2) :141–162, 1989.

[LS92] Fangzen Lin and Yoav Shoham. Concurrent actions in the situation calculus. In William Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 590–595, San Jose, CA, jul 1992. MIT Press.

[LSI96] N. Layaïda and L. Sabry-Ismail. Maintaining Temporal Consistency of Multimedia Documents Using Constraint Networks. In *Multimedia Computing and Networking*, pages 124–135, CA, USA, 1996.

[Mag96] L. Magnin. Modélisation de l'environnement dans les systèmes multi-agents (applications aux robots footballeurs). Thèse de Doctorat, Université de Paris 6, 1996.

[MB01] Meurisse, T. and Briot, J. P. Une approche à base de composants pour la conception d'agents. In Michel Dao et Christophe Dony, editor, *Technique et Science Informatiques (TSI)*, volume 20 of *Numéro Spécial : Réutilisation,*, pages 583–602. Hermès, Paris, April 2001.

[MC91] Thomas W. Malone and Kevin Crowston. Toward an interdisciplinary theory of coordination. Center for Coordination Science Technical Report CCS-TR-120, MIT Sloan School of Management, 1991.

[McD82] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6, 1982.

[MCd96] Bernard Moulin and B. Chaib-draa. An overview of distributed artificial intelligence. In Greg O'Hare and Nick Jennings, editors, *Foundations of Distributed Artificial Intelligence*, chapter 1. John Wiley and Sons, 1996.

[MCHR97] Silvia Miksch, Kenneth Cheng, and Barbara Hayes-Roth. An intelligent assistant for patient health care. In W. Lewis Johnson, editor, *Proceedings of the First International Conference on Autonomous Agents*, New York, 1997. ACM Press.

[MG89] A. Mounir-Alaoui M. Ghallab. Relations temporelles symboliques : Représentations et algorithmes. In *Revue d'Intelligence Artificielle*, volume 3, pages 67–116. 1989.

[MH69] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*. Edinburgh University Press, 1969.

[Mil01] Dave Mills. Network time protocol (ntp). Technical report, NBS Special Publication 432, 2001. Public NTP Time Servers : http ://www.eecis.udel.edu/~mills/ntp.htm.

[MLF96] J. Mayfield, Y. Labrou, and T. Finin. Evaluating KQML as an agent communication language. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI 1037)*, pages 347–360. Springer-Verlag : Heidelberg, Germany, 1996.

[Mou92] B. Moulin. A conceptual graph approach for representing temporal information in discourse,. *Knowledge-Based Systems Journal*, 5(3) :183–192, September 1992.

[MS88] L. Morgenstern and L. Stein. Why things go wrong : A formal theory of causal reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, Philadelphia, PA, 1988.

[New87] I. Newton. Philosophiæ naturalis principia mathematica, 1687. "Le temps absolu, vrai et mathématique, en lui-même et de sa propre nature, coule uniformément sans relation à rien d'extérieur".

[NJFM97] Timothy Norman, Nick Jennings, Peyman Faratin, and Abe Mamdani. Designing and implementing a multi-agent architecture for business process management. In Jörg P. Müller, Michael J. Wooldridge, and Nicholas R. Jennings, editors, *Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures, and Languages : Intelligent Agents III*, volume 1193, pages 261–276. Springer-Verlag : Heidelberg, Germany, 12–13 1997.

[NL95] T.J. Norman and D.P. Long. Alarms : Heuristics for the control of reasoning attention. In RN, editor, *Seventeenth Annual Conference of the Cognitive Science Society.*, volume 95, Pittsburgh, July 1995.

[NL96a] T. J. Norman and D. Long. Alarms : An implementation of motivated agency. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI 1037)*, pages 219–234. Springer-Verlag : Heidelberg, Germany, 1996.

[NL96b] T. J. Norman and D. Long. Alarms : An implementation of motivated agency. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI 1037)*, pages 219–234. Springer-Verlag : Heidelberg, Germany, 1996.

[NNLC99] Hyacinth S Nwana, Divine T. Ndumu, Lyndon C. Lee, and Jaron C. Collis. ZEUS : a toolkit and approach for building distributed multi-agent systems. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 360–361, Seattle, WA, USA, 1999. ACM Press.

[PAC96] Jeremy Pitt, Matthew Anderton, and Jim Cunningham. Normalized interactions between autonomous agents a case study in inter-organizational project management. *Computer Supported Cooperative Work*, 5(2/3) :201–222, 1996.

[Pat96] F. Paterno. Temporal aspects of usability including time in the notion of interactor. *SIGCHI Bulletin vol.28 Number 2*, apr 1996.

[PCL87] H. E. Pattison, D. D. Corkill, and V. R. Lesser. Instantiating descriptions of organizational structures. In M. Huhns, editor, *Distributed Artificial Intelligence*, pages 59–96. Pitman Publishing : London and Morgan Kaufmann : San Mateo, CA, 1987.

[PGS90] A. Potet P. Gaborit and C. Sayettat. Semantics and validation procedures of a multi-modal logic for formalization of multi-agent universes. Stockholm, Sweden, 1990.

[PHR87] William J. Pardee and Barbara Hayes-Roth. Intelligent real-time control of material processing. Technical Report 1, Rockwell International Science Center, Palo Alto, CA, February 1987.

[Pla60] Platon. Timée, 360. 37 d.-38 a., vers 360 av.J-C.

[Pol86] M. E. Pollack. A model of plan inference that distinguishes between the beliefs of actors and observers. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions & Plans — Proceedings of the 1986 Workshop*, pages 279–296. Morgan Kaufmann Publishers : San Mateo, CA, 1986.

[Pro13] M. Proust. A la recherche du temps perdu, 1913. "(...) Ce que nous croyons notre amour, notre jalousie, n'est pas une même passion, continue, indivisible. Ils se composent d'une infinité d'amours successifs, de jalousies différentes et qui sont éphémères, mais par leur multitude ininterrompue donnent l'impression de la continuité, l'illusion de la durée.".

[PSRC98] H. V. D. Parunak, R. Savit, R. L. Riolo, and S. Clark. Dynamic analysis of supply chains. Technical report, Center for Electronic Commerce, ERIM, 1998. Available at http ://www.erim.org/cec/projects/dasch.htm.

[rda94] *NetBSD, System Manager's Manual*, April 1994. http ://www.netBSD.org.

[Reg93] T. Regan. Multimedia in temporal lotos : A lip synchronization algorithm. In proceedings of PSTV XIII, 13th Protocol Specification, Testing and Verification, 1993. North Holland.

[RG85] J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 91–99, Los Angeles, CA, 1985.

[RG95a] A. S. Rao and M. Georgeff. BDI Agents : from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319, San Francisco, CA, June 1995.

[RG95b] Anand S. Rao and Michael P. Georgeff. BDI agents : from theory to practice. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, pages 312–319, San Francisco, CA, 1995. MIT Press.

[Ric90] J. F. Richard. *Les activités mentales : comprendre, raisonner, trouver des solutions.* Armand Colin, 1990.

[Ric01] P.-M. Ricordel. Multi-agent oriented programming. Thèse de Doctorat, INP Grenoble, Laboratoire Leibniz, October 2001. in french.

[Rou95] D. Rousseau. Modélisation et simulation de conversations dans un univers multi-agent. Thèse de Doctorat, Université de Montréal, 1995.

[SA97] Saint-Augustin. Confessions, 396-397. XI, 14, 17.

[Sad91] D. Sadek. Attitudes mentales et interaction rationnelle : vers une théorie formelle de la communication. Thèse de Doctorat, Université de Rennes I, 1991.

[Say90] C. Sayettat. Contributions à la représentation du changement, 1990. HDR Spécialité informatique, Université de Nantes.

[Sch98] Axel T. Schreiner. Distributed objects. http ://www.cs.rit.edu/~ats/java/html/skript/6_3.htmld/, May 1998.

[SD95] J. Sichman and Y. Demazeau. Exploiting social reasoning to deal with agency level inconsistency. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 352–359, San Francisco, CA, June 1995.

[Sea69a] J. R. Searle. *Speech Acts : An Essay in the Philosophy of Language.* Cambridge University Press : Cambridge, England, 1969.

[Sea69b] J. R. Searle. *Speech Acts : An Essay in the Philosophy of Language.* Cambridge University Press, Cambridge, 1969.

[Sea72] J.R. Searle. *Les actes de langage.* 1972.

[Sea79] J. R. Searle. *Expression and Meaning.* Cambridge University Press : Cambridge, England, 1979.

[Sho86] Y. Shoham. What is the frame problem? In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions & Plans — Proceedings of the 1986 Workshop*, pages 83–98. Morgan Kaufmann Publishers : San Mateo, CA, 1986.

[Sho88] Y. Shoham. *Reasoning About Change : Time and Causation from the Standpoint of Artificial Intelligence*. The MIT Press : Cambridge, MA, 1988.

[Sho90] Y. Shoham. Agent Oriented Programming. Technical Report STAN-CS-90-1335, Robotics Laboratory, Comp. Science Dpt, Stanford, 1990.

[Sin94] M. P. Singh. *Multiagent Systems : A Theoretical Framework for Intentions, Know-How, and Communications (LNAI Volume 799)*. Springer-Verlag : Heidelberg, Germany, 1994.

[SJNP98] C. Sierra, N. R. Jennings, P. Noriega, and S. Parsons. A framework for argumentation-based negotiation. *Lecture Notes in Computer Science*, 1365 :177–??, 1998.

[SM88] Y. Shoham and D. McDermott. Problems in formal temporal reasoning. *Artificial Intelligence*, 36, 1988.

[Smi80] R. G. Smith. The contract net protocol. *IEEE Transactions on Computers*, C-29(12), 1980.

[SV85] John R. Searle and Daniel Vandervecken. *Foundations of Illocutionary Logic*. Cambridge University Press, Cambridge, England, 1985.

[TA91] D. Traum and J. Allen. Causative forces in multi-agent planning. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-90)*, pages 89–108. Elsevier Science Publishers B.V. : Amsterdam, The Netherlands, 1991.

[Tra96] David R. Traum. A reactive-deliberative model of dialogue agency. In *Agent Theories, Architectures, and Languages*, pages 157–171, 1996.

[Tsa87] E. P. K. Tsang. Tpl- a temporal planner. In *Artificial Intelligence*. avril 1987.

[Van88] D. Vanderveken. *Les actes de discours*. Pierre Mardaga, 1988.

[VB83] J. F. A. K. Van Bethem. The logic of time. Reidel, 1983.

[Ver80] S. A. Vere. Multilevel counterfactuals for generalizations of relational concepts and productions. *Art. Int.*, 14 :139–164, 1980.

[Wa98] G. Weiss and al. *Multiagent Systems : A modern Introduction to Distributed Artificial Intelligence*. MIT PRESS, 1998.

[Wei93] G. Weiß. Learning to coordinate actions in multi-agent systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 311–316, Chambéry, France, 1993.

[Wei98] C. Weidenbach. The spass and flotter users guide. Technical report, Max-Planck-Institut für Informatik, 1998.

[WSXL99] T. Wagner, J. Shapiro, P. Xuan, and V. Lesser. Multi-level conflict in multi-agent systems. Technical Report UM-CS-1999-017, University of Massachusetts, Amherst, April 1999.

[WW98] Misha Wolf and Charles Wicksteed. Date and time formats. W3C, 1998. This document defines a profile of ISO 8601, the International Standard for the representation of dates and times. ISO 8601 describes a large number of date/time formats. To reduce the scope for error and the complexity of software, it is useful to restrict the supported formats to a small number. This profile defines a few date/time formats, likely to satisfy most requirements.

[ZR89] G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 912–917, Detroit, MI, 1989.

Résumé : Ce mémoire de thèse s'intéresse à la prise en compte de la dimension temporelle au sein des systèmes multi-agents (SMA) dans le contexte de systèmes industriels dynamiques. L'étude s'appuie sur l'approche *Voyelles* afin d'étudier l'impact du temps au sein de chacune des dimensions d'un SMA que cette approche distingue et afin de mettre en place un raisonnement temporel orienté multi-agents. Après avoir analysé les conséquences de la prise en compte du temps dans chacun de ses aspects, un ensemble de méthodes et d'outils sont proposés pour résoudre les problèmes mis en évidence. Le modèle de SMA temporel proposé présente notamment une prise en compte des aspects temporels au sein de l'interaction et de l'organisation ainsi qu'un modèle d'agent permettant de les intégrer. Ces propositions sont ensuite confrontées à des applications concrètes issues du milieu industriel.

Mots-clés : Systèmes Multi-Agents, Intelligence Artificielle Distribuée, Raisonnement Temporel, Interactions temporelles, Organisations temporelles.

Abstract : With industrial systems growing more and more complex, we are led to take explicitly into account the time dimension. In the industrial dynamic systems field, we have to consider the time aspect in the multi-agents systems.
To do so, we lay stress upon the "vowels" approach in order to study the time dimension within each of the MAS dimensions such approach reveals. The current methods and tools don't allow us to take the general time aspect into consideration in a MAS. Once analysed the consequences of the consideration of time in each of its aspects, we suggest a set of methods and tools likely to solve the problems which appear and to implement a multi-agents-driven time reasoning. Next, these propositions have to meet the concrete applications from the industrial environment.

Keywords : Multi-Agent Systems, Distributed Artificial Intelligence, Temporal Reasoning, Temporal Interactions, Temporal Organizations.