



RAPPORT ASA
Année 2002
Le problème de l'emploi du temps

Groupe ASA, PRC IA
Mise en forme : P Mathieu

Table des matières

| | | |
|----------|---|-----------|
| 1 | Enoncé du problème | 7 |
| 1.1 | Introduction | 7 |
| 1.2 | Présentation générale du problème | 7 |
| 1.3 | Scénarii proposés | 8 |
| 1.3.1 | Variante 1 | 8 |
| 1.3.2 | Variante 2 | 8 |
| 1.3.3 | Variante 3 | 9 |
| 2 | Résolution par l’AGL de négociation ANTS | 11 |
| 2.1 | Introduction | 11 |
| 2.2 | Représentation interne d’une négociation | 12 |
| 2.2.1 | Le protocole proposé | 12 |
| 2.2.2 | Dynamique opérationnelle | 13 |
| 2.2.3 | Objets internes | 14 |
| 3 | Résolution par une approche de type marché | 25 |
| 3.1 | Approche générale du problème | 25 |
| 3.1.1 | Le problème à résoudre | 25 |
| 3.1.2 | Représentation de la solution | 25 |
| 3.1.3 | Approche du problème | 26 |
| 3.2 | Réalisation | 26 |
| 3.2.1 | Environnement d’exécution | 27 |
| 3.2.2 | Dynamique de l’exécution | 28 |
| 3.2.3 | Exécution élémentaire sur un problème trivial | 28 |
| 3.2.4 | Développement du prototype | 28 |
| 3.3 | Résolution du problème test N°2 | 29 |
| 3.3.1 | Variante1 | 29 |
| 3.3.2 | Variante 2 | 29 |
| 3.3.3 | Variante 3 et 4 | 30 |
| 3.4 | Conclusion | 30 |
| 4 | Résolution à l’aide de la méthode Voyelles | 33 |
| 4.1 | Introduction | 33 |
| 4.2 | le Modèle de Coordination de VOYELLES | 33 |
| 4.2.1 | Exigences de Coordination | 34 |
| 4.2.2 | Description du Modèle de Coordination | 34 |
| 4.3 | Spécification Multi-Agents selon VOYELLES | 35 |
| 4.3.1 | Analyse | 35 |
| 4.3.2 | Conception | 35 |
| 4.4 | Le problème de l’emploi du temps | 39 |
| 4.4.1 | Fonctionnement | 39 |
| 4.5 | Conclusion | 40 |

| | | |
|----------|--|-----------|
| 5 | Résolution par une métaphore émotionnelle | 45 |
| 5.1 | Contexte | 45 |
| 5.1.1 | Rappel du problème | 45 |
| 5.1.2 | Positionnement de notre approche | 45 |
| 5.2 | Principe de résolution | 46 |
| 5.2.1 | dynamique de la charge cognitive | 47 |
| 5.3 | L'algorithme distribué | 48 |
| 5.3.1 | Modèle d'un agent émotionnel | 48 |
| 5.3.2 | Modèle d'un but | 49 |
| 5.3.3 | Modèle d'un message | 49 |
| 5.3.4 | Comportement par défaut | 49 |
| 5.3.5 | traitements des messages | 50 |
| 5.4 | Implementation | 51 |
| 5.4.1 | Application au problème des emploi du temps | 51 |
| 5.4.2 | Resultats | 53 |
| 5.5 | Conclusion et travaux futurs | 54 |
| | | |
| 6 | L'emploi du temps par MAMOSACO | 57 |
| 6.1 | Introduction | 57 |
| 6.2 | Le problème de l'emploi du temps | 58 |
| 6.3 | Spécification du SMA répondant à la première variante | 58 |
| 6.3.1 | Spécification des agents et de leurs comportements | 58 |
| 6.3.2 | Application | 62 |
| 6.4 | Spécification du SMA répondant à la seconde variante | 64 |
| 6.5 | Conclusion | 64 |
| | | |
| 7 | Résolution via la théorie des AMAS. Le système ETTO | 67 |
| 7.1 | La Théorie des AMAS | 67 |
| 7.1.1 | Les agents AMAS | 67 |
| 7.1.2 | La méthodologie ADELFE | 68 |
| 7.2 | Application d'ADELFE à ETTO | 69 |
| 7.2.1 | Expression des besoins | 69 |
| 7.2.2 | Analyse | 70 |
| 7.2.3 | Conception | 72 |
| 7.3 | Développement et test | 73 |
| 7.3.1 | L'interface de visualisation | 73 |
| 7.3.2 | Résultats de la Variante 1 | 74 |
| 7.3.3 | Résultats de la Variante 2 | 75 |
| | | |
| 8 | Approche multi-agents adaptatifs en DIMA | 79 |
| 8.1 | Introduction | 79 |
| 8.2 | Architecture d'agents | 79 |
| 8.2.1 | Un modèle d'architecture qui dépasse la dichotomie classique | 80 |
| 8.2.2 | Les Composants proactifs | 81 |
| 8.2.3 | Agents interactifs | 82 |
| 8.2.4 | Agents adaptatifs | 83 |
| 8.3 | Principales caractéristiques de l'architecture | 85 |
| 8.4 | DIMA | 86 |
| 8.5 | Variante 1 du problème de l'emploi du temps | 86 |
| 8.5.1 | Analyse | 86 |
| 8.5.2 | Conception et Implémentation | 87 |
| 8.5.3 | Déploiement | 89 |
| 8.6 | Conclusion | 89 |

TABLE DES MATIÈRES

5

9 Conclusion

91

Chapitre 1

Enoncé du problème

Le problème de l'emploi du temps

Cahier des charges réalisé par

Carole Bernon, Marie-Pierre Gleizes, Pierre Glize, Gauthier Picard

Equipe SMAC - IRIT - Toulouse

1.1 Introduction

Dans ce document, nous proposons de réaliser une application capable de résoudre le problème de l'emploi du temps. Ce problème :

- est complexe, c'est typiquement un problème de résolution de contraintes, NP-Complet, dont la solution n'est pas, a priori, connue dans le cas général;
- nécessite, pour fournir une solution, d'être capable de s'adapter pour réagir aux changements dynamiques de l'environnement. Dans la variante 3 proposée ci-dessous, nous mettons en avant le fait que des changements peuvent intervenir en temps réel.
- demande une recherche collective de la solution ;
- n'est pas un problème de simulation, il ne s'agit pas de recréer virtuellement le comportement d'un « organisme » existant mais de fournir plutôt une résolution de problèmes.

Nous pensons que ce type de problème peut nous amener à réfléchir sur certaines problématiques essentielles telles que :

- comment mettre en œuvre toutes les étapes d'une méthodologie orientée (multi-)agent ?
- comment identifier les agents à utiliser dans la résolution d'un problème ?
- comment prendre en compte un environnement ouvert : ici, comment gérer l'ajout ou la disparition de contraintes en temps réel ?
- comment gérer le fait que l'environnement est dynamique et que le système à mettre en œuvre doit être capable de s'adapter en conséquence ?
- comment juger de la « qualité » de la solution ?

1.2 Présentation générale du problème

Les acteurs mis en jeu sont les suivants :

- des enseignants ;
- des groupes d'étudiants ;
- des salles.

Chacun de ces acteurs possède des contraintes devant être remplies (au mieux). Un enseignant possède des contraintes portant sur :

- des disponibilités (jour de la semaine, tranche horaire, ...);

- des compétences (enseignement précis, ...);
- le besoin de matériel pédagogique particulier (rétroprojecteur, vidéoprojecteur, salle de TP, ...).

Un groupe d'étudiants doit suivre un enseignement particulier constitue d'un certain nombre de créneaux horaires pour certains matières d'enseignements (X créneaux de la matière 1, Y de la matière 2, ...).

Une salle est munie ou non d'équipements particuliers (rétroprojecteur, vidéoprojecteur, salle de TP, ...) et peut-être occupée ou non durant une tranche horaire, un certain jour.

On suppose que pour chacun des acteurs, les contraintes sont données sous la forme d'une liste. L'ordre d'une contrainte dans la liste précise son importance relative, ainsi, la première contrainte donnée peut, si cela est possible, être plus facilement relâchée que la dernière.

Le problème à résoudre consiste à concilier toutes ces contraintes pour proposer un emploi du temps sur une certaine durée.

1.3 Scénarii proposés

Afin de pouvoir comparer les différentes solutions mises en œuvre pour résoudre ce problème, nous proposons un scénario typique. Trois variantes sont proposées, elles permettent d'introduire une gradation dans la complexité de l'emploi du temps à établir.

Pour chacune des variantes proposées ci-dessous, nous fournissons une solution possible qui n'est certainement pas unique mais permet de clarifier un peu le problème.

1.3.1 Variante 1

Dans un premier temps, on simplifie en supposant que l'on manipule des créneaux horaires de 2h : 8h-10h, 10h-12h, 14h-16h, 16h-18h et que l'on doit trouver un emploi du temps sur deux jours : j1, j2.

Trois enseignants e1, e2 et e3 enseignent chacun une matière spécifique et leurs impossibilités d'enseignement sont les suivantes :

- e1 ne peut enseigner le jour j1 de 16h à 18h et le jour j2 de 14h à 16h.
- e2 ne peut enseigner le jour j2 de 10h à 12h et le jour j1 de 16h à 18h.
- e3 ne peut enseigner le jour j1 de 14h à 16h et le jour j2 de 8h à 10h.

On considère trois groupes d'étudiants g1, g2 et g3. Chacun d'eux doit suivre, sur ces deux jours, deux enseignements de 2h effectués par chacun des enseignants e1, e2 et e3 (soit, pour chaque groupe, 12h d'enseignement au total).

Pour l'instant, on suppose que les agents n'ont pas à gérer la disponibilité des salles : on dispose d'une salle par groupe. On suppose aussi que les agents ne peuvent pas relâcher de contrainte.

Dans ce cas, une solution pourrait être :

| Jour j1 | | | | Jour j2 | | |
|---------|-----------|-----------|-----------|-----------|-----------|-----------|
| Créneau | Groupe g1 | Groupe g2 | Groupe g3 | Groupe g1 | Groupe g2 | Groupe g3 |
| 8h-10h | e1 | e3 | e2 | e1 | e2 | |
| 10h-12h | e3 | e2 | e1 | | e1 | e3 |
| | | | | | | |
| 14h-16h | e2 | e1 | | e2 | e3 | e1 |
| 16h-18h | e3 | | | e3 | | e2 |

1.3.2 Variante 2

On ajoute des contraintes sur les salles :

- On suppose que trois salles s1, s2 et s3 sont disponibles.
- Seules les salles s1 et s2 sont munies d'un rétroprojecteur.
- La salle s1 n'est pas disponible le jour j1 de 10h à 12h.

- La salle s2 n'est pas disponible le jour j2 de 16h à 18h et de 8h à 10h.
- La salle s3 n'est pas disponible le jour j2 de 16h à 18h et le jour j1 de 14h à 16h.
- Tous les enseignants veulent utiliser un rétroprojecteur au moins une fois pour chaque groupe lors des deux jours.
- On suppose qu'un (ou plusieurs) agent(s) peut(vent) relâcher une(des) contrainte(s) pour proposer une solution.

Dans ce cas, une solution pourrait être :

| Jour j1 | | | |
|---------|-----------|-----------|-----------|
| Créneau | Groupe g1 | Groupe g2 | Groupe g3 |
| 8h-10h | e1/s1 | e3/s2 | e2/s3 |
| 10h-12h | e3/s2 | e2/s3 | e1/s1 |
| | | | |
| 14h-16h | e2/s1 | e1/s2 | |
| 16h-18h | | | e3/s1 |

| Jour j2 | | |
|-----------|-----------|-----------|
| Groupe g1 | Groupe g2 | Groupe g3 |
| e1/s3 | e2/s1 | |
| | e1/s1 | e3/s2 |
| | | |
| e2/s3 | e3/s1 | e1/s2 |
| e3/s2 | | e2/s1 |

À condition de relâcher la contrainte « salle s1 non disponible le jour j1 de 10h à 12h » pour placer e1/s1 à ce créneau et de relâcher la contrainte « salle s2 non disponible j2 de 16h à 18h » pour placer e3/s2 à ce créneau (on pourrait aussi essayer de relâcher la contrainte « e3 non disponible j2 de 8h à 10h »).

1.3.3 Variante 3

On ajoute la possibilité de modifier/ajouter des contraintes en temps réel (via une éventuelle interface graphique). Ainsi, un enseignant pourra signaler qu'il est ou non disponible pour un certain créneau horaire, une salle pourra se libérer ou au contraire devenir occupée, ...

Les contraintes pouvant évoluer dynamiquement, il faut être capable de concevoir un système qui soit capable de s'adapter aux changements sans réinitialisation complète du système multi-agent. Par exemple, au cours de la recherche d'une solution à l'emploi du temps ci-dessus, l'enseignant e1 signale qu'il ne peut plus assurer un cours le jour j1 de 10h à 12h mais qu'à la place, il est disponible le même jour de 16h à 18h.

Comment les agents vont-ils résoudre le problème sans interrompre la recherche de la solution et repartir de zéro ?

Une solution serait de demander à l'enseignant e3 de faire cours au groupe g3 le jour j1 de 14h à 16h pour faire passer e1/s1 du créneau 10h-12h ce jour là, au créneau ainsi libéré.

Chapitre 2

Résolution par l'AGL de négociation ANTS

Philippe Mathieu, Marie-Hélène Verrons
Équipe SMAC

Laboratoire d'Informatique Fondamentale Lille – CNRS upresa 8022

Université des Sciences et Technologies de Lille

e-mail : {mathieu,verrons}@lifl.fr

<http://www.lifl.fr/SMAC/>

2.1 Introduction

Depuis quelques années, les systèmes multi-agents (SMA) ont été étudiés selon différentes approches. Que ce soit pour l'Intelligence Artificielle Distribuée [CL90, MCL88] pour l'algorithmique distribuée [YDIK92] ou même pour la robotique [Bro86], les points d'études principaux ont été la communication et la coopération entre les agents. Si de nombreux environnements multi-agents ont pour objectif de faire coopérer les agents en vue de la réalisation d'un but commun [Mat95] Visiter Hoster [Syk89], d'autres systèmes en revanche utilisent des agents qui ont des intérêts différents [Kle91, MCL88, RZ96] comme dans les systèmes Baazar [Syk89] et Kasbah de P Maes, ce qui nécessite une négociation entre eux. A l'instar d'une négociation entre clients et fournisseurs qui utilisent des règles de négociation pour converger vers une position médiane, les agents informatiques concurrentiels doivent négocier avec leur environnement pour arriver à une position acceptée par tous. La réalisation d'une société d'agents travaillant vers cet objectif pose de nombreux problèmes. Ceux-ci interviennent lorsque l'agent doit choisir entre plusieurs négociations concurrentes et/ou lorsque ces négociations conduisent à des renégociations en cascade. Ces problèmes sont rendus plus difficiles encore par le fait que les agents ont une connaissance incomplète de l'environnement et que cet environnement évolue constamment. Par ailleurs l'utilité d'utiliser un agent au cours des négociations est parfaitement justifié par l'explosion du nombre de messages échangés entre les agents du fait de ce processus en cascade. Dans certains cas le nombre de messages peut être en $O(m^n)$ si n est la profondeur du processus de cascade et m le nombre d'agents impliqués dans une négociation.

Nous avons constaté, suite à nos recherches sur les différents types de négociation, que toutes les négociations sont sensiblement identiques. Formellement, on a des ressources r_i , des participants p_j . Un initiateur propose un contrat sur un ensemble de ressources et une discussion s'en suit pour aboutir à une solution acceptable pour tout le monde. Nous pouvons donc réifier la notion de ressource et la notion de contrat, car finalement, ce qui change, c'est la stratégie.

Dans le cadre de ces travaux sur la négociation nous nous baserons sur la définition consensuelle suivante applicable à de nombreux domaines parmi lesquels se trouvent les ventes aux enchères, les systèmes de prise de rendez-vous, et même dans certains jeux multi-agents.

Définition. La négociation s'effectue sur un *contrat* pour obtenir des *ressources* communes et sur demande d'un agent appelé *initiateur*. Elle réunit l'initiateur et un ensemble d'agents auxquels le contrat est proposé, ces agents sont appelés *participants* et se déroule jusqu'à ce qu'un accord satisfaisant un pourcentage de participants soit trouvé. Les participants cherchent également à obtenir la meilleure solution possible pour eux tout en donnant le minimum d'informations possible aux autres.

La réalisation d'une application de négociation passe par trois phases bien identifiées. La **représentation interne** qui contient la gestion des structures de données et les actes de langage nécessaires aux agents afin de faire évoluer leurs connaissances; la **couche de communication** qui permet aux agents de s'envoyer des messages de manière centralisée si les agents sont sur la même machine ou distribuée s'ils sont sur des machines différentes; le **niveau stratégique** permettant aux agents de raisonner.

Dans cet article nous réaliserons la communication entre les agents via l'API Magique [Ben99], ce qui permet au système de fonctionner sur un réseau hétérogène et distribué. La réussite de la négociation passe bien sûr par de bonnes stratégies. Nous ne nous intéressons pas dans ce papier aux stratégies, qui sont différentes selon le type de négociation effectué. Nous proposons donc des stratégies certes naïves mais néanmoins génériques, que l'utilisateur peut changer facilement.

2.2 Représentation interne d'une négociation

Afin de mettre en œuvre la représentation interne d'une négociation, deux aspects fondamentaux doivent être pris en compte. Le protocole utilisé, qui définit le langage utilisé par les agents pour échanger de l'information et les objets internes qui permettent à chaque agent d'avoir un état d'avancement sur ses négociations en cours.

2.2.1 Le protocole proposé

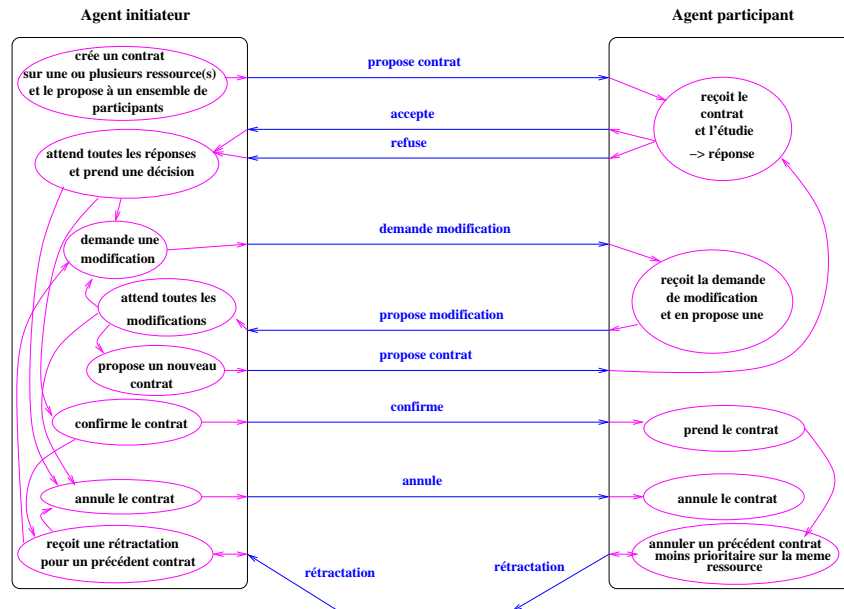


FIG. 2.1 – Le protocole

L'objectif du protocole est de définir les messages que les agents pourront s'envoyer avec la dynamique opérationnelle associée. Le protocole de négociation que nous proposons dans ANTS est

caractérisé par une suite de messages échangés entre un initiateur et des participants (Figure 2.1) comme dans le cadre du Contract Net Protocol [Smi80].

La négociation commence toujours lors de la création d'un contrat par un agent initiateur et de sa proposition aux participants (message *propose contrat*). Puis l'initiateur attend les réponses des participants et lance un timer associé à une réponse par défaut afin d'éviter les deadlocks dus à l'attente de messages. Une fois toutes les réponses reçues, ou lorsque le temps d'attente a expiré, l'agent initiateur prend l'une des trois décisions suivantes :

1. si le seuil minimal d'accords nécessaires est atteint, il confirme le contrat auprès des participants (message *confirme*) . Il se peut alors qu'un contrat moins prioritaire doive être annulé (message *rétractation*).
2. si le nombre maximum de renégociations n'est pas encore atteint, il demande aux participants de proposer une modification pour ce contrat (message *demande de modification*) . L'initiateur lance à nouveau un timer afin d'éviter les deadlocks dus à l'attente des modifications. Lorsqu'il a reçu toutes les modifications (messages *propose modification*), il effectue une synthèse et propose un nouveau contrat, demande à nouveau aux participants d'envoyer une modification, ou annule le contrat le cas échéant.
3. dans les autres cas, il annule le contrat (message *annule*).

Lorsqu'un agent participant reçoit une proposition de contrat, celui-ci l'examine et décide de l'accepter ou non (messages *accepte* et *refuse*). Lorsqu'il reçoit une demande de modification, il envoie ce qui l'arrange le mieux (message *propose modification*).

Ici, la renégociation dénote un tour de demandes de modification pour un contrat qui n'a pas été pris le tour précédent.

Notre protocole utilise deux listes de priorité, l'une pour les participants, l'autre pour les ressources. Ainsi, un participant peut définir des préférences sur le choix du contrat à accepter en cas de conflit.

Un participant peut se rétracter d'un contrat pris précédemment en envoyant le message *rétractation* à l'initiateur du contrat. Lorsqu'un initiateur reçoit une *rétractation*, il regarde si le nombre minimal d'accords est encore atteint. Si ce n'est pas le cas, il demande une modification du contrat aux participants si le nombre maximum de renégociations n'est pas encore atteint. Sinon, il annule le contrat. La renégociation des contrats est donc automatique.

2.2.2 Dynamique opérationnelle

La figure 2.1 représente l'ensemble des messages pouvant être échangés entre un initiateur et un participant. Pour une négociation donnée, seuls certains messages seront échangés. En effet, quand vous avez le choix entre plusieurs réponses, vous ne les envoyez pas toutes. Par exemple, c'est le cas lorsque vous recevez une proposition de contrat, vous pouvez soit l'accepter, soit le refuser, mais vous ne ferez pas les deux en même temps.

Examinons un exemple de négociation faisant intervenir 3 personnes : Paul, Pierre et Jean (Figure 2.2). Pour cet exemple, Paul considère que Jean est plus prioritaire que Pierre. Dans un premier temps, Pierre propose un contrat à Paul qui accepte. Pierre joue le rôle de l'agent initiateur et Paul celui de l'agent participant (Figure 2.1). Pierre crée le contrat et envoie à Paul le message *propose contrat*. Paul reçoit le contrat, l'étudie et envoie le message *accepte* à Pierre pour lui signaler qu'il accepte les termes du contrat. Pierre *confirme* alors le contrat avec Paul. Puis Jean propose un contrat à Paul pour les mêmes ressources. Paul, considérant Jean plus prioritaire que Pierre, envoie un message *accepte* à Jean. Celui-ci confirme le contrat auprès de Paul (message *confirme*). Paul prend alors le contrat avec Jean et annule le contrat pris précédemment avec Pierre (message *rétractation*). Lorsque Pierre reçoit ce message, il décide de demander une modification pour ce contrat à Paul (message *demande de modification*).

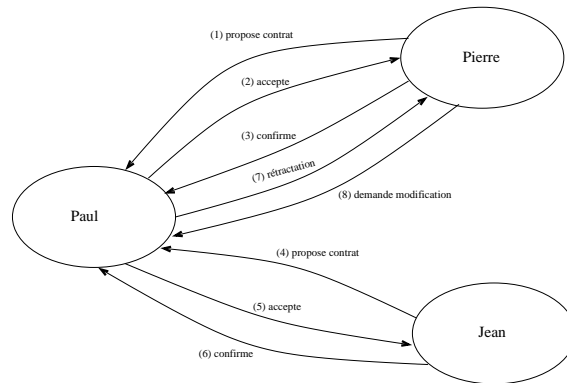


FIG. 2.2 – Communication entre agents

2.2.3 Objets internes

Toute réalisation informatique nécessite une structure de données adaptée. Il faut ici pouvoir coder des négociations simultanées, l'avancement d'une négociation, les initiateurs et les participants et principalement les ressources et les contrats.

Pour implémenter notre protocole, nous avons défini plusieurs objets communs à toute forme de négociation : les ressources et leur représentation, les contrats et leurs propriétés, les buts et les engagements.

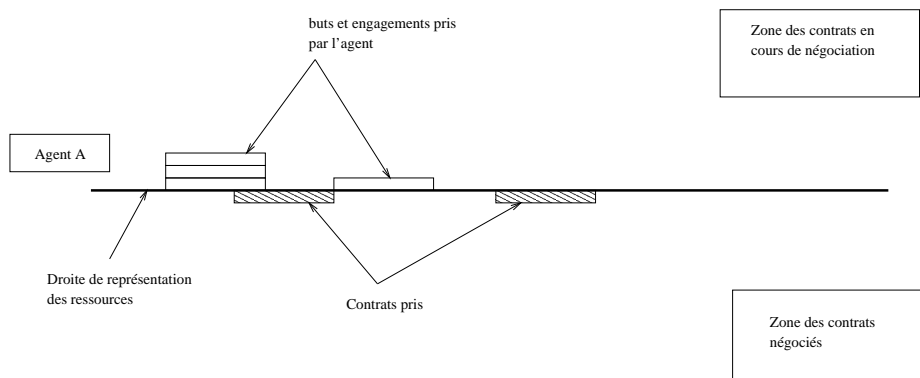


FIG. 2.3 – Représentation des ressources

Les ressources Les ressources sont abstraites et peuvent aussi bien représenter des articles que des tranches horaires ou encore des cageots de fruits. Les ressources sont les objets que chaque agent essaie d'obtenir. N'importe quel objet peut être une ressource, incluant un contrat comme type de donnée, mais pas le contrat en train d'être négocié lui-même.

Les contrats et leurs propriétés Un *contrat* est un objet créé à l'initialisation de la négociation. Il comporte les *ressources* à négocier, l'*initiateur* de la négociation et la *date limite* de réponse pour éviter les deadlocks.

Un contrat sur plusieurs ressources implique que l'obtention de toutes les ressources sont nécessaires pour confirmer le contrat, si toutes les ressources ne peuvent pas être obtenues, la négociation échouera. Tandis que plusieurs contrats sur une seule ressource signifie que l'agent veut obtenir les ressources séparément, c'est-à-dire que si l'agent n'arrive pas à obtenir l'une des

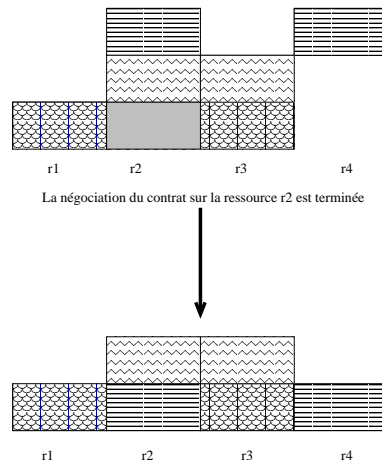


FIG. 2.4 – Comportement de la matrice

ressources, ce n'est pas grave, toutes les négociations sont indépendantes, l'échec de l'une n'a pas d'importance sur la réussite ou l'échec d'une autre.

Le contrat ne contient pas l'ensemble des participants de sorte à préserver leur "vie privée", et donc un participant ne sait pas s'il y a d'autres participants ni leur réponse aux propositions ou leur proposition de modification.

Les *propriétés* du contrat sont constituées du *contrat*, de la liste des *participants* à la négociation, du *déla*i de réponse, de la *réponse par défaut*, du *nombre minimal d'accords* pour confirmer le contrat et du *nombre de renégociations autorisées*. Il mémorise également les participants qui ont accepté le contrat, le nombre d'accords reçus et le nombre de renégociations effectuées.

Ce sont des objets "mémoire" de base qui contiennent les propriétés communes à tout type de négociation.

Les Micro-agents buts et engagements Notre agent négociateur utilise des micro-agents pour gérer ses différentes négociations. Lorsqu'il est initiateur d'un contrat, il crée un micro-agent *but* qui s'occupe de tout le processus de négociation de son contrat. En fait, le but représente l'automate de l'agent initiateur de la Figure 2.1.

Lorsque l'agent reçoit une proposition de contrat, il crée un *engagement* et lui passe le contrat en paramètre. C'est alors celui-ci qui s'occupe de la négociation du contrat, c'est-à-dire qu'il représente l'automate de l'agent participant de la Figure 2.1.

Représentation graphique d'une négociation Afin de représenter les ressources, nous avons défini un modèle graphique tel que le représente la Figure 2.3. Une droite segmentée représente l'ensemble des ressources disponibles, chaque ressource correspondant à un segment. La partie inférieure contient les contrats négociés sur ces ressources, c'est-à-dire les contrats pris par l'agent, les ressources correspondantes n'étant alors plus libres. La partie supérieure contient les buts et engagements qui sont en cours de négociation sur les ressources correspondantes. Il s'agit d'une matrice dont le comportement est calqué sur le jeu Tetris (Figure 2.4). Les contrats qui arrivent s'empilent en cas de conflit sur les ressources. Le but ou l'engagement en cours est celui situé à la surface (base de la colonne ainsi formée), les autres étant en attente. Lors de la fin d'une négociation, le but ou l'engagement est retiré de la surface et celui qui le suivait est alors débloqué, c'est-à-dire que la négociation qu'il représente peut alors commencer. C'est cette matrice qui indique aux objets (but ou engagement) si leur processus de négociation peut commencer (c'est le cas lorsqu'ils se situent à la surface de la matrice). Si un contrat est déjà en cours de négociation

sur les ressources nécessaires pour le contrat arrivant, celui-ci est alors mis en attente. La matrice le réveillera lorsque les ressources seront libérées.

Il peut y avoir des négociations en cours sur des ressources déjà prises car il est possible que l'initiateur du contrat en cours soit plus prioritaire pour l'agent que celui dont le contrat a été pris auparavant.

Gestion simultanée Lorsque l'on veut négocier, il se pose vite le problème des négociations simultanées. En effet, pourquoi traiter toutes les négociations séquentiellement alors que certaines peuvent être traitées simultanément ? C'est le cas lorsque les contrats portent sur des ressources différentes. On peut alors exécuter leur processus de négociation simultanément.

Le fait d'avoir des buts et des engagements qui gèrent la négociation d'un contrat permet d'effectuer plusieurs négociations simultanément, et d'être à la fois initiateur et participant à des contrats. En effet, l'agent est toujours libre pour lancer une négociation, puisqu'il la confie à un but s'il est l'initiateur, ou à un engagement s'il est participant. Afin de savoir si la négociation peut avoir lieu, le but ou l'engagement créé est placé dans la matrice de gestion des ressources (cf. section 2.2.3). Si le contrat géré par le but ou l'engagement en question ne porte pas sur des ressources en cours de négociation, alors il peut être négocié, sinon il est mis en attente jusqu'à la libération des ressources qui sont en cours de négociation.

Conclusion

Cette architecture, implémentée sous forme de l'API ANTS, est une API de négociation de ressources qui sépare les couches de communication, de négociation et de stratégie, la modification de l'une de ces couches ne perturbe pas les autres.

Avantages

- propriétés paramétrables :
 - cardinalité : 1 vers 1, 1 vers n, n vers m
 - type de ressources (individuelles ou communes)
 - simultanéité des négociations
 - nombre de tours de parole
 - nombre de renégociations
 - nombre minimal d'accords nécessaires pour que le contrat soit pris
 - possibilité de se rétracter
 - synchrone / asynchrone
- Renégociation automatique.
- Mécanisme de timer et réponse par défaut pour éviter les deadlocks.
- Mode de réponse manuel ou automatique.
- Un comportement par défaut permet d'effectuer tout de suite une négociation basique.
- Le contrat peut être affiné.
- Possibilité de prendre en compte les résultats des négociations antérieures.
- La description du protocole est abstraite.
- Différents problèmes peuvent être traités :
 - applicatifs : que va-t-on négocier ? (rdv, carottes, ...)
 - stratégiques : l'agent sera-t-il agressif, pénible, ... ?
 - environnementaux : le système est-il centralisé, distribué, ... ?

Limites

- pas d'obligation de contrat pris
- pas de mécanisme de persuasion/d'argumentation.
- pas de négociation multi-step
- pas de négociation combinée

Le problème des emplois du temps

Les acteurs :

- les enseignants e_1, e_2, e_3
- les groupes d'étudiants g_1, g_2, g_3

Les enseignants ont des contraintes de disponibilité.

2 jours, 4 créneaux horaires par jour.

Chaque groupe doit voir 2 fois chaque enseignant.

Le problème est de trouver un emploi du temps satisfaisant les contraintes de disponibilité des enseignants.

Création d'emplois du temps avec ANTS

Comment implémenter ANTS ?

- Implémenter l'interface Communicator
- Implémenter l'interface InitiatorStrategy
- Implémenter l'interface ParticipantStrategy
- Paramétrage du système et des ressources via ants.xml

Dynamique de l'application

- Les utilisateurs fonctionnent en mode asynchrone.
- Chaque enseignant saisit son emploi du temps en temps réel.
- Les groupes sont en mode automatique.
- Les utilisateurs donnent des priorités aux ressources et aux autres utilisateurs.
 - Lorsque tous les enseignants ont la même priorité pour les groupes d'étudiants, le système fonctionne en mode "premier arrivé, premier servi".
 - Lorsque les enseignants ont des priorités différentes, les contrats pris par un enseignant moins prioritaire peuvent être déplacés.

Création de l'application

Il faut définir dans le fichier ants.xml :

- les ressources, $2*4=8$ créneaux horaires
- les agents, $3+3=6$ agents
- les stratégies à utiliser, celles par défaut
- le communicateur, communicateur Magique
- plus les paramètres du transparent 5

The screenshot shows a software window titled 'enseignant_3' with a menu bar containing 'Contract retraction', 'Messages', 'Manual', and 'ResView'. Below the menu bar are three tabs: 'Parameters', 'ContractsTaken', and 'ContractCreation'. The 'Parameters' tab is active and contains the following fields:

- Default answer :** A dropdown menu set to 'yes'.
- min agreements needed :** An empty text input field.
- nb minutes waiting* :** An empty text input field.
- nb times to renegotiate :** An empty text input field.
- resources* :** A list of time slots: j1 8h-10h, j1 10h-12h, j1 14h-16h, j1 16h-18h, j2 8h-10h, j2 10h-12h, j2 14h-16h, j2 16h-18h.
- participants* :** A list of participants: enseignant_1, enseignant_2, groupe_1, groupe_2, groupe_3.

At the bottom right of the window is a 'Create' button.

Exemple d'emploi du temps trouvé en mode premier arrivé, premier servi

| | groupe 1 | groupe 2 | groupe 3 |
|----------|----------|----------|----------|
| J1 8-10 | e1 | e3 | e2 |
| J1 10-12 | e2 | e1 | e3 |
| J1 14-16 | | e2 | e1 |
| J1 16-18 | e3 | | |
| J2 8-10 | e1 | | e2 |
| J2 10-12 | | e1 | e3 |
| J2 14-16 | e3 | e2 | |
| J2 16-18 | e2 | e3 | e1 |

Exemple d'emploi du temps non trouvé en mode premier arrivé, premier servi

- 2 enseignants, 1 groupe, 5 créneaux
- Les enseignants doivent voir 2 fois le groupe

```

<?xml version="1.0"?>
<!DOCTYPE ants SYSTEM "ants.dtd" >
<ants>
<negotiation-type>planning</negotiation-type>
<resources-list>
<resource>8h-10h</resource>
<resource>10h-12h</resource>
<resource>14h-16h</resource>
<resource>16h-18h</resource>
</resources-list>
<agents-list>
<agent><name>e1</name><address>localhost</address></agent>
<agent><name>e2</name><address>localhost</address></agent>
<agent><name>e3</name><address>localhost</address></agent>
<agent><name>g1</name><address>localhost</address></agent>
<agent><name>g2</name><address>localhost</address></agent>
<agent><name>g3</name><address>localhost</address></agent>
</agents-list>
<default-communicator>
fr.lifl.ants.magique.MagiqueCommunicator
</default-communicator>
<default-initiator-strategy>
fr.lifl.ants.strategy.DefaultInitiatorStrategy
</default-initiator-strategy>
<default-participant-strategy>
fr.lifl.ants.strategy.DefaultParticipantStrategy
</default-participant-strategy>
<nbRounds>20</nbRounds>
<nbRenegotiations>3</nbRenegotiations>
<minAgreements>100%</minAgreements>
<answer-delay>10</answer-delay>
<default-answer value="refuse"/>
<simultaneity value="deferred"/>
<retraction-allowed value="true"/>
<nb-modifications-by-round>5</nb-modifications-by-round>
<magique>
<skill>
<class>planning.PlanningNegotiationSkill</class>
</skill>
</magique>
</ants>

```

FIG. 2.5 – Fichier XML pour l'application de création d'emplois du temps

– Contraintes :

| e1 | |
|-------|----------|
| | groupe 1 |
| 8-9 | |
| 9-10 | |
| 10-11 | ////// |
| 14-15 | ////// |
| 15-16 | |

| e2 | |
|-------|----------|
| | groupe 1 |
| 8-9 | |
| 9-10 | ////// |
| 10-11 | ////// |
| 14-15 | |
| 15-16 | |

| | groupe 1 |
|-------|----------|
| 8-9 | e1 |
| 9-10 | |
| 10-11 | |
| 14-15 | e2 |
| 15-16 | e1 |

1. e1 choisit ses 2 créneaux
2. e2 n'a plus qu'une seule possibilité : 14-15. Il ne pourra donc pas voir le groupe deux fois.

Une solution était :

| | groupe 1 |
|-------|----------|
| 8-9 | e1 |
| 9-10 | e1 |
| 10-11 | |
| 14-15 | e2 |
| 15-16 | e2 |

Ce problème est résolu si la priorité de e2 est supérieure à la priorité de e1.

Une solution est-elle toujours trouvée en mode dernier arrivé, premier servi ?

Le fait de régler les priorités entre enseignants ne résoud pas automatiquement le problème, même lorsqu'il y a une solution. En effet, il n'y a pas d'obligation de contrat pris et un enseignant ne déplace pas automatiquement un de ses cours avec un groupe pour pouvoir prendre ce créneau avec un autre groupe. L'enseignant doit créer un nouveau contrat pour le cours avec l'autre groupe et déplacer l'ancien en cas de succès.

Prenons l'exemple de la variante 1 du sujet.

Décidons que e1 parle en premier, puis e2, puis e3.

Les priorités sont : $p(e1) < p(e2) < p(e3)$.

| | groupe 1 | groupe 2 | groupe 3 |
|----------|----------|----------|----------|
| J1 8-10 | e1 | e2 | e3 |
| J1 10-12 | e2 | e1 e3 | e1 |
| J1 14-16 | | e1 | e1 e2 |
| J1 16-18 | e3 | | |
| J2 8-10 | e1 | e1 | e2 |
| J2 10-12 | e3 | e1 | e1 |
| J2 14-16 | e2 | e3 | |
| J2 16-18 | e1 | e2 | e1 e3 |

Conclusion

ANTS n'est pas adapté aux applications où le contrat doit obligatoirement être pris. Dans ANTS, on examine toutes les possibilités de ressources libres avant d'annuler le contrat mais on ne remet pas en cause un contrat pris pour pouvoir en placer un autre. Il faut donc que chaque

enseignant vérifie s'il a bien pris tous ses cours.

Par contre, ANTS est adapté aux changements dynamiques : ajout ou retrait de participants (enseignants ou groupes), un cours supprimé est automatiquement déplacé.

Bibliographie

- [Ben99] NourEddine Bensaïd. *MAGIQUE, une architecture multi-agents hiérarchique*. Thèse de Doctorat, Université de Lille 1, Mai 1999.
- [Bro86] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1) :14–23, 1986.
- [CL90] P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42 :213–261, 1990.
- [Kle91] M. Klein. Supporting conflict resolution in cooperative design systems. *IEEE Transactions on System, Man, Cybernetics*, 21(5) :1379–1390, 1991.
- [Mat95] P. Mathieu. Travail coopératif et communication avancée. In *Actes des journées de travail sur la notion de Multi-agents*, Université des sciences et technologies de Lille, 1995. Groupe Ganymède de la Région Nord-Pas de Calais. Publication 163.
- [MCL88] R.A. Meyer, S.E. Conry, and V.R. Lesser. Multistage negotiation in distributed planning. In A. Bond and L. Gasser, editors, *Reading In Distributed Artificial Intelligence*, pages 367–386, Los Altos, CA, 1988. Morgan Kaufmann Publishers.
- [RZ96] J.S. Rosenschein and G. Zlotkin. Mechanism for automated Negotiation in state Oriented Domains. *Journal of Artificial Intelligence Research*, 5 :163–238, 1996.
- [Smi80] Reid G. Smith. The Contract Net Protocol : high-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, C-29(12) :1104–1113, December 1980.
- [Syk89] K.R. Sykara. Multiagent compromise via negotiation. In L. Gasser and M.N. Huhns, editors, *Distributed Artificial Intelligence*, volume 2, pages 119–137, Los Altos, CA, 1989. Morgan Kaufmann Publishers.
- [YDIK92] Yokoo, Durfee, Ishida, and Kuwabara. Distributed constraint satisfaction for formalizing distributed problem solving. In *Proceedings of the twelfth International Conference on Distributed Computing Systems*, Yokohama, Japan, June 1992.

Chapitre 3

Résolution par une approche de type marché

Jean-Paul A Barthès
Université de Technologie de Compiègne
UMR 6599 HEUDIASYC , BP 349
60206 COMPIEGNE
Email : jean-paul.barthes@utc.fr

3.1 Approche générale du problème

3.1.1 Le problème à résoudre

Le système fait intervenir :

- des enseignants (E_i),
 - des groupes d'étudiants (G_j),
 - des unités de valeurs ou cours (C_n)
 - des salles de cours (S_k),
 - des créneaux horaires (T_l).
- l'emploi du temps à réaliser est périodique sur deux jours.

Les contraintes sont les suivantes :

- chaque groupe d'étudiants a un cursus à suivre qui se traduit par un certain nombre de séances qui correspondent à des cours. Il peut y avoir plusieurs séances par cours ;
- les enseignants peuvent enseigner plusieurs cours ;
- les enseignants ont des plages horaires pendant lesquelles ils ne souhaitent pas enseigner ;
- les salles peuvent être spécialisées ou contenir du matériel indispensable à certains cours ;

L'emploi du temps à réaliser est périodique. Une solution sera trouvée lorsque tous les groupes d'étudiants pourront suivre tous les cours requis. Dans ce cas, on aura une affectation de créneaux horaires, enseignants, groupes d'étudiants, cours, et salle.

Note : il n'est pas sûr a priori qu'une telle affectation existe.

3.1.2 Représentation de la solution

La Figure 1 donne une représentation graphique du problème et d'une solution d'affectation. À gauche les groupes d'étudiants, à droite les enseignants, au centre les salles, en hauteur le temps, en gris les moments pendant lesquels les enseignants ne peuvent enseigner. L'hypothèse est que chaque enseignant donne un cours différent qui doit être enseigné une fois chaque jour à chaque groupe.

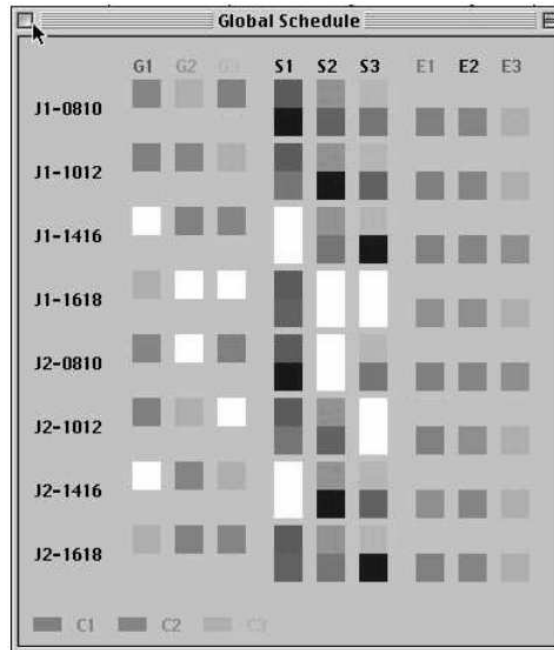


Figure 1 - Résultat d'une affectation

3.1.3 Approche du problème

Le problème est piloté par la demande des groupes d'étudiants qui agissent en fonction de leur curriculum. L'idée générale est une régulation par les coûts. Les groupes d'étudiants achètent les services d'un enseignant qui à son tour loue une salle.

Le point de vue du groupe d'étudiants

Le groupe d'étudiants doit suivre un cursus représenté par des cours. Chaque cours doit avoir un nombre donné de séances. Le groupe d'étudiants va essayer d'obtenir un enseignant et une salle pour toutes ses séances, en faisant une offre aux enseignants qui donnent le cours à un prix minimal (au départ 0). S'il n'y arrive pas, il va augmenter le prix qu'il est prêt à payer.

Le point de vue de l'enseignant

Un enseignant reçoit un appel d'offres de la part de groupes d'étudiants. Il va essayer de louer une salle dans un créneau horaire qui l'arrange. Il essaiera d'abord avec les créneaux libres, puis en dénonçant les contrats moins intéressants (avec peut-être une restriction pour les contrats sur le même cours avec le même groupe¹).

Le point de vue des salles

Les salles sont purement passives. Elles sont ou non disponibles.

3.2 Réalisation

La réalisation fait intervenir plusieurs types d'agents et des objets extérieurs. Les agents sont les enseignants et les groupes d'étudiants, les objets extérieurs sont les cours, les salles et les créneaux horaires.

L'exécution du système est assez difficile à suivre. Aussi, pour expliquer ce qui se passe nous avons utilisé le simulateur sur des configurations simples en analysant les mécanismes en détail,

¹lorsque le cours doit être donné plusieurs fois pendant la période déterminée.

comme lors de l'approche précédente. Commençant par un cas trivial le système est progressivement compliqué pour arriver à la configuration du problème 1.

3.2.1 Environnement d'exécution

Les informations sont distribuées dans plusieurs agents :

- un agent administrateur qui définit les cursus à suivre, la disponibilité des salles, les charges d'enseignement. Son rôle est préliminaire et peut être ignoré dans un premier temps. Il pourrait ensuite modifier dynamiquement la charge d'enseignement d'un enseignant ou la disponibilité des salles en fonction de la situation générale.
- des agents enseignant, qui ont des charges d'enseignement, des périodes où ils ne souhaitent ou ne peuvent enseigner
- des agents groupes d'étudiants qui ont un cursus à suivre, et des moments où ils ne peuvent être en cours
- un agent emploi du temps qui regroupe toutes les informations sur les salles et qui affiche la situation courante

agent administrateur

Son rôle n'étant pas crucial pour le déroulement de l'affectation, il ne sera pas décrit ici.

agent enseignant

Il possède une structure permettant de spécifier le service qu'il doit faire, les cours à enseigner, pour chaque cours le nombre de séances à faire, et une version de l'agenda indiquant les moments où il est déjà pris (cours, groupe, salle, créneau) et les créneaux pendant lesquels il ne peut intervenir.

cours

Un cours est représenté comme une liste de séances. Chaque séance donne le créneau horaire, la salle, l'enseignant, le coût. Un même cours peut être donné plusieurs fois (e.g., à des groupes différents).

agent groupe d'étudiants

Il possède une structure spécifiant les cours qu'il doit prendre et les séances correspondantes, et une version de l'agenda indiquant les moments où il est déjà pris (enseignant, cours, groupe, salle, créneau) et les créneaux pendant lesquels il ne peut suivre des cours.

agent emploi du temps (room control)

Il possède une version de l'agenda indiquant l'occupation des salles (créneau, salle, cours, enseignant, groupe).

agenda

La structure d'agenda étant commune à beaucoup de monde, elle est définie séparément. L'agenda est une table dont les colonnes représentent les salles et les lignes les créneaux horaires.

Une entrée comprendra les informations suivantes :

- cours : identifiant du cours
- enseignant : identifiant de l'enseignant qui donne ce cours
- groupe : identifiant du groupe d'étudiants qui suit le cours
- coût : prix d'utilisation de ce créneau horaire
- coût-minimal : prix de location minimal de la salle pour ce créneau horaire (un prix élevé indiquera que la salle n'est pas disponible) ou bien coût d'intervention d'un enseignant (servant à exprimer les préférences et les indisponibilités) ou encore coût d'assistance pour un groupe d'étudiants (servant à exprimer les préférences ou les indisponibilités).

3.2.2 Dynamique de l'exécution

Le fonctionnement de chaque type d'agent va être précisé progressivement sur une série d'exemples de plus en plus complexes.

Le système est actionné par les agents groupes d'étudiants qui vont intervenir tant que leur cursus ne sera pas complètement assuré. Pour cela ils possèdent un but qui consiste à envoyer des appels d'offres tant que leur situation n'est pas réglée. Un appel d'offres est dirigé vers l'ensemble des enseignants (en pratique c'est un broadcast auquel seuls les enseignants savent répondre).

Un enseignant qui reçoit un appel d'offres regarde s'il enseigne le cours en question et, si oui, demande quelles sont les salles disponibles et à quel coût. La demande est faite à l'agent emploi du temps qui retourne une liste de possibilités assortie des coûts correspondants. L'enseignant retourne la liste de salles avec leur coût d'utilisation (incluant le sien). Le groupe choisit la meilleure possibilité (coût le moins cher) et confirme auprès de l'enseignant et du Room-Control de l'agent emploi du temps. Si la salle a été prise entre-temps, la transaction est annulée et la demande abandonnée.

3.2.3 Exécution élémentaire sur un problème trivial

Dans une version très élémentaire (PB0 variante 1), nous faisons intervenir un seul enseignant, quatre créneaux horaires, une seule salle et un seul groupe d'étudiants. Le but est d'expliquer le fonctionnement élémentaire du système, ce qui est résumé sur la Figure 2.

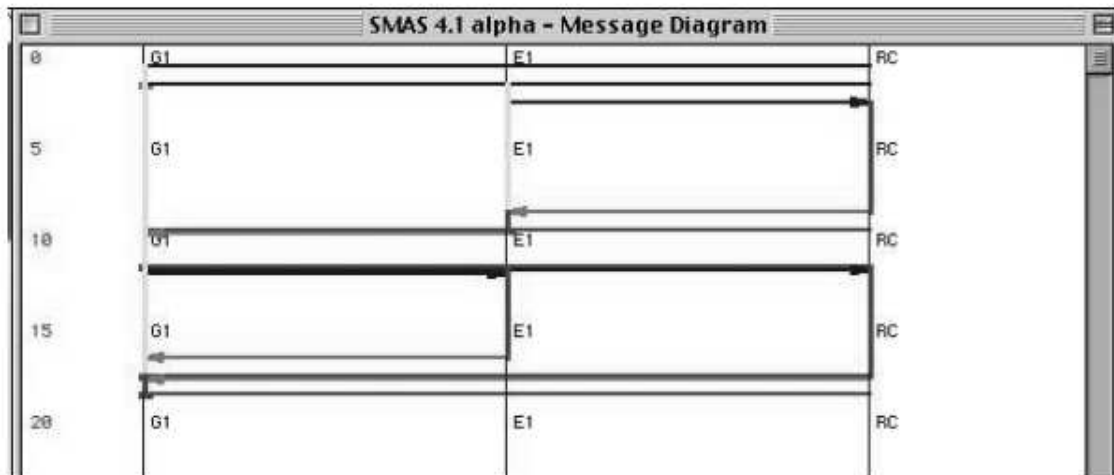


Figure 2 - Exécution élémentaire

Dans ce cas de figure, nous avons un agent enseignant E1, un agent G1, un agent room-control (RC).

Au départ l'agent G1 détecte qu'il lui manque une session du cours C1. Il active donc un message en broadcast pour rechercher un enseignant pour ce cours, en précisant quels sont les créneaux qui lui conviendraient et les salles permises (SI).

E1 reçoit le message, vérifie ses propres créneaux et demande à RC s'il a des propositions et à quel coût. RC retourne une liste de possibilités qu'E1 retourne à G1.

G1 prend la moins chère, accepte et confirme à E1 et à RC. La situation n'ayant pas changé entre temps, le créneau est accepté.

Le système s'arrête car le but de G1 est satisfait. Le résultat apparaît sur la Figure 3.

3.2.4 Développement du prototype

La démarche consiste à compliquer progressivement le problème en introduisant successivement :

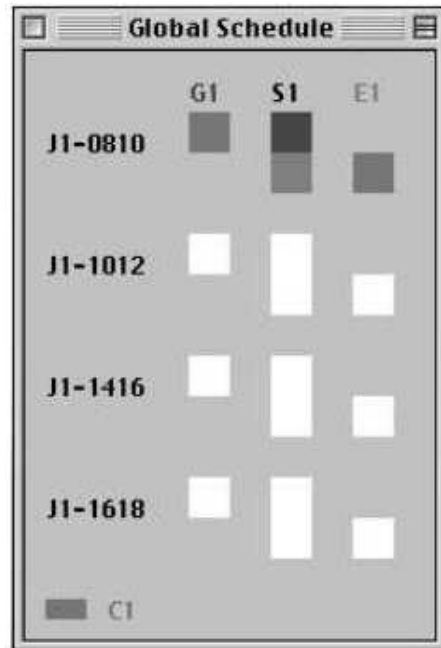


Figure 3 - Emploi du temps

- une contrainte sur l'enseignant qui ne souhaite pas enseigner par exemple le matin. La façon d'exprimer cela est d'attribuer un coût aux créneaux du matin. On leur donnera par exemple un coût de 10.
- une contrainte sur le groupe d'étudiants qui n'a pas envie ou ne peut pas aller en cours une partie de l'après-midi. On peut indiquer cette contrainte sous forme d'un coût particulier de la même manière que pour l'enseignant.
- des conflits entre l'enseignant qui ne veut pas enseigner le matin et les étudiants qui ne peuvent pas suivre le cours l'après-midi. La résolution du problème se fait de la même façon que précédemment au détriment de l'enseignant puisque ses contraintes sont moins fortes que celle du groupe d'étudiants (hiérarchisation des contraintes). La différence avec le cas précédent est que cette fois l'affectation a un coût pour le groupe (10) qui est celui de forcer la contrainte de l'enseignant.
- le découpage en deux journées séparées, en proposant un mécanisme permettant de répartir deux sessions du même cours dans l'emploi du temps dans chacune des journées.

Le fonctionnement du système peut être représenté sous la forme d'un diagramme de type AUML comme indiqué sur la Figure 4 que nous ne détaillerons pas.

3.3 Résolution du problème test N°2

3.3.1 Variante 1

La même approche appliquée au problème test n°2 dans sa variante 1 donne les résultats indiqués en début de ce document sur la Figure 1. La difficulté est faible compte tenu de l'admissibilité de nombreuses solutions sans qu'aucune des contraintes ne soit remise en cause.

3.3.2 Variante 2

Les trois salles ne sont plus affectées mais tous les groupes peuvent les utiliser. En revanche deux salles seulement S1 et S2 sont munies d'un rétroprojecteur, et chaque enseignant veut utiliser

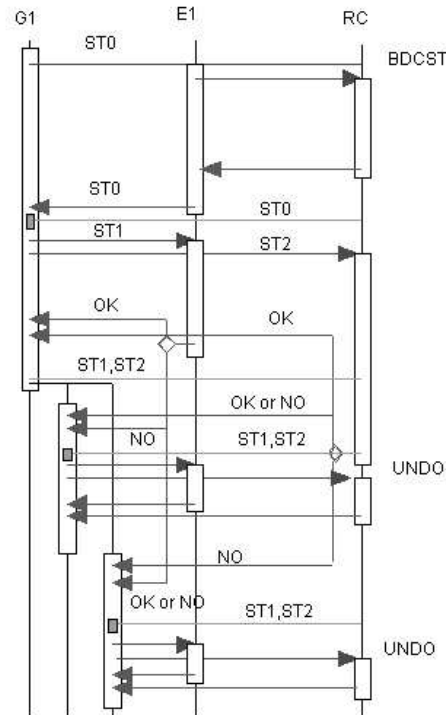


Figure 4 - Diagramme AUML des échanges entre agents

un rétroprojecteur au moins une fois.

Cette dernière contrainte est difficile à modéliser sous forme d'un coût. En effet si un enseignant donne déjà une session à un groupe dans une salle avec un rétroprojecteur, alors peu importe où le cours suivant sera donné. En revanche si la première session est dans une salle sans rétroprojecteur, le coût associé à l'utilisation de cette dernière pour donner la session suivante devra être fortement augmenté, ce qui n'est pas très facile à modéliser.

De plus dans cette variante il faut hiérarchiser les contraintes et déterminer s'il est plus important qu'un enseignant relaxe une de ses contraintes ou s'il est plus important que l'on relaxe la contrainte du rétroprojecteur.

3.3.3 Variante 3 et 4

La modification de contraintes en temps réel (simulé) et l'arrivée ou le retrait de groupes d'étudiants ou d'enseignants ne pose pas de problème particulier pour cette approche, si ce n'est que l'on n'aboutira pas forcément à une solution optimale au sens du coût global.

3.4 Conclusion

L'approche du problème se fait en plusieurs phases. Une première phase d'analyse permet de préciser tous les concepts utilisés dans le problème. Une deuxième phase préside au choix de l'approche générale adoptée (approche par les coûts). Une troisième phase définit les agents utilisés, puis les structures de données correspondantes et les opérations sur ces structures pour chacun des agents. Ensuite le prototype est développé suivant une approche en spirale à partir de cas simples en résolvant progressivement des cas plus compliqués.

Au cours des développements, les compétences des agents sont modifiées progressivement, les mécanismes de gestion de contraintes sont introduits, si bien qu'en fin de compte, l'on obtient un

ystème générique de résolution de ce type de problèmes d'affectation et que la spécification du problème à résoudre tient en quelques lignes.

Chapitre 4

Résolution à l'aide de la méthode Voyelles

João L.T. da Silva Yves Demazeau
Laboratoire LEIBNIZ-IMAG,
46, Avenue Felix Viallet
38031 Grenoble, France
{Joao-Luis.Tavares, Yves.Demazeau}@imag.fr

4.1 Introduction

Dans la Gestion d'Agendas distribués, chaque Agenda est représenté par un agent autonome agissant au service d'un utilisateur particulier. Le problème principal est de trouver un plan local à travers un consensus global sur des contraintes distribuées entre plusieurs agents. Dans notre contexte, les contraintes se rapportent aux préférences de l'utilisateur, les dépendances entre activités individuelles (locales à l'Agenda) et les exigences de coordination externes, comme la disponibilité de temps/ressources et les interactions avec d'autres Agendas.

De façon générale, l'application fournit un outil pour gérer le calendrier et l'agenda des activités d'un utilisateur individuel ; organiser et prévenir l'utilisateur de ses activités ; traiter la coordination avec d'autres agents gestionnaires (en trouvant une solution mutuelle pour une demande de rendez-vous entre d'autres agents-agenda).

Une approche de coordination multi-agents de la gestion d'agendas s'explique par leur caractéristique distribuée. Par ailleurs, la décentralisation du contrôle permet le respect de la privauté de l'utilisateur. Pour cette problématique, nous proposons un modèle de coordination décentralisée sur le Paradigme VOYELLES [4, 3] et nous illustrons la spécification multi-agents d'un gestionnaire d'agendas.

4.2 le Modèle de Coordination de Voyelles

Notre modèle de coordination décentralisé est fondé sur le paradigme VOYELLES[4, 3], se concentrant en particulier sur l'interaction entre plans et la dépendance sociale. Le paradigme VOYELLES décrit une méthodologie de Programmation Orientée Multi-Agents pour la construction des systèmes multi-agents. Cette approche est fondée sur un principe d'assemblage de composants pour raisonner en termes d'Agents, d'Environnements, d'Interactions et d'Organisations.

Ce modèle est orienté au SMA et intègre des approches de coordination à chaque composant multi-agents. Au niveau Organisation, nous retenons un mécanisme de raisonnement social [10] et nous l'étendons pour traiter la planification par rapport au niveau Interaction. Les relations entre plans sont empruntées de l'approche de Martial [7] pour la coordination des plans multi-agents

selon une taxonomie d'interactions entre plans. Dans ce cas, nous avons employé des relations de plans aux niveaux Agent et Organisation par l'extension d'un Réseau de Dépendances (DEPNET) de [9]. Au niveau Environnement, nous avons associé une description des activités, des ressources et de la représentation du monde à travers l'approche TÆMS [2]. Dans la section suivante, nous présentons une rapide vue d'ensemble de notre modèle de coordination décentralisé sous VOYELLES¹.

4.2.1 Exigences de Coordination

Notre idée centrale est fondée sur les interactions (positives et négatives) entre plans, actions et buts, ainsi que sur la spécification des dépendances. Les dépendances sont définies au travers des classes qui décrivent des exigences de coordination concernant les ressources, les buts, les plans et les actions. Par exemple, une exigence d'action est fréquemment liée à des *contraintes de simultanéité* (Mutex) ou à des *conflits de ressources* (*producteur/consommateur*); en ce qui concerne les mécanismes de coordination, le premier peut être géré par planification, tandis que le dernier est une affaire de synchronisation.

Dans les SMAs, ces dépendances représentent des contraintes à plusieurs niveaux dans les composants multi-agents : des *contraintes personnelles* au niveau d'Agent ; des *contraintes situationnelles* au niveau d'Environnement ; des *contraintes relationnelles* au niveau Interaction et des *contraintes organisationnelles* au niveau Organisation. Nous avons proposé une approche de coordination [1] qui vise à décentraliser la dynamique d'un SMA en distribuant le contrôle entre les composants de base selon leurs spécifications et compétences. Dans notre travail, des exigences de plans et de buts sont traitées au niveau Agent, tandis que des exigences d'actions et des ressources sont prises en compte dans le composant Environnement.

4.2.2 Description du Modèle de Coordination

Coordination au niveau Agent : à ce niveau, le problème de coordination est principalement concerné par la synchronisation des plans, des dépendances entre actions/ressources, de l'évaluation de buts et l'optimisation de plans. Nous avons tenu compte des *contraintes personnelles* produit à travers le raisonnement interne de l'agent, en ce qui concerne les interactions entre plans et les dépendances entre les tâches. Pour la coordination dans \mathcal{A} , nous avons défini une représentation de planification hiérarchique avec la description des relations complémentaires. Cette représentation est appliquée à une description externe des accointances pour la dépendance sociale [9].

Coordination au niveau Environnement : la gestion des ressources et des *contraintes situationnelles* au niveau des tâches de plusieurs agents coopérants est traitée dans le composant \mathcal{E} . Pour résoudre une certaine tâche et les dépendances de ressources, nous utilisons une description des relations contenue dans la structure de tâches fondée sur les mécanismes de coordination de GPGP [2]. Ainsi, nous décrivons au niveau de la planification certaines relations possibles rapports tel que : la *Permission*, la *Facilitation*, l'*Annulation* et etc., lesquels sont représentés dans la structure de tâches. Les contraintes de ressources sont pris en compte à travers des dépendances entre plans et font partie de la description de l'action qui exige ces ressources. Ainsi, les actions de coordination du style synchronisation sont déclenchées dès la détection d'une relation action/ressource.

Coordination au niveau Interaction : ce niveau représente les *contraintes relationnelles* en ce qui concerne la communication multi-agents : la spécification de l'échange des messages, la gestion de protocoles et les exigences de négociation. Pour atteindre les exigences de coordination par négociation, nous utilisons l'ensemble des relations entre plans [7] appliquées à une gestion de protocoles [6]. Cela nous mène à la description de besoins de communication à travers une intersection entre la structure de tâches (\mathcal{E}) et le réseau de dépendance (\mathcal{O}). De plus, certaines

¹Une description plus détaillée de la spécification du modèle peut être trouvée dans [1].

contraintes détectées dans ces opérations d'intersection peuvent guider le choix de protocoles adéquats.

Coordination au niveau Organisation : la notion d'Organisation tient compte des rôles des interactions et de la dépendance sociale entre agents [9]. Un rôle est une abstraction du comportement d'un agent qui agit réciproquement avec d'autres. Ce comportement est décrit par des buts globaux et des plans qui jouent ces rôles dans un SMA en employant la même structure de description externe employée dans le modèle d'agent. Au niveau social, les rôles interagissent réciproquement à travers leurs relations et leurs dépendances. Ces relations déterminent l'existence d'une liaison entre deux rôles et le besoin de l'interaction entre les agents. En utilisant cette description externe des compétences et des buts des autres, nous sommes capables de prendre en compte les relations plan/action/ressource. Ainsi, ces relations engendrent une coordination implicite à travers la formation de coalition s'appuyant sur les complémentarités entre agents. De plus, ces coalitions peuvent influencer le contrôle local de chaque composant multi-agents dans l'évaluation des priorités des plans et de l'ordonnancement des tâches.

Dans la section suivante, nous allons spécifier la Gestion d'Agenda Distribuée selon notre modèle de coordination multi-agents.

4.3 Spécification Multi-Agents selon Voyelles

4.3.1 Analyse

Le problème est celui de la Gestion des Agendas distribués, qui se situe dans les domaines classiques de la résolution de contraintes et de l'allocation des ressources distribuées. Typiquement, le problème est la satisfaction d'un but commun dans un contexte physique et/ou temporel, prenant en compte des contraintes locales pour atteindre un but consistant. La solution demande la mise en œuvre de la coordination et de la planification.

Dans ce contexte, un agenda représente son utilisateur et prend en compte ses préférences personnelles pour gérer les dépendances entre les tâches locales et satisfaire les contraintes distribuées lors d'une prise de rendez-vous. Afin de garantir un certain degré de privauté des utilisateurs, nous voulons éviter d'échanger la totalité des agendas pendant la négociation d'un rendez-vous. Une approche décentralisée permet ce contrôle local et flexible, malgré la quantité élevée d'interactions. Cette caractéristique nous amène à détecter les dépendances au niveau interaction (négociation et modification de plans).

4.3.2 Conception

L'application est focalisée sur la brique agent, car les problèmes de résolution de contraintes et allocation des ressources demandent une solution orientée sur la coordination et la planification. C'est à ce niveau que les contraintes locales sont prises en compte lors du processus de négociation pour atteindre une solution globale consistante. Une approche classique de prise de rendez-vous sera plutôt centrée sur la brique Interaction, à cause de la nature dynamique des communications dans la négociation d'une plage horaire. Dans notre contexte, l'Agenda offre plus de fonctionnalités que la simple prise de rendez-vous et doit également gérer la coordination interne entre les événements individuels de l'utilisateur. Dans la suite, nous présentons la spécification de l'Agenda sous chaque composant Voyelles.

Spécification du composant Agent

Dans le composant (\mathcal{A}) nous mettons en œuvre des heuristiques de prises de rendez-vous *meeting schedule* et de gestion de contraintes (préférences personnelles de l'utilisateur et des engagements déjà pris). La coordination parmi des agents pour la réunion de la planification est définie en termes de dépendances calculées visant une possible approche de modifications des plans et des ressources (décalage des dates et le changement de ressources indisponibles).

Quand l'Agenda reçoit un but de son utilisateur, un plan est choisi pour satisfaire ce but qui relie les dépendances entre plans et les contraintes de l'agent à satisfaire. Par exemple, dans les dépendances tâche/sous-tâche, la tâche *ConfirmMeeting* dépend de la relation au niveau environnement pour la tâche *VerifyCalendar* afin d'exécuter le plan *ScheduleMeeting*. L'agent déduit qu'une contrainte situationnelle implémente une exigence de coordination et active un composant distribué pour exécuter la tâche associée au niveau de l'environnement.

Nous montrons dans [1] que les buts sont atteints par des plans associés formés par un ensemble d'actions complètement instanciés. Les buts, comme *ScheduleMeeting* sont représentés par une hiérarchie "ET". La Figure 4.1 (a) montre que pour réaliser le plan *ScheduleMeeting*, la tâche *VerifyCalendar* doit obtenir une plage horaire appropriée; la tâche *ConfirmMeeting* doit trouver un consensus pour que la tâche *AppendMeeting* atteigne un *planning* respectant les exigences.

Les événements sur l'Agenda de l'utilisateur sont décrits par une approche de planification hiérarchique, selon le modèle de coordination au niveau (A)gent. Ainsi, dans un premier niveau de négociation, les agents échangent d'abord des plans abstraits. Si les contraintes ne sont pas encore résolues, les agents continuent la négociation en raffinant seulement les plans abstraits qui doivent être détaillés. Par exemple, quand un agent commence un cycle de négociation, il bloque d'abord une plage horaire sur son calendrier. Ensuite, un plan *AppendMeeting* est engagé par l'agent contenant une tâche *VerifyCalendar*, comme illustré dans la Figure 4.1 (a). Le Calendrier vérifie s'il y a des plages horaires disponibles et, s'il trouve un conflit temporel, les plans sont encore raffinés. Dans ce cas, l'action *blockInterval* représente une dépendance entre l'agent et l'environnement (le Calendrier).

Les préférences représentées par contraintes

Les contraintes sont définies en fonction des préférences de l'utilisateur et des engagements pris. Dans ce contexte, un *rendez-vous* est défini comme un futur engagement conjointement accepté par tous les participants sous ces contraintes. Chaque utilisateur crée ses propres événements (engagements) selon l'état de son agenda pour des plages horaires sélectionnées. Ces engagements peuvent être des tâches individuelles ou des réunions, qui décrivent un but réalisable par leurs Agendas. Les préférences de l'utilisateur sont classées par préférences temporelles en définissant des priorités sur les plages horaires et les leurs activités personnelles et professionnelles.

L'utilisateur définit son propre niveau de priorité sur les engagements en assignant une combinaison d'urgence et d'importance à chaque activité prévue. Les préférences temporelles sont définies en fonction des horaires de travail de chaque jour de la semaine, les heures complémentaires, les week-ends et les vacances. Ces derniers définissent comment les contraintes temporelles seront considérées dans la négociation sur des temps de travail complémentaires. De plus, des préférences contextuelles sont définies par la responsabilité dans la tâche et la structure organisationnelle.

Spécification du composant Environnement

L'environnement est défini selon le groupe d'agents et la définition du calendrier. A ce niveau, le calendrier représente des contraintes d'actions et ressources entre agents. Ces contraintes sont décrites par la disponibilité temporelle des plages horaires du calendrier, sous la forme : *indisponible*, *occupé* ou *libre*. Une plage horaire *indisponible* est un intervalle qui possède un *engagement* privé déjà pris par l'utilisateur et il ne peut pas être relâché dans la négociation. Un créneau *occupé* est alloué à un *engagement* public et peut être négociable ou relâché selon les préférences personnelles de l'utilisateur. Finalement, un créneau *libre* peut accepter un nouveau *engagement*.

Dans notre modèle, nous supposons l'existence d'une description des ressources et les interactions avec des actions permises par l'environnement à travers une hiérarchie de structure de tâches. La Figure 4.1 (b) illustre une structure de tâches partielle pour vérifier la disponibilité des ressources (ici, le créneau horaire, mais aussi des salles et des équipements). Dans ce travail, nous employons un modèle d'automate à états finis pour décrire l'état des ressources (les pré-conditions et post-conditions des créneaux) et une relation *Mutex* entre les tâches et les ressources.

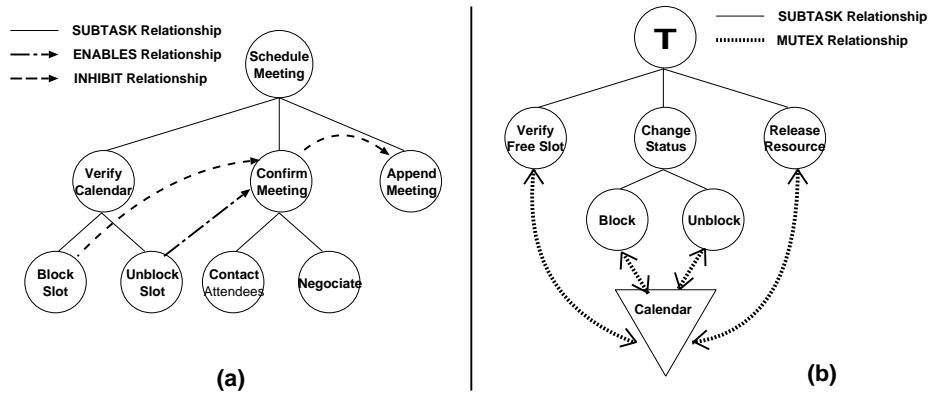


FIG. 4.1 – (a) Représentation du plan SCHEDULEMEETING ;)b) Structure partielle de tâches et ressources.

Spécification du composant Interaction

Le composant (\mathcal{I}) est responsable de la gestion des protocoles de communication dans le sens où il doit choisir le protocole adéquat. De plus, une certaine cohérence doit être assurée pendant l'échange des messages. Par exemple, dans une prise de rendez-vous, lorsqu'un participant ne peut pas recevoir le message, tous les messages associés doivent être annulés. Le gestionnaire de protocoles doit fournir correctement l'atomicité des transactions.

Selon les contraintes de dépendance, le contrôle d'interaction peut correctement traiter avec des protocoles distincts pour différents types d'échange d'information. Par exemple, lors de la connexion de l'agent, le contrôleur d'Interaction déclenche un mécanisme de coordination initial pour la mise à jour d'information (*information gathering*), ce qui met à jour la description externe de chaque agent.

La brique Interaction possède également un composant distribué, qui accepte et émet des événements de la même forme que pour la brique Agent. La gestion de protocoles est prise en compte par rapport la situation de l'agent : protocole de connexion, *information gathering* au début d'une contrainte non résolue, protocole de négociation, etc.

La gestion de protocoles est faite à travers un modèle d'ingénierie de protocoles basée sur la notion de micro-protocoles [6]. Le modèle porte sur la réutilisation de protocoles exprimés par le formalisme CPDL (*Communication Protocol Description Language*). Dans cette approche, un protocole est obtenu par un assemblage de micro-protocoles qui à son tour regroupe un ou plusieurs performatifs basés sur le langage ACL (*Agent Communication Language*). L'un des avantages d'utiliser ce modèle est la gestion dynamique des protocoles utilisés selon les besoins de l'agent. Pour l'infrastructure de communication nous avons choisi d'utiliser une sous-couche de JATLite [5], plus particulièrement les sous-couches de routage et communication.

Dans notre application, nous adaptons le protocole d'apprentissage coopératif de Sian [8] (cf. Figure 4.2) pour atteindre un simple consensus. Sur ce protocole, un agent élabore une hypothèse qu'il croit être vraie pour la partager avec un groupe d'agents associés. Chaque agent évalue cette hypothèse contre ses propres croyances et ensuite la confirme, la modifie ou la rejette. Quand le groupe atteint un consensus, chaque agent met à jour sa base de connaissance avec cette nouvelle information. Dans notre cas, l'hypothèse représente une demande de réunion, qui doit être acceptée ou rejetée par tous les participants en cas de non-consensus. Plusieurs demandes de réunion peuvent être impliquées avec des groupes d'agents différents. Par conséquent, le modèle d'Interaction doit traiter avec l'ordonnancement simultané.

Nous supposons que seulement l'information appropriée doit être échangée pour construire des plans locaux. Ici, nous prenons une spécification traditionnelle de la prise de rendez-vous pour représenter le contenu du message. Dans notre approche, le protocole de négociation est décrit de

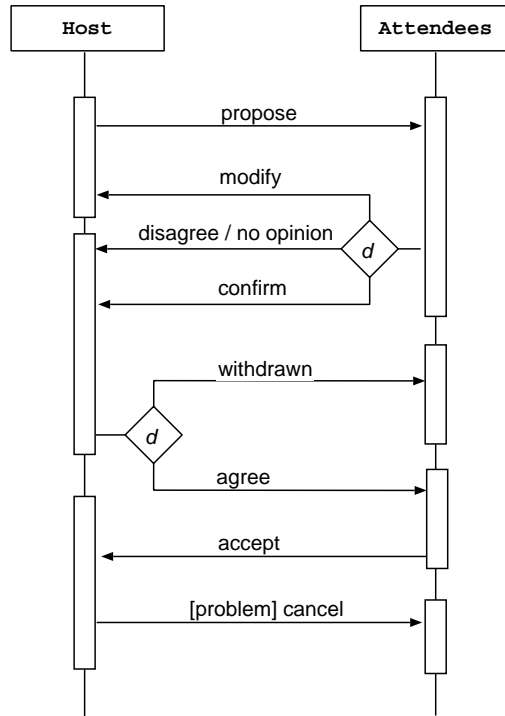


FIG. 4.2 – Spécification UML du protocole de Sian.

la manière suivante :

$$\text{Negotiate}(h_i, A_i, m_i) \rightarrow \text{propose}(h_i, A_i, m_i)$$

Formellement, nous représentons une spécification de réunion par le quadruplet :

$$m_i = (S_i, l_i, w_i, T_i)$$

Le but d'un ensemble d'agents (A) est de trouver une prise de rendez-vous (demande de réunion) (m_i) qui soit globalement cohérente. La demande de réunion est proposée par le *Host* ($h_i \in A_i$) qui interagit avec un ensemble de participants ($A_i \subseteq A$) pour trouver un plan consensuel. La réunion doit être fixée dans l'intervalle de temps (S_i) avec une durée exigée (l_i). L'intervalle de temps est défini par une paire $\langle \text{date}, \text{time} \rangle$, pour les intervalles de début et de fin, par exemple, ($S_i = \{\langle d_{start}, t_{start} \rangle, \langle d_{end}, t_{end} \rangle\}$). Une priorité est assignée à la réunion (w_i), en étant directement liée aux contraintes locales, pour être évaluée et résolue pendant la coordination. Facultativement, un ensemble de temps libre (T_i) est proposé par le *Host* en prévoyant de possibles intervalles disponibles pour la réunion.

Spécification du composant Organisation

Les exigences de coordination, liées aux contraintes organisationnelles (cf. Section 4.2.1), sont concernées par la description externe des rôles et la structure organisationnelle.

Le composant (\mathcal{O}) tient compte des relations sociales entre les participants pour représenter la hiérarchie organisationnelle et les responsabilités entre les activités selon les rôles joués par les agents. L'agent appartient à un groupe (un ensemble de rôles) qui définit la structure organisationnelle de la société d'agents. Nous supposons que la hiérarchie organisationnelle et la responsabilité influencent les préférences pendant une négociation pour trouver un consensus. En effet, le rôle qu'un agent joue dans un groupe (la hiérarchie organisationnelle) peut influencer la recherche

d'un *schedule* cohérent. Par exemple, une réunion avec le directeur présente une priorité plus élevée que celle avec un collègue, tandis que la responsabilité sur une tâche comme *PrepareLecture* peut être plus importante qu'une tâche *OrganizeDesktop*. Ainsi, dans l'Organisation (\mathcal{O}), une brique groupe/rôle générique gère la structure organisationnelle sur les groupes d'agents dans les Meetings (groupe thématique) et dans l'institution (priorité dans la hiérarchie).

Au niveau de la coordination, comme chaque agent est accessible aux autres participants par une description externe, des dépendances sociales [9, 10] sont détectées par rapport à leurs buts et leurs plans. De cette façon, la brique (\mathcal{O}) gère la formation des groupes thématiques. Par exemple, dans le contexte de l'emploi du temps, le rôle d'enseignant est décrit pour le comportement *TeachClassX*. Son plan initial est donc, *verifyCalendar(T), allocateRoom(R), meetingSchedule(X, R, T)*, où ce "contexte thématique" offre sa maximale priorité (hiérarchique).

4.4 Le problème de l'emploi du temps

Le problème consiste à concilier des contraintes (temporelles et de ressources) pour proposer un emploi du temps sur une certaine durée entre plusieurs acteurs (enseignantes et étudiants). Dans ce scénario, les buts sont préétablis en fonction d'un plan pédagogique et des contraintes portant sur :

- des disponibilités temporelles ;
- des compétences ;
- le besoin des ressources spécifiques (matériel pédagogique).

La variante de base à ce problème simplifie la mise en œuvre en proposant des créneaux horaires de 2h (8h-10h, 10h-12h, 14h-16h, 16h-18h), sur deux jours (j_1, j_2).

Trois enseignants e_1, e_2 et e_3 enseignent chacun une matière spécifique et leurs impossibilités d'enseignement sont les suivantes :

- e_1 ne peut enseigner le jour j_1 de 16h à 18h et le jour j_2 de 14h à 16h ;
- e_2 ne peut enseigner le jour j_2 de 10h à 12h et le jour j_1 de 16h à 18h ;
- e_3 ne peut enseigner le jour j_1 de 14h à 16h et le jour j_2 de 8h à 10h.

On considère trois groupes d'étudiants g_1, g_2 et g_3 . Chaque groupe d'étudiants (g_i) doit suivre un enseignement particulier, selon aussi un plan pédagogique préétabli, en ayant des contraintes sur le nombre des cours et de créneaux horaires. Pour cette variante du problème, chaque groupe doit suivre deux enseignements de 2h, de chacun des enseignants, sur deux jours ; c'est-à-dire, 12h d'enseignement pour chaque groupe.

On suppose que le système n'a pas à gérer la disponibilité des salles : on dispose d'une salle par groupe. On suppose également que les acteurs ne peuvent pas relâcher de contrainte.

4.4.1 Fonctionnement

Les agents sont les enseignants (e_i) et les groupes d'étudiants (g_j), ici représentés par leurs respectifs agents Agendas.

Lorsque chaque enseignant (e_i) fixe un engagement dans son Agenda, les dates choisies sont fixées sur son calendrier, tout comme ses contraintes ; par exemple, l'enseignant (g_1) est indisponible dans ces créneaux :

$$\neg available(e_1\{\langle j_1, 16 - 18 \rangle, \langle j_1, 12 - 14 \rangle, \langle j_2, 14 - 16 \rangle, \langle j_2, 12 - 14 \rangle\})$$

Pour l'expérimentation de ce scénario, dans le contexte général de l'Agenda, on fixe ces contraintes comme "personnelles", avec une priorité maximale qui évite le relâchement des contraintes.

A partir de là, on dira que les enseignants ont fixés des engagements passifs. C'est-à-dire, dans un premier temps, il n'y a pas encore une dépendance envers d'autres acteurs du type prise de rendez-vous. Cet état sera actif lorsque les groupes d'étudiants explicitent leurs désirs envers les enseignements disponibles.

Chaque groupe d'étudiants (g_j) choisit individuellement et fixe ses créneaux par rapport au cours et l'enseignant prévu. Comme notre approche est à la fois décentralisée et générale à ce scénario, c'est à chaque groupe que revient la responsabilité de fixer les contraintes. Ainsi, un groupe va fixer un engagement envers chaque enseignant tout en respectant sa contrainte individuelle de remplir 12h au total.

D'après le composant (\mathcal{O}) de notre approche de conception, les enseignants et les groupes d'étudiants font partie d'une organisation commune. Cette structure est représentée par la liste d'accointances des agents Agendas. Une fois qu'un engagement est fixé avec un membre de cette liste, automatiquement une contrainte ($\mathcal{A}-\mathcal{I}$) est déclenchée lui permettant de commencer un processus de négociation. D'après le protocole de négociation, chaque Agenda des groupes d'étudiants fait une offre en multicast (les enseignants concernés). Chaque Agenda des enseignants recevant l'offre commence à examiner son calendrier pour fixer le "rendez-vous" avec le groupe d'étudiants.

La planification évolue en tenant compte des contraintes au niveau des composantes multi-agents. Dans la phase de négociation, le module de coordination tient compte des contraintes envers les buts et les participants de la demande de rendez-vous. Dans notre cas, la négociation se tient à plusieurs entre chaque enseignant (e_i) et les groupes d'étudiants (g_j) en ce qui concerne les ressources (temps et salles). Dans ces cas, pour éviter des blocages successifs et garantir le consensus final (équilibre global), une contrainte organisationnelle désigne aux groupes d'étudiants que les enseignants forment un super ensemble E . Cet ensemble E , défini dans notre modèle comme un "groupe thématique", a la priorité sur les autres groupes du même niveau. Lorsqu'un groupe d'étudiants envoie un *MeetingRequest* à E , chaque agent du groupe E vérifie son calendrier (*VerifyCalendar*) et bloque les créneaux négociables. Après avoir trouvé un consensus auprès des groupes d'étudiants, les créneaux des enseignants sont débloqués et pris (si tel est le cas) et ils sont prêts à recevoir des nouvelles demandes.

Par exemple, le groupe g_1 établit son plan et l'agent déclenche une négociation avec les enseignants pour le plan suivant :

Groupe g_1 :

- **Buts** : {AttendClass(e_1), AttendClass(e_2), AttendClass(e_3)};
- **Plan initial** : { $j_{e_1, 08-10_i}$, $j_{e_2, 10-12_i}$, $j_{e_3, 14-16_i}$ };

Lorsque E reçoit ce plan, l'agent e_3 répond tout de suite à g_1 l'indisponibilité de $j_{e_3, 14-16_i}$. L'agent g_1 annule la négociation et établit un nouveau plan fondé sur un préclassement local. Son nouveau plan est :

Groupe g_1 :

- **Buts** : {AttendClass(e_1), AttendClass(e_2), AttendClass(e_3)};
- **Plan initial** : { $j_{e_1, 08-10_i}$, $j_{e_2, 14-16_i}$, $j_{e_3, 10-12_i}$ };

Dans ce deuxième temps, g_1 reçoit un *accept* de tous les enseignants, E , et procède à la suite du plan *ConfirmMeeting* et *AppendMeeting*, quand E aussi finalise *AppendMeeting* et débloque les créneaux.

4.5 Conclusion

Nous avons illustré la spécification d'un problème d'emploi du temps à travers une application de gestion d'agendas distribués. A son tour, le gestionnaire d'agendas est basée sur un modèle de coordination multi-agents conçue selon l'approche Voyelles. Notre spécification n'était pas dirigée à ce problème et pour cela nous présentons la spécification de la gestion d'agendas et comment nous l'avons appliquée à ce scénario spécifique.

La variante pour l'utilisation des salles et le relâchement des contraintes peut être appliquée, tout en modifiant le statut des préférences de l'utilisateur. On marque alors les plages indisponibles comme publiques avec différents degrés d'importance par rapport aux autres engagements de l'agent. Ainsi, de façon systématique, le mécanisme de coordination prend en compte aussi ces plages, définies comme *occupées* au niveau de l'environnement et non plus *indisponibles*. Ces plages, avec une importance inférieure par rapport aux engagements des groupes d'étudiants, seront sujettes à de nouvelles négociations pour un possible déplacement (relâchement des contraintes). Pour les salles, le plus simple est de définir un gestionnaire de salles, représenté par un agent Agenda. Il sera un élément de plus dans les négociations et ajoutera des nouvelles contraintes dans le mécanisme de coordination.

Bibliographie

- [1] J.L. Tavares da Silva and Y. Demazeau. Vowels co-ordination model. In Cristiano Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 1129–1136, Bologna, Italy, 2002. ACM Press.
- [2] Keith Decker and Victor Lesser. Generalized partial global planning. *International Journal on Intelligent Cooperative Information Systems*, 1(2) :319–346, June 1992.
- [3] Y. Demazeau. Steps towards multi-agent oriented programming. In *1st International Workshop on Multi-Agent Systems (IWMAS'97)*, Boston, 1997.
- [4] Yves Demazeau. From cognitive interactions to collective behavior in agent-based systems. In *Proceedings of the First European conference on cognitive science*, Saint Malo, France, April 1995.
- [5] Heecheol Jeon, Charles Petrie, and Mark R. Cutkosky. JATLite : A java agent infrastructure with message routing. *IEEE Internet Computing*, 4(2) :87–96, 2000.
- [6] Jean-Luc Koning and Marc-Philippe Huet. A component-based approach for modeling interaction protocols. In H. Kangassalo, H. Jaakkola, and E. Kawaguchi, editors, *Proceedings of 10th European-Japanese Conference on Information Modelling and Knowledge Bases*, Saariselkä, Finland, May 8–11 2000. IOS Press.
- [7] Frank von Martial. *Coordinating plans of autonomous agents*, volume 610 of *Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1992.
- [8] S. S. Sian. Adaptation based on cooperative learning in multi-agent systems. 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 257–272, Amsterdam, The Netherlands, 1991. Elsevier Science Publishers B.V.
- [9] Jaime Simão Sichman, Rosaria Conte, Cristiano Castelfranchi, and Yves Demazeau. A social reasoning mechanism based on dependence networks. In A. G. Cohn, editor, *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pages 188–192, Chichester, August 8–12 1994. John Wiley and Sons.
- [10] Jaime Simão Sichman and Yves Demazeau. Exploiting social reasoning to deal with agency level inconsistency. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 352–359, San Francisco, CA, USA, June 12–14, 1995. AAAI Press.

Chapitre 5

Résolution par une métaphore émotionnelle

Pierre De Loor – Pierre Chevallier
Laboratoire d'Ingénierie Informatique
Ecole Nationale d'Ingénieurs de Brest,
Technopôle Brest Iroise,
BP 30815 ; 29608 Brest Cedex, France,
[deloor,chevallier]@enib.fr,
<http://www.enib.fr/LI2>

Nous proposons une technique de résolution distribuée de problèmes utilisant des heuristiques basée sur la métaphore d'agents émotionnels. Cette métaphore s'inspire du comportement d'un groupe d'humains plus ou moins susceptibles. Ils sont capables de s'engager pour répondre à une requête diffusée dans "l'environnement", mais également capables d'annuler leurs décisions si la "tension" globale monte. Ils possèdent donc une perception de l'ambiance globale (niveau d'avancement de la résolution) au travers des messages de requêtes circulant dans l'environnement. Ils possèdent également une perception de leur influence sur cette résolution. Cette métaphore permet une résolution du "problème des emploi du temps" dynamique et donc adaptative.

5.1 Contexte

5.1.1 Rappel du problème

Cette section a pour objectif de rappeler brièvement le problème : définir un emploi du temps concernant des groupes d'élèves devant suivre une série de cours de différentes matières. Chaque professeur enseigne une matière et possède des créneaux horaires durant lesquels il est disponible. Ce problème est réputé difficile car non polynomial. De plus, nous cherchons à obtenir une résolution distribuée, adaptative et dynamique du problème. Il faut que les différents intervenants puissent, durant la résolution, modifier leurs contraintes (changement de disponibilité des professeurs par exemple, apparition d'un nouveau groupe d'élèves ...). L'algorithme doit alors adapter la solution en cours et ne pas recalculer entièrement celle-ci.

5.1.2 Positionnement de notre approche

Il existe différents algorithmes distribués de résolution [DR94], [Yok01]. Ceux-ci sont généralement caractérisés par une communication entre des variables proposant des valeurs. Ces valeurs peuvent être refusées par d'autres variables en fonction des contraintes qui les relient ; ce mécanisme est un retour-arrière asynchrone (*Asynchronous Backtracking Algorithm*). D'autres

algorithmes procèdent par évaluation de la *valeur* d'une solution (nombre de contraintes résolues pour une affectation aléatoire de valeur) et tentent de minimiser celle-ci par le biais du calcul d'états *voisins* (ces algorithmes ne sont pas sans rappeler la technique de recuit simulé ou de *hill-climbing*, mais ils sont distribués) : ce sont les algorithmes "*Asynchronous Weak-Commitment*" et "*Distributed BreakOut*". D'autres algorithmes proposent une analyse pré-alable des contraintes (*Distributed Consistency Algorithm*) de restreindre l'espace de recherche avant le début de celle-ci. Les algorithmes "*Handling Multiple Variables*" et "*Handling Over-Constrained Situations*" généralisent ces approches à la gestion d'un ensemble de variables par un agent et aux systèmes sur-contraints. Lorsque les interdépendances entre les variables s'accroissent, la résolution de problèmes est abordée sous un angle différent. On s'intéresse davantage à l'autonomie de décision des agents et aux mécanismes de négociation (*contract net protocol* [SD81]) qu'à une résolution systématique et déterministe. Ils utilisent alors des métaphores comme par exemple dans [WYHW01], où l'attribution de tâches se fait par le biais d'une vente aux enchères organisée entre agents. Un autre principe couramment utilisé est celui de l'éco-résolution [Fer99] comme par exemple dans [GCG99] qui propose une méthodologie basée sur l'identification de situation non coopératives.

Notre approche peut être vu comme une technique d'éco-résolution appliquée à la résolution de contraintes, elle se distingue par les points suivants :

- Les heuristiques utilisées dans les approches CSP classiques sont basées sur des critères mathématiques *a priori*. En particulier, les critères définissant le déclenchement d'un retour-arrière sont définis quantitativement et de façon déterministe, ils correspondent souvent à un nombre de contraintes en échec et n'associent aucune sémantique à celles-ci. Les algorithmes de négociation sont généralement destinés à la recherche d'une solution à des problèmes consistants, pour lesquels un retour arrière n'est pas nécessaire. Notre hypothèse est qu'un aspect qualitatif appuyé par une métaphore émotionnelle pourrait améliorer la recherche de solutions dans des problèmes non-consistants. Elle se justifie par le constat qu'un groupe d'humain est capable de résoudre ce genre de problème de façon pragmatique. J.F Dorcier [Dor99] préconise d'ailleurs l'utilisation de critères psychologiques pour contribuer à l'amélioration de la résolution de problèmes en intelligence artificielle.
- Nous cherchons à obtenir une solution adaptative, "en-ligne", acceptant toute modification des contraintes (nouveaux cours, retraits d'enseignants, ...), sans ré-initialiser la recherche de solution, mais en adaptant la solution actuelle.
- Nous ne cherchons pas une solution optimale. Le problème peut d'ailleurs être non solvable. L'objectif est de trouver une solution "acceptable", voire d'évaluer des facteurs permettant d'orienter la modification du problème pour que celui-ci devienne solvable.
- Nous cherchons à obtenir une solution dont la mise en oeuvre limite les mécanismes d'acquiescement ou de synchronisation. Nous privilégions la diffusion générale et l'absence d'acquiescement. La complexité des algorithmes de négociation est souvent liée à la nécessité d'acquiescement qui provoque une synchronisation de la résolution, une mémorisation et une connaissance de l'état des différents agents et souvent une centralisation de l'information par un agent particulier (commissaire priseur dans la vente aux enchère par exemple). Sans acquiescement, les algorithmes locaux des agents sont simplifiés, la parallélisation est améliorée (symétrie de comportement) et le système est plus robuste et adaptatif. En contrepartie, il faut mettre en place un mécanisme de régulation des émissions des requêtes. Notre approche propose que ce mécanisme soit propre à chaque agent et calculé en fonction de critères émotionnels individuels.

5.2 Principe de résolution

Nous proposons une résolution basée sur des agents dits *émotionnels* car s'appuyant sur un modèle comportemental intégrant la notion de *charge cognitive* et de *seuil de tolérance*. La *charge cognitive* dépend de la perception des agents et de leurs actions. Ici, la perception correspond à la réception de messages et les actions sont la recherche d'une solution à un but requis par les autres agents. Les agents possèdent donc des buts (ou *problèmes*) personnels et des *compétences*

relatives à ces buts. Ils peuvent envoyer et recevoir des requêtes relatives à ces buts. Lorsqu'un agent perçoit une requête concernant un but dont il a la compétence, cette requête devient pour lui un *but requis*. S'il peut l'atteindre, il diffusera une proposition. Les contraintes correspondent à des conflits possibles entre les *buts personnels* et les *buts requis*. Lorsque la *charge cognitive* d'un agent dépasse le *seuil de tolérance*, celui-ci provoque une *crise*. Elle correspond à une remise en cause des engagements pris par l'agent. Cette remise correspond à un retour arrière partiel. Pour ce faire, il diffuse des messages d'annulation de ses engagements.

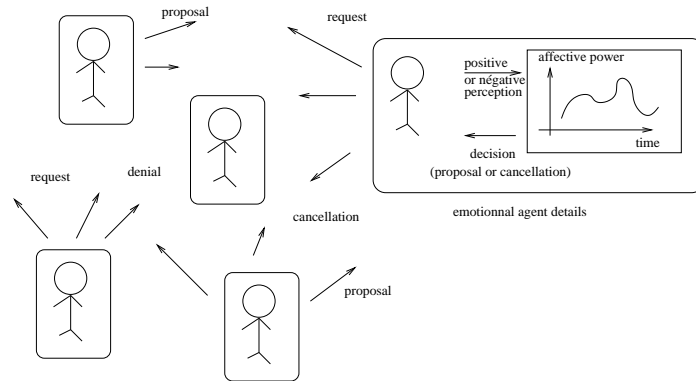


FIG. 5.1 – Principe

La métaphore émotionnelle permet d'effectuer un mécanisme de retour-arrière local appliqué aux agents les plus *gêneurs*. Pour favoriser la recherche de solutions dans le cas où plusieurs agents seraient gêneurs, voire où la situation relative de chacun d'eux provoquerait un (ou des) inter-blocage(s), deux facteurs importants sont intégrés dans le mécanisme de résolution :

- la perception de la charge cognitive globale : Les messages émis par les agents sont diffusés et perçus par l'ensemble des autres agents. La perception d'un message de requête ou de refus augmente la charge cognitive, cette charge est d'autant plus accrue que l'agent est concerné par le message et qu'il ne peut y répondre.
- la relaxation de la charge cognitive locale : lors d'une crise, la charge cognitive de l'agent concerné est annulée. Ceci laisse la possibilité, en cas de blocage fort, que d'autres agents passent leur propre seuil de tolérance et entrent en crise à leur tour. Plus techniquement, cette relaxation oriente le retour-arrière de la recherche en fonction d'une heuristique basée sur la notion de *hiérarchie dynamique des gêneurs*.

Ainsi, le mécanisme de recherche de la solution possède un comportement dynamique auto-adaptatif, scrutant les solutions permettant de réduire la charge cognitive globale des agents. Lorsqu'une solution existe et qu'elle est trouvée, les charges cognitives des différents agents atteignent leur minimum. Lorsqu'il n'existe pas de solution, un résultat partiel est tout de même obtenu et correspond à la recherche d'une diminution de la charge cognitive globale.

5.2.1 dynamique de la charge cognitive

La charge cognitive dépend de la perception et des actions de l'agent. Perception et actions peut être décomposée selon deux types d'information (figure 5.1) :

- les informations positives : elles diminuent la charge cognitive d'un agent. Elles peuvent être décomposées de la façon suivante :
 - le *succès personnel* : l'agent a réussi à résoudre un (ou des) but(s), personnel ou requis.
 - la *décharge de responsabilité* : l'agent fait une requête car il ne peut résoudre un but personnel. Dans ce cas, il se repose sur la communauté. Précisons que contrairement à certains protocoles de type *contract net protocol*, l'agent n'opère cette décharge que pour ses buts personnels et non sur des buts requis.

- le *succès coopératif* : un agent lui a fait une proposition qui permet de résoudre un de ses buts personnels.
- la *crise* : l’agent annule toutes les solutions qu’il avait auparavant proposées. La crise est un évènement particulier qui va provoquer la remise à zéro de la charge cognitive de l’agent et une remise en question de la solution en cours de recherche.
- les informations négatives : elles augmentent la charge cognitive d’un agent. Elles peuvent être décomposées de la façon suivante :
 - les *requêtes* : toute requête est un signe que “quelque chose n’est pas résolu au sein du système multi-agents” et va augmenter la charge cognitive. Si l’agent peut répondre à cette requête, le *succès personnel* consécutif (voir les informations positives) diminue la charge cognitive. S’il n’est pas compétent pour répondre à cette requête, il ne fait rien. S’il est compétent mais qu’il ne peut résoudre le problème lié à la requête, il en résulte un *échec personnel*.
 - les *échecs personnels* : l’agent est incapable de résoudre un problème, qu’il lui soit propre ou posé par un autre agent.
 - les *refus* : l’agent perçoit les refus diffusés par les autres agents. Il distingue les refus le concernant directement (annulant une proposition qu’il a faite) et qui accroissent davantage sa charge cognitive que les refus ne le concernant pas. Cependant le fait qu’elle croisse dans les deux cas reflète la perception de la “mauvaise ambiance” générale au sein du système (ou de la communauté d’agents). Cette proposition part de la constatation que la communication indirecte peut être un mécanisme fondamental lors de la résolution de problèmes [JDS00].

La figure 5.2 représente un exemple d’évolution de la charge cognitive d’un agent émotionnel durant la résolution d’un problème.

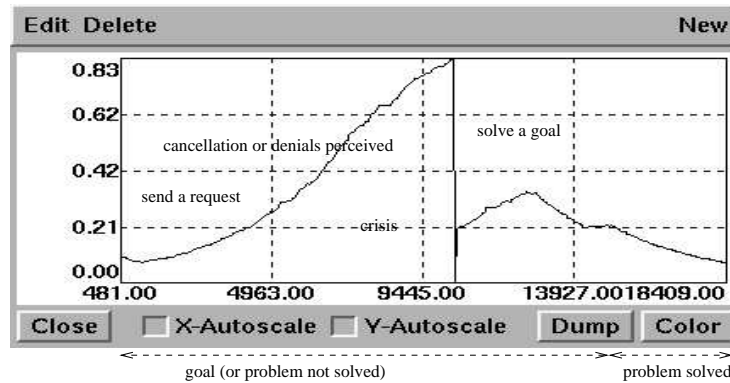


FIG. 5.2 – Exemple d’évolution de la charge cognitive d’un agent.

5.3 L’algorithme distribué

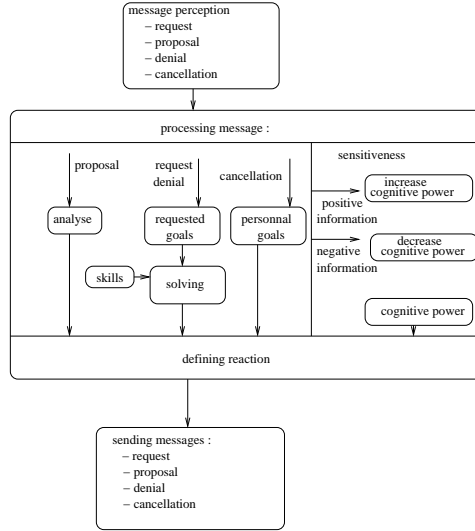
L’algorithme de résolution est distribué au sein de chaque agent sous la forme d’une boucle perception-décision-action telle que celle représentée en figure 5.3. Un traitement particulier est dédié à chaque type de message reçu. La charge cognitive est modifiée en fonction de ces traitements et du comportement par défaut de l’agent.

5.3.1 Modèle d’un agent émotionnel

Un agent émotionnel A est formalisé par un 8-uplet :

$$\langle \psi^A, \alpha_s^A, \alpha_c^A, \rho^A, \gamma^A, S_s^A, R_g^A, P_g^A \rangle$$

- $\psi^A \in [0,1]$ est un réel caractérisant la *charge cognitive*.

FIG. 5.3 – *emotional agent newel loop.*

- $\alpha_s^A \in]0,1[$ est un réel représentant le *taux de sensibilité personnel*.
- $\alpha_c^A \in]0,1[$ est un réel représentant le *taux de sensibilité collective*.
- $\rho^A \in]0,1[$ représente le *seuil de requête requirement*.
- $\gamma^A \in]0,1[$ représente le *seuil de crise* ($\gamma^A > \rho^A$).
- S_s^A est l'*ensemble de compétences*. (informations symboliques).
- R_g^A est l'*ensemble des buts requis*.
- P_g^A est l'*ensemble des buts personnels*.

5.3.2 Modèle d'un but

Un but g est un 3-uplet $\langle P^g, S^g, A^g, V^g \rangle$:

- P^g est un prédicat du premier ordre exprimant le but. Cette expression symbolique dépend du problème.
- S^g est la compétence requise pour atteindre ou *améliorer* le but (résoudre P^g).
- A^g est la solution totale ou partielle permettant d'atteindre le but. Cette solution sera initialement vide et sera progressivement enrichie par les différents intervenants capables de contribuer à la résolution du but.
- V^g est un booléen correspondant à la valeur de P^g .

5.3.3 Modèle d'un message

Un message M est un 3-uplet $\langle T^M, E^M, G^M \rangle$:

- $T^M \in \{\text{'request'}, \text{'proposal'}, \text{'cancellation'}, \text{'refusal'}\}$ est le type de message.
- E^M est l'agent émetteur du message.
- G^M est un but (le sujet du message) .

5.3.4 Comportement par défaut

Le comportement par défaut d'un agent émotionnel A est exécuté tant qu'aucun message n'est reçu :

$$\forall \text{ goal } g \in P_g^A$$

if ($V^g == false$)
 $\psi^A = \psi^A + \alpha_s^A * (1 - \psi^A)$

if ($\psi^A > \rho^A$) (requirement)
 \forall goal $g \in P_g^A$
 if ($V^g == false$)
 broadcasting the message $\langle 'request', A, g \rangle$
 $\psi^A = \psi^A - (\alpha_s^A * \psi^A)$

if ($\psi^A > \gamma^A$) (crisis)
 \forall goal $g \in R_g^A$
 broadcasting the message $\langle 'cancellation', A, g \rangle$
 $R_g^A = \phi$
 $\psi^A = 0$

$\psi^A = \psi^A - f(\alpha_s^A, \psi^A)$ (default relaxation)

Résumer : chaque but personnel non atteint accroît la *charge cognitive* proportionnellement à sa *sensibilité personnelle* (α_s^A). Si sa *charge cognitive* dépasse le *niveau de requête*, des requêtes relatives à ces buts personnels sont envoyées. Dans ce cas, la *charge cognitive* diminue. Un tel mécanisme évite toute procédure d'acquiescements souvent coûteuse en message, en temps d'attente et en mémorisation. En effet, c'est la *charge cognitive* qui régule le lot des requêtes. En fait plusieurs politiques peuvent être utilisées. Par exemple, l'envoi d'une requête peut être associée à une probabilité dépendant de la *charge cognitive* et du *seuil de requête*. Il est aussi possible de n'envoyer qu'une requête pour un but tiré aléatoirement plutôt qu'un ensemble de requêtes pour chacun de ces buts. L'influence de la politique sur les performances de l'algorithme reste actuellement l'objet d'études. Quand la *charge cognitive* dépasse le *seuil de crise*, l'agent annule tous ses engagements envers les autres. Pour cela, il diffuse un message d'annulation relatif aux buts incluant dans l'ensemble de ses buts requis. Dans ce cas, sa charge cognitive est remise à zéro. Ainsi, si d'autres agents ont une *charge cognitive élevée*, ils pourront entrer en crise avant que le précédent n'y entre à nouveau. Ce mécanisme peut être vu comme un retour-arrière local qui passe parmi les agents les plus généreux à un instant donné de la résolution. Ce retour-arrière, basé sur une métaphore psychologique, introduit une hiérarchisation dynamique de généreux. Enfin, f est une fonction positive reflétant la tendance naturelle à l'apaisement de la *charge cognitive* lorsqu'il n'y a pas de problème particulier. (La définition de cette fonction sera décrite plus tard). Elle est importante pour évaluer la convergence de l'algorithme : lorsque la *charge cognitive* de chaque agent décroît, c'est qu'une solution est trouvée.

5.3.5 traitements des messages

– traitement d'un message $\langle 'request', A', g \rangle$
 if ($A \neq A'$) \wedge ($S^g \notin S_s^A$)
 $\psi^A = \psi^A + \alpha_c^A * (1 - \psi^A)$ (climat social)
 else
 solving(P^g)
 if P^g is soluble with a solution s
 $A^g = s$
 $V^g = true$
 $R_g^A = R_g^A \cup g$
 broadcasting the message $\langle 'proposal', A, g \rangle$
 $\psi^A = \psi^A - (\alpha_c^A * \psi^A)$

Résumer : si l'agent en a la compétence, il tente de résoudre le but grâce à une méthode abstraite *solving* dépendante du problème. Le succès de cette méthode retourne une solution symbolique telle que définie dans le modèle des buts (voir 5.3.2 et l'exemple).

- traitement d'un message $\langle 'proposal', A', g \rangle$
 - if $(A \neq A') \wedge \{\exists g' \in P_g^A | P^g == P^{g'}\}$
 - $analyse(A^{g'})$
 - if ($A^{g'}$ is acceptable)
 - $A^g = A^{g'}$
 - $V^{g'} = true$
 - $\psi^A = \psi^A - (\alpha_s^A * \psi^A)$
 - else
 - broadcasting the message $\langle 'refusal', A, g \rangle$
 - else
 - $\psi^A = \psi^A - (\alpha_c^A * \psi^A)$ (public mood)

Résumer : un agent A est concerné par une proposition si celle-ci résulte d'une de ses requêtes. Cependant, la *charge cognitive* d'un agent est décrémentée même s'il n'est pas concerné par cette proposition. Ceci reflète sa perception du *climat social*. Avant d'accepter une proposition, l'agent doit analyser celle-ci par le biais d'une méthode abstraite dépendante du problème.

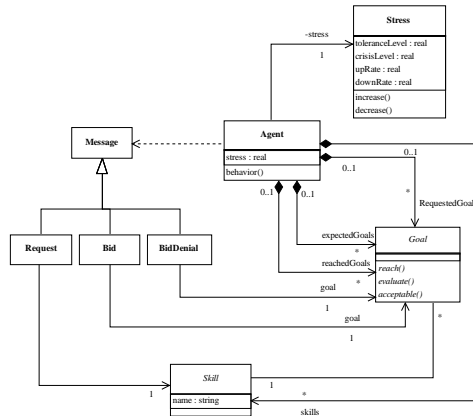
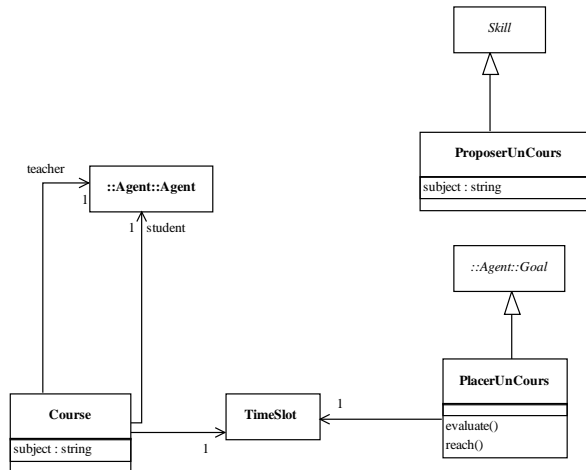
- traitement d'un message $\langle 'refusal', A, g \rangle$
 - if $(A \neq A') \wedge \{\exists g' \in R_g^A | P^g == P^{g'}\}$
 - $\psi^A = \psi^A + (\alpha_s^A * \psi^A)$
 - $R_g^A = R_g^A - g'$
 - else
 - $\psi^A = \psi^A + (\alpha_c^A * \psi^A)$ (public mood)
- traitement d'un message $\langle 'cancellation', A, g \rangle$
 - if $\{\exists g' \in P_g^A | P^g == P^{g'}\}$
 - $\psi^A = \psi^A - (\alpha_s^A * \psi^A)$
 - $V_g' = false$
 - else
 - $\psi^A = \psi^A - (\alpha_c^A * \psi^A)$ (public mood)

5.4 Implementation

Nous avons procédé à une implémentation des agents émotionnels grâce à la plateforme **oRis** et avons instancié cette implémentation au problème des emplois du temps. **oRis** est un langage interprété intégrant un simulateur et permettant la spécification rapide d'objets actifs communicants. Il est particulièrement adapté à la l'implémentation d'architecture d'agent et renvoyons le lecteur à [HTRC02] pour plus de précision à son sujet. La figure 5.4 représente le diagramme de classe des agents émotionnels.

5.4.1 Application au problème des emplois du temps

Le problème des emplois du temps a fait l'objet de différentes études [Mar98], [KAD97]. Notre approche se distingue de celle-ci par le fait que la recherche est distribuée, complètement asynchrone et adaptative. Par contre, contrairement à [KAD97] nous ne proposons pas pour l'instant de démarche d'optimisation qui fait l'objet de nos recherches actuelles. La figure 5.5 résume la spécialisation des classes des agents émotionnels pour ce problème.

FIG. 5.4 – *emotional Agent's class.*FIG. 5.5 – *TimeTable Class.*

Groupe d'élèves

Les élèves possèdent les attributs suivants :

- **Buts personnels** : la recherche d'un professeur et d'un horaire pour une liste de cours à faire. Ils possèdent donc une liste de ces cours (dont chacun porte sur une matière) et tant qu'un professeur et un horaire n'est pas affecté à un cours, celui-ci est considéré comme un but non atteint.
- **Buts requis** : aucun.
- **Compétences** : aucune (dans le cadre de la résolution de ce problème).
- **Messages Emis**
 - **requête de cours** : il s'agit d'un message réclamant un professeur et un horaire pour un cours d'une matière définie.
 - **refus cours** : si une proposition est faite mais qu'ils ne peuvent l'accepter (car le créneau est déjà pris ou qu'un autre professeur a fait une offre), ils diffusent un refus concernant ce cours.

Tous les messages sont perçus par les élèves (et influent sur leur charge cognitive), mais certains d'entre-eux impliquent un traitement particulier :

- Proposition de cours
- Annulation de cours

Le diagramme de classe complet d'un `GroupeEleve` sera prochainement fournit.

Professeurs

Les professeurs possèdent les attributs suivants :

- **Buts personnels** : aucuns
- **Buts requis** : l'affectation d'un professeur et d'un horaire à un cours.
- **Compétences** : capacité à faire un cours d'une matière donnée.
- **Messages Emis**
 - **proposition de cours** : si une requête de cours est faite par un groupe d'élève, que le professeur est compétent pour faire ce cours et qu'il possède un crénaux de disponible, il envoit une proposition.
 - **annulation de cours** : l'annulation d'un cours correspond aux messages d'annulation d'engagement, émis en cas de crise.

Tous les messages sont perçus par les professeurs (et influent sur leur charge cognitive), mais certains d'entre-eux impliquent un traitement particulier :

- Requête d'un cours
- Refus d'un cours

Le diagramme de classe complet d'un `GroupeEleve` sera prochainement fournit.

5.4.2 Resultats

Nous avons appliqué l'algorithme sur un problème dont nous avons fait croître la complexité. En fait, la complexité d'un tel problème n'est pas facile à définir. Elle dépend certe du nombre de cours à faire, mais essentiellement du rapport entre la taille de l'espace de recherche et le nombre de solutions acceptables dans cet espace. En d'autres termes, le nombre de cours ou le nombre d'élèves accroît la difficulté du problème, mais le nombre de professeurs la décroît et surtout leurs disponibilité commune (crénaux horaire dispo commun à chaque professeurs) l'accroît. De même, si la résolution du problème implique que tous les crénaux disponibles de tous les professeurs sont pris, le problème est plus difficile que s'il leur reste un nombre de crénaux disponible à la fin de la résolution. Nous avons donc testé notre algorithme pour différents cas. Généralement, nous tentons de placer 80 cours, concernant 3 à 4 groupes d'élèves et 3 à 4 professeurs. Le temps de convergence dépend fortement des crénaux disponibles et communs des professeurs. La figure 5.6 montre l'évolution des charges cognitives des agents durant la résolution d'un de ces problème. Les temps indiqués en abscisse sont des temps logiques. Quand toutes les charges cognitives décroissent, le problème est résolu. Nous avons implémenté l'algorithme dans une futur version d'oRis beaucoup plus rapide (C++) et pouvons placer par exemple 160 cours en quelques secondes sur un problème difficile (relativement à la définition donnée précédemment).

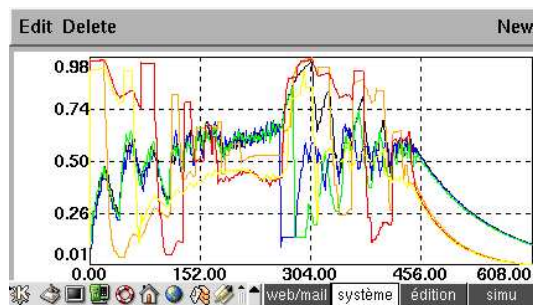


FIG. 5.6 – *cognitive power evolution of the agent during the timetabling problem solving.*

Nous pouvons également montrer la dynamicité offerte par oRis : sur la figure 5.7 une solution a été trouvée puis, successivement, durant la simulation, nous avons

- ajouté un cours à faire. Ceci augment la charge cognitive des élèves, ce qui provoquera une réaction des professeurs et une nouvelle succession de crises. Le problème n’avait alors plus de solution.
- ajouté des créneaux disponibles pour un professeur. Dans ce cas, la solution en cours s’améliore rapidement et l’algorithme converge à nouveau vers un emploi du temps correct.

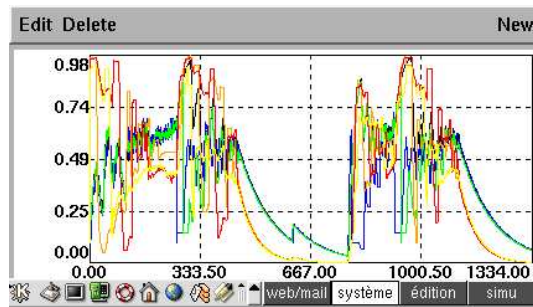


FIG. 5.7 – *dynamic adaptation.*

5.5 Conclusion et travaux futurs

Nous avons proposé l’élaboration d’un modèle “d’agents émotionnels” pour la résolution distribuées de problèmes, appliqué au problème des emploi du temps distribué. Montré que les résultats sont convainquant. Dernièrement, nous avons implémenté un des algorithmes distribué réputé pour être très efficace (l’AWC [Yok01]). Nous pouvons distinguer deux éléments importants permettant de mesuer l’apport de notre approche :

- Nous obtenons des performances équivalentes voir meilleures selon certains cas. Nous expliquons ce résultat par le faite que l’AWC est un algorithme de recherche systématique, certe optimisé, mais pas à l’aide de critères métaphoriques mais sur des critères purement qualitatifs n’apportant aucune sémantique particulière aux contraintes et aux agents du problème.
- La modélisation du problème par le biais de notre approche est naturelle et se prête à la dynamicité du problème. Au contraire dans une approche de type AWC, la recherche des variables et des contraintes est une étape complexe. Le problème est d’abord vu globalement avant d’être distribué... Ceci implique qu’une modification dynamique du problème nécessite une redéfinition du problème global.

Notons également que si le problème est sans solution, notre algorithme trouve tout de même une solution incomplète et oscille progressivement pour tenter de l’améliorer sans arrêt. Ceci permettra une convergence très rapide dès qu’une relaxation de contrainte sera effectuée. Nous travaillons sur différents points :

- l’auto-relaxation de contraintes par les agents lorsqu’aucune solution ne semble émerger.
- l’auto-régulation du stress par le biais de méthodes d’apprentissage, permettant de modifier automatiquement les paramètres de l’algorithme pour optimiser la vitesse de convergence de celui-ci quel que soit le problème.
- l’élaboration d’un langage plus générique de spécification des différents messages, un peu comme proposé par [TT01]

Bibliographie

- [Dor99] J. F. Dortier. Espoirs et réalités de l'intelligence artificielle. *Le cerveau et la pensée*, pages 69–77, 1999.
- [DR94] E. H. Durfee and J. Rosenschein. Distributed problem solving and multiagent systems : Comparisons and examples. In M. Klein, editor, *Proceedings of the 13th International Workshop on DAI*, pages 94–104, Lake Quinalt, WA, USA, 1994.
- [Fer99] J. Ferber. *Multi-Agent System : An Introduction to Distributed Artificial Intelligence*. Harlow : Addison Wesley Longman, 1999.
- [GCG99] M.P. Gleize, V. Camps, and P. Glize. A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. In *Proceedings of the Fourth European Congress on Systemic*, Valence, 1999.
- [HTRC02] Fabrice Harrouet, Jacques Tisseau, Patrick Reignier, and Pierre Chevaillier. oRis : un environnement de simulation interactive multi-agents. *Revue des sciences et technologie de l'information, série Technique et science informatiques (RSTI-TSI)*, 21(4) :499–524, 2002.
- [JDS00] B. Pavard J. Dugale and J.L. Soubie. A pragmatic development of a computer simulation of an emergency call centre. In Rose Dieng et al., editor, *Designing Cooperative Systems, Frontiers in Artificial Intelligence*. IOS Press, 2000.
- [KAD97] Rainer Staudte Kay Anke and Werner Dilger. Producing and improving time tables by means of constraint and multi-agent systems. In *The AAAI-97 Workshop on Constraints and Agents*, 1997.
- [Mar98] Hugo Terashima Marín. Combinations of gas and csp strategies for solving the examination timetabling problem, 1998.
- [SD81] R. G. Smith and R. David. Frameworks for cooperation in distributed problem solving. 11(1) :61–70, June 1981.
- [TT01] T. Torroni and Francesca Toni. Extending a logic based onto-toone negotiation framework to one-to-many negotiation. In P. Omicini, A. Petta and R. Tolksdorf, editors, *LNAI*, 2203, pages 105–118, Berlin, 2001. ESAW 2001, Springer-Verlag.
- [WYHW01] William E. Walsh, Makoto Yokoo, Katsutoshi Hirayama, and Michael P. Wellman. On market-inspired approaches to propositional satisfiability. In *IJCAI*, pages 1152–1160, 2001.
- [Yok01] Makoto Yokoo. *Distributed Constraint Satisfaction*. Springer, 2001.

Chapitre 6

L'emploi du temps par MAMOSACO

Emmanuel ADAM Rene MANDIAU
Groupe RAIHM - LAMIH UMR CNRS 8530
Université de Valenciennes et du Hainaut Cambrésis
Le Mont Houy, 59309 Valenciennes - FRANCE

6.1 Introduction

Un système multi-agent peut être qualifié d'Organisation Multi-Agent (OMA) lorsque les agents qui le composent possèdent une connaissance des autres et qu'ils suivent des règles fixées par le système global tout en gardant leurs capacités d'autonomie ([2]; [6]). Il est possible de distinguer actuellement deux modèles d'organisations multi-agents : les organisations émergentes dont la structure émerge d'actions individuelles des agents qui la composent, et les organisations à support d'activités dont la structure plus ou moins flexible et adaptative est prédéfinie.

Depuis plusieurs années, nos recherches s'orientent vers l'étude et la conception d'OMA ([5]; [4]; [1]) dans deux grandes familles d'applications : dans le contexte des transports terrestres et dans la modélisation des organisations humaines. Afin de répondre au cahier des charges proposé, nous axons cet article sur la problématique liée aux organisations humaines.

La spécification d'une organisation agent adaptée à une organisation humaine est une démarche pluridisciplinaire. En effet, il est nécessaire, afin de mieux appréhender la complexité des organisations humaines, de s'inspirer des résultats de disciplines telle que la sociologie. Et, afin de représenter, de mettre à plat, le fonctionnement de l'organisation humaine et multi-agent, les outils et méthodes du génie logiciel sont bien entendu indispensables.

L'étude et la recherche du modèle organisationnel général applicable à la plupart des systèmes complexes que nous avons effectuées ont mené au choix du modèle holonique. Ce modèle définit tout système complexe en tant que hiérarchie flexible, c'est-à-dire en tant que structure pyramidale dont les membres possèdent des niveaux de responsabilités et une relative autonomie. Les règles organisationnelles définissant ce type de système peuvent être intégrées dans les organisations multi-agents. Le système multi-agent peut donc ainsi posséder une structure proche de l'environnement d'application (l'organisation humaine).

Afin de spécifier les organisations multi-agents holonique (OMAH), nous avons défini une méthodologie basée sur la méthode MAMOSACO, Méthode Adaptable de Modélisation de Systèmes Administratifs COMplexes. Cette méthode, développée pour permettre la modélisation et la simulation du fonctionnement des organisations administratives complexes permet de représenter les coopérations entre les agents d'une OMAH et les acteurs d'une organisation humaine. Nous proposons de définir l'OMAH en deux étapes, une étape de description des caractéristiques individuelles des agents jouant un rôle dans l'OMAH, et une étape de

spécification des processus se déroulant au sein de l'OMAH et entre l'OMAH et l'organisation humaine dans laquelle elle évolue.

Suite au rappel du problème de l'emploi du temps, nous présenterons notre méthodologie basée sur la méthode MAMOSACO. Enfin, les résultats issus de l'application de notre méthodologie sont présentés.

6.2 Le problème de l'emploi du temps

Rappelons brièvement le contexte de cet article. Dans un objectif de recenser et des tester les plateformes et les méthodologies orientées multiagents existante, il a été proposé de les confronter sur un ensemble de problème dont celui de la gestion d'un emploi du temps par un ensemble de trois professeurs. Ce sujet, proposé par Carole Bernon, Marie-Pierre Gleizes, Pierre Glize et Gauthier Picard (de l'équipe SMAC de l'IRIT, Toulouse) peut être résumé par ces conditions en ce qui concerne la variante 1 :

- Trois enseignants (e1, e2 et e3) doivent proposer chacun deux enseignements de deux heures, répartis sur deux jours (j1 et j2), à trois groupes d'étudiants (g1, g2, g3).
- Les créneaux sont donc des créneaux de deux heures, s'étalant sur deux jours, de 8h à 18h, avec une pause de deux heures entre 12h et 14h.
- Les enseignants possèdent ces contraintes :
 - e1 ne peut enseigner le jour j1 de 16h à 18h et le jour j2 de 14h à 16h,
 - e2 ne peut enseigner le jour j2 de 10h à 12h et le jour j1 de 16h à 18h,
 - e3 ne peut enseigner le jour j1 de 14h à 16h et le jour j2 de 8h à 10h.

En ce qui concerne la variante 2, on suppose que trois salles s1, s2 et s3 sont disponibles pour ces enseignements, cependant :

- la salle s1 n'est pas disponible le jour j1 de 10h à 12h,
- la salle s2 n'est pas disponible le jour j2 de 16h à 18h et de 8h à 10h,
- la salle s3 n'est pas disponible le jour j2 de 16h à 18h et le jour j1 de 14h à 16h.
- Seules les salles s1 et s2 possèdent un rétroprojecteur.
- Tous les enseignants veulent utiliser un rétroprojecteur au moins une fois pour chaque groupe sur les deux jours.
- Les contraintes les moins fortes (citées en second lieu) peuvent être levées.

6.3 Spécification du SMA répondant à la première variante

La méthodologie que nous avons mise au point permet la spécification et la conception d'organisations multi agent à structure pyramidale dont chaque agent a un rôle bien déterminé et est responsable d'un ensemble agents. L'application de notre méthode méthodologie au problème qui nous intéresse dans cet article est donc peu " naturelle ". Cependant notre méthode MAMOSACO (Méthode Adaptable de Modélisation de Systèmes Administratifs Complexes) est suffisante pour modéliser/spécifier les échanges d'informations entre les agents représentant les enseignants ou le responsable de salle.

6.3.1 Spécification des agents et de leurs comportements

Pour répondre à un problème, nous posons tout d'abord les rôles, limités, dans le cadre du contexte de cet article, au rôle d'enseignant et au rôle de gestionnaire, ou de responsable, de l'emploi du temps. Ce dernier rôle affiche simplement les choix effectués par les agents enseignants. Notre méthodologie se propose de définir les rôles à l'aide d'une grille (cf. tableau 6.1) de spécification de rôle.

La grille 6.1 permet donc de définir pour chaque agent les fonctions relatives :

TAB. 6.1 – Grille de spécification de rôle adaptée de la grille d’analyse de Ferber [3]

| Fonctions/Dimensions | Sociale | Environnementale | Personnelle |
|----------------------|--|--|---|
| Représentationnelle | Représentation du groupe, des rôles, des autres | Représentation du monde | Représentation de soi, de ses capacités |
| Organisationnelle | Planification des actions sociales, des communications | Planification des actions dans l’environnement | Contrôle des planifications, méta-planification |
| Interactionnelle | Description interactions agent-société, performatifs | Mécanismes de perception et d’action par rapport à l’environnement | Auto-communication, auto-action |
| Productive | Tâches de gestion, d’administration, de coordination, de négociation | Tâches d’analyse, de modification et de création | Auto-modification, apprentissage |
| Conservative | Conservation de la société, des relations, du réseau d’acointances | Conservation des ressources, défense et entretien du territoire | Conservation de soi, réparation, entretien |

- aux connaissances, procédurales ou non (la fonction représentationnelle). Ces connaissances sont plus ou moins détaillées selon la position de l’agent dans l’organisation,
- à la planification des actions (la fonction organisationnelle). Chaque agent étant autonome, il doit pouvoir gérer ses propres actions,
- aux interactions (la fonction interactionnelle),
- à la maintenance (la fonction conservative), ceci pour maintenir la stabilité de l’agent et de l’organisation en général,
- aux actions spécifiques au rôle de l’agent (la fonction productive). Par exemple, l’agent effectue l’interface avec l’utilisateur ou gère des données situées sur le poste de travail.

Ces fonctions sont décrites relativement à l’environnement de l’agent, aux autres agents et à l’agent lui-même.

Il était donc peu intéressant de décrire le rôle joué par les agents qui n’interagissent ni n’agissent avec l’environnement ou avec l’organisation.

Une fois les rôles définis, nous proposons de spécifier les interactions entre les agents à l’aide des modèles de la méthode MAMOSACO que nous avons définie, afin de modéliser des processus administratifs complexes de type workflow, par intégration de plusieurs méthodes du génie logiciel suite à leur comparaison. Cette méthode est supportée par un atelier de modélisation et de simulation. Cette méthode permet de représenter les acteurs des processus administratifs complexes selon leurs degrés de responsabilité dans le processus, et permet surtout de représenter les flux d’informations entre ces acteurs. Appliquée aux organisations multi agent, les modèles permettent de spécifier les échanges entre les agents logiciels qui forment l’organisation.

MAMOSACO se compose de quatre principaux modèles, un modèle de données, un modèle de flux de données, un modèle de traitement des données et un modèle dynamique :

- en ce qui concerne le modèle des données, celui proposé par UML permet de représenter à la fois la structure des informations (des documents) ainsi que les relations qui existent entre

- elles (appartenance, agrégation, héritage). Dans le cas de la spécification d'une forgnisation multi-agent, ce modèle permettra de définir la structure des agents ;
- le modèle des flux de données, représentant les flux d'informations entre les acteurs, est basé sur le modèle des actigrammes de la méthode SADT, que nous avons adapté afin de représenter les niveaux de responsabilité entre acteurs et par extension entre agents ;
 - si le modèle d'activité représente tous les flux possibles entre les acteurs, il ne représente pas les conditions dans lesquelles ces flux suivent un chemin précis. Cette notion de condition se trouve dans le modèle des traitements. Ce modèle doit également faire ressortir la coopération entre les acteurs, et bien sûr, les liens de hiérarchie et/ou de responsabilité. Nous utilisons à ce propos le modèle des traitements de la méthode OSSAD qui permet en outre de représenter la coopération et les notions de hiérarchie. A l'instar du modèle d'activité, ce modèle peut être utilisé pour la spécification de l'OMA. Aux acteurs correspondent alors les agents, classés par niveau croissant de responsabilité et traitant des données pour les transmettre à d'autres agents sous certaines conditions ;
 - la méthode MAMOSACO se compose d'un quatrième modèle, basé sur les réseaux de Petri (RdP) paramétrés. La modélisation par ces RdP implique de définir, au niveau des transitions (niveau général), au niveau des places (niveau local) et au niveau des acteurs (niveau personnel) des règles de fonctionnement sous la forme préconditions-actions-postconditions.

Des quatre modèles qui constituent la méthode, tous ne doivent pas être réutilisés dans le cadre de la seule spécification d'une OMA. Ainsi, dans le cadre de ce problème, nous n'avons utilisé que le modèle de traitement de la méthode. La figure 6.1 présente le fonctionnement et les échanges effectués entre nos agents enseignants.

Tout d'abord, chaque agent communique ses contraintes aux autres agents, puis choisit en fonction des contraintes de l'ensemble ses créneaux (si un créneau ne peut pas être pris par un autre agent et ne fait pas parti de ses propres contraintes, alors ce créneau est choisi par l'agent). Lorsque les créneaux horaires sont choisis, ils sont transmis aux accointances. Le premier agent ayant envoyé ses choix les garde. Le second modifie ses créneaux en fonction des choix du premier puis envoie ses horaires au troisième agent ayant transmis ses choix. Ce dernier établit donc son emploi du temps en fonction de ses contraintes et des choix des deux autres agents enseignants. Lorsque qu'un agent enseignant a fixé définitivement ses créneaux, il les envoie à l'agent gestionnaire qui affiche l'emploi du temps sur deux jours.

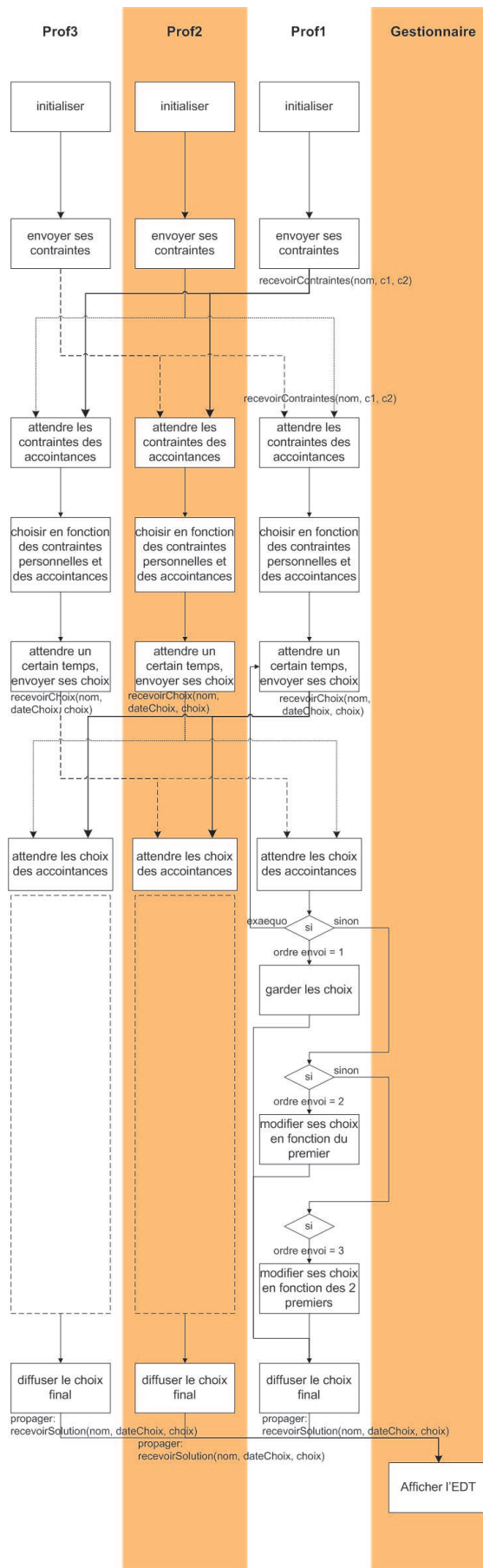


FIG. 6.1 – Choix des créneaux dans le cadre de la variante 1

6.3.2 Application

La structure des OMA que nous développons suit une architecture holonique, donc pyramidale. C'est pourquoi nous nous tournons vers la plateforme MAGIQUE [7] (de l'équipe SMAC du Lifi, Lille) particulièrement adaptée à nos besoins. Cette plateforme permet de créer des agents vides, qui possèdent par défaut, les compétences nécessaires à la transmission de messages entre agents. Nous avons donc développé une compétence " personnelle " (c'est-à-dire associée à chaque agent enseignant) de gestion d'emploi du temps, appelée ProfSkill, ainsi qu'une compétence associée à l'agent gestionnaire, appelée GestionSkill. La figure 6.2 présente le modèle objet simplifié des classes permettant la création du SMA répondant à la première variante.

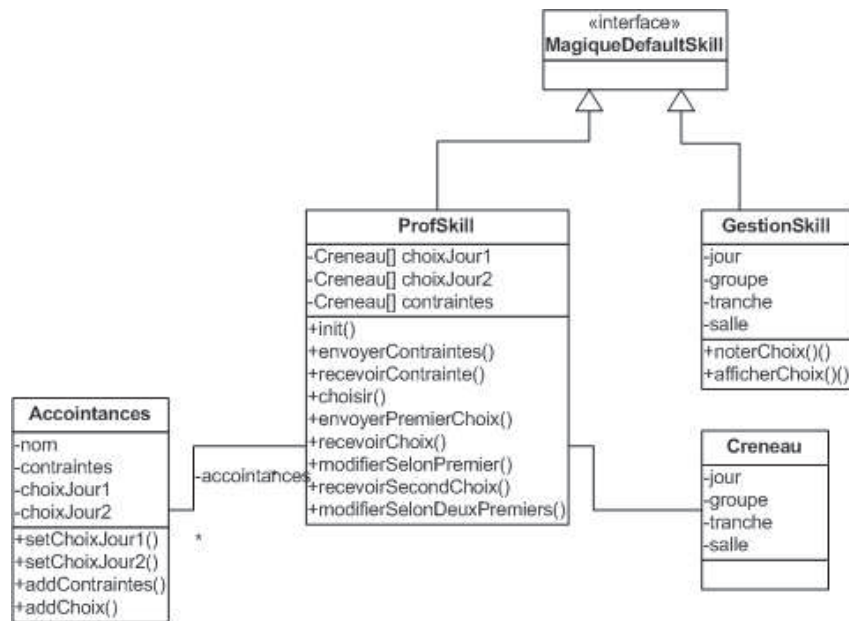


FIG. 6.2 – Modèle objet simplifié des classes répondants à la première variante

Quatre agents sont donc nécessaires : trois agents enseignant et un agent gestionnaire jouant également le rôle de " boss ". Ce rôle est prédéfini dans MAGIQUE et permet à l'agent qui le possède de propager les messages à l'ensemble des agents qu'il a sous sa responsabilité.

Nous avons associé à chaque agent une interface homme-machine relativement basique car constituée d'une zone de texte déroulante. Il est bien sûr possible de " lancer " les agents sur des machines ou plateformes différentes, cependant, pour des raisons de simplicité dans les tests ou l'exécution, les agents sont activés séparément, mais en local par l'utilisateur par l'intermédiaire d'une fenêtre de dialogue.

Les résultats obtenus par notre SMA diffèrent selon l'ordre dans lequel agents enseignant e1, e2 et e3 prennent leurs décisions. Les figures suivantes correspondent au cas où e2 choisit ses créneaux en premier lieu, e1 étant le dernier agent enseignant à avoir proposé ses choix.

Quant à l'agent gestionnaire, il synthétise les résultats et affiche ceci :

```

super@127.0.0.1
demarrerSimu
demarrerSimu
demarrerSimu
reception des choix de prof1@127.0.0.1:4444
reception des choix de prof2@127.0.0.1:4444
reception des choix de prof3@127.0.0.1:4444
  
```



FIG. 6.3 – Résultats de l’agent e2 (appelé prof2)



FIG. 6.4 – Résultats de l’agent e3 (appelé prof3)



FIG. 6.5 – Résultats de l’agent e1 (appelé prof1)

| creneaux | G1 | G2 | G3 |
|----------|-------|-------|-------|
| 8 - 10 | prof1 | prof2 | prof3 |
| 10 - 12 | prof3 | prof1 | prof2 |

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

| | | | |
|------------------------------|-------|-------|-------|
| 12 - 14 | prof2 | | prof1 |
| 14 - 16 | | prof3 | |
| creneaux | G1 | G2 | G3 |
| 8 - 10 | prof2 | | prof1 |
| 10 - 12 | prof1 | prof3 | |
| XXXXXXXXXXXXXXXXXXXXXXXXXXXX | | | |
| 12 - 14 | | prof2 | prof3 |
| 14 - 16 | prof3 | prof1 | prof2 |

6.4 Spécification du SMA répondant à la seconde variante

En ce qui concerne la seconde variante, nous n'avons pas totalement terminé l'implémentation. Le SMA répondant à ces nouvelles contraintes étend le SMA précédent. Nous avons augmenté la compétence de l'agent Gestionnaire afin qu'il puisse distribuer les salles aux agents enseignants après qu'ils lui aient envoyé leurs emplois du temps. En cas d'impossibilité à distribuer les salles, du fait des contraintes imposées sur celles-ci, l'agent gestionnaire les relâche. S'il est toujours impossible de distribuer les salles, un message est envoyé à l'agent enseignant dont la demande pose problème afin qu'il relâche une de ses contraintes et recompose son emploi du temps (en tenant compte bien sûr des emplois du temps des autres agents).

6.5 Conclusion

Bien que conçue pour la spécification et la conception d'organisation multi-agent pour aider les acteurs des organisations humaines, notre méthodologie peut s'appliquer en partie à des problèmes plus 'simples' dans la structure de l'organisation.

La troisième variante demandant à modifier les contraintes sans réinitialisation du système n'a pas été abordée. Cependant, comme les agents que nous proposons possèdent des informations sur les créneaux pris par leurs accointances, ils peuvent modifier leurs emplois du temps de façon autonome, sans réinitialisation du système.

La plate-forme Magique correspond bien à notre méthodologie. Néanmoins, les fonctionnalités de communication n'étant pas orientées groupe, mais orientées compétence, quelques courtes adaptations ont été réalisées pour répondre au problème et permettre la diffusion de messages au groupe des agents jouant le rôle d'enseignant.

Bibliographie

- [1] E. ADAM. *Modèle d'organisation multi-agent pour l'aide au travail coopératif dans les processus d'entreprise : application aux systèmes administratifs complexes*. PhD thesis, Université de Valenciennes et du hainaut Cambrésis, 2000.
- [2] B. CHAIB-DRAA, R. MANDIAU, and P. MILLOT. Trends in Distributed Artificial Intelligence. *Artificial Intelligence Review*, 6 :35–66, 1992.
- [3] J. FERBER. *Les systèmes multi-agents : vers une intelligence collective*. IIA, Paris - FRANCE, intereditions edition, 1995.
- [4] E. LE STRUGEON. *Une méthodologie d'auto-adaptation d'un système multi-agents cognitifs*. PhD thesis, Université de Valenciennes et du hainaut Cambrésis, 1995.
- [5] R. MANDIAU. *Contribution à la modélisation des univers multi-agents : génération d'un plan partagé*. PhD thesis, Université de Valenciennes et du hainaut Cambrésis, 1993.
- [6] R. MANDIAU. *Modélisation et évaluation d'organisation multi-agents, Habilitation à diriger des Recherches*. Hdr, Université de Valenciennes et du hainaut Cambrésis, 2000.
- [7] J.C. ROUTIER, P. MATHIEU, and Y. SECQ. Dynamic Skills Learning : a support to agent evolution. In *Proceedings of AISB'01*, ISBN 1 902956 17 0, pages 25–32, York, 2001.

Chapitre 7

Résolution via la théorie des AMAS. Le système ETTO

Marie-Pierre Gleizes¹, Carole Bernon¹, Valérie Camps², Gauthier Picard¹, Sylvain Peyruqueou³, Pierre Glize¹

7.1 La Théorie des AMAS

Un système multi-agent adaptatif est un système multi-agent qui est capable de changer son comportement en cours de fonctionnement pour l'ajuster dans un environnement dynamique, soit pour réaliser la tâche pour laquelle il a été conçu, soit pour améliorer sa fonction ou ses performances.

La théorie des AMAS garantit qu'un système est fonctionnellement adéquat (il réalise la fonction souhaitée) si les échanges entre les agents qui le composent sont coopératifs, ce qui entraîne les conditions suivantes :

- un signal doit être interprété sans ambiguïté;
- toute interprétation doit être informative (pas de redite,...);
- les conclusions doivent être utiles à autrui.

La particularité de la théorie des AMAS réside dans le fait que l'on ne code pas la fonction globale au sein d'un agent. Grâce à la capacité des agents à s'auto-organiser, le système est capable de s'adapter par lui-même et réalise une fonction qui n'est pas codée dans l'agent. L'objectif de l'auto-organisation est de permettre l'évolution de l'existant en fonction du contexte de façon à assurer la viabilité du système.

7.1.1 Les agents AMAS

Tous les agents que nous considérons sont composés de cinq parties contribuant à leur comportement :

- **Les compétences** : ce qu'est capable de faire l'agent ou quels savoir-faire il peut apporter à la collectivité. Dans un système d'information tel que l'Internet, un agent qui représente un utilisateur peut rechercher de l'information, communiquer avec les autres utilisateurs, ...
- **Une représentation de soi, des autres ou de l'environnement** : ce que l'agent connaît à propos de lui-même, des autres agents et de son environnement. Dans l'Internet, par exemple, un agent lié à un utilisateur connaît ses types d'actions possibles et probablement sait que les autres agents de même type sont capables d'agir de manière similaire, il connaît aussi les ressources qu'il peut trouver dans son environnement.

¹IRIT - Université Paul Sabatier, Toulouse

²L3I - Université de la Rochelle, la Rochelle

³ARTAL Technologies - Parc Technologique du Canal, Ramonville Saint Agne

- **Une attitude sociale** : elle permet à l'agent de modifier ses interactions avec les autres. Cette attitude sociale se fonde sur ce que nous appelons "coopération" : si un agent détecte une Situation Non Coopérative (SNC), il agit pour revenir à un état dit coopératif. Dans l'Internet, par exemple, un agent A1 pourrait se voir demander une information par un agent A2. Si A1 ne possède pas la réponse (il n'a pas l'habitude de traiter de telles demandes et un tel imprévu constitue une SNC pour lui) mais pense que A3 la possédera peut-être, A1 aidera A2 en relaxant la requête à A3 (pour revenir à une situation qu'il juge coopérative).
- **Un langage d'interaction** : ce langage est nécessaire à l'agent pour communiquer de manière directe ou non. Dans l'Internet, les agents communiquent directement par passage de messages. La communication indirecte se fait, par exemple, par le biais d'une modification de l'environnement commun des agents.
- **Les aptitudes** : elles représentent la capacité qu'un agent possède pour raisonner sur ses représentations et sur ses connaissances ; par exemple, interpréter un signal ou une requête.

7.1.2 La méthodologie ADELFE

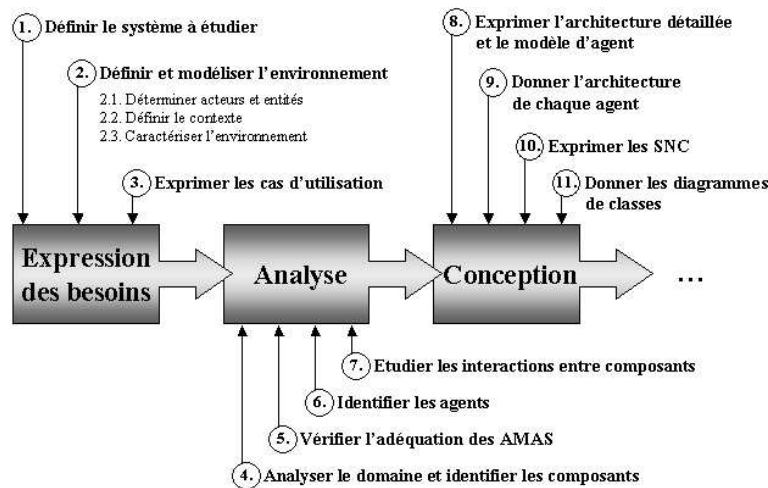


FIG. 7.1 – Schéma récapitulatif des premières étapes d'ADELFE

ADELFE est l'acronyme de "Atelier pour le Développement de Logiciels à Fonctionnalité Emergente". En créant ADELFE, notre but est, en effet, d'aider tout développeur à concevoir des logiciels à fonctionnalité émergente et pas seulement ceux spécialistes des systèmes multi-agents adaptatifs.

La méthodologie ADELFE suit le RUP [IGJ] et y adjoint quelques étapes spécifiques à la conception de systèmes adaptatifs. Actuellement, ADELFE est focalisé sur les phases d'expression des besoins, l'analyse et la conception.

L'expression des besoins est divisée en deux sous-parties : les besoins préliminaires et les besoins finals. Les besoins préliminaires représentent un travail d'intercompréhension et/ou une description consensuelle du problème du cahier des charges entre clients, utilisateurs et concepteurs sur ce que doit être et ce que doit faire le système, ses limites et ses contraintes. Les besoins finals fournissent un modèle d'environnement. Les objectifs sont de définir une vue du système, de transformer cette vue en un modèle de cas d'utilisation, d'organiser et de gérer les besoins (fonctionnels ou non) et leurs priorités. De manière pratique, à ce stade, le concepteur doit définir le système étudié, d'une part, et modéliser formellement son environnement, d'autre part.

L'analyse doit dégager une compréhension du système, de sa structure en terme de composants et repérer si la théorie des AMAS est nécessaire. La spécificité d'ADELFE implique que tout

concepteur n'a pas forcément besoin d'elle car toutes les applications ne requièrent pas l'emploi de systèmes multi-agents adaptatifs. L'analyse débute par une étude du domaine, ou analyse du domaine. Un premier diagramme de classe, préliminaire, du système et de son environnement est établi afin de construire une première vue statique du système. La dynamique du système est représentée au travers d'un ensemble de diagrammes de séquence entre entités actives. Nous augmentons l'analyse classique d'une nouvelle étape, l'adéquation des AMAS, afin de vérifier si les modèles d'environnement et de système précédemment établis correspondent à un modèle AMAS. Une autre étape supplémentaire est l'identification des agents. Toutes les entités actives ne sont pas forcément des agents. De fait, l'analyste doit identifier les agents en accord avec différents critères (comme l'autonomie par exemple) et avec les interactions avec l'environnement.

La conception doit définir une architecture détaillée pour le système en terme de paquetages, de sous-systèmes, d'objets et d'agents. Ce sont des activités importantes d'un point de vue multi-agent du fait qu'une caractérisation récursive du système multi-agent est obtenue à ce stade. Cela implique différents processus de conception pour les différents niveaux du système identifiés. Une architecture agent permet au concepteur de doter les agents de modèles de compétences, de représentations du monde (de lui, des autres et de son environnement), d'aptitudes, d'attitudes sociales et d'un langage d'interaction. A ce stade, une liste des situations non coopératives est établie. Le modèle de conception est composé de classes (dans des diagrammes de classes), de classes d'agents et d'interactions. Une conception détaillée résulte de diagrammes d'état-transition (pour modéliser le comportement des objets et des agents) et de diagrammes d'activités.

7.2 Application d'ADELFE à ETTO

L'emploi du temps appartient à la classe des "problèmes à satisfaction de contraintes", c'est un problème très complexe et connu de tous. Le cahier des charges contient quatre variantes qui permettent d'introduire une gradation dans la difficulté de l'emploi du temps à établir. Cette partie présente ce que nous avons obtenu, pour chaque étape, en appliquant la méthodologie ADELFE à ce problème de l'emploi du temps..

7.2.1 Expression des besoins

Dans un premier temps, le système que l'on souhaite mettre en place doit permettre de constituer un emploi du temps à partir d'un ensemble de salles, de professeurs et d'élèves, chacune de ces entités étant soumise à des contraintes. Par la suite, les contraintes et les entités pourront évoluer.

On identifie ainsi des entités actives :

- **Les professeurs** parce qu'ils ont la possibilité de modifier eux-mêmes leurs contraintes (dynamique et autonomie) et qu'ils vont interagir avec le système.
- **Les élèves**, pour les mêmes raisons.
- **Le responsable des enseignements** car c'est lui qui va devoir choisir les cours à tenir dans la période pour laquelle l'emploi du temps va être établi. Il peut aussi faire des modifications en cours de fonctionnement. Il agit sur le système.
- **Le responsable des salles**, pour des raisons assez semblables ; son action sur le système étant de modifier les contraintes sur les salles.

et des entités passives :

- **Les salles**, puisqu'il s'agit de simples ressources ; l'évolution de leurs caractéristiques est décidée par le responsable des salles.
- **Le PPN** ou Plan Pédagogique National. Il recense tous les cours qui devront être assurés dans l'année pour chaque formation. C'est la 'bible' du responsable des enseignements, une ressource donc.

Les interactions entre les entités actives et le système ont toutes la même forme. Une phase d'initialisation précède de possibles mises à jour ; le système restituant un nouvel emploi du temps chaque fois. Pour les professeurs et les élèves, ces interactions portent sur leurs contraintes, les

responsables, quant à eux, interagissent respectivement au sujet des cours et des salles. Les diagrammes de séquences qui représentent ces interactions ont une forme sensiblement identique.

En utilisant les termes proposés dans la méthodologie, l'environnement se caractérise de la façon suivante :

- **Dynamique** : l'évolution des entités actives ne dépend pas du système, elle sont imprévisibles de son point de vue.
- **Accessible** : le système peut obtenir toute l'information qu'il souhaite sur l'état de l'environnement.
- **Non-déterministe** : le système n'est pas capable de connaître quels seront les effets de ses actions sur les entités actives.
- **Continu** : le nombre d'interactions entre le système et les entités est non fini.

Chaque entité active peut initialiser et modifier ce sur quoi elle agit et visualiser ensuite le nouveau résultat.

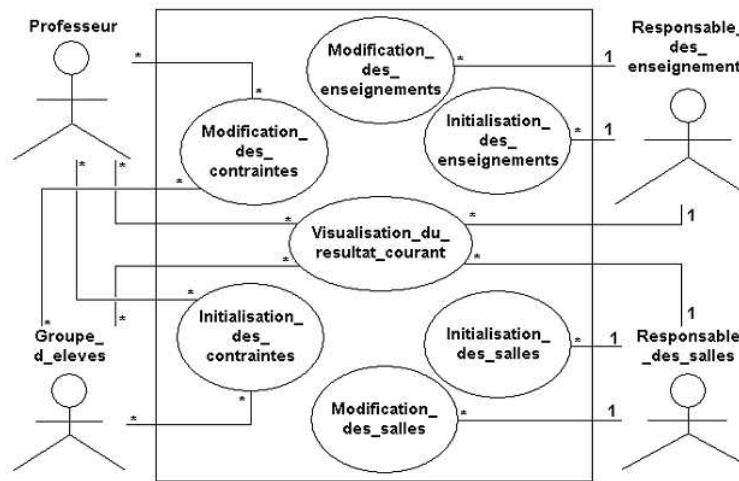


FIG. 7.2 – Diagramme des cas d'utilisation

7.2.2 Analyse

à ce niveau, il est nécessaire de décomposer le système en entités. Parmi les entités extérieures en interaction avec le système que nous avons déterminées plus haut, certaines nécessitent des représentants à l'intérieur même du système. Nous identifions dans un premier temps en tant qu'entités :

- les professeurs, les élèves et les salles de manière évidente ;
- **le gestionnaire des salles** pour servir d'intermédiaire entre l'acteur "responsable des salles" et les salles du système ;
- **le gestionnaire des enseignements** en tant qu'interface entre le "responsable des enseignements" et le système ;
- et **le PPN** comme référence pour le "gestionnaire des enseignements".

Dans un deuxième temps, nous cherchons à déterminer quelles autres entités peuvent être utiles au système :

- **les contraintes** semblent indispensables puisque beaucoup d'entités y sont soumises ;
- **un gestionnaire des contraintes** pour chaque entité possédant des contraintes ;
- **une grille** pour regrouper les salles et représenter le résultat. La grille présentée dans le cahier des charges possède ainsi trois dimensions : les salles, les créneaux horaires et les jours.

Nous appelons **Cellule** chaque intersection des dimensions de la grille ; c'est le constituant élémentaire de la grille.

Le questionnaire d'ADELFE sur l'adéquation fournit le résultat suivant :

- Résultat textuel au niveau global du système ETTO : " *Certains aspects de votre application justifient le développement par les AMAS. Un suivi précis de la méthodologie permettra de les identifier.*"
- Résultat textuel au niveau local des composants : " *Certains composants de votre système justifient le développement par les AMAS. Vous devez employer la méthodologie ADELFE sur ceux-ci.*"

L'outil d'adéquation se prononce clairement pour l'utilisation des AMAS au niveau global. En outre, il estime que certaines entités du système méritent, elles aussi, un développement AMAS. Par conséquent, il faudra revenir à cette étape plus tard en ne considérant plus que le niveau local.

Nous identifions ensuite deux catégories d'agents, les professeurs et les groupes d'élèves. Leurs caractéristiques sont tout à fait similaires et leur but individuel parfaitement symétrique : les professeurs cherchent les élèves et inversement. Nous considérons donc qu'il s'agit de deux classes distinctes d'un même type d'agent générique que nous appelons **RepresentativeAgent** (RA).

Pour répondre de nouveau au questionnaire de l'adéquation, nous considérons que le système est un RepresentativeAgent que nous avons déterminé à l'étape précédente. L'environnement à prendre en compte devient alors celui d'un professeur ou d'un élève et nous obtenons :

- Résultat textuel au niveau global : " *Certains aspects de votre application justifient le développement par les AMAS. Un suivi précis de la méthodologie permettra de les identifier.*"
- Résultat textuel au niveau local : " *Les caractéristiques que vous attribuez aux composants du système indiquent nettement qu'il est inutile de leur employer la technologie des AMAS.*"

Les professeurs et les groupes d'élèves doivent respectivement assurer et recevoir plusieurs cours chacun (atteindre plusieurs objectifs). La décomposition apparaît : un agent par cours. C'est le **BookingAgent** (BA) qui se charge pour le RepresentativeAgent de trouver un partenaire et de réserver un lieu. Il y a autant de BA que le RA a de cours à assumer ou à recevoir. En outre, on dit que les BA issus d'un RA sont ses "fils". Evidemment, le RA est alors leur "père". Tous les agents issus d'un même père sont des "frères".

Après cette nouvelle décomposition, le diagramme de classes préliminaire a évolué. Voici sa nouvelle version :

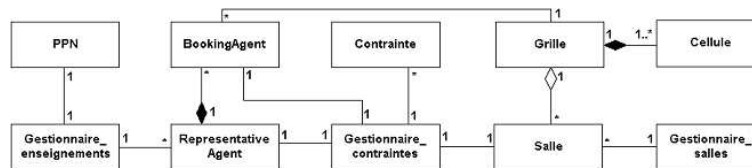


FIG. 7.3 – Diagramme de classes préliminaire évolué.

RepresentativeAgent et BookingAgent possèdent un gestionnaire de contraintes. En effet, les contraintes propres à un professeur (la matière qu'il enseigne, ses disponibilités, ...) sont mémorisées dans son RepresentativeAgent, mais les BookingAgent possèdent eux-aussi des contraintes qui leur sont propres ; par exemple, tous les cours assurés par le professeur n'ont pas la même durée et ne nécessitent pas le même matériel.

Nous avons vu au début de l'analyse que le gestionnaire des enseignements permet de choisir les cours qui devront être assurés sur la période pour laquelle l'emploi du temps est établi. Dans le système, cela se transcrit par la création de BookingAgent. Lorsque le responsable des enseignements décide qu'un cours doit être suivi par un groupe d'élèves, le gestionnaire des enseignements informe son RepresentativeAgent afin qu'il crée un BookingAgent pour se charger de

ce cours. Toutefois, si le gestionnaire souhaite que le cours soit effectivement suivi, il faut aussi qu'il intègre au système un professeur compétent pour le donner (ou plusieurs mais certains seront alors inutiles). En conclusion, nous pouvons dire que le gestionnaire des enseignements prend son information dans le PPN (même si pour l'instant c'est le responsable des enseignements qui gère la chose) et décide ensuite des professeurs et des élèves à instancier.

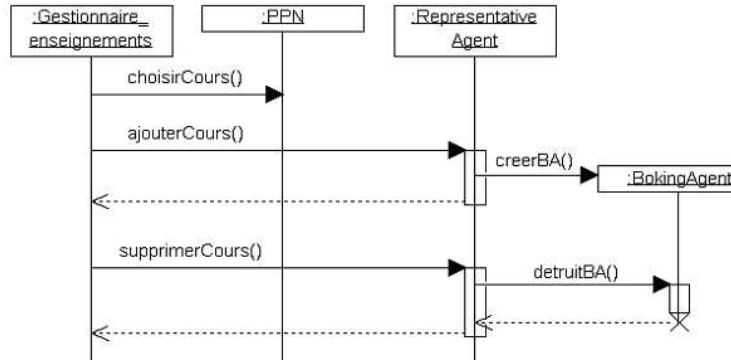


FIG. 7.4 – Diagramme de séquences – Gestion des enseignements.

7.2.3 Conception

Dans un premier temps, nous décomposons l'architecture en paquets assez généraux. Après l'analyse précédente, la décomposition suivante en cinq paquets nous semble la plus pertinente :

- un paquetage **agent** pour gérer tout ce qui concerne les BA et le RA ;
- un paquetage **grille** pour gérer les différentes dimensions constitutives de la grille et ainsi la structurer ;
- un paquetage **contrainte** qu'il faut externaliser pour en permettre l'accès à la fois aux salles et aux agents ;
- un paquetage **interface** grâce auquel l'utilisateur pourra intervenir sur le système pour jouer le rôle de responsables des salles et des enseignements.

Deux niveaux d'agents étant déterminés, il est nécessaire d'établir leurs situations non coopératives respectives :

- **RepresentativeAgent**. Bien que les RepresentativeAgent soient vus comme des agents AMAS, ils n'ont pas de situations non coopératives spécifiques. La conception de ces agents n'est pour autant pas remise en cause, mais toutes leurs actions, leurs interactions et leur évolution sont en fait reléguées au niveau inférieur, celui des BookingAgent.
- **BookingAgent**. Les BookingAgent n'ayant pas de fonction clairement établie sur laquelle chercher des exceptions, les SNC sont déterminées en considérant le but à atteindre. En se focalisant sur cet objectif et en cherchant tout ce qui pourrait en empêcher l'atteinte, nous déterminons les situations non coopératives fondées sur l'incompétence, l'improductivité, le conflit et l'inutilité. L'incompréhension et l'ambiguïté sont absentes car le système est conçu de manière à ce que les agents interprètent correctement les signaux provenant de leur environnement.

Au-delà des situations non coopératives, grâce aux scénarios imaginés, cette étape nous permet de déterminer le fonctionnement de l'agent et notamment la manière dont il effectue des réservations et des partenariats. Chaque contrainte est affectée d'un poids qui détermine la difficulté que doit avoir l'agent à la relâcher : plus le poids est important, plus le coût du relâchement de la contrainte est élevé. Au niveau local, cela signifie que chaque BookingAgent cherche à minimiser son coût personnel. Toutefois, en cas de conflit, l'intérêt personnel passe au second plan et c'est le BA qui coûte le moins cher à la collectivité qui l'emporte.

Un BookingAgent sans réservation, du moment que la salle est compatible avec ses besoins, doit la réserver quel que soit le coût des réservations qu'il doit relâcher. Par la suite, il ne réservera que les salles lui permettant d'améliorer son score courant. Il en va de même pour les partenariats. Cela permet de trouver une solution non optimale très rapidement, la suite du traitement sert uniquement à améliorer le score global, donc à améliorer la solution.

Afin de générer le diagramme de classes général, il faut déterminer les liens entre les diagrammes de chaque paquetage :

- Agent-grille : la grille représente l'environnement dans lequel les BookingAgent évoluent. Toutefois, ils y évoluent en se déplaçant de cellule en cellule. Il paraît donc logique de relier la classe BookingAgent à la classe Cellule.
- Agent-contrainte : la liaison est évidente, nous avons déjà vu précédemment que les BA et les RA possédaient des contraintes. Il y a donc un lien entre BookingAgent et GestionnaireContraintes, et entre RepresentativeAgent et GestionnaireContraintes.
- Grille-contrainte : comme les agents, les salles et plus généralement les objets dimensionnants peuvent avoir des contraintes. La classe ObjetDimensionnant est donc reliée à GestionnaireContraintes.
- Interface-agent : depuis une étape précédente, nous avons déterminé que le gestionnaire des enseignements était relié à la classe RepresentativeAgent afin de faire instancier le nombre de BookingAgent nécessaire.
- Interface-grille : nous avons vu avec la mise en place du paquetage interface que GestionnaireObjDim était chargé de la gestion de tous les objets dimensionnants, et donc lié à la classe ObjetDimensionnant.

On obtient alors le diagramme suivant :

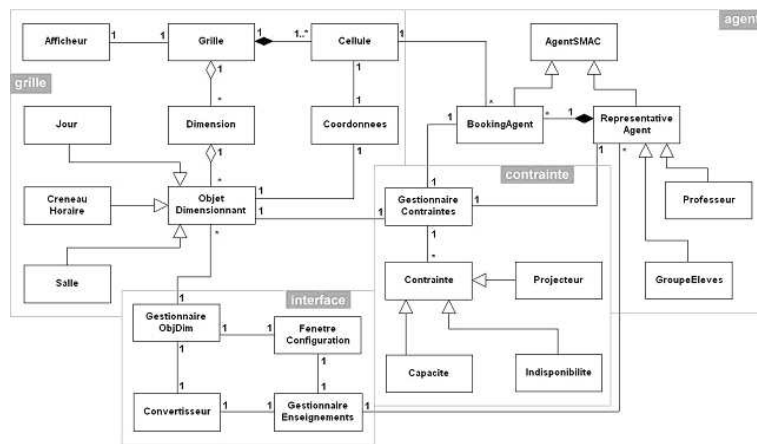


FIG. 7.5 – Diagramme de classes général.

Toutes les étapes d'ADELFE sont maintenant développées et on peut noter que la partie concernée par les AMAS est assez réduite.

7.3 Développement et test

7.3.1 L'interface de visualisation

ETTO est codé en JAVA. Ce langage facilite le portage de l'application et est compatible avec OpenTool[©] ce qui facilitera à terme l'intégration. L'interface est en HTML car il suffit de produire des fichiers texte pour la génération de l'affichage. L'interface de contrôle est à base de

JAVA SWING pour définir le nombre de cycles que le système doit effectuer avant de lui proposer une nouvelle photo de l'état du système. Un bouton lui permet aussi de faire avancer l'application pas à pas.

La grille étant générique, l'utilisateur, à l'initialisation de la résolution, doit indiquer quelles sont les dimensions d'affichage de la grille avant de spécifier dans l'ordre les dimensions suivant lesquelles éclater l'affichage des multiples grilles. Une partie de la fenêtre est réservée à l'affichage de l'état des agents. Elle indique les contraintes relâchées par les agents et celles auxquelles ils sont soumis. Cela permet de vérifier la justesse de la solution trouvée.

Les configurations initiales sont décrites dans de simples fichiers texte formatés. Celui qui porte le bon nom est automatiquement chargé par l'application au démarrage.

7.3.2 Résultats de la Variante 1

Voici une des solutions que trouve ETTO (tab. 1 et 2) :

| Jour 1 | Salle 1 | Salle 2 | Salle 3 |
|-------------|---------------------------------|---------------------------------|---------------------------------|
| 08h00-10h00 | E1_0_1450 G2_0_0 **E1_0** | E3_0_1500 G1_0_0 **E3_0** | E2_0_1460 G3_0_0 **E2_0** |
| 10h00-12h00 | E3_0_1470 G1_0_0 **G1_0** | E2_0_1490 G2_0_0 **E2_0** | E1_0_1480 G3_0_0 **G3_0** |
| 14h00-16h00 | E1_0_1480 G1_0_0 **G1_0** | E2_0_1450 G3_0_0 **G3_0** | |
| 16h00-18h00 | | E3_0_1470 G3_0_0 **G3_0** | |

TAB. 7.1 – Premier jour de la première variante.

| Jour 2 | Salle 1 | Salle 2 | Salle 3 |
|-------------|---------------------------------|---------------------------------|---------------------------------|
| 08h00-10h00 | E2_0_1460 G2_0_0 **E2_0** | | E1_0_1470 G1_0_0 **E1_0** |
| 10h00-12h00 | E3_0_1480 G2_0_0 **G2_0** | E1_0_1440 G_0_0 **E1_0** | |
| 14h00-16h00 | | E2_0_1460 G1_0_0 **G1_0** | E3_0_1460 G2_0_0 **E3_0** |
| 16h00-18h00 | E2_0_1480 G1_0_0 **E2_0** | E1_0_1460 G2_0_0 **G2_0** | E3_0_1470 G3_0_0 **G3_0** |

TAB. 7.2 – Deuxième jour de la première variante.

Une première analyse permet de rapidement constater que tous les BA ont réussi à trouver un partenaire et une place où s'établir. Pour conclure que la solution est optimale, il faut toutefois vérifier que toutes les contraintes sont vérifiées. Cela se fait par l'intermédiaire des coefficients qui apparaissent après chaque réservation et représentant le coût total de celle-ci. Dans le cas présent, tous sont nuls et confirment que la solution est optimale.

Les coefficients qui suivent le nom des agents correspondent, pour le premier, au coût des contraintes pour occuper le créneau, le second étant la valeur du coefficient évoluant au cours du temps que nous avons vu plus en amont. On constate que le premier coefficient est nul pour tous les agents, ce qui est normal puisque toutes les contraintes sont vérifiées. Par contre, le deuxième nombre n'est pas nul pour les enseignants alors qu'il l'est pour les élèves. Cela est dû au fait que seuls les professeurs possèdent des contraintes.

Cette variante fonctionne parfaitement et donne des résultats satisfaisants. Le nombre de cycles donnant une solution optimale s'établit en moyenne autour de 140. D'un point de vue temporel, la résolution est instantanée.

7.3.3 Résultats de la Variante 2

Voici une des solutions que trouve ETTO pour la variante 2 (tab. 3 et 4) :

| Jour 1 | Salle 1 | Salle 2 | Salle 3 |
|-------------|--|-----------------------------------|----------------------------------|
| 08h00-10h00 | E3_0_19970 G2_0_0 **G2_0** | E1_0_24645 G3_0_-1 **E1_0** | E2_0_22056 G1_0_0 **E2_0** |
| 10h00-12h00 | | E2_0_22128 G2_0_-1 **G2_0** | E3_0_19990 G3_0_0 **E3_0** |
| 14h00-16h00 | E2_0_22092 G3_0_-1 **G3_0** | E1_0_23160 G2_0_-1 **E1_0** | |
| 16h00-18h00 | E1_12000_24300 G1_0_0 **E1_12000** | E3_0_20010 G3_0_-1 **E3_0** | |

TAB. 7.3 – Premier jour de la deuxième variante.

| Jour 2 | Salle 1 | Salle 2 | Salle 3 |
|-------------|-----------------------------------|-----------------------------------|--|
| 08h00-10h00 | E2_0_22104 G1_0_-1 **G1_0** | | E1_0_23355 G2_0_0 **E1_0** |
| 10h00-12h00 | E3_0_19990 G2_0_-1 **E3_0** | E1_0_24510 G1_0_-1 **G1_0** | E2_0_22080 G3_11004_0 **G3_11004** |
| 14h00-16h00 | E2_0_21960 G2_0_0 **G2_0** | E3_0_19990 G1_0_-1 **E3_0** | E1_12195_24285 G3_0_0 **E1_12195** |
| 16h00-18h00 | E3_0_19920 G1_0_0 **G1_0** | | |

TAB. 7.4 – Deuxième jour de la deuxième variante.

On constate de nouveau que tous les agents ont trouvé un endroit où se placer en compagnie d'un partenaire. Cependant, dans ce cas, tous les coefficients vus précédemment ne sont pas nuls. Dans les trois salles légèrement grisées, les réservations ont un coût qui confirme qu'il n'y pas de solution optimale, les agents étant obligés de relâcher des contraintes pour pouvoir compléter l'emploi du temps.

Le coefficient -1 que l'on observe signale que l'agent possède une contrainte qui ne peut être relâchée. On a, en effet, imposé que l'utilisation des projecteurs soit obligatoire. Ce sont donc les contraintes de disponibilité qui sont relâchées.

Bibliographie

- [CMPSG02] Bernon Carole, Gleizes Marie-Pierre, Peyruqueou Sylvain, and Picard Gauthier. Adelfe, a methodology for adaptive multi-agent systems engineering. In *ESAW'02, Engineering Societies in the Agents World*, September 2002.
- [IGJ] Jacobson Ibar, Booch Grady, and Rumbaugh James. *The Unified Software Development Process*.
- [Jac95] Ferber Jacques. *Les systèmes multi-agents : vers une intelligence collective*. Inter-Editions, 1995.
- [JCMPP00] Casteran Jean-Christophe, Gleizes Marie-Pierre, and Glize Pierre. Des méthodologies orientées multi-agent. In Editions Hermès, editor, *Journées Francophones en Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, pages 191–207, Septembre 2000.
- [MPP02] Gleizes Marie-Pierre and Glize Pierre. Abrose : Multi agent systems for adaptive brokerage. In *Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems 27-28 May 2002, Toronto (Ontario, Canada) at CAiSE'02*, 2002.

Chapitre 8

Approche multi-agents adaptatifs en DIMA

Zahia Guessoum

Object et Agents pour Systèmes d'Informations et de Simulation (OASIS)

Laboratoire d'Informatique de Paris 6 (LIP6)

Case 169, 4 place Jussieu, 75252 PARIS Cedex 05

e-mail : Zahia.Guessoum@lip6.fr

8.1 Introduction

Cette dernière décennie a vu fleurir un très grand nombre de modèles, de langages et d'architectures dans le but de développer des systèmes à plusieurs composantes autonomes (appelées agents) pouvant coopérer entre elles. Ainsi, de nombreux travaux sur les architectures d'agents ont tenté d'identifier les différentes fonctions et composantes (perception, communication, délibération,...) d'un agent et la façon dont elles sont organisées pour fournir le comportement global d'un agent. Ces différentes composantes sont ensuite réifiées pour faciliter la spécification d'un agent. Les architectures d'agents proposées offrent une solution à une partie des problèmes que posent le développement des applications multi-agents. En effet, le problème lié à la difficulté d'implémenter les systèmes multi-agents reste entier notamment par un non-spécialiste des systèmes multi-agents. La difficulté de réalisation d'un système multi-agents s'explique principalement de par la diversité, la richesse des paradigmes multi-agents, et la complexité des concepts sous-jacents (coordination, interaction, organisation, etc.). Cette complexité rend l'utilisation de la majorité des outils existants toujours difficile pour ceux qui ne sont pas les concepteurs et souvent presque impossible pour les non-spécialistes des systèmes multi-agents. L'objectif de ce rapport est de montrer que la modularité et la réutilisabilité peuvent apporter une solution à ces problèmes. Nous proposons une plate-forme basée sur une architecture modulaire et une bibliothèque de composants. Elle est fondée sur l'extension des facilités de modélisation et d'implémentation de langages à objets. Nous voulons rendre un environnement à objets plus approprié aux problèmes de systèmes multi-agents en lui incorporant des représentations spécifiques et des structures de contrôle. Plus concrètement, dans cet article nous décrivons l'extension d'un modèle d'objets en modèles d'agents génériques et modulaires (incarnés par la plate-forme DIMA -Développement et Implémentation de Systèmes Multi-Agents). Nous illustrons ensuite l'utilisation de DIMA par l'exemple de l'emploi du temps.

8.2 Architecture d'agents

Les systèmes multi-agents existants sont classés en général en deux groupes : les systèmes cognitifs et les systèmes réactifs. Les systèmes multi-agents cognitifs sont fondés sur la coopération d'agents capables, à eux seuls, d'effectuer des opérations complexes [DEM 90] [HAY 95] [JEN 92].

Chaque agent dispose d'une capacité de raisonnement, d'une aptitude à traiter des informations diverses liées au domaine d'application, et d'informations relatives à la gestion des interactions avec d'autres agents et l'environnement. Dans un système multi-agents réactif [BRO 86] [FER 92], le comportement complexe du système émerge de la coexistence et de la coopération d'agents au comportement simple. Les agents réactifs ne disposent que d'un protocole de communication réduit. Ils répondent uniquement à une loi de type stimulus/réponse. Récemment, une synthèse s'est amorcée entre ces deux orientations et on parle alors d'agents hybrides [FER 95] [MÜL 94]. Ces agents combinent des propriétés cognitives et réactives généralement logées dans des modules différents. Ce type d'architecture pose le problème du contrôle à deux niveaux : la coordination classique entre agents, et la coordination entre modules au sein même de l'agent. Les architectures hybrides apportent une solution au problème de l'intégration des aspects réactifs et cognitifs. Ces architectures présentent des qualités indéniables de génie logiciel. Leur organisation modulaire permet entre autre l'intégration de capacités très hétérogènes en terme de programmation. Cette hétérogénéité est nécessaire pour répondre à l'intégration des tâches très diversifiées qu'a à remplir un agent depuis le raisonnement jusqu'à l'interaction qui peuvent faire appel à des concepts de système des plus spécialisés. L'intérêt est de pouvoir réutiliser des méthodes existantes notamment des méthodes d'intelligence artificielle [GUE 01]. La modularité introduit beaucoup de souplesse dans la gestion des capacités en autorisant à moindre coût l'échange de modules dans un but d'évolutivité ou de test. Une architecture modulaire fait de l'agent un système ouvert [GUE 01]. Cependant, les architectures hybrides sont complexes et ne seraient pas appropriées à la modélisation d'agents dont le comportement est très simple. Par exemple, la conception d'un agent avec Interrap (voir [MÜL 94]) nécessite la conception des trois modules correspondant aux trois couches (comportement, planification, coopération) de l'architecture. Il est donc difficile d'utiliser cette architecture pour concevoir un agent dont le comportement est très simple (par exemple, basé sur le principe de l'éco-résolution [FER 92]). Ces architectures hybrides ne résolvent pas le problème de granularité. Elles ne permettent pas la conception de systèmes multi-agents dont la granularité des agents est variable.

8.2.1 Un modèle d'architecture qui dépasse la dichotomie classique

Le modèle d'architecture d'agents DIMA propose de décomposer chaque agent en différents composants dont le but est d'intégrer des paradigmes existants notamment des paradigmes d'intelligence artificielle. Un agent peut ainsi avoir un ou plusieurs composants qui peuvent être réactifs ou cognitifs (voir figure 1). Ce modèle permet de dépasser la dichotomie classique réactif/cognitif en permettant de définir des agents hétérogènes au sein d'une même application. Chaque agent est composé d'un ou de plusieurs composants.

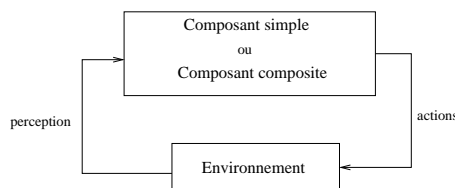


FIG. 8.1 – L'architecture d'agents de DIMA

La modularité offre ainsi plusieurs avantages à cette architecture :

- Possibilité de définir des agents à granularité variable.
- Possibilité de définir des agents à structure adaptative. Chaque agent peut dynamiquement changer ses composants ainsi que les relations entre ces différents composants.
- Possibilités d'intégrer différents modèles d'agents.
- Possibilité de définir des bibliothèques de composants réutilisables.
- Etc.

Le modèle d'architecture d'agents DIMA est donc fondé sur la conclusion suivante : les travaux sur les architectures d'agents ont engendré plusieurs résultats intéressants. Il est très difficile de comparer ces architectures dans le but de trouver la meilleure architecture. Chacune est bien appropriée à une catégorie d'agents. Une bonne plate-forme intègre les différentes architectures d'agents existantes ainsi que d'éventuelles nouvelles architectures. Ainsi, le modèle d'architecture d'agents proposé par DIMA peut être vu comme un modèle 'ouvert', une proposition d'agent minimal est faite permettant, par raffinements successifs, d'ajouter des fonctionnalités fournies par les différentes bibliothèques de la plate-forme. La force de DIMA réside donc à la fois dans sa proposition de modèle d'architecture d'agents modulaire, mais également dans les différentes bibliothèques disponibles. Chaque agent est un composant simple ou composant composite [MEU 01] qui gère l'interaction de l'agent avec son environnement et qui représente son comportement interne (voir figure 1). L'environnement regroupe tous les autres agents ainsi que les entités qui ne sont pas des agents. Dans les sections suivantes, nous présentons d'abord les composants proactifs, la brique de base de cette architecture. Ensuite nous décrivons les différents modèles d'agents qui sont fondés sur ces composants proactifs.

8.2.2 Les Composants proactifs

Un composant proactif représente une entité autonome et proactive. Le noyau de base de DIMA est un framework de composants proactifs. Il est composé de plusieurs classes et de leurs méthodes. Nous avons choisi un ensemble minimal de classes et de méthodes définissant les fonctionnalités des composants proactifs. Ces fonctionnalités peuvent être étendues dans les sous-classes. Ce noyau minimal est principalement composé de la classe ProactiveComponent (voir Figure 8.2).

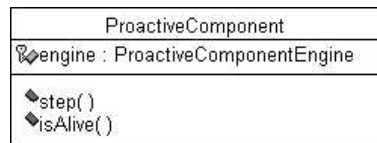


FIG. 8.2 – Le *Contract Net*

Contrairement aux objets dont l'activité est restreinte au traitement des messages envoyés par les autres objets, le composant proactif a différentes compétences et son activité n'est pas restreinte à l'envoi et à la réception de messages. Une instance de la classe ProactiveComponent décrit :

- Le but du composant proactif, il est implicitement ou explicitement décrit par la méthode `isAlive()`.
- Les compétences de base (ou comportements) du composant proactif, une compétence est une suite d'actions qui permettent de changer l'état interne de l'agent ou envoyer des messages aux autres agents. Dans notre implémentation une compétence pourrait être implémentée sous forme d'une méthode Java.
- Le méta-comportement, il définit la manière dont les compétences sont sélectionnées, séquencées et activées en fonction du but.

Différents paradigmes (automate, règles de production,...) sont réutilisés pour définir de nouvelles classes de composants proactifs. Par exemple, le comportement d'un composant proactif peut être modélisé par un ATN (Augmented Transition Network) dont les états correspondent aux différents états d'un composant proactif. A chaque état est associée une ou plusieurs transitions. Une transition est définie par une condition et une action. L'action de transition est l'activation d'une compétence.

Le composant proactif peut être considéré comme une bonne brique de base pour construire des agents. Par exemple, nous pouvons implémenter avec ProactiveComponent des agents réactifs tels qu'ils sont décrit par J. Ferber et A. Drogoul [?]. Les comportements d'un agent réactif, dans ce cas, correspondent aux comportements de base de l'éco-résolution tels que : rechercher satisfaction,

TAB. 8.1 – Main methods of ProactiveComponent

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

fuire, agresser. La méthode `step()` est fondée sur le principe de l'éco-résolution. Elle est fonction des quatre états suivants d'un agent (voir [?]) et les actions qui leur sont associées :

- satisfait : l'agent ne fait rien.
- rechercheSatisfaction : l'agent cherche à se satisfaire.
- rechercheFuite : l'agent a été agressé et tente de fuir.
- fuite : l'agent fuit.

La méthode `step()` implémente donc les règles suivantes :

- Si le but de l'agent n'est pas satisfait, l'agent recherche satisfaction.
- Quand l'agent se satisfait, il s'endort.
- Quand l'agent ne peut se satisfaire, il recherche les gêneurs parmi ses accointances et les agresse (leur demande de fuir).
- Si l'agent est agressé, il essaie de fuir.

8.2.3 Agents interactifs

Pour définir des agents interactifs, les composants proactif (voir section précédente) ont été enrichis d'un module de communication (voir Figure 8.3). Ce dernier gère les interactions avec les autres agents. Les comportements de ces agents interactifs intègrent notamment deux types d'actions :

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

Chaque agent interactif a deux composants :

- une boîte aux lettres pour stoker ses messages,
- un module de communication pour gérer les messages envoyés et reçus.

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

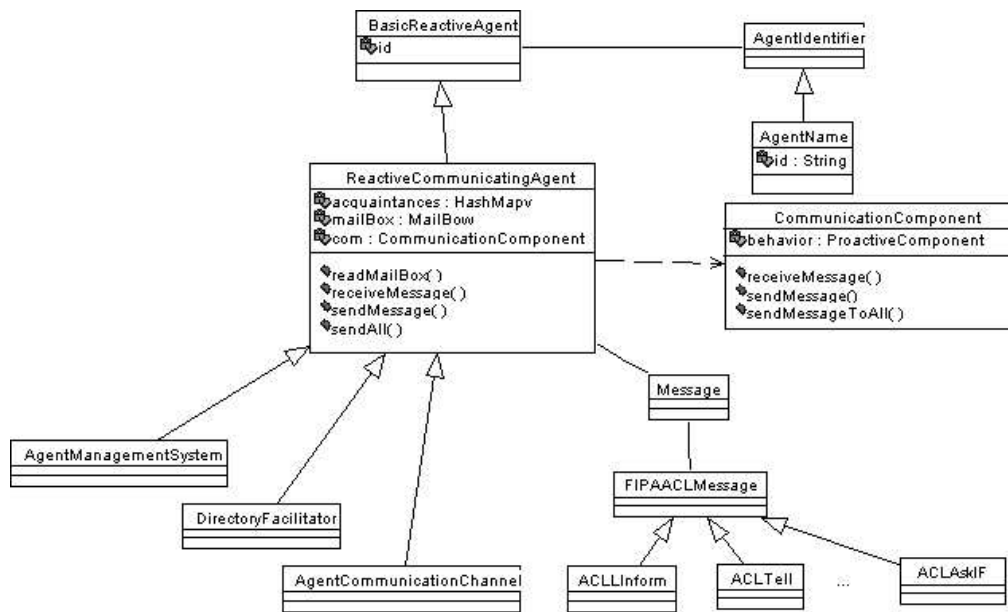


FIG. 8.3 – Composants de base d'un agent interactif et agents de l'architecture FIPA

8.2.4 Agents adaptatifs

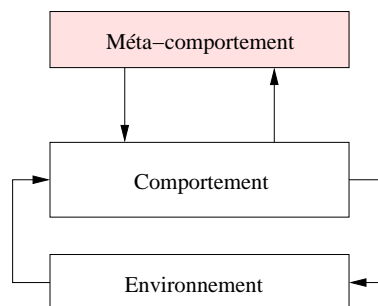


FIG. 8.4 – Architecture d'agents adaptatifs

Plusieurs chercheurs ont souligné l'intérêt de la représentation explicite et séparée du contrôle ou de l'aspect réflexif des architectures de méta-niveaux (voir par exemple [?], [?], [?] et [?]). En adoptant, cette représentation explicite et séparée du contrôle, nous proposons d'introduire un méta-comportement dans notre architecture d'agents (voir Figure 8.4). Ce méta-comportement donne à chaque agent la capacité de prendre des décisions appropriées au sujet de contrôle ou d'adapter son comportement avec le temps à des nouvelles circonstances. Il fournit à l'agent un mécanisme d'auto-contrôle pour adapter dynamiquement ses comportements selon son état interne et celui de son environnement.

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

- un ensemble de conditions destinées à être utilisées par le méta-comportement pour tester le contexte de l'agent,

- un ensemble d’actions qui peuvent être utilisées par le méta-comportement pour modifier l’état de l’agent et/ou l’état de son environnement.

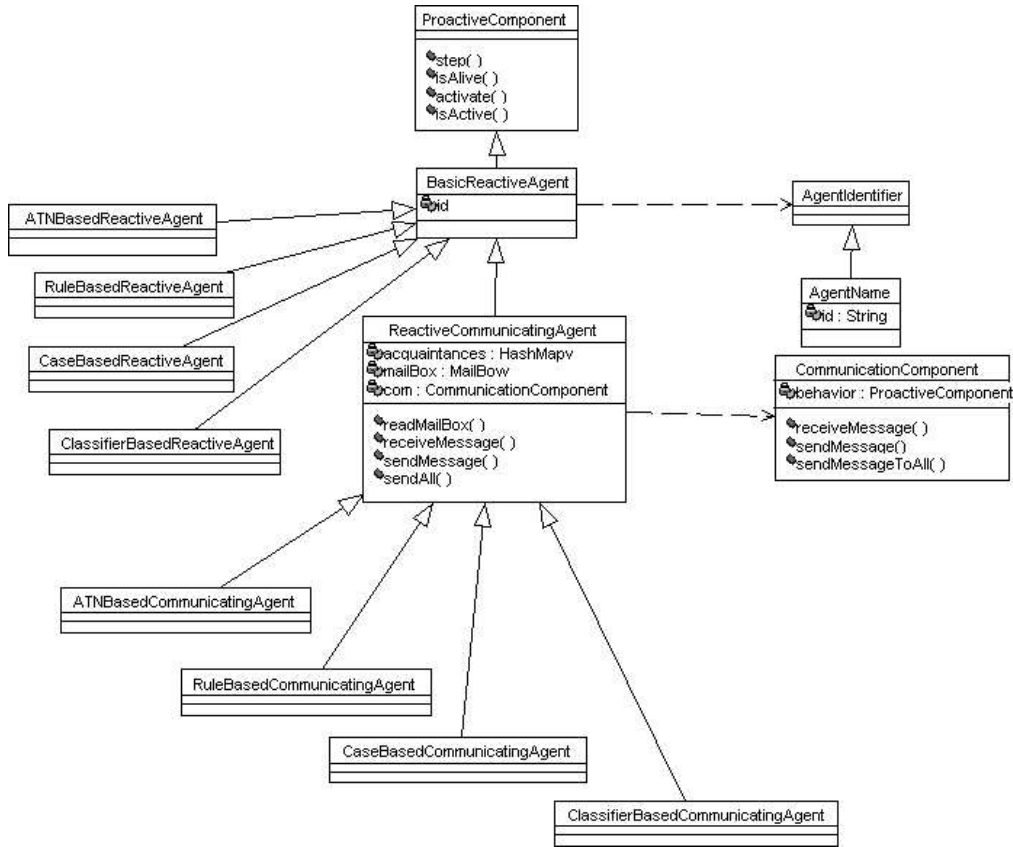


FIG. 8.5 – Exemples de classes d’agents

Cette structure est très générale et peut être utilisée avec n’importe quel système décisionnel (ATN, base de règles, etc.), il suffit qu’il fournisse un lien entre les conditions et les actions. Figure 8.5 donne des exemples de classes d’agents adaptatifs. Dans la classe ATNBasedProactiveComponent, le méta-comportement est décrit par un ATN et la méthode `step()` permet d’activer une transition dont la condition est vérifiée.

```

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

```

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

```

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

```

Par exemple dans le modèle éco-résolution, le méta-comportement d’un agent peut-être décrit par un automate dont les états sont : satisfait, rechercheSatisfaction, rechercheFuit, fuite et les transitions correspondent aux règles (voir Figure 8.6).

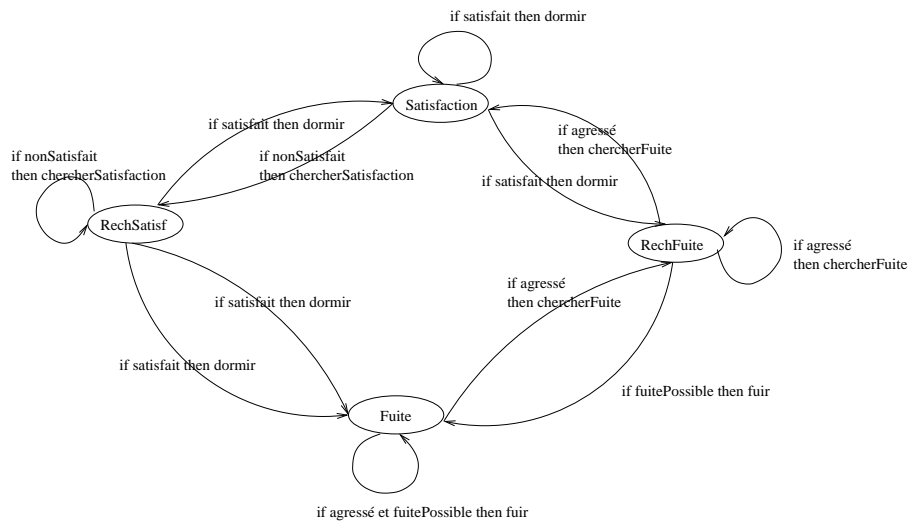


FIG. 8.6 – Méta-Comportement d'un agent dans le modèle de l'éco-résolution

Un méta-comportement a ainsi deux principaux rôles :

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

Ces deux rôles peuvent être combinés pour avoir à la fois des agents qui adaptent leur structure et leur comportement. Ce modèle peut être utilisé hors-ligne pour faciliter la conception d'un agent. Il suffit donc de simuler l'évolution de l'environnement des agents et d'attendre que les bases de règles se stabilisent. Les agents peuvent ensuite être activés avec la base de règles ainsi obtenue.

8.3 Principales caractéristiques de l'architecture

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

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

- Hétérogénéité : l'hétérogénéité est une propriété très importante dans les systèmes multi-agents. Cependant, la majorité des systèmes existants ne fournissent pas la possibilité de réaliser des agents hétérogènes. Différents types d'hétérogénéité sont considérés :
 - multiplicité des plates-formes d'exécution,
 - multiplicité des formalismes de représentation des connaissances,
 - multiplicité des modèles d'agents.

8.4 DIMA

DIMA est un environnement de développement de systèmes multi-agents dont le développement a débuté en 1993. DIMA est basée sur le modèle d'architecture proposé dans ce chapitre. La première version de DIMA a été implémentée en Smalltalk-80 et a été ensuite portée en JAVA (voir <http://www-poleia.lip6.fr/guessoum/dima.html>).

DIMA est principalement caractérisée par son architecture d'agents modulaire qui permet de dépasser la dichotomie classique en offrant la possibilité de développer des applications multi-agents dont la granularité des agents est variable. DIMA offre en effet des bibliothèques (classes JAVA) offrant les briques de base pour construire des modèles d'agents divers. Ces différentes briques ont pour but d'offrir à l'utilisateur une grande variété de paradigmes (par exemple automate, règle de production, etc.) d'une part, et d'autre part, une implémentation des différentes propositions conceptuelles introduites par la communauté multi-agents (BDI, KQML, ACL, etc.). Les bibliothèques de DIMA regroupent des classes qui peuvent être réutilisées et/ou adaptées pour construire facilement des agents. Ces classes peuvent être instanciées ou sous-classées pour implémenter un agent ou une de ses composantes.

DIMA offre également un ensemble de frameworks (sous forme de packages) pour les différents paradigmes (règles, ATN, classeurs, ...). Par exemple, un framework (l'ensemble de classes du package `Gdima.tools.automata`) d'ATN est proposé. Pour implémenter un ATN avec ce framework, il suffit de spécifier les différents états et les différentes transitions.

DIMA a été utilisée pour développer plusieurs applications réelles (Ventilation artificielle [?], simulation des modèles économiques [?], simulation d'un marché électrique, le filtrage de l'information sur les réseaux, etc.) par les auteurs mais également par d'autres personnes n'ayant pas participé à son développement. Ces applications peuvent être des simulations, des résolutions de problèmes ou des systèmes de contrôles ayant éventuellement des contraintes temps-réel. Par ailleurs des *benchmarks* (factorielle, proies/prédateurs, ventes aux enchères, ...) sont proposés pour illustrer l'utilisation des différents modèles et bibliothèques.

8.5 Variante 1 du problème de l'emploi du temps

Dans DIMA, le développement d'un système multi-agents nécessite les principales étapes de développement :

- Analyse,
- Conception,
- Implémentation,
- Déploiement.

Les paragraphes suivants illustrent ces différentes étapes par la variante 0 de l'exemple de l'emploi du temps.

8.5.1 Analyse

Dans DIMA, l'analyse commence par la modélisation du domaine. Cette modélisation du domaine est similaire aux processus d'élaboration de modèles conceptuels dans les méthodes objets. Son implémentation nécessite la définition d'un ensemble de classes JAVA. La figure suivante donne les classes qui ont été identifiées pour le problème de l'emploi du temps sans tenir compte des agents, les classes `Professeur` et `GroupeEtudiant` seront transformées en classes d'agents.

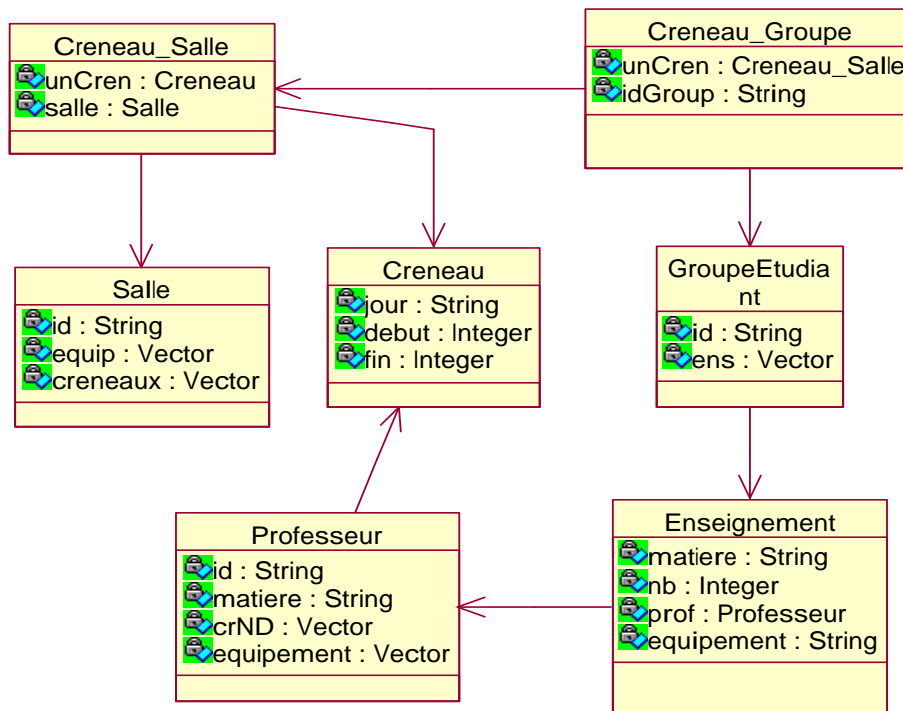


FIG. 8.7 – Domaine de l'emploi du temps

Pour l'analyse multi-agents du problème de l'emploi du temps, nous avons utilisé la méthode Cassiopée. Le principe fondateur de la méthode Cassiopée est d'articuler l'analyse et la conception d'un système multi-agents autour de la notion d'organisation et de la notion de rôle. Nous avons identifié les rôles suivants : Enseignant et GroupeEtudiants. Nous avons ensuite défini le graphe de dépendances et les interactions entre ces deux rôles. Le protocole d'interaction utilisé entre ces deux agents est le contract net. Ce graphe a ensuite été analysé pour identifier les agents. L'analyse est basée sur plusieurs critères tels que le degré de dépendance et la distribution. Chaque rôle a été ainsi associé à un agent.

8.5.2 Conception et Implémentation

Dans DIMA, un système multi-agents est un ensemble d'agents et un ensemble d'objets représentant le domaine. Les principales étapes pour la conception et l'implémentation d'un agent sont les suivantes :

- détermination de l'ensemble de ses compétences (ou comportements),
- détermination du type de l'agent (réactif ou adaptatif) et son méta-comportement. Si l'ensemble des stimuli auxquels l'agent doit réagir sont connus à la conception ainsi que l'ensemble des règles de réaction (Si stimulus et état alors action), l'agent est réactif, sinon ils est adaptatif.

Pour la variante 1 de l'emploi du temps, les agents sont réactifs et leur méta-comportement est basé sur le principe de l'éco-résolution proposé par Jacques Ferber et Alexis Drogoul [Fer 92]. Nous avons, en effet, choisi deux sous classes de ATNBasedCommunicatingAgent : ParticipantCNPAgent et InitiatorCNPAgent.

Ces deux classes implémentent les deux rôles (Participant et Initiateur) du contract net. Elles utilisent les deux ATN correspondant à ces deux rôles et implémentent les différentes compétences (méthodes) utilisées par ces ATN (voir Figure 8.8 et Figure 8.9). Les méthodes utilisées dans ces deux ATN (sendCFP, noMail, readAllMessages, decisionProcess, isAccepted) sont donc

génériques. Les deux classes Enseignant et GroupeEtudiant sont respectivement sous classes de ParticipantCNPAgent et InitiatorCNPAgent.

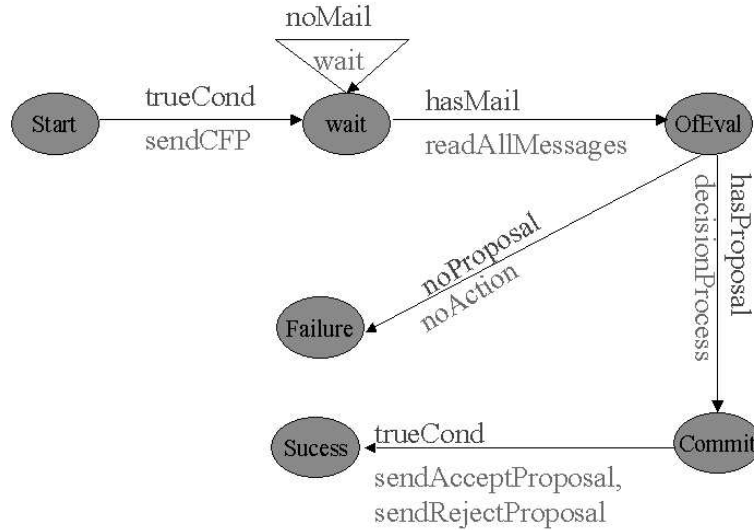


FIG. 8.8 – L'ATN décrivant le l'Initiateur du CNP

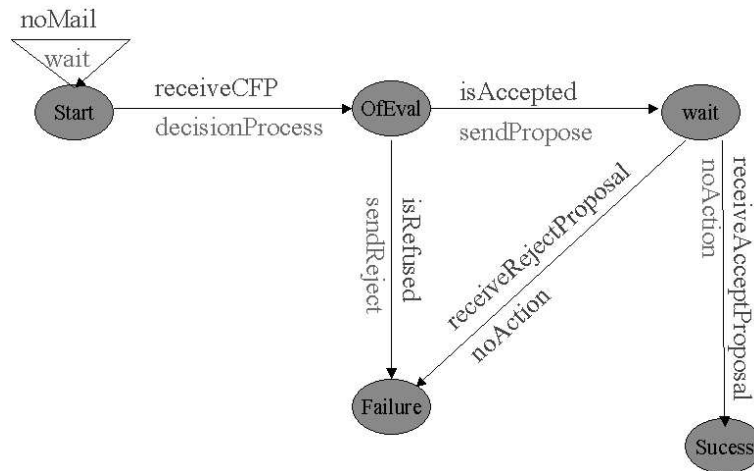


FIG. 8.9 – L'ATN décrivant le Participant du CNP

Leur méta-comportement est fondé sur le principe de l'éco-résolution. Il est fonction des quatre états suivants d'un agent (voir [?]) et les actions qui leur sont associées :

- satisfait : l'agent ne fait rien.
- rechercheSatisfaction : l'agent cherche à se satisfaire.
- rechercheFuite : l'agent a été agressé et tente de fuir.
- fuite : l'agent fuit.

Le comportement peut-être donc décrit comme suit :

- Si le but d'un agent n'est pas satisfait, l'agent rechercheSatisfaction.
- Quand un agent se satisfait, il s'endort.
- Quand un agent ne peut se satisfaire, il recherche les gêneurs parmi ses accointances et les agresse (leur demande de fuir).

Pour les agents Enseignants :

- satisfait : s'il n'a plus d'appel à propositions des groupes des autres agents (groupeEtudiant).
- rechercheSatisfaction : Répond à l'appel à proposition des autres agents.
- rechercheFuite : l'agent a été agressé et tente de fuir.
- fuite : l'agent fuit en annulant un cours choisi aléatoirement.

Pour les agents GroupeEtudiants :

- satisfait : si toutes les contraintes sont remplies.
- rechercheSatisfaction : envoie des appels à propositions aux enseignants (Contract Net). S'il n'a pas de propositions, il agresse les enseignants en leur envoyant un message " agresse".
- Les GroupeEtudiants ne sont jamais agressés. Ils ne cherchent donc pas à fuir. ‘

Le cycle de vie de chacun de ces deux types d'agents (Enseignant et GroupeEtudiant) implémente l'algorithme d'éco-résolution correspondant.

8.5.3 Déploiement

Après avoir modélisé le domaine et l'ensemble des agents, DIMA offre deux possibilités : 1) exécution des agents sur une seule machine, 2) répartition des agents sur plusieurs machines. L'exécution des agents sur une seule machine nécessite l'initialisation du réseau de communication (accointances) avant l'activation des agents en utilisant la méthode : `activate()`

La répartition des agents DIMA sur plusieurs machines s'appuie sur le middleware DarX [Marin 01]. La répartition des agents nécessite :

- itialisation du serveur DarX sur les différentes machines;
- activation des agents en utilisant la méthode : `activateWithDarx(activation_URL, activation_PORT)`

8.6 Conclusion

Aujourd'hui, l'intérêt d'une méthodologie de conception et d'implémentation de systèmes multi-agents n'est plus à démontrer. De nombreux travaux de recherche continuent à explorer les possibilités que fournirait une méthodologie [DEM 98] [DRO 98] [FER 98] [WOO 00]. Toutefois, la plupart de ces propositions se base sur une architecture d'agent purement 'conceptuelle', l'idée n'étant pas de fournir un moyen à partir d'une spécification d'un problème donné, de déduire une solution opérationnelle, mais de se restreindre à l'agentification du problème.

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

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

Chapitre 9

Conclusion

Le problème d'Emploi du temps que nous avons décrit dans cet article présente de nombreux avantages pour la comparaison de plateformes. Il nécessite la mise en place de plusieurs mécanismes dont les agents ont fréquemment besoin comme l'autonomie, la distribution, ou le raisonnement propre à chacun. Ce problème est donc bien adapté à la comparaison de plateformes que nous venons d'effectuer. Chacun peut maintenant, après avoir vu ces plates-formes à l'œuvre, se faire une idée plus précise des avantages et inconvénients de chacune, ainsi que leurs facilités de mise en œuvre.