Des agents intelligents dans un environnement de communication multimédia : Vers la conception de services adaptatifs

Romaric Charton

U.F.R. Sciences & Techniques Mathématiques, Informatique et Automatique
Ecole Doctorale IAE+M Lorraine
Département de Formation Doctorale en Informatique

N° attribué par la bibliothèque
|__|__|__|__|__|__|__|__|__|__|

Des agents intelligents dans un environnement de communication multimédia : vers la conception de services adaptatifs

THESE

Présentée et soutenue publiquement le 2 décembre 2003

pour l'obtention du

Doctorat de l'Université Henri Poincaré - Nancy 1

Spécialité Informatique

par

Romaric CHARTON

Composition du Jury

Président du jury :	Dominique MERY	Professeur de l'Université Henri Poincaré, Nancy I
Rapporteurs :	Anne NICOLLE	Professeur de l'Université de Caen
	Abderrafiaa KOUKAM	Professeur de l'Université de Technologie de Belfort-Montbéliard
Examinateur :	Brahim CHAIB-DRAA	Professeur de l'Université de Laval
Directeurs de thèse :	Jean-Paul HATON	Professeur de l'Université Henri Poincaré, Nancy I Membre de l'Institut Universitaire de France
	Anne BOYER	Maître de Conférences de l'Université Nancy 2
Invité :	Michel LEDERMAN	Co-fondateur, DIALOCA.

Laboratoire Lorrain de Recherche en Informatique et ses Applications - UMR 7503

Table des Matières

Table des Illustrations

Table des Tableaux

Remerciements

Je souhaiterais rendre ici hommage à toutes ces personnes formidables qui m'ont toujours entouré et soutenu et sans qui cette thèse n'aurait jamais vu le jour.

Je remercie, tout d'abord, Dominique Méry pour avoir suivi mes travaux de recherche en tant que référent interne et pour m'avoir fait l'honneur de présider mon jury de soutenance.

Mes pensées vont ensuite à Abderrafiaa Koukam pour l'intérêt qu'il a toujours porté à mes travaux depuis notre rencontre et pour avoir accepté d'être rapporteur de ce travail. Merci également pour son soutien et ses conseils.

A Anne Nicolle, pour le regard avisé qu'elle a bien voulu porter sur mon travail en tant que rapporteur. Merci aussi pour ses remarques pertinentes qui m'ont permis d'affiner mes modèles.

A Brahim Chaib-Draa, qui m'a fait le plaisir de participer à mon jury comme examinateur. Merci également pour tous ses commentaires concernant mes travaux.

A Anne Boyer, pour m'avoir fait l'honneur de diriger ma thèse, mais aussi pour tant d'autres choses. Merci pour sa disponibilité de toute heure, pour nos longues discussions, pour ses conseils et pour tous ses encouragements. Souvent, je sortais de réunion avec le plein de travail, mais aussi avec le plein d'idées et de moral.

A Jean-Paul Haton, pour l'attention portée à mes travaux et pour ses conseils avisés. Merci également pour ses cours et ses présentations que j'ai eu la chance et le plaisir de pouvoir suivre et qui n'ont fait que renforcer ma passion pour l'intelligence artificielle.

A François Charpillet, qui m'a accueilli chaleureusement dans son équipe de recherche. Merci pour toute l'aide et le soutien qui ont rendu cette thèse possible et qui m'ont permis d'avancer dans mes travaux.

A Vincent Chevrier, pour m'avoir parlé un jour de ses travaux sur les systèmes multi-agents et pour être depuis resté toujours disponible, passionné et passionnant.

A Olivier, merci pour tant de discussions théoriques, pratiques et variées qui pouvaient commencer au hululement de la hulotte dans la neige ou se terminer bien plus tard dans la potamobile.

A Régis, pour toutes nos aventures nancéennes, parisiennes, allemandes et autres. Finalement, depuis que l'on est monté ce soir là dans ce train sans queue ni tête, on en a fait de la route, ne trouves-tu pas ?

A David, notre expert en pêche à la truite à grands coups de bichocos et de tux, qui nous apportait toujours un peu de Marseille dans le bureau et qui depuis est parti rejoindre la Californie.

A Laurent, pour ses explications claires et détaillées sur certains algorithmes obscurs et sur d'autres mystères informatiques. Merci à celui qui avait toujours une réponse à la question Java du jour.

A Alain, pour ses conseils, ses relectures et ses corrections qui, telle l'épée de d'Artagnan, corrigeaient mes formules et ma prose comme il se devait.

A Christine, Vincent, Loïc, Simon, Raghav, Daniel, Eric, Franck, ainsi que les autres membres de l'équipe Maia sans oublier les anciens comme Makram, Rédimé, Iadine, Bruno ou Nono. Merci pour leur aide, leur dynamisme, leur bonne humeur et leur amitié.

A Martine, notre formidable assistante de projet, tant pour sa disponibilité que pour sa gentillesse.

A nos voisins de l'équipe Parole, comme Armelle, David, Fabrice, Yassine et nos deux Vincents et à tous les autres de RFIA.

Je remercie Michel Lederman pour son enthousiasme dans notre collaboration et dans nos projets depuis le tout début. Merci pour l'intérêt qu'il a toujours porté à mes travaux. Merci également à Thierry Brizzi et Yvan Maillet pour m'avoir initié aux arcanes d'Unimédia, à Dominek Saelens qui s'est également penché sur mes travaux, à Michel Repiso, ainsi que toute l'équipe Dialoca.

Grâce à vous tous, je garderai de ces années de travail le souvenir de très bons moments

A Sylvie, mon épouse, qui, armée d'une patience d'ange, m'a soutenu tout du long et n'a pas hésité à participer aux nombreuses relectures.

A Claire, ma fille, qui est déjà dans le monde des blocs et qui a gentiment a accordé à son papa du temps pour terminer sa thèse.

A toute ma famille, ma belle-famille et mes amis qui ont accepté avec compréhension ces longues années de sacrifice et qui m'ont, eux aussi, soutenu et aidé autant qu'ils le pouvaient.

Avec une petite pensée nostalgique pour Thérèse Devine : c'était il y a plus de vingt ans maintenant. Et puis, chic, on va enfin pouvoir se parler ! (l'intéressé comprendra).

La liste pourrait encore être longue, mais j'entends déjà des remarques sur le nombre de pages... aussi je prie ceux qui ne se retrouveraient pas dans ces lignes de ne pas m'en tenir rigueur.

Je vous dis à toutes et à tous MERCI !

P.S. En bon informaticien qui se respecte, je ne saurais terminer sans penser à toutes les machines qui ont eu la gentillesse de m'épargner, le plus souvent, les bugs tant redoutés pendant la période de rédaction...

Introduction

Au cours des dernières décennies, les ordinateurs n'ont pas cessé de se complexifier et la quantité des données qu'ils sont capables de manipuler est devenue gigantesque. Dans tous les domaines, l'utilisation des ordinateurs est devenue très courante pour calculer, mais aussi concevoir, traiter du son, de la vidéo, rechercher des données, échanger des messages et bien d'autres tâches des plus simples aux plus complexes. Cependant, même si les systèmes informatiques sont présents où que nous allions et même s'ils sont capables d'aller fouiller les grands annuaires mondiaux en quelques secondes, de nombreux défis subsistent. Par exemple, comment faciliter la recherche d'informations pertinentes pour un utilisateur donné ou l'aider à effectuer les traitements dont il a besoin ? Une solution consiste à l'assister en tenant notamment compte de ses préférences et de ses habitudes.

Force est de constater que les interfaces des systèmes informatiques, bien souvent conçues pour les uns, restent obscures ou inadaptées pour les autres. Ainsi, une personne n'ayant jamais utilisé de moteur de recherche sur Internet, se trouve souvent désemparée face à un champ vide, suivi d'un bouton sur lequel il est écrit "ok". Ceci est dû au fait qu'elle ne sait pas quelle démarche adopter, alors qu'elle a une idée plus ou moins précise de ce qu'elle veut. A l'inverse, une personne expérimentée n'éprouve aucune difficulté : elle remplit le champ, parfois même trop vite et au détriment des erreurs de frappe, puis elle clique sur le bouton. Une fois les réponses affichées, la personne expérimentée balaye de l'œil l'ensemble des résultats et identifie très vite celui qu'elle pense être adéquat. Notre utilisateur novice, quant-à-lui, peut avoir des difficultés à s'y retrouver dans ce foisonnement de données. Cela se manifeste d'autant plus lorsque les résultats sont exprimés dans une langue qui ne lui est pas familière ou sans explications sur les nombres qui les accompagnent... tout ceci sans parler des multiples publicités et autres fenêtres qui s'ouvrent parfois en cascade.

Et pourtant, cet utilisateur va tirer parti de cette expérience s'il a une autre recherche à effectuer et qu'il persévère : il se souviendra des actions qu'il a faites, des échecs qu'il a rencontrés et il va tenter de s'en sortir le mieux possible. Il va par ailleurs chercher à se documenter sur le système, de façon globale, puis sur des fonctions particulières. En pratique, plus il sera confronté à une situation donnée, mieux il s'en sortira. Au fil du temps, il pourra même devenir, lui aussi, un expert du système.

On assiste là à un phénomène curieux : si le système est conçu pour fournir facilement des informations à l'utilisateur, il nécessite finalement que cet utilisateur s'adapte au système, au fur et à mesure qu'il l'utilise, alors que ce devrait-être le contraire. Le problème réside dans le fait que l'utilisateur est sensé maîtriser le système en question et posséder toute la connaissance nécessaire à l'accomplissement de sa tâche. Les systèmes informatiques restent toujours des outils complexes et pour réaliser une action donnée, tout utilisateur, même expérimenté, peut avoir besoin d'être guidé à un moment ou à un autre. De plus, un système est souvent prévu pour une utilisation dans un contexte bien défini, alors qu'en pratique, les hypothèses de fonctionnement ne sont pas toujours réunies.

La tâche de rendre les systèmes robustes aux aléas et accessibles au grand public incombe au concepteur d'application qui doit prévoir tous les cas possibles. Bien souvent, une telle exhaustivité n'est hélas pas réaliste. Ce constat dénote un réel besoin de médiation entre le système et les utilisateurs, c'est-à-dire une intervention active pour faciliter leurs relations. Diverses solutions de médiation sont possibles. Le plus fréquemment, c'est au travers de la formation des utilisateurs que le problème est résolu : le formateur explique le fonctionnement du système à l'utilisateur et il le guide dans son apprentissage. Cette solution ne peut cependant être mise en place que pour des usages réguliers, car il est difficile de former un grand nombre d'utilisateurs, surtout s'ils sont occasionnels. Une solution alternative consiste à rendre le système capable de s'adapter, mais cela suppose que le système puisse être modifié, ce qui est loin d'être toujours possible.

Nous proposons une conception d'assistants adaptatifs qui se souviennent des caractéristiques et des préférences des utilisateurs et qui les aident à interagir avec les systèmes. Nous avons cherché à obtenir un assistant qui tente de comprendre ce que l'utilisateur veut faire et qui cherche à l'amener vers la façon de réaliser cette tâche qu'il estime être la meilleure, en fonction de toutes les informations disponibles. Par ailleurs, nous considérons que ces échanges utilisent des moyens de communication dans des environnements réels où de nombreux aléas peuvent survenir. Les supports de transmission et les dispositifs utilisés transforment l'information, ce qui peut être la cause de bruits et d'erreurs qui ne peuvent pas être négligés. De même, la prise en compte des comportements humains nécessite de pouvoir pallier d'éventuels manques de précision. C'est pourquoi, un aspect important que nous avons cherché à prendre en compte est l'incertitude, tant liée aux informations que le système reçoit comme entrée, qu'aux résultats des actions qu'il effectue. Enfin, notre solution de médiation vise plus généralement à faire travailler conjointement un ensemble hétérogène d'intervenants en estompant leurs différences de langage, de connaissance ou de raisonnement, qu'ils soient spécialistes, fournisseurs d'informations, utilisateurs, etc.

1. Contexte de la thèse

Cette thèse s'est déroulée sous la direction d'Anne Boyer, maître de conférences à l'Université Nancy 2, et de Jean-Paul Haton, professeur à l'Université Henri Poincaré. Les travaux de recherche ont été menés au LORIA (Laboratoire Lorrain de Recherche en Informatique et ses Applications) dans l'équipe MAIA (MAchine Intelligente Autonome) dirigée par François Charpillet, directeur de recherche INRIA, qui a amplement participé à leur encadrement. L'équipe Maia s'intéresse à l'étude du comportement d'agents, qui sont des entités autonomes disposant de capacités de décision, mais aussi de moyens de percevoir et d'agir qui leur sont propres. Les travaux de recherche portent autant sur l'interprétation des phénomènes perçus par les agents, qu'à la production d'actions appropriées. L'équipe a acquis une expertise en matière d'apprentissage, de planification et d'intelligence collective, notamment au niveau de la modélisation de comportements prenant en compte l'incertitude des observations et des actions. Les champs d'application sont nombreux : le domaine de la télémédecine, la biologie du comportement, les robots mobiles, etc.

La thèse s'est inscrite dans le cadre d'une convention Cifre, en collaboration avec la société Dialoca. La société Dialoca a été créée en octobre 1996 par Michel Lederman et Michel Repiso, sous le nom de MIC2. Localisée à Paris, son activité concerne l'édition de solutions logicielles basées sur la reconnaissance vocale. A cet effet, Dialoca propose la plate-forme Unimédia, réalisée par Thierry Brizzi et Yvan Maillet, qui permet l'accès aux moyens de communication actuels (téléphone, web, e-mail, SMS...) au travers d'une approche uniformisée. Cet environnement offre différents services, tels que la consultation d'une base de données par téléphone avec synthèse et reconnaissance de la parole ou la conversation en temps réel entre un vendeur et un client, en utilisant plusieurs médias.

Dirigés au départ vers des applications de commerce électronique, nos objectifs étaient de trouver comment accueillir un client non-spécialiste et de déterminer ses besoins en lui posant des questions. Une application typique consistait à déterminer, à partir d'un vaste catalogue, quelles étaient les meilleures offres que l'on pouvait proposer à un client en le sollicitant un nombre restreint de fois sur une borne interactive. Il s'agissait également de conseiller ce client, en lui fournissant des conseils sur les différents produits et de mener une stratégie commerciale. Après avoir donné lieu à un démonstrateur, l'étude s'est ensuite étendue et généralisée à l'ensemble de la palette des services de communication multimédia. Les objectifs étaient notamment de pallier l'aspect statique des services et le besoin d'exhaustivité lors de leur conception. Il s'agissait d'envisager ce que peuvent apporter la modélisation et la production de comportements au domaine de la communication multimédia. Nous nous sommes donc intéressés à la conception de services adaptatifs aux utilisateurs et capables de réagir aux divers évènements pouvant survenir lors de leur exécution. Il s'agit d'un problème complexe, car l'environnement dans lequel se déroulent les services est très réactif, ouvert et évolutif. Par ailleurs, l'implication des humains dans le système et l'utilisation de technologies comme la reconnaissance vocale nous amènent à tenir compte du caractère incertain des phénomènes qui sont en jeu. Nos travaux ont été mis en œuvre dans le cadre d'applications réelles que nous avons pu modéliser avec notre approche afin de la valider.

2. Objectifs

Cette thèse expose nos travaux de recherche visant à favoriser la coopération entre différents intervenants, humains ou artificiels, qui communiquent au travers d'un système informatique. Elle se positionne dans le cadre de l'intelligence artificielle avec une approche agent. Cette approche permet de modéliser les différents intervenants, en tenant compte de l'aspect distribué des problèmes, tant du point de vue physique que du point de vue fonctionnel. Avec cette modélisation nous pouvons mieux appréhender la dynamique des interactions et chercher à coordonner les comportements des agents pour réaliser des tâches applicatives. La thèse soulève un bon nombre de questions difficiles :

- *Comment faciliter les échanges entre des intervenants différents les uns des autres ?*

  De par leur nature, les agents que nous considérons n'ont pas les mêmes capacités, n'utilisent pas les mêmes langages de communication et ne disposent pas des mêmes connaissances. Dépasser ces différences pour réussir à les faire interagir devient alors un vrai défi et c'est pourquoi ***l'hétérogénéité*** des agents doit être explicitement prise en compte pour la conception des systèmes coopératifs.

- *Comment détecter et prendre en compte les phénomènes d'incertitude, de bruit et d'imprécision et plus généralement les problèmes qui peuvent survenir lors d'une interaction ?*

  Les agents communiquent au travers d'une suite de transmissions et de transformations de l'information qui peut parfois s'avérer très complexe. Tout au long de cette chaîne, diverses altérations peuvent survenir et remettre en cause le bon déroulement des interactions. Il est important que les agents soient sensibles à ces problèmes et puissent ***réagir*** de façon appropriée. De la même façon, ils doivent disposer des moyens nécessaires pour pouvoir ***prendre les meilleures décisions possibles, alors qu'ils ne disposent que d'informations incomplètes ou incertaines***.

- *Comment produire un comportement en cherchant à répondre à un maximum de cas de figure sans pour autant adopter une démarche exhaustive ?*

  La conception des systèmes informatiques se heurte souvent au problème de la ***complétude***. Pour les agents, cela se traduit fréquemment en imposant une façon de se comporter et d'interagir qui soit robuste face à un maximum de situations possibles. Si cette solution assure la maîtrise des situations normales, elle ne convient plus quand les cas d'exception deviennent trop fréquents. Lorsque les agents sont plongés en environnement non déterministe, une telle rigidité n'est plus appropriée et il faut qu'ils soient capables d'***agir avec un comportement adaptatif***.

- *Comment déterminer ce qu'un intervenant souhaite obtenir ?*

  Pour pouvoir travailler ensemble, les agents doivent être en mesure de se comprendre mutuellement. En particulier, celui qui a un besoin doit réussir à l'exprimer, mais ceci n'est pas toujours réalisable. Une possibilité est que les autres agents soient capables de l'***aider à formuler ce besoin*** et pour cela, il faut que ces derniers puissent ***reconnaître ses buts***.

- *Comment personnaliser les actions effectuées à l'intention d'un intervenant pour par exemple l'assister ou lui fournir des conseils ?*

  Pour un agent, être capable de comprendre les intentions des autres est un premier pas nécessaire vers une interaction personnalisée, mais ensuite, qu'en est-il des interactions ? L'agent doit pouvoir ***adapter son comportement aux autres*** et ceci ne peut se faire que s'il dispose des connaissances nécessaires, en termes de caractéristiques, de préférences ou de comportement. La question est alors de trouver comment ***modéliser les autres agents*** et ceci reste encore un problème ouvert à l'heure actuelle, surtout lorsqu'il s'agit d'interactions complexes avec des humains.

Pour répondre à ces questions la thèse aborde plusieurs domaines spécifiques de l'intelligence artificielle. Au niveau des systèmes multi-agents, elle s'intéresse à résoudre le problème de l'hétérogénéité et plus particulièrement aux moyens d'abstraire la nature des agents. Comme nos travaux nous ont amenés à introduire des agents logiciels, la thèse montre comment nous avons cherché à définir le comportement d'agents médiateurs avec des méthodes de planification. La thèse touche ensuite le vaste problème de l'adaptation qu'elle tente d'aborder sur des sujets comme la

modélisation des autres agents et le suivi des interactions pour déceler et prévenir les risques d'échec. Elle présente des outils de modélisation tels que les Processus de Décision Markoviens, conçus pour la prise de décision en environnement incertain, ainsi que des méthodes d'apprentissage. Elle montre enfin comment nous utilisons ces derniers pour atteindre nos objectifs.

3. Organisation du document

Partie 1. Contexte et problématique

Dans le premier chapitre, nous expliquons les bases de l'approche agent et nous la replaçons dans le cadre des travaux sur l'intelligence artificielle distribuée. Nous décrivons les systèmes multi-agents (SMA), ainsi que les concepts d'interaction et d'organisation. Nous nous intéressons au problème d'interaction avec des agents qui ne sont pas forcément connus, c'est pourquoi nous focalisons notre étude sur les problèmes liés aux SMA hétérogènes (hSMA). Nous expliquons que nous considérerons les humains comme des agents dans un SMA en soulignant l'intérêt, mais aussi les difficultés que cela apporte. Nous présentons ensuite notre problématique de recherche : notre but est de faire coopérer des agents de nature très différente alors qu'ils n'ont pas a priori de structure, ni de langage commun.

Dans le chapitre 2, nous décrivons le domaine de la communication multimédia, puis l'approche de la société Dialoca avec laquelle nous avons collaboré. Nous détaillons alors la plate-forme Unimédia de Dialoca en commençant par ses composants (moteurs, pilotes, applications, etc.) et en poursuivant avec une description des services qu'elle permet de mettre en oeuvre. Nous expliquons que nous avons identifié un certain nombre de problèmes en interaction avec les experts Dialoca, notamment le besoin d'une approche exhaustive pour la conception des services et la difficulté de faire évoluer ou de réutiliser un service existant. Nous montrons que ce cadre applicatif correspond bien à notre problématique de coopération dans un hSMA, c'est pourquoi nous l'utilisons pour tenter de répondre à ce besoin de services évolutifs, à la fois sur le plan théorique et pratique.

Partie 2. La solution théorique

Dans le chapitre 3, nous revenons sur notre problématique de recherche qui est la coopération multi-agent hétérogène. Nous nous plaçons dans le cadre d'une architecture logicielle qui met en œuvre un ensemble d'agents en englobant aussi bien les hommes, les machines que les robots et nous rappelons notre objectif qui est de définir des schémas d'interaction. Nous présentons nos partitions de l'ensemble des agents considérés, afin de prendre en compte l'hétérogénéité. Nous distinguons d'une part les agents physiques des agents virtuels et d'autre part les agents contrôlés, partiellement contrôlés et non contrôlés. Ceci permet notamment d'identifier ceux avec lesquels il faut interagir sans forcément pouvoir agir directement sur eux. Nous revenons ensuite sur les notions d'interaction et d'organisation, en montrant que les services de communication peuvent être vus comme des schémas d'interaction avec des rôles bien définis. En nous focalisant sur l'interaction entre les rôles utilisateurs et les autres rôles, nous identifions les manques qu'il faut combler. Nous présentons ensuite notre solution théorique qui consiste à introduire un médiateur afin de faciliter l'interaction. Un médiateur intervenant entre deux agents va devoir les comprendre, les assister, traduire les informations, etc. Nous présentons enfin un ensemble de classes de services que nous avons pu construire à partir de ce rôle de médiateur. Afin de concevoir un tel médiateur, nous identifions deux grandes questions : comment produire un comportement pour fournir un service et comment adapter le comportement du médiateur en fonction des agents non contrôlés avec lesquels il interagit ?

Dans le chapitre 4, nous présentons l'approche de planification classique. Nous montrons qu'avec ce premier outil, il est déjà possible de définir un service simple. Après avoir souligné les limites dues aux hypothèses de la planification classique (comme le besoin d'un environnement statique, la certitude sur l'effet des actions, etc.), nous présentons des méthodes avancées, comme la planification conditionnelle et la planification hiérarchique. Nous nous penchons ensuite sur les modèles stochastiques qui ont retenu notre attention, car ils permettent de travailler sur des environnements incertains et évolutifs. Nous décrivons la façon dont nous pouvons utiliser la modélisation stochastique avec des MDP et nous présentons comment il est possible d'apprendre un comportement par renforcement.

Dans le chapitre 5, nous abordons le problème de l'adaptation au sein des services sous deux principaux aspects. En début de chapitre, nous expliquons comment l'étude de l'adaptation d'un service selon des comportements d'agents donnés nous a amenés à nous intéresser au domaine de la modélisation des utilisateurs. Le chapitre s'intéresse ensuite à la prise en compte d'imprévus et d'anomalies dans les services et plus généralement dans un système multi-agent hétérogène.

Dans le chapitre 6, nous nous focalisons sur le problème de la médiation dans une classe de service particulière : la recherche d'informations. Nous décrivons tout d'abord ce que sont les systèmes de dialogue et en quoi les travaux d'apprentissage de stratégies de dialogues à base de modèles stochastiques nous ont inspirés pour les comportements de médiation. Nous décrivons ensuite une spécification permettant de concevoir un médiateur à base de MDP et nous illustrons la présentation de nos travaux à l'aide d'une application de réservation de vol d'avion.

Partie 3. La solution applicative

Dans le chapitre 7, nous replaçons notre étude de la coopération entre agents hétérogènes dans le cadre applicatif d'Unimédia. En étudiant les services Dialoca, nous nous sommes aperçus qu'ils pouvaient être représentés par des automates à états avec une mémoire. Au vu de la complexité et des besoins d'exhaustivité, nous discutons de la difficulté de conception des services. Nous revenons également sur la robustesse face aux évolutions de l'environnement. Nous expliquons ensuite dans quelle mesure notre modélisation des services est adéquate, puis, dans la suite du chapitre, nous présentons l'architecture conceptuelle que nous avons définie en décrivant chacun des niveaux qui la composent.

Dans le chapitre 8, nous expliquons comment notre approche a été mise en œuvre au travers des différentes maquettes et outils logiciels que nous avons réalisés. Nous détaillons notamment comment nous avons conçu l'implantation d'un "corps" pour les agents contrôlés. Nous présentons également notre implantation d'agent médiateur, sur lequel nous avons expérimenté le problème de réservation. Après avoir exposé les résultats obtenus, nous formulons un certain nombre de critiques pour lesquelles nous proposons différentes améliorations.

Partie I. Contexte et Problématique

Chapitre 1 - Agents et Systèmes Multi-Agents

Dans ce chapitre, nous présentons le cadre théorique dans lequel s'inscrivent nos travaux. Nous expliquons tout d'abord comment une vision distribuée de l'intelligence artificielle est apparue pour faire face à la complexité des tâches à réaliser. Nous présentons ensuite les concepts d'agent et de Système Multi-Agent auxquels nous ferons largement appel tout au long de cette thèse. Notre intérêt pour cette branche de l'intelligence artificielle distribuée est motivé par notre besoin de faire collaborer des humains et des logiciels. De par leur capacité à accomplir des tâches de façon autonome, nous pensons, en effet, qu'une modélisation à base d'agents, permet de bien appréhender la délégation des tâches et de formaliser les échanges au travers des notions d'organisation et de coopération.

1. L’approche agent

1.1. L'Intelligence artificielle distribuée

L'un des objectifs de l'Intelligence Artificielle (IA) est la définition de systèmes capables de représenter des connaissances, de raisonner, de planifier des actions afin de résoudre des problèmes pouvant être très complexes. Elle cherche à obtenir des résultats comparables à ce que feraient des êtres humains dans des cas similaires, mais sans forcément utiliser les mêmes moyens.

Pour résoudre certaines tâches simples, il est tout à fait possible de considérer un système dit intelligent qui agit seul dans son environnement. C'est particulièrement vrai lorsqu'on cherche, par exemple, à résoudre un jeu de taquin comme le problème illustré Figure 1. Il s'agit d'une image découpée en petits carrés que le joueur doit reconstituer. Il y a une case vide et les seuls mouvements autorisés consistent à faire glisser les pièces du puzzle horizontalement ou verticalement. Dans ce problème, on suppose que le joueur est isolé et que les pièces ne bougent que lorsque celui-ci les déplace.

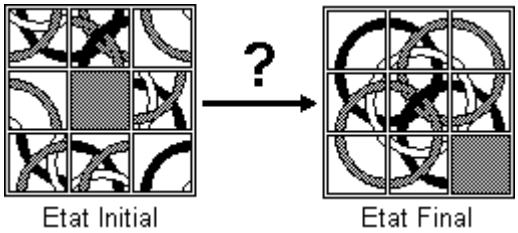


Figure 1 - Un problème dans le jeu du Taquin

Cependant, en dehors de ces problèmes simplifiés qui pourraient être qualifiés de cas d'école et que l'on sait bien souvent résoudre, on se retrouve souvent dans les cas réels avec un système qui n'est plus seul à agir sur l'environnement. Les effets de ses actions peuvent alors se trouver combinées à celles

d'autres entités avec des phénomènes d'influence *(Pearl 2000)*. Il est parfois possible d'étendre le principe du système intelligent unique afin de pouvoir traiter ces problèmes. Cependant, un tel système devient finalement très complexe, car il doit gérer les connaissances provenant de divers champs d'expertise et/ou se représenter son environnement de façon globale. Au lieu de compliquer cet agent unique, il est aussi possible de construire un système à base d'entités plus simples où chacune s'occupe d'une partie du problème, tout en ayant au final une complexité globale moindre. Certains problèmes peuvent par exemple être décomposés en terme d'espace où chaque entité est chargée de travailler sur un sous-espace éventuellement localisé, en terme de points de vue, de fonctionnalités ou encore en terme de domaines d'expertise où chaque entité possède sa propre spécialité.

C'est donc avec ces motivations que sont apparues au début des années 1970, les approches visant à distribuer l'intelligence artificielle avec le premier système à tableau noir HEARSAY *(Reddy et al. 1976)*. Avec l'arrivée de la programmation objet, le nombre croissant d'architectures matérielles, de systèmes et de langages, permettant d'effectuer des calculs en parallèle et avec l'interconnexion des machines au travers des réseaux informatiques, sont apparus trois axes de distribution de l'intelligence artificielle *(Labidi et Lejouad 1993)* :

- Le premier axe, l'Intelligence Artificielle Parallèle (IAP) vise la parallélisation des approches existantes en IA, avec des travaux comme *(Kanal et al. 1994)*. Il traite des techniques de mise en oeuvre sur les architectures parallèles et les systèmes massivement connectés.

Ce premier axe n'apporte pas vraiment de nouveau modèle de raisonnement, à l'inverse des deux suivants qui sont considérés par *(Bond et Gaesser 1998)* dans leur définition comme relevant de l'Intelligence Artificielle Distribuée (IAD).

- Le second axe, la Résolution Distribuée de Problèmes (RDP) *(Durfee et al. 1989)* vise la décomposition de problèmes entre un certain nombre de modules spécialisés qui coopèrent en partageant leurs connaissances et qui concourent chacun à la résolution.
- Le troisième axe concerne les ***Systèmes Multi-Agents*** (SMA) *(Ferber 1995)*, *(Demazeau et Briot 2001)* et *(Wooldridge 2002)*. Il considère que de petites entités interagissant fortement peuvent produire un tout intelligent amenant finalement à une vue sociale de l'IA. C'est alors qu'est apparu le terme agent, afin de qualifier les entités considérées dans cette approche. Depuis, le terme agent a connu de multiples définitions et nombreux sont les travaux utilisant une approche dite agent pour résoudre des problèmes divers. L'apparition, au niveau global, de phénomènes issus des interactions entre les agents se retrouve quant à elle au travers de la notion d'émergence et constitue l'un des grands thèmes des recherches sur les SMA. Notons que, comme l'ont montré *(Drogoul et Dubreuil 1993)*, il est possible de résoudre le problème du taquin avec une approche multi-agent.

La Figure 2 illustre la différence entre l'approche IA classique et l'approche IAD des SMA.

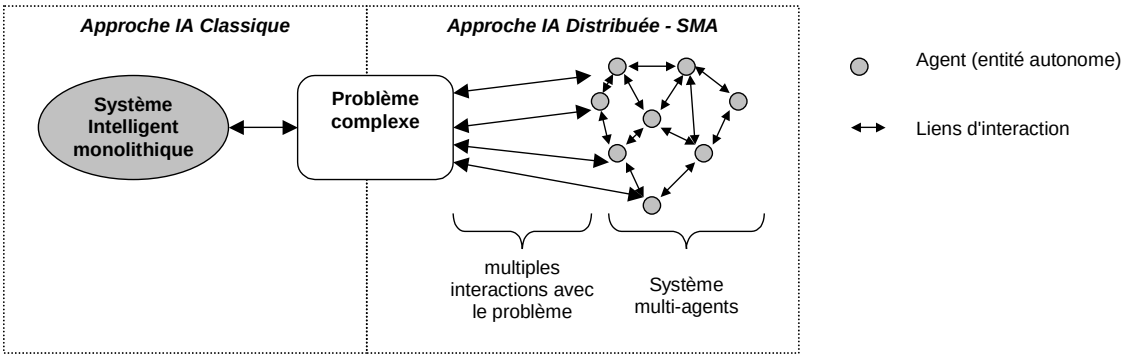


Figure 2 - Comparaison entre les approches IA et IAD / SMA

Nos travaux prennent place dans le domaine de l'IAD et peuvent être appréhendés comme partant de l'axe RDP vers l'axe SMA. Nous avons voulu passer d'une conception descendante qui prédétermine le déroulement des systèmes de façon globale à une conception évolutive, où il s'agit plutôt de

coordonner les entités considérées pour résoudre des problèmes de façon ascendante, avec l'idée que ces entités apprennent à se coordonner dans des limites bien définies.

1.2. Qu'est-ce qu'un agent ?

D'un point de vue étymologique, le mot **agent** vient du latin "agere" qui signifie agir. Littéralement, l'agent est donc celui qui agit.

Il existe, à l'heure actuelle, encore plusieurs définitions de ce qu'est un agent et aucune d'entre-elles n'est totalement admise. Certaines définitions très générales se permettent de qualifier d'agent des entités qui sont exclues par d'autres définitions plus spécifiques. Nous nous appuierons, à la base, sur une première définition simple, telle que celle donnée par *(Russell et Norvig 1995)* :

*Un **agent** est tout ce qui peut être vu comme **percevant** son environnement au travers de **capteurs** et **agissant** sur cet **environnement** au travers d'**effecteurs**.*

Cette première définition peut amener à voir l'agent, pour l'instant, comme une sorte de boîte noire fonctionnelle abstraite prenant en entrée diverses données (les perceptions) et donnant en sortie des signaux de commandes (les actions) dans un système plus vaste (le monde dans lequel l'agent est placé). A cette étape, nous appellerons cette fonction, un ***comportement***. La définition peut être étendue en introduisant les concepts de rationalité et de rationalité limitée *(Russell et Norvig 1995)* :

- *Un **agent rationnel** idéal cherche à **maximiser une mesure de performance** en fonction de ce qu'il perçoit et de la connaissance dont il dispose.*
- *La **rationalité limitée** est d'**agir de façon appropriée** quand on ne dispose pas d'assez de **temps** pour effectuer tous les calculs nécessaires.*

La seconde définition met l'accent sur une contrainte temporelle, mais nous pouvons fort bien l'étendre en indiquant que l'agent doit avoir la capacité à gérer des ressources (connaissances, mémoire, temps, énergie, etc.), mais aussi à tenir compte du contexte, c'est à dire des variabilités de l'environnement. Dans les définitions données par *(Zilberstein 1995)*, nous retrouvons plus généralement la question de l'optimisation du comportement de l'agent :

- *Un **agent rationnel** est un agent dont le comportement représente la solution à un **problème d'optimisation**.*
- *Comme les **ressources disponibles** pour l'agent **sont limitées** (en termes de puissance de calcul, de mémoire, etc.), le comportement résultant peut être imparfait, ce qui mène à une distinction entre **rationalité limitée** et rationalité parfaite.*

La mesure d'optimalité peut se traduire par la définition d'un espace de satisfaction ou d'utilité dans lequel l'agent considéré est positionné avec un système de récompenses/pénalités, en fonction des actions qu'il accomplit, des buts qu'il atteint, des obstacles qu'il percute, etc. L'agent rationnel est alors en quelque sorte motivé par l'atteinte d'un idéal, qui est la plus grande utilité qu'il puisse espérer.

Cependant, entre cette "boîte noire fonctionnelle motivée" et la notion d'agent, il manque encore l'un des aspects les plus importants qui va, entre autres, différencier un agent d'un objet : le concept d'autonomie.

- *Un système est **autonome** lorsque son **comportement** est déterminé par sa propre **expérience**. (Russell et Norvig 1995)*
- *La caractéristique essentielle des agents est qu'ils sont **autonomes** : capables d'**agir de façon indépendante**, en exhibant un **contrôle** sur leur état interne [...] Les agents incarnent une plus forte notion d'autonomie que les objets et en particulier, ils décident pour eux-mêmes s'ils doivent ou non accomplir une action à la demande d'un autre agent. (Wooldridge 2002)*

L'autonomie, telle qu'elle est définie ici, porte essentiellement sur la fonction de comportement et il s'agit, bien sûr, là d'un point de vue discutable, car il nous semble difficile qu'un agent puisse tout apprendre de zéro. Nous pensons plutôt qu'un agent est capable d'exhiber, d'une part, des comportements acquis et d'autre part des comportements innés. De la même façon, nous considérons

que les moyens de perception et d'action utilisés par l'agent peuvent être, eux aussi, innés ou acquis. Ceci nous place donc sur une ligne, avec à une extrémité des agents complètement asservis, ayant "perçu un paquetage complet" et à l'autre extrémité, des agents, presque laissés à eux même, qui font avec ce dont ils disposent sans compter sur aucune aide extérieure. Nous pourrions dire que les premiers, issus d'une conception descendante, sont astreints à accomplir les tâches qui leur ont été affectées. A l'inverse, les seconds relèveraient davantage d'une vision ascendante, car ils sont capables de se donner des buts et donc des motivations propres, d'acquérir de nouvelles capacités et finalement de se compléter les uns les autres pour atteindre ces buts.

Il existe bien d'autres définitions, comme *(Ferber 1995)*, dans lesquelles sont introduites différentes notions dont nous ne discuterons pas ici, telles que la mobilité ou la reproduction. Nous reviendrons sur les aspects de compétences par la suite dans ce chapitre et nous discuterons très largement au cours de cette thèse sur les capacités des agents à communiquer. Nous retiendrons cependant un aspect important : le concept d'agent peut désigner des entités qui existent de façon concrète dans un univers physique ou de façon abstraite dans un univers informatique.

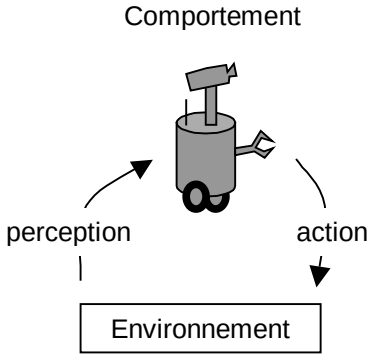


Figure 3 - Boucle sensori-motrice entre un agent et son environnement

Dans notre approche, nous allons considérer comme agent une entité réelle ou virtuelle qui perçoit et agit dans un environnement de façon autonome.

Avec cette définition, illustrée Figure 3, nous nous plaçons volontairement à un niveau suffisamment général pour pouvoir l'appliquer, entre autres, aux exemples suivants :

- à l'utilisateur humain d'un système informatique. En effet les êtres humains correspondent à cette définition dans le sens où ils disposent de moyens d'acquérir et d'utiliser des connaissances, où ils perçoivent ce qui les entoure par les organes sensoriels, et où ils agissent sur cet environnement par action musculaire.
- à un robot qui explore la surface de Mars. Celui-ci peut aussi être considéré comme un agent, car il doit gérer l'énergie dont il dispose, il peut percevoir son environnement au moyen de caméras et d'autres capteurs ou appareils de mesure et il agit essentiellement au travers des moteurs qu'il contrôle.
- à certains programmes informatiques que l'on appellera alors des agents logiciels. On retrouve d'ailleurs souvent l'appellation d'agents intelligents *(Wooldridge et Jennings 1995)* et *(Gini 1997)* qui devraient être qualifiés d'artificiels, car ces logiciels sont conçus à partir d'approches issues de l'intelligence artificielle. *(Maes 1994)* donne un bon nombre d'exemples d'agents, capables de trier les messages électroniques, de sélectionner des articles sur les forums de discussions Internet ou encore de planifier des réunions. Les capteurs et les actionneurs des agents logiciels représentent alors les parties de programme qui gèrent respectivement les entrées et les sorties. Ainsi, un agent qui filtre des e-mails analyse les messages reçus (il perçoit), puis les classe ou les supprime (il agit) selon leur forme ou leur contenu (il décide).

Notre définition n'impose pas de contraintes supplémentaires en ce qui concerne la nature du comportement, car selon le type de l'agent considéré, il peut être de type stimuli-réponse, résulter d'un raisonnement, etc.

1.3. Typologies des agents

Il existe diverses classifications qui s'appuient sur les propriétés trouvées en étudiant les diverses définitions pour les agents. Nous avons choisi de présenter ici deux typologies qui ont des points de vue complémentaires. Les schémas Figure 4 et Figure 5 présentent respectivement à gauche une typologie établie par *(Franklin et Graesser 1996)* selon la nature des agents et à droite, la partie de la typologie de *(Nwana 1996)* qui porte sur les propriétés des agents. Les zones grisées montrent comment se situent nos travaux vis à vis de chacune des classifications.

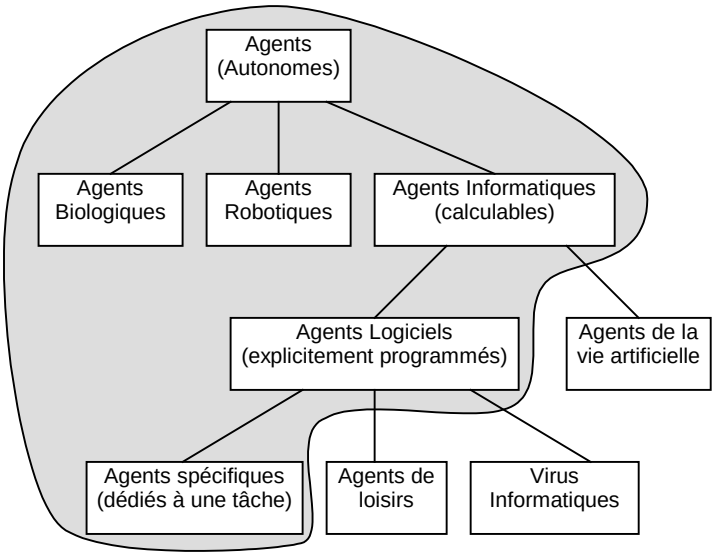


Figure 4 - Typologie selon la nature des agents *(Franklin et Graesser 1996)*

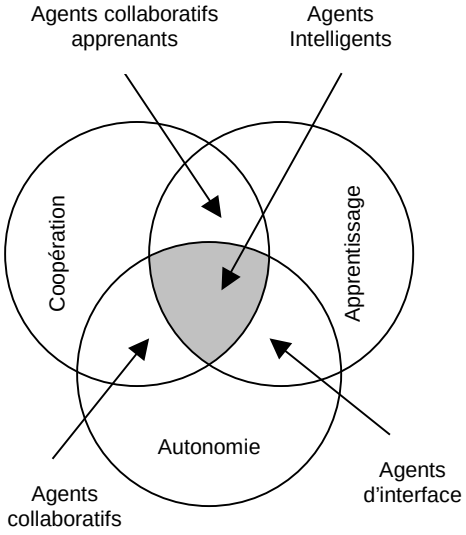


Figure 5 - Typologie selon les propriétés des agents *(Nwana 1996)*

On peut également s'intéresser à classer les agents en fonction de la complexité du mécanisme de raisonnement utilisé. On dira que plus un agent est complexe, plus son comportement est difficile à calculer et plus il est coûteux en ressources (temps, volume, énergie...).

Parmi les plus simples, nous distinguons tout d'abord les ***agents réactifs***. Les exemples biologiques que nous assimilons à cette catégorie, même si ce choix est parfois discutable, sont la fourmi, le termite, voire plus petit comme la cellule. En informatique, il peut s'agir de micro-programmes comme les éléments d'un automate cellulaire ou les virus informatiques. Les agents réactifs ont des capacités de raisonnement très limitées et se comportent suivant des règles stimulus/réponses simples avec une priorité donnée. Les agents réactifs perçoivent le plus souvent l'environnement de façon partielle : ils sont seulement sensibles à ce qui se passe dans leur proche voisinage. Ces agents si petits et sans grande capacité d'action peuvent paraître très limités. Cependant, il suffit de considérer une colonie de fourmis pour découvrir comment ces limites individuelles peuvent être dépassées avec l'apparition de comportements collectifs tels que la collecte de nourriture ou la construction du nid. Les études entomologiques ont d'ailleurs inspiré divers travaux en informatique comme l'optimisation par algorithmes de type fourmis *(Dorigo et al. 1999).*

En ajoutant, d'une part, de la mémoire, des connaissances et en complexifiant la fonction de comportement, d'autre part, on obtient des agents plus évolués. Ils deviennent alors progressivement capables de raisonner. Ils peuvent utiliser un modèle de leur environnement et des mécanismes calqués sur la pensée avec des approches symboliques ou d'effectuer des calculs complexes en suivant une approche numérique. En effet, ils deviennent capables, non seulement, de se représenter leur environnement, mais aussi de prévoir ses évolutions. Lorsqu'ils sont en mesure de prendre en compte les effets de leurs actions à plus long terme, ils peuvent alors planifier et utiliser des stratégies de plus en plus élaborées pour arriver à leurs fins. On peut également les doter d'une capacité à apprendre un

modèle, de sorte à ce qu'ils soient capables d'améliorer leurs performances dans le temps. Ils deviennent, à eux seuls, capables de résoudre des problèmes de plus en plus complexes.

Parmi les agents les plus évolués, on retrouve bien sûr, dans le domaine biologique l'exemple de l'homme. En informatique, on a affaire à des systèmes, eux aussi très complexes : ***les agents "intelligents"***, capables par exemple de combiner des informations issues de diverses sources de connaissances ou de chercher à comprendre les autres agents avec lesquels ils interagissent. L'illustration typique se retrouve dans les agents à architecture BDI *(Rao et Georgeff 1995)* implantés dans des systèmes comme PRS, COSY ou IRMA. Ces agents sont en mesure de se fixer leurs propres buts (Intention) parmi les états qu'ils souhaitent atteindre (Desire) et selon les faits qu'ils croient être vrais dans l'état courant, c'est-à-dire leur représentation du monde qui les entoure (Beliefs). Cependant, au fur et à mesure que l'on ajoute des besoins fonctionnels, on se rend compte que ces agents deviennent très difficiles à réaliser du fait de leur complexité. Nous retrouverons cependant des notions similaires lorsque nous parlerons d'état et de but en abordant la production de comportements dans le chapitre 4. De même, la notion de rôle que nous introduirons peut, elle aussi, être rattachée à la notion d'engagement et donc d'intention. C'est donc à ce niveau de complexité que nous nous plaçons, avec des agents intelligents qui peuvent être de nature très différente.

1.4. Les propriétés des environnements

Les mondes dans lesquels les agents évoluent sont appelés des **environnements**. Les environnements peuvent être de nature très différente allant du plus simple au plus complexe. Il peut s'agir d'espaces concrets (comme le terrain sur lequel se déplace un véhicule) ou abstraits (comme les réseaux informatiques ou les espaces d'information). Lorsque la nature de l'environnement le permettra, nous utiliserons le terme d'***agent situé*** pour désigner des entités dont les perceptions, les influences ou plus généralement l'existence sont restreintes à une partie de l'environnement. Si de plus, la nature de l'environnement permet la définition d'une distance, le sous-espace décrivant la zone dans laquelle l'agent peut percevoir et agir est appelé ***localité***. Ces deux notions prennent, par exemple, tout leur sens dans le cas des humains. Notre corps peut être à un instant donné situé dans telle rue, dans tel bâtiment, ou plus précisément, dans telle salle, etc. De même, notre champ de vision est limité en portée et en angle d'ouverture, nous ne pouvons par ailleurs percevoir que les objets qui sont proches de nous, etc. Introduire les concepts de ***situation*** et de ***localité*** permet bien souvent de simplifier la complexité spatiale, cependant, comme nos travaux portent sur la communication, nous n'en ferons qu'un usage limité et plutôt métaphorique. Dans le Tableau 1, nous avons listé les propriétés données par *(Russell et Norvig 1995)* afin de caractériser les environnements.

Tableau 1 - Propriétés des environnements *(Russell et Norvig 1995)*

Accessible	Si les organes sensitifs d'un agent lui donnent accès à l'état complet de l'environnement, alors l'environnement est dit accessible à cet agent. Dans ce cas, les capteurs détectent tous les aspects qui interviennent dans le choix d'une action. L'agent n'a pas à maintenir d'état interne pour garder une trace du monde.
Déterministe	Dans un environnement déterministe, l'état du monde à l'instant suivant est complètement déterminé par l'état courant et par les actions effectuées par les agents.
Episodique	Dans un environnement épisodique, la vie de l'agent est divisée en épisodes. Ce qui se passe dans l'épisode ne dépend pas de ce qui s'est passé dans les épisodes précédents.
Dynamique	Si l'environnement évolue pendant que l'agent décide quelle action accomplir, alors on dit que l'environnement est dynamique pour cet agent, sinon il est statique. Les environnements statiques évitent donc à l'agent de regarder comment est le monde pour sélectionner une action. L'agent n'a donc pas à se soucier du passage du temps.
Discret	Un environnement est dit discret s'il y a un nombre limité de perceptions et d'actions distinctes possibles. Dans un jeu d'échecs, l'environnement est discret car seul un nombre fixé de mouvements est possible à chaque tour. A l'inverse, la conduite d'un véhicule est continue car divers éléments comme la vitesse appartiennent à un intervalle de valeurs continues.

2. Les Systèmes Multi-Agents

Au début de ce chapitre, nous avons donné les motivations qui ont poussé les chercheurs en intelligence artificielle vers une vision distribuée des systèmes. Ainsi, plutôt que de considérer un agent unique, compliqué, difficile à maintenir et apparaissant finalement comme une ressource critique, ils ont appliqué le principe "diviser pour régner".

Les systèmes multi-agents s'appuient sur le principe suivant : au lieu d'avoir un seul agent en charge de l'intégralité d'un problème, on considère plusieurs agents qui n'ont chacun en charge qu'une partie de ce problème. La solution au problème initial est alors obtenue au travers de l'ensemble des comportements individuels et des interactions, c'est à dire par une résolution collective, comparable à celles observées dans les systèmes biologiques *(Deneubourg 1991)*.

2.1. Définition

Dans *(Ferber 1995)*, un système multi-agent est défini de la façon suivante :

On appelle système multi-agent (ou SMA), un système composé des éléments suivants :

1. *Un environnement **E**, c'est-à-dire un espace disposant généralement d'une métrique.*
2. *Un ensemble d'objets **O**. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans **E**. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*
3. *Un ensemble **A** d'agents, qui sont des objets particuliers (**A** $\subset$ **O**), lesquels représentent les entités actives du système.*
4. *Un ensemble de relations **R** qui unissent des objets (et donc des agents) entre eux.*
5. *Un ensemble d'opérations **Op** permettant aux agents de **A** de percevoir, produire, consommer, transformer et manipuler des objets de **O**.*
6. *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.*

La Figure 6 illustre cette définition dans un système composé d'agents robotiques.

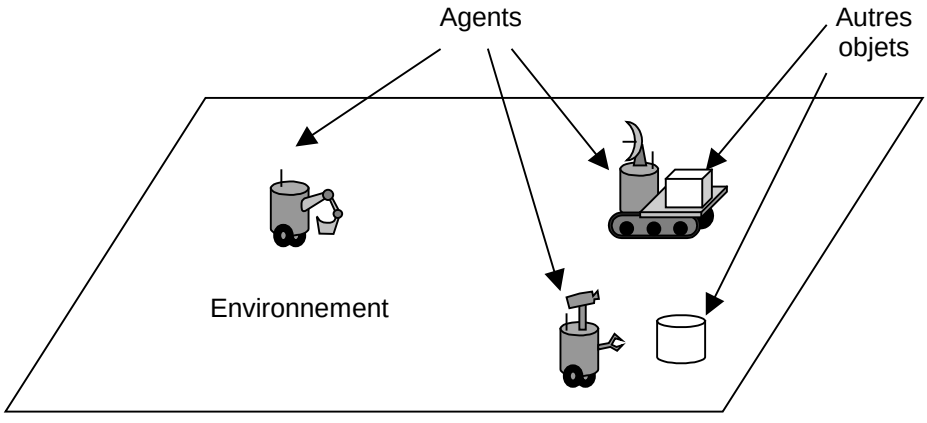


Figure 6 - Un exemple de Système Multi-Agent

En général, pour les problèmes qui s'y prêtent, la résolution avec une approche multi-agent donne au moins la même qualité et les mêmes performances qu'une résolution classique. Du fait de la distribution, on peut aussi compter sur une meilleure robustesse face aux dégradations. La perte ou la défaillance d'un individu ne remet pas forcément en cause les performances globales du système. En effet, selon le problème auquel on est confronté, il est possible de jouer sur la granularité du SMA, c'est-à-dire sur le compromis entre la complexité d'un agent et le nombre d'agents utilisés. On pourra

avoir des SMA composés de quelques individus, par exemple, spécialisés dans un domaine particulier ou répartis géographiquement. En cas de défaillance, le fonctionnement d'un tel système peut être certes perturbé, mais il n'est vraiment remis en cause que si des agents "incontournables" sont atteints. On pourra à l'inverse avoir des SMA composés d'un très grand nombre de petits agents réactifs, comme les fourmis qui vivent en colonies de plusieurs millions d'individus. Avec autant d'agents "peu coûteux", le dysfonctionnement d'un nombre limité d'agents est facilement corrigé, car d'autres individus peuvent combler le manque.

Un bon nombre de domaines comme la biologie ou la sociologie ont contribué au succès des SMA, d'une part en les utilisant pour simuler des phénomènes naturels et d'autre part en tant que sources d'inspiration pour l'amélioration des modèles et le cheminement vers une programmation orientée agent. C'est par ailleurs cette idée de programmation agent, proposée par Yohav Shoham, qui est également visée par des travaux comme *(Demazeau 1995)* lorsqu'ils présentent leur approche MAGMA, où ils posent les trois hypothèses suivantes :

1) Equation Déclarative *SMA = Agents + Environnement + Interactions + Organisation*

2) Equation Fonctionnelle *Fonction (SMA) = Σ fonction (Agents) + Fonction Collective*

3) Principe de Récursivité *Un SMA est considéré comme un agent à un niveau plus élevé*

La première équation, aussi appelée *Voyelle*, indique non seulement que l'on a besoin d'avoir des agents et un environnement, mais également que les agents doivent interagir et être organisés. La seconde équation indique que les capacités du système peuvent dépasser la somme des capacités individuelles par l'effet de la collectivité. Enfin, la troisième nous invite à voir un agent, lui-même comme un système multi-agent *(Marcenac 1997)* et d'aller encore plus loin, tel que nous l'enseigne la biologie : les cellules forment des tissus qui composent des organes, les organes quant à eux s'assemblent en un organisme, aussi, l'homme, avec ses milliards de cellules ne serait-il pas finalement à la fois un macro-agent et un système multi-agent ?

En examinant ces définitions, nous nous apercevons qu'à la base, le concept de SMA est applicable indépendamment de la nature des agents, pour peu qu'un plus petit dénominateur commun soit défini. Elles nous amènent également à distinguer d'un côté les cas pour lesquels nous appréhendons un phénomène comme un SMA pour mieux le comprendre (pour la simulation multi-agent) et d'un autre côté, les cas pour lesquels nous concevons un SMA pour résoudre un problème (au sens de la programmation agent). Dans notre thèse, nous supposons qu'au moins une partie des agents existe (les utilisateurs, les experts, etc.) et qu'il faut en quelque sorte compléter le système pour que les agents puissent se comprendre et travailler ensemble.

En début de chapitre, lorsque nous avons défini ce qu'était un agent isolé nous avons décrit ses propriétés et ses capacités individuelles. Nous allons maintenant décrire ce qui se passe lorsque l'on est en présence de plusieurs agents en présentant les phénomènes d'interaction et la notion d'organisation qui rendent possible la résolution collective de tâches dans les systèmes multi-agents.

2.2. L'interaction dans les SMA

2.2.1. Qu'est-ce que l'interaction ?

Le fait d'avoir plusieurs agents actifs au même moment dans le système implique de nouveaux phénomènes :

1) Les effets des actions de chaque agent et les lois d'évolution de l'environnement se combinent, et peuvent parfois s'amplifier, s'annuler, se perturber...
2) Les agents peuvent percevoir les effets provoqués par les actions des autres.

Ainsi, un agent peut avoir connaissance du fait qu'il n'est pas seul dans son environnement. Ces deux remarques nous amènent à la notion d'interaction qui peut être définie de la façon suivante :

*Une **interaction** est une mise en relation dynamique de deux ou plusieurs agents par le biais d'actions réciproques. (Ferber 1995)*

Dans *(Chevrier 2002)*, nous retrouvons deux définitions de l'interaction qui nous paraissent les plus appropriées pour le domaine de la communication sur lequel nous nous sommes penchés :

L'interaction faible *considère l'acte isolément. Ce mode correspond à une action ou influence unidirectionnelle et se place dans une perspective monologique de l'interaction. Il s'apparente plus à des capacités de communication que de réelles capacités à interagir.*

L'interaction forte *considère l'acte comme emprunt de réciprocité et correspond à une co-action ou une influence mutuelle. Elle se situe dans la perspective dialogique où l'établissement d'une réelle communication entre deux agents implique négociation, interprétation et conflit.*

Ainsi on peut dire qu'un individu qui écrit une lettre et qui l'envoie effectue un acte d'interaction faible. En revanche, l'interaction forte couvrira non seulement le fait qu'un expéditeur envoie une lettre, mais également le fait que le destinataire la reçoive, la lise et en comprenne le contenu. L'interaction dénote alors la présence d'une certaine influence de l'expéditeur sur le destinataire : la lettre n'est pas envoyée pour rien, mais elle l'est intentionnellement dans le but d'informer le destinataire et éventuellement de le faire réagir.

Dans son cours sur les systèmes multi-agents, *(Boissier 1999)* explique que l'interaction peut se faire selon trois degrés de complexité :

- sans communication explicite, par l'effet de contraintes ou de dépendances,
- au travers d'une communication primitive par des signaux dont l'interprétation est fixée (ce type de communication s'effectue alors par échange de messages ou de plans) ou
- par une communication sophistiquée.

Notre idée est que les types d'interactions peuvent se répartir sur une échelle selon les capacités sociales des agents. A une première extrémité de l'échelle se situent des agents interagissant presque de façon involontaire, car ils ne se soucient pas des autres. A l'autre extrémité, nous trouvons des agents bien plus sociaux qui connaissent tous les autres et qui communiquent avec eux de façon explicite, élaborée et totalement maîtrisée. Dans les communications évoluées, nous retrouvons les conversations dans des langages d'interaction, ACL *(FIPA 2002)* ou KQML *(Chalupsky et al. 1992)*, qui sont basés sur les actes de langage *(Searle 1969)*. Ces approches mettent en œuvre des aspects linguistiques reflétant davantage la sémantique des interactions, mais concernent les communications entre agents logiciels. De telles interactions peuvent aussi utiliser des ***protocoles d'interaction***, comme l'approche COOL *(Barbuceanu et Fox 1995)*.

Dans *(Boissier 1999)*, nous retrouvons la définition d'un ***protocole d'interaction*** comme étant *un schéma commun de conversation utilisé pour exécuter une tâche. C'est une stratégie de haut niveau gouvernant les interactions entre agents permettant de faciliter leur dialogue.*

Pour nos travaux, nous considérons que les agents connaissent, en partie, leurs partenaires et qu'ils sont capables de communiquer au travers de médias bien identifiés, comme nous l'expliquerons dans les chapitres suivants. Etant davantage focalisés par les aspects de prise de décision dans l'interaction, nous ne nous sommes pas appuyés sur les actes de langage, mais une approche plus linguistique est tout à fait possible. En revanche, nous nous sommes davantage orientés du point de vue comportemental. Ceci nous a amenés à l'apprentissage des stratégies de médiation avec des modèles probabilistes, comme nous l'expliquerons dans le chapitre 3.

L'interaction est le moteur des systèmes multi-agents, car même si elle n'est pas forcément complexe, elle est toujours présente pour leur donner une dynamique. Ce caractère incontournable explique qu'un nombre croissant de travaux ont porté sur la modélisation de l'interaction, afin de permettre la conception, la validation et le contrôle des systèmes multi-agents.

2.2.2. Modélisation des interactions

Les interactions entre agents sont souvent modélisées avec des approches logiques, mais aussi avec des outils comme des graphes de transition d'automates à états finis *(Winograd et Florès 1986)*. La Figure 7 représente un exemple d'interaction modélisé par un automate. Dans cet exemple, un état de

l'automate représente l'état d'un système à deux agents A et B et une transition représente une interaction. Les réseaux de Pétri sont également utilisés pour la modélisation des interactions, comme on le retrouve dans les travaux de *(El Fallah-Seghrouchini et al. 1999)*. Les places du réseau représentent les états internes de l'agent et les messages en cours d'acheminement, alors que les transitions représentent des synchronisations sur les messages et des conditions d'application d'actions. Le même exemple d'interaction est modélisé à l'aide d'un réseau de Pétri sur la Figure 8.

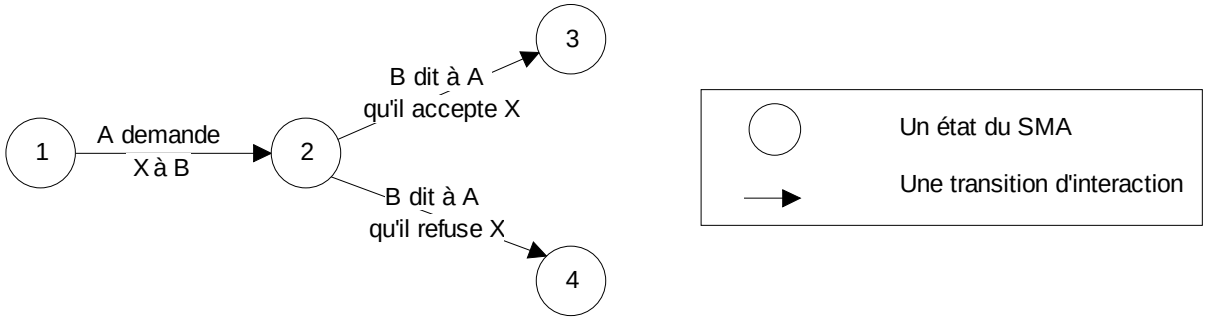


Figure 7 - Un exemple d'automate à états modélisant une interaction

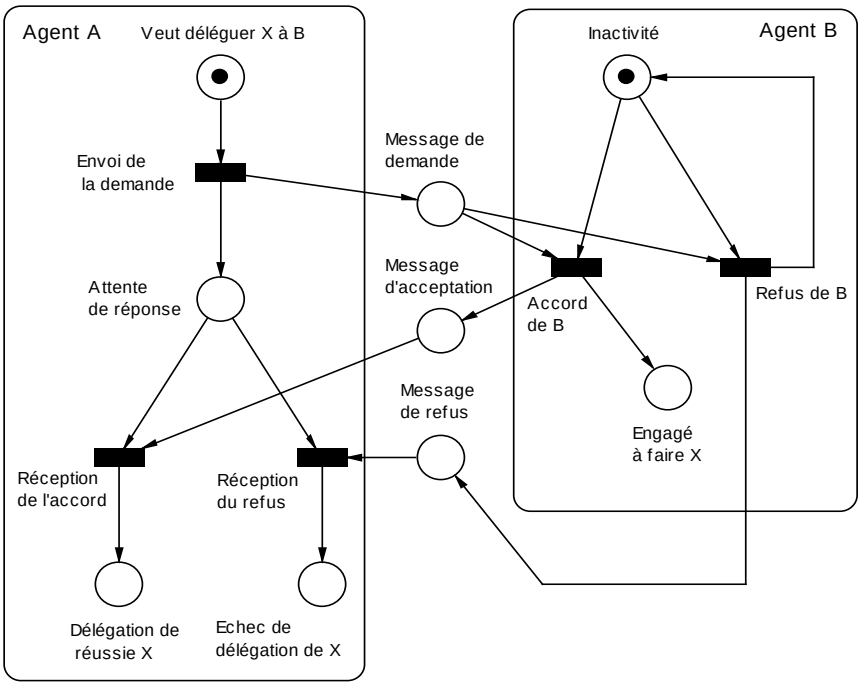


Figure 8 - Un exemple de réseau de Pétri modélisant la même interaction que la Figure 7

Modéliser les interactions permet d'une part à un observateur de comprendre la façon dont elles se déroulent et d'autre part de concevoir le comportement d'un agent qui respecte, par exemple, un protocole de communication. Dans un système chaotique, les interactions se produisent de façon désordonnée et sans objectif particulier. En revanche, si le système est doté d'un cadre d'interaction suffisamment rigide pour orienter les interactions vers un but donné, mais aussi suffisamment souple pour éviter que le moindre "grain de sable" remette tout en cause, le système a alors plus de chances d'atteindre ce but. C'est avec l'organisation des agents et la coordination des interactions qu'il devient possible de donner ce cadre aux systèmes multi-agents.

2.2.3. Les situations d'interaction et la coordination

La façon dont se réalisent les interactions permet d'obtenir diverses situations au niveau du système multi-agent. Le Tableau 2 montre la classification des situations d'interaction donnée dans *(Ferber 1995)* en fonction de la compatibilité des buts des intervenants, de la disponibilité des ressources et de la capacité des agents à atteindre leur but en termes de compétences individuelles.

Tableau 2 - Classification des situations d'interaction *(Ferber 1995)*

Buts	***Ressources***	***Compétences***	***Types de situation***	***Catégorie***
Compatibles	Suffisantes	Suffisantes	Indépendance	Indifférence
Compatibles	Suffisantes	Insuffisantes	Collaboration simple	Coopération
Compatibles	Insuffisantes	Suffisantes	Encombrement	
Compatibles	Insuffisantes	Insuffisantes	Collaboration coordonnée	
Incompatibles	Suffisantes	Suffisantes	Compétition individuelle pure	Antagonisme
Incompatibles	Suffisantes	Insuffisantes	Compétition collective pure	
Incompatibles	Insuffisantes	Suffisantes	Conflits individuels pour des ressources	
Incompatibles	Insuffisantes	Insuffisantes	Conflits collectifs pour des ressources	

On peut remarquer que selon les combinaisons, des conflits peuvent apparaître, mais ce n'est pas forcément négatif pour la résolution du problème. Dans certains cas comme l'éco-résolution *(Dury et al. 1999)*, la solution passe en effet par la mise en compétition des agents. Dans nos travaux, nous nous sommes concentrés sur la coopération entre les agents humains et les agents logiciels. Cependant, la mise en compétition de fournisseurs d'informations, avec l'utilisateur comme "ressource critique" aurait, elle-aussi, pu être envisagée.

De façon concrète, l'ensemble de ces situations sont gérées au sein d'un système multi-agent par des mécanismes de ***coordination***. Nous en donnons ici deux définitions :

*La **coordination** est la gestion des dépendances entre les activités. (Malone et Crowston 1994)*

*La **coordination** est le processus de construction de programmes en assemblant les parties actives; un modèle de coordination est la colle qui lie des activités séparées en un ensemble. (Gelernter et Carriero 1992)*

Ainsi, la coordination détermine en quelque sorte quelles sont les règles de bon fonctionnement du système auquel les agents appartiennent. Selon la nature des agents et des interactions, plusieurs formes de coordination sont possibles. La coordination peut être imposée à un agent par une sorte de contrat, comme le respect d'un protocole d'interaction donné. Elle peut aussi être apprise, lorsque l'agent trouve un intérêt à participer à la collectivité. Notre objectif étant de faire collaborer des agents, de tels mécanismes sont indispensables. Toutefois, comme il n'est pas toujours possible d'imposer aux agents de se plier à un contrat donné, dans la solution que nous avons retenue, ce sont les agents que nous ajoutons au système qui prennent en charge la coordination.

La coordination est fortement liée aux relations qui peuvent s'établir entre les agents et plus généralement à la façon dont les agents sont organisés.

2.3. Le concept d'organisation

Lorsque nous avons décrit ce qu'était un agent, nous avons indiqué qu'il cherchait à maximiser une espérance de récompense qui peut être assimilée à une satisfaction. Ainsi, il n'agit et, a fortiori, n'interagit pas gratuitement : il est motivé, car il cherche à satisfaire des objectifs. Supposons donc que les interactions entre agents ne se déroulent pas au hasard. Dans ce cas, pourquoi interagissent-ils et surtout qu'est-ce qui conduit à l'obtention d'un tout utile, cohérent et performant ? Par ailleurs, qu'en est-il des buts : les agents se les sont-ils fixés ou leur ont-ils été imposés ?

On peut tenter de répondre à ces questions en envisageant une conception descendante, dans laquelle un problème de départ est analysé et le plus souvent réduit en sous-problèmes plus simples. En

appliquant des principes comme "*diviser pour régner*" on poursuit jusqu'à l'obtention d'une solution composée d'éléments simples que l'on sait résoudre. Appliquée aux systèmes multi-agents et donc à la résolution collective de problèmes, cette conception descendante conduit à spécifier précisément ce que chaque agent doit faire, ce que chaque agent doit savoir, etc. Dans ce cas, un ensemble de buts à plus ou moins long terme peuvent être imposés aux agents et ceux-ci doivent alors s'y tenir.

Il est aussi possible d'envisager une conception ascendante, où les agents sont dotés de diverses compétences et connaissances ou en font l'acquisition. Bien que cela paraisse moins conventionnel, selon la richesse des interactions, il est tout à fait possible d'observer une diminution du désordre, voire une convergence vers des structures, avec la réalisation de tâches collectives.

2.3.1. Pourquoi est-ce difficile de définir ce qu'est une organisation ?

Afin de résoudre ces problèmes, un nombre croissant de chercheurs en intelligence artificielle distribuée travaillent sur l'organisation et sur l'auto-organisation dans les systèmes multi-agents avec toujours de fortes inspirations provenant des travaux en sociologie et en biologie. Dans *(Nicolle 2000)*, nous retrouvons plusieurs motivations pour l'étude des organisations dans le cadre des systèmes multi-agents. Elle peut entrer en jeu pour former des groupes (robots footballeurs, robots fourrageurs), pour effectuer des simulations et pour une raison qui prime dans nos travaux : permettre l'interaction avec des humains. Dans un premier temps, nous pouvons partir de la définition suivante pour le terme ***organisation*** :

Manière dont les différents organes ou parties d'un ensemble complexe, d'une société, d'un être vivant sont structurés, agencés; la structure, l'agencement eux-même (Larousse 1998).

Nous définissons par ailleurs le concept d'***auto-organisation*** comme désignant la capacité d'un ensemble composé à se doter d'une organisation sans intervention extérieure au système autrement dit, l'apparition d'un phénomène d'organisation autonome.

Comme l'indiquent *(Carley et Gasser 1999)*, les organisations humaines conditionnent les actions des individus car elles englobent les normes, la culture, etc. L'un des objectifs est notamment d'organiser les agents socialement et de découvrir la clef de la résolution collective de problèmes. Ceci est motivé par le fait qu'une société d'agents organisée dispose d'une capacité d'action étendue et permet par la coordination des agents d'atteindre des buts irréalisables par un seul individu.

(Carley et Gasser 1999) soulignent que la réponse classique à la question "Qu'est-ce que l'organisation ?" est "on le sait quand on la voit" et ils indiquent qu'il n'existe pas de définition unique. Ils donnent en revanche les caractéristiques générales des organisations :

- *Elles s'attachent aux techniques de résolution de problème à grande échelle.*
- *Elles mettent en cause plusieurs agents, qu'ils soient humains ou artificiels.*
- *Elles sont généralement engagées dans une ou plusieurs tâches et donc dirigées par des buts (qui peuvent néanmoins changer ou ne pas concerner tous les membres de l'organisation).*
- *Elles sont capables d'affecter et d'être affectées par leur environnement.*
- *Elles disposent d'une culture comprenant des connaissances et possèdent des capacités distinctes des agents eux-mêmes.*
- *Elles ont un statut distinct de celui des agents.*

Les auteurs précisent également que les organisations visent le dépassement des limites du simple agent en termes de cognition, de physique (on veut aller au-delà de la localité espace-temps) et statut (comme l'importance du caractère institutionnel : "l'état", "l'administration",...). Cependant, une mauvaise organisation peut au contraire nuire à la performance et à la flexibilité, et augmenter les coûts, en terme de ressources, de temps de décision, de coûts de coordination, etc.

Pour établir plus clairement ce qu'est une organisation, nous proposons de regarder la définition de l'organisation utilisée par *(Ferber 1995)* pour les systèmes multi-agents et empruntée à E. Morin :

*Une **organisation** peut être définie comme un agencement de relations entre composants ou individus qui produit une unité, ou système, dotée de qualités inconnues au niveau des composants ou individus.*

L'organisation lie de façon interrelationnelle des éléments ou événements ou individus divers qui dès lors deviennent les composants d'un tout. Elle assure solidarité et solidité relative, donc assure au système une certaine possibilité de durée en dépit de perturbations aléatoires.

Les relations dont il est question portent aussi bien sur l'aspect spatial, les liens de communication que sur la subordination, etc. L'auteur précise, en outre, qu'une organisation n'est pas statique et à titre d'illustration, nous pouvons citer l'exemple des travaux de *(Foisel 1998)* qui proposent des approches pour la gestion des préférences d'interactions et qui ouvrent donc la voie à la réorganisation dans les systèmes multi-agents.

(Ferber 1995) introduit également le terme de ***structure organisationnelle*** pour désigner la partie invariante d'une organisation. Une organisation donnée (dite ***organisation concrète***) devient alors une instance d'une structure organisationnelle.

Nous nous sommes intéressés au point de vue fonctionnel de l'organisation. Une question qui se pose est de déterminer la part que prend un agent dans l'organisation, c'est à dire l'utilité qu'il a vis-à-vis des autres agents de l'organisation et cela mène à la notion de ***rôle***. Décrire le ***rôle*** d'un agent dans un système organisé, c'est expliquer de quelle façon il est fonctionnellement impliqué dans ce système : qu'est-ce qu'il lui apporte ?

Afin d'illustrer notre propos, prenons l'exemple d'une structure organisationnelle hiérarchique, qui est typique des sociétés humaines : il est possible de retrouver des rôles spécialisés dans des domaines étroits aux feuilles et des rôles de coordination au niveau des nœuds qui deviennent de plus en plus généraux, au fur et à mesure que l'on va vers la racine.

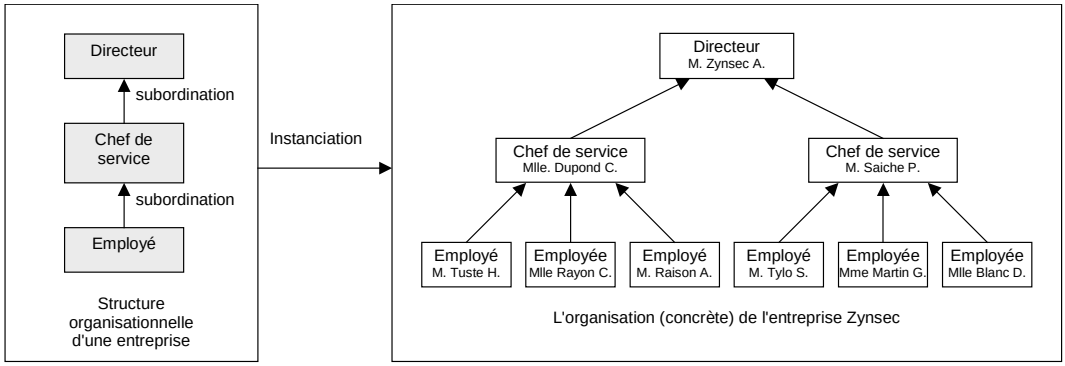


Figure 9 - Un exemple d'organisation hiérarchique

Sur la Figure 9, nous avons représenté de façon très simplifiée l'organisation hiérarchique d'une entreprise. A gauche, nous donnons la structure organisationnelle qui définit les rôles et les dépendances de subordination de façon abstraite. La partie droite représente l'organisation d'une entreprise fictive dans laquelle chaque personne est associée à un rôle donné. Les personnes ayant le rôle d'employé vont se focaliser sur des travaux précis et plutôt à court terme. Les chefs de services vont avoir une influence sur un domaine plus grand et devront superviser les travaux de leurs subordonnés à moyen terme, alors que le directeur aura pour tâche d'amener l'entreprise à des objectifs à long terme. Cette organisation permet à la société de survivre, car chacun y tient un rôle précis. Par ailleurs, les liens entre les rôles décrivent ici la relation de subordination et définissent quelles sont les interactions possibles : un directeur va essentiellement s'adresser aux différents chefs de service, mais beaucoup plus rarement à un employé donné. De même pour un chef de service : bien qu'il ait à dialoguer avec ses employés et avec le directeur, il ne s'adresse pas à eux de la même manière, etc. Beaucoup de paramètres de l'interaction varient, comme le niveau de détail, la fréquence ou la forme.

Les aspects de rôles, d'organisation et d'interaction se retrouvent dans de nombreux travaux et particulièrement dans le modèle Aalaadin, présenté dans *(Gutknecht et Ferber 1998)* et *(Gutknecht 2001)*, qui nous a servi comme source d'inspiration pour la définition des classes de services du chapitre 3.

2.3.2. Le modèle Agent Groupe Rôle (AGR)

Le modèle organisationnel Aalaadin est *basé sur l'association de trois concepts clés : l'agent, le groupe et le rôle, utilisés simultanément pour décrire des organisations concrètes d'agents.*

- *Agent :* *entité autonome communicante qui joue des rôles au sein de différents groupes.*
- *Groupe :* *terme générique pour qualifier une communauté d'agents en relation (par interaction, par partage d'un environnement, par un but ou une ontologie commune,...)*
- *Rôle :* *représentation abstraite d'une fonction du groupe pouvant contraindre le comportement de l'agent, et incarnée dans un ou des comportements spécifiques par l'entité.*

Notons que dans le modèle AGR, un même agent peut jouer différents rôles et il participe alors à des interactions spécifiques à chacun de ses rôles. De même, un même rôle peut être joué par plusieurs agents. *(Gutknecht 2001)* présente également, un modèle organisationnel générique, montré Figure 10, qui permet d'abstraire les structures d'organisation. Il ajoute pour cela les notions suivantes :

- *La structure organisationnelle définie comme un ensemble de structures de groupe participant à la résolution. Elle définit une structure sociale abstraite complète (par exemple, un laboratoire).*
- *La structure de groupe, qui se définit comme une description abstraite d'un groupe (par exemple : une équipe). Elle identifie la totalité des rôles, l'environnement, et les schémas d'interaction qui sont à isoler au sein d'un groupe.*
- *Les **schémas d'interaction** identifiés entre rôles du groupe impliqués. On y précise le rôle initiateur de l'interaction et les autres rôles pouvant y apparaître. Les conversations qui auront lieu entre agents jouant ces rôles se conformeront à ces interactions.*

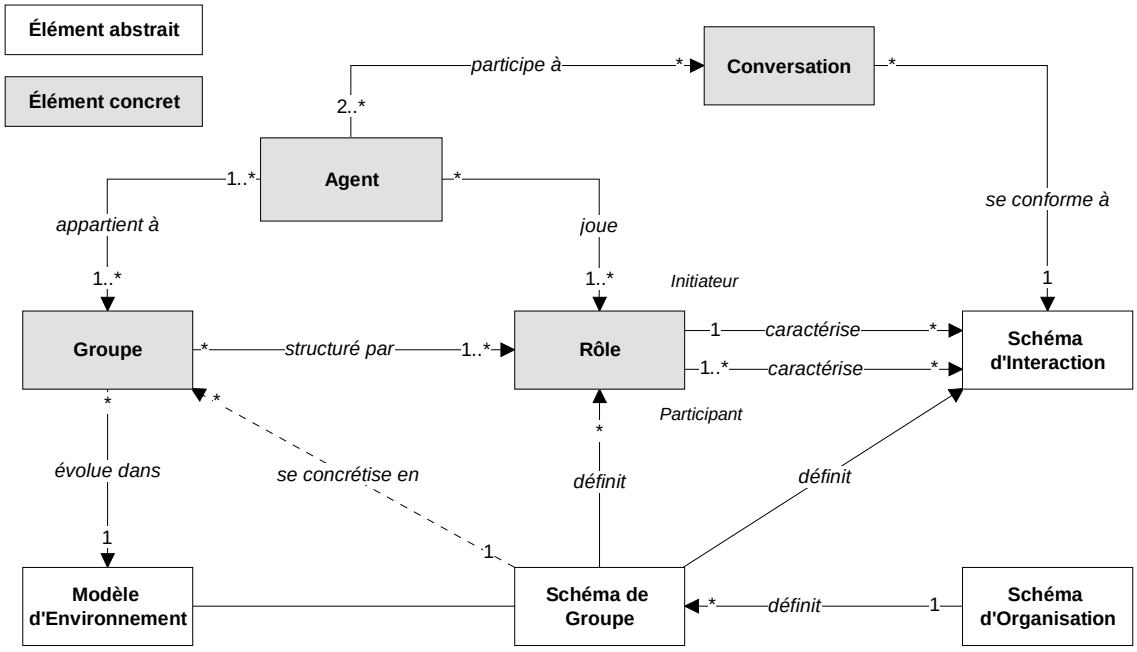


Figure 10 - Le Modèle Agent Groupe Rôle (AGR) Etendu *(Gutknecht 2001)*

2.3.3. Satisfaction individuelle et tâche collective

En dotant un système multi-agent de schémas organisationnels avec les schémas d'interactions correspondants, il est possible de coordonner les interactions et de faire en sorte que l'organisation obtenue réalise une tâche de façon collective. Le fait qu'un agent, autonome par nature, se conforme à un schéma d'interaction et joue un rôle particulier dans l'achèvement d'une tâche commune peut paraître au départ incompatible. Cependant, si l'achèvement de la tâche collective profite directement à l'agent, par le biais d'une récompense ou indirectement, en le rapprochant de ses buts individuels, cela

ne peut en quelque sorte que le conforter dans son rôle. La tâche collective et les récompenses correspondant à son accomplissement peuvent, bien sûr, être imposées par le concepteur du système. Une alternative est que les agents puissent rendre collectifs leurs buts individuels en les déléguant aux autres. Il est alors possible d'imaginer un schéma d'interaction qui anime le système multi-agent et dans lesquels les besoins des uns sont décomposés, transformés en requêtes et transmis d'agent en agent jusqu'à leur résolution. C'est d'ailleurs l'idée qui se retrouve dans la planification multi-agent dont nous parlerons dans le chapitre 4.

L'organisation permet de structurer les interactions et de faire un grand pas en direction de la résolution collective de problèmes distribués. Cependant, la force des systèmes multi-agents s'appuie sur le nombre des agents et sur la fiabilité des échanges. Des problèmes peuvent toujours apparaître si les communications sont soumises à des aléas, si les agents sont en nombre limité et à plus forte raison s'ils sont différents les uns des autres.

2.4. L'hétérogénéité des agents

En dehors des hypothèses faites dans les études théoriques sur les systèmes multi-agents, il est très rare que deux agents soient identiques en tous points. Nous définissons l'**hétérogénéité** dans un système multi-agent par la présence de différences entre les agents. La différence de nature est peut être la plus grande et si on regarde la typologie de Franklin et Graesser Figure 4, on peut se retrouver avec un système comprenant des agents biologiques et des agents logiciels... Ensuite, même si on prend deux agents logiciels, ils peuvent être conçus de façon très dissemblable. Le premier pourra être construit sur une architecture à base de règles, alors que le second pourra être un agent BDI. Enfin, même s'ils sont du même type, ils n'auront pas forcément les mêmes connaissances et les mêmes capacités. Tout ceci montre que si on veut les faire interagir ensemble dans le même SMA, il faut remédier à toutes sortes d'incompatibilités plus ou moins perceptibles.

Dans nos travaux, nous avons été amenés à nous intéresser spécifiquement à des problèmes faisant intervenir un ou plusieurs agents humains. Cela ajoute de nombreuses difficultés, comme celles soulignées dans les travaux de *(Grislin-LeSturgeon et Péninou 1998)* sur les interactions Homme-SMA. Tout d'abord, la tâche de modélisation est très difficile, car elle nécessite de prévoir au maximum les comportements qu'un humain peut adopter. Ensuite, il est bien souvent ardu de comprendre ce qu'un agent humain veut, de savoir ce qu'il connaît sur le domaine et de déterminer comment l'aider. Même s'il s'agit d'agents difficiles à prendre en compte, ce sont eux qui sont à l'origine du SMA et la plupart du temps une décision humaine doit être prioritaire sur celle d'un agent logiciel. Par ailleurs, on dispose actuellement d'un choix important de moyens pour faire communiquer des agents, dans la mesure où ils peuvent utiliser un protocole commun. Entre des agents logiciels on dispose en effet, d'un large panel de langages et de protocoles comme KQML *(Chalupsky et al. 1992)* ou le langage ACL de la FIPA *(FIPA 2002)*. Une complication de taille est que les protocoles de communication classiques entre agents sont inadaptés pour la communication avec les humains et ne sont pas forcément compatibles avec les langages qui peuvent êtres imposés. On voit en effet davantage de problèmes se soulever lorsque l'un des agents est un humain parce qu'il y a un fossé de langage. De plus cet agent humain peut fort bien ne pas être un spécialiste, voire être un novice pour le domaine considéré. Ainsi, il peut faire des erreurs et s'exprimer de façon incertaine ou même être incapable de décrire ce qu'il veut de façon simple... En considérant cela, on conçoit que prendre en compte des humains dans un SMA est encore un challenge très difficile.

3. Problématique générale

Nous allons maintenant nous baser sur les différents concepts que nous venons de présenter afin de décrire les problèmes auxquels nous avons cherché à apporter notre contribution. Au cours de nos travaux de recherche, nous nous sommes intéressés à des SMA impliquant des agents variés en nombre et en nature tels que des humains, des programmes informatiques mais aussi des robots.

Notre problématique générale relève donc d'une vision des SMA à l'échelle humaine et nous pouvons la formuler de la façon suivante :

Trouver un modèle et des méthodes permettant de coordonner un ensemble d'agents hétérogènes pour réaliser coopérativement une tâche dans un environnement incertain.

Une grande partie de la difficulté réside dans la conception des agents qui ont à communiquer avec les humains ou qui ont plus généralement à agir dans le monde réel, car ils doivent prendre en compte le caractère complexe et incertain des interactions et être capables de faire le lien avec les agents existant déjà dans le système. C'est précisément ce qui nous a poussés à nous focaliser par la suite sur les approches probabilistes.

4. Conclusion

Dans ce chapitre, nous avons décrit ce qu'était l'approche agent en la replaçant dans la lignée des travaux sur l'intelligence artificielle distribuée. Nous avons ensuite exposé les bases des systèmes multi-agents en montrant que leur force réside dans la richesse des interactions. En considérant l'aspect hétérogène, nous avons ensuite formulé un problème de recherche théorique concernant la coopération hétérogène que nous avons tenté de résoudre au long de cette thèse.

Le chapitre suivant va présenter en détail le cadre applicatif de notre thèse qui est la communication multimédia. Nous y décrivons les services proposés par la société Dialoca, en mettant en avant les besoins qui ont été identifiés et en montrant que notre problématique s'y retrouve pleinement.

Chapitre 2 - Les services de communication multimédia

Dans ce chapitre, nous abordons le domaine d'application dans lequel s'est déroulée notre thèse. Nous décrivons tout d'abord le thème de la communication multimédia, puis nous présentons la société Dialoca, en définissant la notion de service. Nous décrivons ensuite la plate-forme Unimédia et la palette de services qu'elle offre à ses utilisateurs. Dans la suite du chapitre, nous recensons les différents besoins qui ont été exprimés par Dialoca puis nous montrons que la problématique générale que nous avons proposée dans le chapitre 1 est tout à fait pertinente pour ce domaine d'applications.

1. La communication multimédia

Communiquer, c'est *être en relation, en rapport, en correspondance avec quelqu'un. Ce qui permet de joindre, deux choses, deux lieux. (Larousse 1998)*

Lorsque l'on parle de communication, l'aspect de liaison joue donc un aspect très important. Nous utiliserons le mot ***média*** afin de caractériser un support de communication :

Média (pl. médias) vient du latin *medium* qui signifie milieu ou centre. *Un média est un support de diffusion de l'information constituant à la fois un moyen d'expression et un intermédiaire transmettant un message. (Larousse 1998)*

On peut citer plusieurs médias classiques comme l'air, l'eau, le vide, les conducteurs électriques, etc. Les médias sont caractérisés par un certain nombre de propriétés comme la matière qui les constitue (métal, air, liquide...) ou encore leur topologie décrivant la structure spatiale du média (lien, réseau, volume...). Une information peut être transmise de multiples façons, mais le plus souvent, il s'agit d'une modification concernant une propriété du média. Les médias sont loin d'être équivalents et en ce sens, ils se comportent différemment vis-à-vis des messages qu'ils véhiculent. De même, la communication sera totalement différente en fonction de la façon dont le média est modifié. Pour ce qui nous concerne, nous ne considérerons que les médias avec lesquels un système informatique peut être interfacé. Par ailleurs, nous nous permettrons, à la faveur de quelques tropes, de parler de médias évolués comme le fax, le web, la voix (éventuellement transmise par téléphone).

A ce niveau, il faut émettre une remarque importante : si le média peut être modifié par le message, l'inverse est également vrai, mais cela va encore plus loin, car d'autres phénomènes extérieurs au média peuvent entrer en jeu. Ainsi, selon le média qui le véhicule, un message peut être altéré, subir des ajouts de bruits, des déformations et il peut même être purement et simplement détruit.

Disposer d'un média, n'est cependant qu'un commencement avant de pouvoir communiquer. En effet, ceux qui modifient un média, appelés émetteurs, et ceux qui perçoivent ces modifications du média, appelés récepteurs, ne sont pas forcément conscients qu'il s'agit là d'une communication. Communiquer nécessite donc de savoir comment utiliser le média pour transmettre, mais aussi pour recevoir un message. Il faut ensuite que le message transmis soit dans un langage commun aux différents intervenants. Une fois qu'émetteurs et récepteurs peuvent se comprendre, il faut alors ajouter

divers mécanismes et protocoles comme l'adressage, la diffusion, la sécurité, etc. C'est pour toutes ces raisons que la complexité de la communication est réduite à l'aide d'une approche par couches d'abstraction (dite de layering). Globalement, on peut dire que, plus on descend vers les couches basses, donc vers le concret, plus des phénomènes physiques entrent en jeu et ajoutent des risques d'altération (bruits, déformations, etc). C'est pourquoi, plus on remonte vers les couches abstraites, plus on se dote de moyens de corriger ces altérations. Le grand avantage de l'abstraction est qu'elle permet d'utiliser une fonctionnalité fournie par le niveau inférieur, sans se préoccuper de la façon dont elle se déroule réellement. Un exemple typique d'architecture en couches d'abstraction est le modèle OSI établi par l'International Standard Organization (ISO) pour les réseaux informatiques *(ISO 1994)*. Nous reviendrons sur cette notion de couche d'abstraction lorsque nous décrirons l'architecture que nous avons proposée dans le chapitre 7. Quoi qu'il en soit, moins on dispose de mécanismes assurant la qualité de la communication, plus l'incertitude et l'incomplétude du message transmis peuvent être grandes.

Communiquer n'est pas chose aisée, surtout si l'on cherche à mettre en œuvre plusieurs médias et passer ainsi à la communication ***multimédia***.

Multimédia *: qui utilise ou qui concerne plusieurs médias; Ensemble des techniques et des produits qui permettent l'utilisation simultanée de plusieurs modes de représentation de l'information. (Larousse 1998)*

Nous soulignons que, telle que nous allons l'appréhender, la communication multimédia traitera davantage du transport de l'information sur de multiples canaux, plutôt que des produits de grande distribution classiquement appelés logiciels multimédia ou encore des ateliers de production multimédia.

2. Présentation de Dialoca et de ses activités

La société Dialoca, anciennement MIC2, a été créée en octobre 1996. Ses activités s'articulent autour de la notion de suite logicielle, avec à la base, une plate-forme de communication appelée Unimédia, et au-dessus, un bouquet de services multimédia qui ne cesse de se renforcer. Ces services sont principalement axés autour du dialogue en langage naturel avec l'utilisation d'outils comme la reconnaissance et la synthèse de parole, avec des applications de téléphonie, mais ils font également intervenir le web, le courrier électronique, etc.

L'activité principale de Dialoca *(Lederman 2000)* est de proposer à ses clients des services Unimédia et leur hébergement. Toutefois, un client peut administrer lui-même son système. A cet effet, Unimédia dispose également d'un langage de scénarisation, décrit dans *(Assadourian et Maillet 2000)* et *(Sachet et Maillet 2000)*, permettant au client le développement de ses propres applications. Dans une seconde activité, Dialoca propose la conception d'applications spécifiques au cas par cas.

2.1. Qu'est-ce qu'un service Unimédia ?

Nous allons tenter de donner une définition au terme ***service*** que nous allons largement utiliser par la suite.

L'utilisation courante du terme ***service*** désigne *l'usage que l'on peut faire de quelque chose* ou *ce que l'on fait pour être utile à quelqu'un à titre onéreux ou non. (Larousse 1998)*

On voit que derrière cette notion, on retrouve en fait la fourniture d'une fonction utile. Par ailleurs, le synonyme le plus commun désigne la notion d'aide que l'on apporte. Ainsi, un service est la délégation de la réalisation d'une tâche à un tiers. Les raisons de la délégation peuvent être de nature très variée, en effet il peut s'agir d'actions que l'on ne peut ou que l'on ne souhaite pas effectuer. Il se peut également que le tiers soit à même de réaliser la tâche avec une qualité, une expertise, un volume ou une cadence que l'on ne peut atteindre soi-même. Le terme service sera interprété en se référant à la tâche réalisée et non pas à la partie d'un organisme qui réalise un service. Il faut donc, pour parler de service, déterminer les participants, décrire les domaines concernés ainsi que les fonctionnalités proposées.

2.2. Domaines des services

Le champ d'application des services de communication multimédia est très large. Il touche de nombreux domaines dans le monde du travail (commerce, économie...) mais on peut l'étendre potentiellement à l'enseignement et à la formation mais aussi plus généralement, à la culture, à la vie pratique, etc.

A titre d'exemple, voici quelques-unes des applications proposées par la société *(Dialoca 2002)* :

- l'automatisation de l'accueil téléphonique dans l'entreprise,
- l'automatisation partielle des centres d'appels,
- le développement des nouveaux services pour les serveurs vocaux interactifs,
- l'accès vocal aux sites web,
- l'interactivité vocale avec les informations de l'entreprise pour ses membres nomades.

Nous allons dans un premier temps, décrire la plate-forme logicielle Unimédia puis, dans un second temps, présenter l'ensemble des services proposés par Dialoca que nous avons étudiés en détail.

3. La plate-forme Unimédia

L'architecture d'Unimédia est présentée dans la Figure 11. Unimédia est construit autour d'une bibliothèque de fonctions *(Ricart et Brizzi 1998)* qui implante un protocole de communication : le Protocole MultiMédia Unifié (PMMU). Ce protocole, basé sur TCP/IP, permet l'échange de trames contenant des messages typés avec des paramètres associés. Il permet de mettre en liaison quatre types d'éléments appelés **ressources Unimédia**, selon un principe client-serveur :

- les **pilotes**, permettant d'effectuer des entrées et des sorties d'informations déclinées selon divers médias,
- les **applications** de haut niveau, contrôlées par un interpréteur de scripts appelé Application Générique,
- les **moteurs**, permettant de transformer l'information et
- les **modules d'administration**, permettant la gestion des ressources Unimédia, la configuration de la plate-forme, la trace d'évènements, etc.

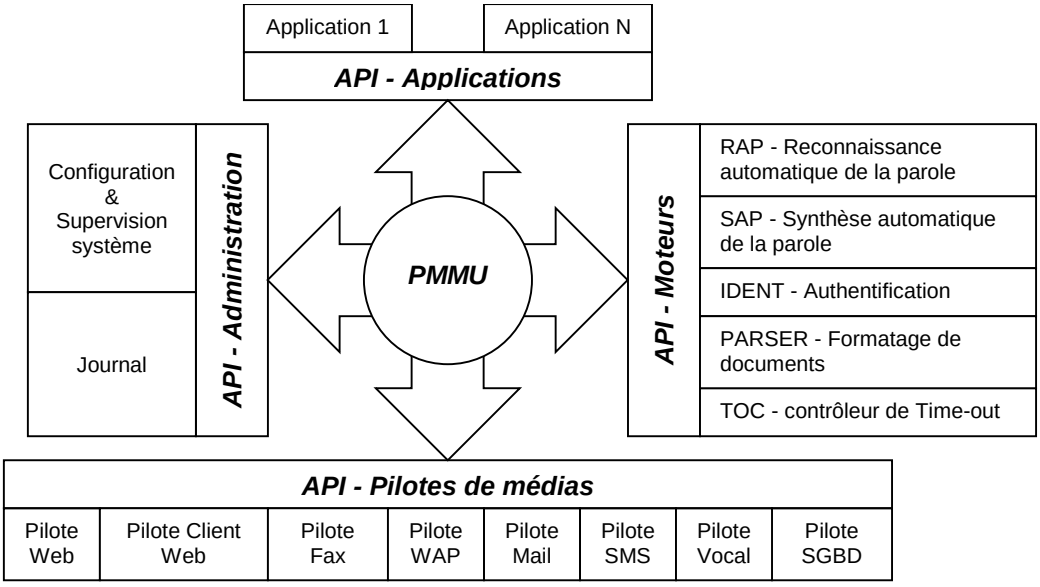


Figure 11 - Architecture d'Unimédia

3.1. Les différents flux

Le protocole PMMU permet de véhiculer trois types de flux :

- Tout d'abord, un flux de contrôle partant des applications vers les pilotes et les moteurs. Ce flux contient la description des actions que ces ressources Unimédia doivent effectuer. A l'inverse, les pilotes peuvent générer des événements qui remontent vers les applications, par exemple pour indiquer qu'un utilisateur a raccroché son combiné téléphonique.
- Ensuite, on trouve les flux de données, qui peuvent être d'une taille assez conséquente, car ils contiennent notamment des échantillons de parole. Ces flux relient surtout entre eux les pilotes et les moteurs.
- Enfin, les flux d'administration qui permettent la gestion de la plate-forme. Ceux-ci proviennent essentiellement des modules de configuration et de gestion.

3.2. Les Pilotes

Un pilote est un processus gérant un média particulier. Les différents pilotes disponibles sont dans le Tableau 3.

Tableau 3 - Les pilotes Unimédia

Pilote	***Description***
Web	Le pilote web, basé sur un serveur http, permet de créer des documents hypertexte dynamiquement, à partir de modèles HTML, de transmettre des vidéos et des images. Il utilise un mécanisme de suivi de sessions pour gérer les actions d'un utilisateur sur le site web. Il permet également la gestion de communications point-point, mais aussi point-multipoint.
Client Web	A l'inverse du pilote web, cette application émule un navigateur avec le protocole HTTP et peut alors se connecter sur des sites Internet afin de remplir des formulaires connus et de récupérer des données.
Fax	Ce pilote permet la réception et l'envoi de documents par fax selon un modèle donné.
Wap	Ce pilote est basé sur un serveur Wap, ce qui permet l'accès aux documents WML par des appareils de téléphonie mobile compatibles.
E-Mail	Ce pilote s'interface avec des serveurs de messagerie électronique SMTP et POP3 afin de pouvoir envoyer et recevoir des e-mails. A la réception d'un e-mail, les différents champs du message sont analysés et retournés, avec le corps du message, en direction des applications.
SMS	Ce pilote, basé sur un modem GSM permet la gestion des messages que l'on peut émettre et recevoir depuis un téléphone portable.
Vocal	Le pilote vocal permet la gestion de la téléphonie sur une architecture de type Dialogic. Il est utilisé lorsqu'une interaction vocale avec les utilisateurs est nécessaire. Il gère les appels entrants et sortants et permet, d'une part, la mise en liaison directe entre deux humains, mais aussi une interaction homme-machine. Il permet également la lecture et la diffusion de messages vocaux.
SGBD	Le pilote de base de données permet l'accès aux sources de données via ODBC.

Les différents pilotes permettent (si c'est possible) de décliner sur le média correspondant des actions telles que :

- la gestion des connexions avec les appels, les réponses et les déconnexions,
- les envois de données en provenance des applications,

- les sélections de choix dans des listes et
- les enregistrements de messages.

L'approche choisie par les concepteurs d'Unimédia est de rendre les applications les plus indépendantes possible des médias, de sorte à ne devoir y traiter que de l'information de "haut niveau". Ainsi, les spécificités d'un média sont restreintes aux limites du pilote qui gère ce média. Le plus souvent, un pilote est paramétré au moyen de fichiers de modèles, décrivant comment dérouler une action dans un cas précis. Par exemple, lorsque l'on réalise une action d'envoi vers un pilote vocal, celui-ci a besoin de connaître l'intitulé à envoyer.

3.3. Les Moteurs

Les moteurs sont des processus permettant d'effectuer des traitements sur les données qu'ils reçoivent. Les différents moteurs sont décrits dans le Tableau 4.

Tableau 4 - Les moteurs Unimédia

Moteur	***Description***
RAP	Ce moteur permet d'effectuer une reconnaissance automatique de la parole (RAP) à partir de données sonores en utilisant une grammaire et des dictionnaires phonétiques donnés. Il retourne la séquence de mots la plus probable.
SAP	Ce moteur permet, à partir de données textuelles, d'effectuer la synthèse automatique de la parole (SAP) à destination d'un fichier vocal ou d'un flux sonore dirigé vers le pilote vocal.
PARSER	Ce moteur a pour but l'analyse structurelle de documents formatés (HTML, XML, VoiceXML...)
TOC	Ce moteur permet la gestion des limites temporelles (Time Out) lors des interactions
IDENT	Ce moteur permet l'authentification des utilisateurs par une signature sonore avec des technologies de type AudioSmartCard.

D'autres moteurs sont venus renforcer l'ensemble existant, comme le moteur de débruitage et le moteur d’identification thématique.

3.4. L'Application générique

Les différents services Unimédia sont réalisés à l'aide de scripts qui sont des enchaînements d'étapes. Une application Unimédia résulte de l'interprétation d'un script au travers de l'Application Générique, dont l'architecture est donnée Figure 12. Les scripts peuvent être lus depuis des fichiers écrits dans un langage spécifique ou être chargés depuis une base de données.

Les étapes sont en fait les briques de base d'un service de communication. A chaque étape, une action est effectuée en direction d'une ressource de la plate-forme. Les actions permettent par exemple :

- d'envoyer des données sur le média (*send*),
- de récupérer des données textuelles depuis le média (*getstrings*),
- de sélectionner des éléments (*select*),
- de fixer une variable de connexion dans le pilote (*cnxvarset*), etc.

Une fois l'action envoyée à la ressource, cette ressource utilise un fichier de modèle pour décliner l'action selon le média utilisé. Une action n'est pas obligatoirement bloquante, ce qui fait que l'application peut se poursuivre pendant que d'autres actions s'exécutent. Après l'exécution de l'action, la ressource mise en œuvre retourne les divers résultats (*user inputs*) ainsi qu'un code de retour (*action response*). De nombreux évènements peuvent se produire dans la plate-forme et l'application peut

recevoir des messages particuliers (*event*). Par exemple, lorsqu'un utilisateur raccroche son combiné, le pilote vocal crée un événement de déconnexion (*disconnect*) qu'il envoie à l'application. En fonction des codes de retour, des évènements ou du contenu des variables internes, la suite du déroulement de l'application en cours d'exécution est donnée par un branchement à une autre étape du script. Toutefois, certaines étapes peuvent être spécifiquement attachées à des évènements particuliers. En cas de déclenchement, le cours normal de l'application est alors interrompu et détourné vers ces étapes spécifiques. Les actions peuvent également définir et manipuler un certain nombre de variables qui sont stockées dans l'interpréteur dans des zones appelées contextes.

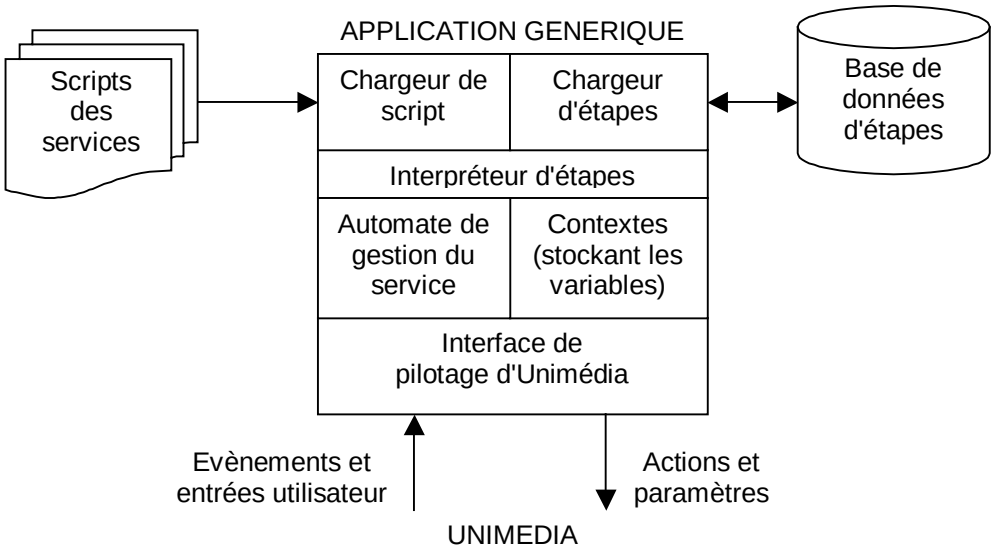


Figure 12 - Architecture de l'Application Générique

3.5. L'Administration d'Unimédia

La plate-forme Unimédia nécessite l'utilisation d'un certain nombre d'outils permettant

- de gérer indépendamment chacune des ressources (configuration, démarrage, arrêt, etc.),
- de surveiller la charge des différents serveurs, la charge réseau, les ports utilisés, et
- de surveiller l'activité de chaque ressource au travers de traces.

4. Les services Dialoca

Un service Dialoca est une application Unimédia permettant de réaliser une tâche au travers des pilotes et des moteurs disponibles. Il met en œuvre un certain nombre d'opérations de communication et crée une chaîne informationnelle entre les différents intervenants, qu'ils soient humains ou non. Nous allons décrire en détail les services proposés par Dialoca pour en tirer une analyse comparative. Pour chacun d'entre eux, nous présenterons les principales fonctionnalités, la façon dont ils se déroulent et les ressources Unimédia qu'ils utilisent.

4.1. Le Service de télé-réunion RMS

RMS signifie Remote Meeting Services. Ce service permet de faciliter des réunions distantes. Il gère une réunion qui se déroule par téléphone et permet l'échange de documents Microsoft PowerPoint sélectionnés. Les intervenants choisissent tout d'abord, à l'aide d'un logiciel FTP, quelles sont les différentes diapositives qu'ils souhaitent présenter et ils les transfèrent dans la banque de documents

sur le serveur RMS. La banque de documents peut néanmoins être complétée, en cours de session. Le service se déclenche lorsqu'un utilisateur se connecte sur le serveur web associé au centre de télé-réunion. Cet utilisateur reçoit sur son navigateur la page de connexion au service RMS sur laquelle il doit entrer ses paramètres de connexion ainsi que son numéro de téléphone. Une fois l'utilisateur connecté, une liste indiquant l'état de chaque correspondant possible s'affiche. En effet, un utilisateur peut être déconnecté, en attente de réunion (connecté) ou encore en cours de réunion (en ligne). L'utilisateur peut alors choisir de commencer une session RMS avec les correspondants de son choix. Les intervenants sont mis en relation téléphonique et ils ont accès sur leur navigateur aux documents de la banque d'arguments qu'ils peuvent transmettre à leurs interlocuteurs. Les correspondants peuvent terminer leur session aussi bien depuis leur navigateur qu'en raccrochant leur téléphone.

Le service RMS utilise le pilote vocal pour la gestion des appels téléphoniques et il utilise la mise en liaison directe de deux postes téléphoniques (appelée aboutement) afin que les intervenants puissent dialoguer oralement. En parallèle, il utilise le pilote web pour la transmission des éléments visuels. Il met en œuvre le pilote de base de données pour gérer les abonnés, ainsi qu'un serveur de données pour la mise en cache des éléments visuels.

4.2. Les services de notification (e-Nots et e-SMS)

Nous allons maintenant décrire deux services relativement proches du point de vue des fonctionnalités. Il s'agit en effet de deux services traitant de la réception de messages asynchrones (e-mail et SMS) qui permettent le filtrage et la redirection des messages reçus.

4.2.1. La notification de messages électroniques e-Nots

Ce service est attaché à un serveur de messagerie électronique et il permet la notification d'e-mails par téléphone et/ou par mini-messages SMS avec une possibilité de réponse vocale.

Le service e-Nots est activé lors de la réception d'un e-mail. Le pilote de mail récupère le message, il découpe les champs de l'en-tête, retrouve le corps du message et les attachements éventuels, puis il active l'application en lui transmettant ces informations. L'application recherche dans la base des abonnés les coordonnées du destinataire du message à partir de son adresse. Un filtrage est alors appliqué, en fonction des options de l'abonné, selon l'expéditeur et l'objet du message. L'application peut générer un mini-message et l'envoyer sur le téléphone portable de l'abonné. Elle peut également l'appeler par téléphone pour lui diffuser le message synthétisé vocalement et éventuellement enregistrer une réponse. Dans ce cas, l'application retourne un message électronique à l'expéditeur avec le message enregistré comme pièce jointe.

L'application e-Nots utilise donc tout d'abord le pilote e-mail pour recevoir les messages à notifier et pour l'envoi des réponses avec, en pièce jointe, un fichier vocal enregistré. Il met également en œuvre une base de données des abonnés contenant les champs suivants :

- le nom de l'abonné,
- le numéro de téléphone sur lequel la notification vocale est effectuée,
- le numéro de téléphone portable sur lequel la notification SMS est effectuée,
- la demande (ou le refus) de notification vocale et
- la demande (ou le refus) de notification SMS.

L'application utilise par ailleurs le pilote vocal couplé aux moteurs de synthèse et de reconnaissance de parole pour la notification par téléphone (l'appel, la lecture du message, l'enregistrement de la réponse éventuelle, etc.), mais aussi le pilote SMS pour la notification par mini-message.

4.2.2. Le service e-SMS

Le service e-SMS permet de gérer la réception de mini-messages. L'abonné peut, selon les options qu'il a paramétré sur le site web de configuration, choisir d'être notifié par e-mail et/ou par fax. Il peut

également recevoir une notification téléphonique avec une écoute sélective. Les mini-messages peuvent être filtrés selon l'émetteur et traités différemment selon des tranches horaires définies. Il est par ailleurs possible de choisir des numéros de téléphone différents pour la notification.

Cette application met en œuvre les pilote SMS, e-Mail, fax, vocal et web. Elle utilise également les moteurs de synthèse et de reconnaissance de la parole.

4.3. Le service VOS

Le service VOS (Virtual Operator Service) permet aux utilisateurs d'appeler par téléphone un conseiller virtuel qui converse oralement avec eux de sorte à obtenir, par exemple, des informations boursières, mais aussi des conseils, notes d'actualité, recommandations, etc. Concrètement, il s'agit d'un serveur vocal permettant d'accéder à une base de données ainsi qu'à une banque d'informations complémentaires fournies par des experts et de recevoir les résultats, soit directement par téléphone, soit sur le média choisi (par fax, e-mail, SMS ou encore sur une page web).

Cette application met tout d'abord en œuvre le pilote vocal avec la synthèse et la reconnaissance de la parole pour l'interaction avec l'utilisateur. Il utilise ensuite le pilote de base de données, pour accéder aux informations, et éventuellement les autres pilotes médias (fax, mail, SMS et web), pour transmettre les données.

4.4. Le Service MAS - Centres d'expertise

Le service MAS (Multimédia Assistance Service) permet à un utilisateur de pouvoir contacter un interlocuteur humain afin d'obtenir, en temps réel, une assistance et des conseils à partir d'une borne interactive. L'utilisateur peut interroger ce conseiller, tout en visualisant les informations que ce dernier lui envoie sur l'écran de la borne. Le poste du conseiller, illustré Figure 13, est constitué d'une interface hypertexte qui lui permet de puiser les informations dans une banque documentaire tout en dialoguant avec l'utilisateur par une liaison téléphonique établie par l'application MAS avec la borne. Le conseiller peut également guider son interlocuteur au travers de sites Web, de catalogues de produits, etc. S'il est en contact avec un client, le conseiller peut également accéder à un système d'information contenant, par exemple, le compte de celui-ci.

L'interface du conseiller est composée de plusieurs parties. Il a tout d'abord accès à une hiérarchie d'éléments. Pour un service clientèle, l'interface contiendra par exemple un ensemble de domaines avec les questions/réponses les plus fréquentes pour chaque domaine. Elle pourra aussi contenir des catégories et des sous-catégories de produits avec des vignettes indiquant les éléments visuels correspondants qui sont disponibles. Le conseiller a accès à un historique des visuels transmis, il peut configurer l'application très facilement et changer, par exemple, la langue ou sélectionner un pays différent.

Le service se déclenche lorsqu'un poste (une borne ou un télé-conseiller) se connecte sur le serveur web associé au centre d'expertise. Une première partie du service sert d'initialisation et elle récupère notamment l'identification de l'appelant et le mémorise comme connecté. Le flux de contrôle du service se divise alors en deux selon la nature du poste. S'il s'agit d'une borne, l'écran d'accueil/attente est affiché puis le conseiller est appelé par téléphone. Une fois que le conseiller répond, il est mis en contact avec la borne et obtient le contrôle de celle-ci. S'il s'agit d'un poste de conseiller, alors le service demande au conseiller de s'enregistrer puis il le met en attente d'une connexion d'une borne. Une fois qu'un contact est établi avec une borne, l'interface du conseiller est affichée, permettant à ce dernier d'agir sur la borne au moyen d'un ensemble de commandes, tout en dialoguant au téléphone. Si l'utilisateur raccroche ou que l'on perd le contact avec la borne, alors le service signale cet évènement au conseiller puis il se réinitialise. Ce service a notamment été utilisé pour le support d'un centre d'appel de conseils diététiques du groupe Danone.

Le service met en jeu le pilote web pour la gestion de la partie visuelle de la borne interactive mais aussi pour l'interface de commande du conseiller. Il utilise également le pilote vocal pour la gestion des appels téléphoniques et l'aboutement des liaisons téléphoniques de sorte à ce que le conseiller et

l'utilisateur puissent dialoguer oralement. Il utilise enfin le pilote base de données pour le stockage des informations sur les états des postes.

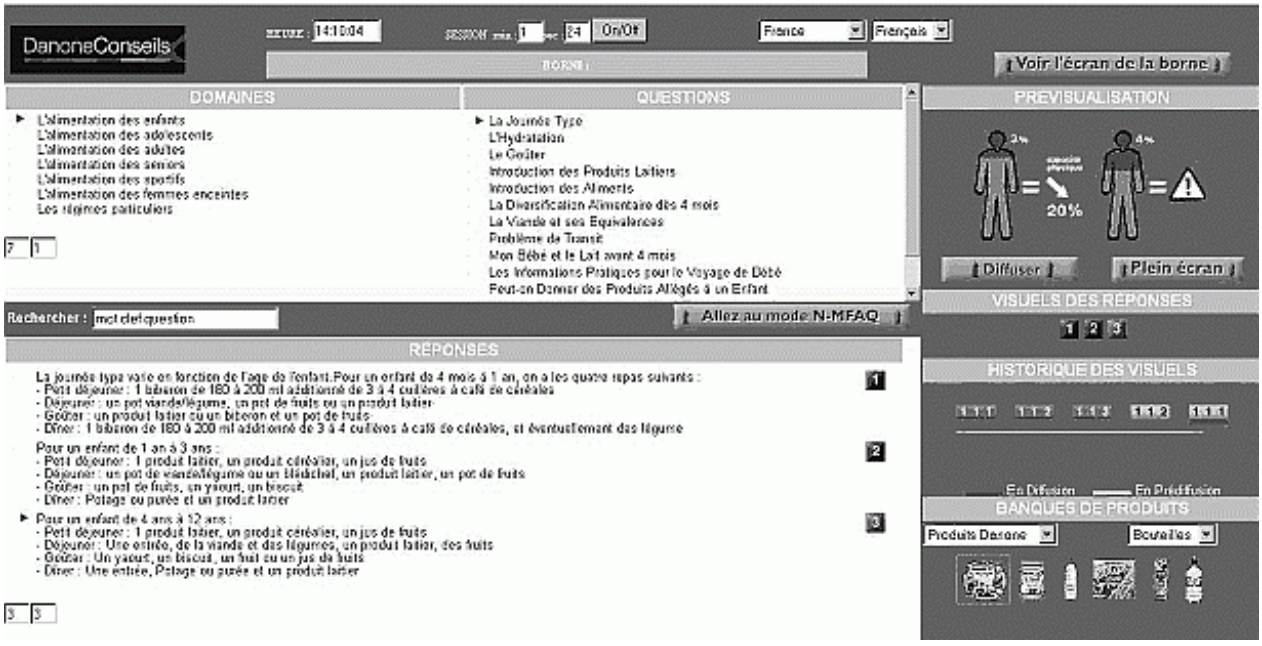


Figure 13 - Poste d'un conseiller MAS

(Reproduit avec l'aimable autorisation du groupe Danone)

4.5. Le service E-Vots - L'accès vocal à un site web

Le service E-Vots vise la navigation et l'exploitation de sites web exclusivement par téléphone, sans nécessiter ni ordinateur, ni fonction de type WAP ou autres. L'interface vocale permet aux utilisateurs de formuler leurs demandes d'informations et d'obtenir en retour les données et les textes.

Le service se comporte, en pratique, comme un "internaute virtuel". L'application nécessite un partenariat avec le site Web concerné afin que la structure des pages consultées soit connue et que le module de navigation puisse extraire les informations pertinentes. Ce module permet d'une part l'analyse et l'extraction des données disponibles sur le site Web, mais il permet également de transmettre des informations au travers de formulaires existants. Pour le site partenaire, il faut noter que le service e-Vots est perçu comme un internaute ordinaire. Par exemple, pour le remplissage d'un formulaire HTML, le service se déroule de la façon suivante : tout d'abord l'utilisateur formule sa requête qui est analysée et traduite sous une forme exploitable par le service. Le service se connecte alors sur le site web, puis il récupère le formulaire HTML. Il soumet la requête et récupère les résultats. Il synthétise alors un message sonore, à partir de ces résultats et il le diffuse enfin à l'utilisateur. Ce service a notamment été appliqué dans le domaine boursier afin de permettre le passage d'ordres par téléphone en passant par un site web partenaire.

Ce service nécessite l'utilisation du pilote vocal pour gérer les appels des utilisateurs puis des moteurs de reconnaissance et de synthèse de la parole. Il utilise également le pilote client web qui se charge de toute la navigation sur le site web partenaire (envoi de requêtes http et analyse des documents HTML).

4.6. Analyse comparative

Le Tableau 5 présente une comparaison des différents services selon les objectifs, les éléments nécessaires à leur fonctionnement et les actions de communication qu'ils mettent en œuvre.

Tableau 5 - Caractéristiques des services de la palette Dialoca

Service	*Objectifs*	*Eléments nécessaires*	*Actions de communication*
RMS	Gérer une télé-réunion en temps réel avec un support vocal et visuel	Les participants de la télé-réunion ayant chacun un navigateur et un téléphone	• connexion et déconnexion • synchronisation web-téléphone • diffusion de visuels
E-Nots	Notifier les courriers électroniques par fax, SMS ou téléphone (avec possibilité d'enregistrer une réponse vocale)	• Les abonnés, avec leur compte courrier, et les médias pour les joindre (téléphone, SMS, fax...) • La base de données des abonnés • L'émetteur avec son adresse mail	• réception de messages • accès aux informations de l'abonné • envoi d'une notification • connexion et déconnexion • synthèse et reconnaissance vocale • enregistrement de message
e-SMS	Gérer la réception et la retransmission de messages SMS reçus	Les abonnés, avec leur compte sur le serveur SMS et un moyen de notification (e-mail, télécopie, téléphone)	• réception de messages • filtrage • envoi d'une notification
MAS	Faciliter le travail d'un télé-conseiller et lui offrir un support d'argumentation à l'aide d'éléments multimédia	• Le conseiller, son navigateur et son poste téléphonique • Les clients du service de conseil • Les bornes de consultation • Une source de données alimentée par des experts	• connexion et déconnexion • synchronisation web-téléphone • diffusion de visuels • recherche dans la source de données
VOS	Accueillir des clients et permettre la consultation de sources de données expertes, répondre aux questions avec la possibilité d'envoyer des documents	• Les sources de données consultées • Les clients, avec leur téléphone • Les fournisseurs de données • Les destinataires des documents, avec leur e-mail ou leur fax	• connexion et déconnexion • synthèse et reconnaissance vocale • fourniture de l'information par les experts • transmission des données • envoi de données formatées
e-Vots	Effectuer des transactions vocales sur un site web à partir d'un téléphone	• Les abonnés • Le site web du partenaire	• connexion et déconnexion • synthèse et reconnaissance vocale • émulation de navigation web • synthèse des résultats

Nous avons étudié d'autres services très spécifiques, en dehors de la gamme décrite dans *(Lederman 2000)*, comme la démonstration de réservation de billet d'avion pour une compagnie aérienne sur laquelle nous reviendrons par la suite. On peut également citer le service Pandora qui permet d'automatiser une prise de commande par téléphone au travers d'un dialogue où le client indique oralement toutes les informations nécessaires.

En observant les caractéristiques des services du Tableau 5 et la représentation des éléments mis en oeuvre dans la Figure 14, nous nous apercevons que les services proposés, se répartissent en deux grandes catégories :

- D'une part, des services qui facilitent la communication entre les clients/abonnés, comme RMS, e-Nots, et e-SMS. A ce niveau, Unimédia sert de plaque tournante : l'information arrive par l'un des intervenants et repart vers les autres, sous une forme différente ou par diffusion avec d'éventuels retours.

- D'autre part, des services qui facilitent l'accès à l'information et qui permettent des transactions au travers d'une interface, souvent vocale et téléphonique, éventuellement accompagnée d'un poste hypertexte. Il s'agit de MAS, VOS, e-Vots et Pandora qui se distinguent par l'intervention d'un humain au cours du service et par la nature des sources d'informations utilisées. L'information est, la plupart du temps, renseignée par des experts et stockée dans une base de données pour un accès asynchrone. En fait, les applications Pandora et e-Vots servent de passerelle vers d'autres services proposés par des partenaires.

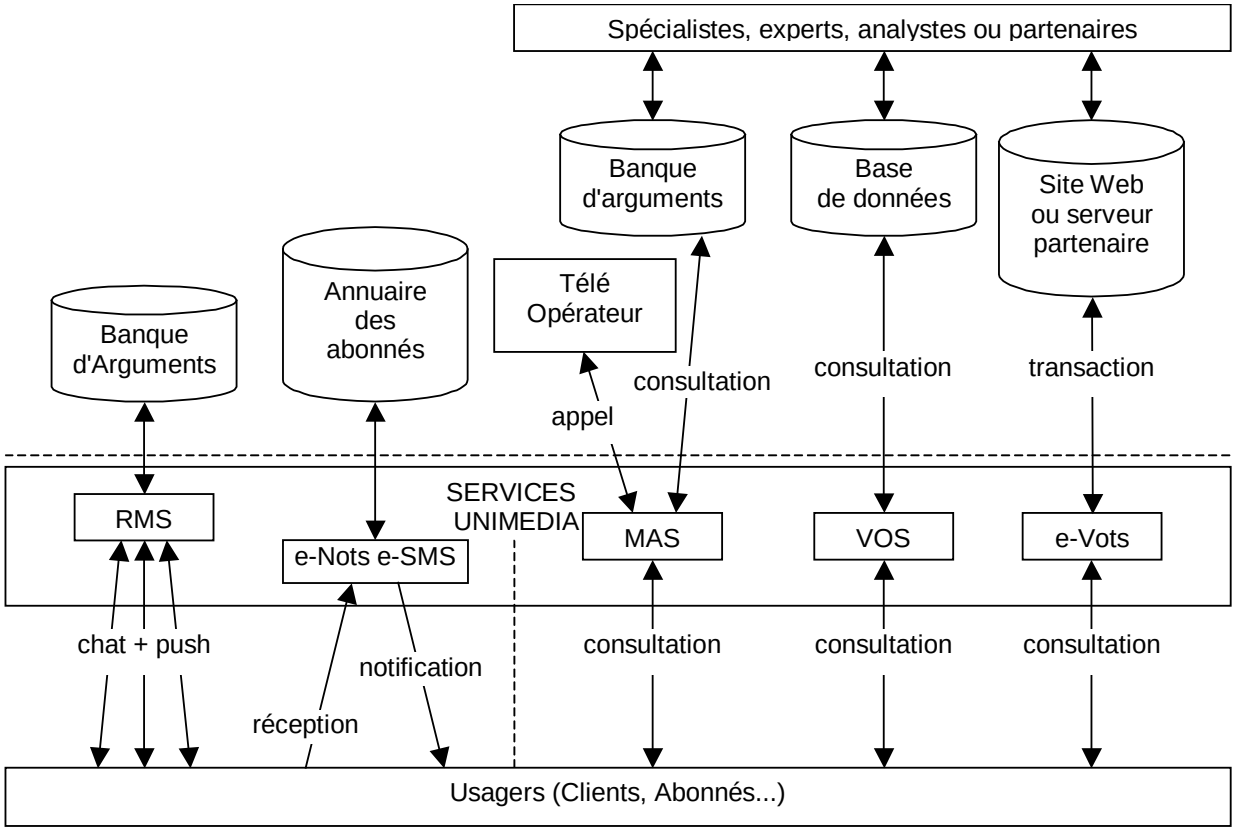


Figure 14 - Partition des services Dialoca

5. Identification des besoins

Lors de notre étude des différents services et en interaction avec les experts Dialoca, nous avons pu identifier un certain nombre de problèmes et de besoins que nous avons regroupés en deux principaux thèmes :

1) Permettre l'évolution des services de la plate-forme Unimédia avec une généralisation des services existants.

Concevoir un nouveau service pour Unimédia nécessite une approche exhaustive réalisée par un expert Dialoca qui doit prévoir tous les cas possibles. Nous verrons dans le chapitre 7 que les scripts obtenus sont très complexes et qu'il est, par conséquent, difficile de les rendre résistants aux aléas d'utilisation. Par ailleurs, faire évoluer ou réutiliser un service existant est bien sûr possible, mais cela nécessite un nombre non négligeable de modifications et surtout une nouvelle intervention experte. Sur ce point, ce qui a motivé nos travaux est l'idée de concevoir un service, sans avoir nécessairement à le décrire dans son intégralité, mais de façon plus abstraite.

2) *Permettre aux services de s'adapter aux utilisateurs, aux ressources Unimédia, aux contextes, mais également aux divers aléas qui peuvent survenir, en prenant notamment en compte des cas particuliers d'utilisation.*

Malgré une approche qui couvre la plupart des cas, la complexité des services fait qu'il n'est pas possible de prendre en compte tous les problèmes et incidents pouvant survenir pendant l'exécution. Même lorsqu'un service se déroule correctement, les médias et les ressources par lesquels transite l'information sont rarement d'une qualité irréprochable. Ainsi, elle se voit altérée de diverses manières. Nous pouvons illustrer cela dans un service comme VOS, qui est basé sur une interaction à l'aide d'une ressource téléphonique. L'utilisation du téléphone couplée à la reconnaissance automatique de la parole peut augmenter le risque de mauvaise reconnaissance de mots car la bande passante du téléphone dégrade le signal audio. Dans le cas d'un dialogue par téléphone, il serait ainsi plus judicieux que l'application pose des questions qui appellent des réponses simples, comme "oui" ou "non", plutôt que de demander par exemple le nom d'une rue (il y a bien souvent un très grand nombre de réponses possibles et, de plus, celles-ci peuvent être complexes).

Les ressources et les médias utilisés ne sont pas les seules sources d'aléas et d'incertitude. Les différents intervenants d'un service, qui rappelons-le sont bien souvent des humains, sont eux-même des sources d'incertitude, d'ambiguïté et autres erreurs. Ainsi, toujours au niveau d'une reconnaissance vocale, il faut prendre en compte les caractéristiques de la personne qui parle. L'idéal est de savoir s'il s'agit d'un homme ou d'une femme, de connaître son âge mais aussi de pouvoir passer outre le problème des accents. Par ailleurs la voix dépend aussi de l'état physiologique de la personne. En effet, nous ne parlons pas de la même manière lorsque nous sommes malades, fatigués, angoissés, etc. Ainsi, pour un locuteur donné, dans un contexte précis, certains mots vont être plus ou moins faciles à reconnaître, aussi vaut-il mieux éviter de poser des questions qui risquent de conduire à des réponses non reconnaissables.

Nous venons d'énoncer diverses sources de difficultés liées à l'utilisation d'un média particulier, mais pour une interaction avec des humains, il ne faut pas oublier qu'il nous arrive d'être indécis, d'hésiter ou de changer d'avis. Il nous arrive également de nous tromper, de ne pas avoir la connaissance nécessaire pour répondre à une question ou encore de ne pas vouloir répondre. Dans le cas d'une application de transactions boursières utilisant le service VOS, supposons qu'un utilisateur appelle le serveur vocal et veuille passer un ordre d'achat. Cet utilisateur va devoir formuler sa demande en termes clairs sur le domaine complexe de la bourse alors qu'il peut être totalement novice et n'avoir par conséquent qu'une connaissance très partielle. Il s'agit donc là d'un problème de taille que de vouloir fournir des services adaptatifs avec des informations incertaines ou incomplètes.

Ainsi, pour Dialoca, les objectifs de la collaboration étaient d'enrichir leur offre et d'améliorer la qualité des services en utilisant des solutions issues de l'intelligence artificielle avec, plus particulièrement, des approches à base d'agents. Comme nous allons le voir, l'utilisation des systèmes multi-agents était motivée par le fait que les services mènent finalement à des problèmes d'interaction entre les différents intervenants et ceci, dans des environnements pouvant être distribués tant du point de vue spatial que du point de vue des domaines d'expertise mis en œuvre. De plus, une modélisation à base de systèmes multi-agents permet de se diriger vers des systèmes plus ouverts, plus dynamiques et plus évolutifs, où l'effort cognitif de conception est allégé.

6. Adéquation à notre problème de recherche

Nous allons maintenant expliquer de quelle façon la problématique générale sur les systèmes multi-agents hétérogènes peut être instanciée pour la réalisation de services adaptatifs dans le domaine de la communication multimédia.

6.1. Correspondance en terme d'agents

Les agents dont nous parlerons sont les intervenants des communications s'effectuant dans le cadre des applications Unimédia. Nous considérons notamment qu'Unimédia va se comporter, lui-même, comme

un agent logiciel communicant, comme le montre la Figure 15. En effet, si nous examinons en détail les différentes caractéristiques des agents, nous pouvons établir les relations suivantes:

- Les perceptions de l'agent sont obtenues au travers des pilotes média, qui traduisent l'ensemble hétérogène des messages entrants en informations manipulables par les services avec une forme unifiée.
- Les actions de l'agent correspondent aux messages d'actions envoyés au cours du déroulement des services aux différents pilotes, afin que ceux-ci les déclinent sur les médias disponibles.
- Les scripts des services Dialoca correspondent au comportement même de l'agent. En effet, ce sont eux qui indiquent quelle action effectuer en prenant en compte tous les évènements perçus.
- Cet agent dispose également d'une rationalité limitée, dans le sens où il doit chercher à satisfaire les fonctionnalités attendues par les services.

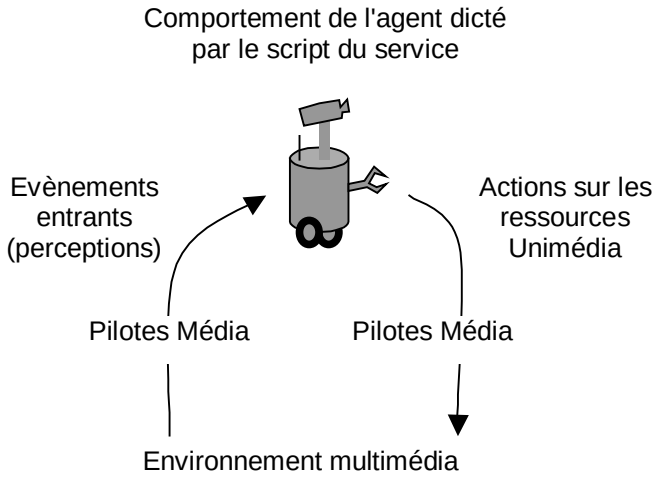


Figure 15 - Boucle sensori-motrice instanciée pour Unimédia

Parmi les autres agents, nous retrouvons bien-sûr les humains, tels que les utilisateurs finaux, les conseillers, les experts, les administrateurs, etc. Nous considérons également les fournisseurs d'informations comme des agents. Si on prend l'exemple du service E-Vots, le site web sera en fait considéré comme une partie visible d'un agent logiciel.

6.2. Correspondance en terme d'environnements

En examinant le monde dans lequel se déroulent les services de communication multimédia, tels que ceux fournis par la société Dialoca, nous pouvons trouver comment est constitué l'environnement des agents : les services Unimédia se situent dans le cadre de transports et de traitements des informations, qu'elles soient physiques, comme les signaux audio ou virtuelles, comme les requêtes envoyées aux bases de données. On retrouve également des objets qui peuvent être fort complexes, comme des pages hypertexte. Ainsi, les environnements que nous allons considérer sont loin d'être des mondes faciles à modéliser. Ils doivent permettre de prendre en compte, d'une part, des agents logiciels et des agents physiques, mais aussi un grand nombre d'éléments comme les divers milieux de transport relatifs au domaine de l'information... Il est tout aussi difficile d'établir une métrique, ce qui explique que l'on ne pourra définir des notions de position et de distance que dans certains cas bien précis. Ainsi, on pourra rarement utiliser la notion de localité, qui s'avère souvent bien utile, car elle permet de limiter la complexité lors de la modélisation des systèmes.

Nous avons cherché à caractériser les environnements de communication multimédia à partir de critères *(Russell et Norvig 1995)* et nous obtenons les propriétés suivantes :

- ils ne sont pas accessibles, car la perception des agents est partielle et qu'ils ne disposent pas forcément de toutes les informations nécessaires pour prendre des décisions,
- ils sont indéterministes, car les actions entreprises par les agents n'ont pas de succès garanti,
- d'une façon générale, ils ne sont pas épisodiques, car les interactions qui se sont passées dans des sessions antérieures peuvent avoir une influence sur le déroulement d'un service à l'instant présent. Cependant, nous considérerons que seuls certains éléments des interactions passées sont utiles pour le futur.
- ils sont dynamiques, car ils sont ancrés dans une réalité qui reflète l'activité humaine et celle des autres agents et
- ils sont a priori continus, car les agents peuvent effectuer un grand nombre d'actions. Nous chercherons cependant à abstraire les perceptions et les actions pour nous ramener à des ensembles finis afin de limiter la complexité.

7. Conclusion

Dans ce chapitre, nous avons présenté le domaine de la communication multimédia ainsi que l'approche Dialoca, avec ses différents services. Que ce soit en terme d'environnement ou d'agents, nous avons pu également constater que ce cadre applicatif correspondait bien à notre problématique de collaboration dans un système multi-agent hétérogène. Nous pouvons donc exprimer nos objectifs de la façon suivante :

Faciliter la conception et le contrôle de services de communication multimédia en les exprimant comme des problèmes de coopération à résoudre par un ensemble d'agents hétérogènes.

La correspondance que nous avons décrite dans la fin de ce chapitre va se poursuivre dans le chapitre suivant pour constituer notre solution théorique. Nous y précisons comment les services de communication multimédia peuvent prendre place au niveau des schémas d'interactions et dans l'organisation entre les différents agents. Après avoir présenté cette solution théorique, nous montrerons, dans le chapitre 7, comment elle peut être mise en œuvre dans ce cadre applicatif.

Partie II. La solution théorique

Chapitre 3 - Modélisation des services par une coopération multi-agent hétérogène

Dans ce chapitre, nous proposons des éléments de solution pour notre problématique générale de coopération multi-agent hétérogène et son application aux services de communication multimédia :

Trouver un modèle et des méthodes permettant de coordonner un ensemble d'agents hétérogènes pour réaliser coopérativement une tâche dans un environnement incertain.

Faciliter la conception et le contrôle de services de communication multimédia en les exprimant comme des problèmes de coopération à résoudre par un ensemble d'agents hétérogènes.

Nous revenons, dans un premier temps, sur le problème de l'hétérogénéité que nous avons distingué selon deux aspects (nature et contrôle), afin de proposer une nouvelle classification de l'ensemble des agents. A partir de cette classification, nous nous proposons de voir les services de communication multimédia comme les cadres d'une coopération multi-agent. Nous examinons le cas particulier de la coopération entre un utilisateur et un fournisseur de service dans lequel nous identifions un besoin de médiation. Nous présentons notre solution théorique qui est l'introduction d'agents contrôlés comme médiateurs puis, après une extension au cas plus général, nous définissons un ensemble de rôles qui nous permettent une modélisation des services à l'aide de schémas d'interaction. Nous présentons enfin l'ensemble des classes de service que nous avons défini, en nous servant des rôles comme briques de base.

1. Classification d'un ensemble d'agents hétérogènes

Nous avons indiqué au chapitre 1 qu'il était possible de distinguer des agents selon leurs capacités, leur taille, leur architecture, leurs perceptions, etc. Dans notre cas, afin de pouvoir appréhender les différences de communication dues à l'hétérogénéité dans un système multi-agent et pour comprendre quels types d'adaptation étaient nécessaires, nous avons défini une classification d'un ensemble ***A*** d'agents, selon deux critères :

1) Nous avons partitionné l'ensemble ***A*** en considérant d'une part les ***agents physiques*** $\boldsymbol{A_P}$, tels que des humains ou des robots, et d'autre part des ***agents logiciels*** $\boldsymbol{A_L}$. Cette partition permet de savoir si la qualité de la communication entre deux agents peut être affectée de façon physique, comme le passage du numérique à l'analogique, etc.
2) Nous avons également partitionné l'ensemble ***A*** selon le contrôle qu'il était possible d'avoir sur les agents en distinguant trois classes d'agents. Nous nous sommes placés du point de vue d'un concepteur cherchant à former un système multi-agent avec l'idée de ***contrôle***, qui correspond à la capacité à influencer le comportement d'un agent.

Nous avons distingué les classes suivantes :

a) Les ***agents contrôlés*** $\boldsymbol{A_C}$: Dans cette classe d'agent, on considère soit que le concepteur participe à la conception du comportement, soit qu'il peut fortement l'influencer. Ces agents peuvent tout de même garder une certaine autonomie quant aux détails d'exécution. On fait, par ailleurs, l'hypothèse que l'état de ces agents est totalement connu.

b) Les ***agents partiellement contrôlés*** $\boldsymbol{A_{PC}}$: Ces agents sont considérés comme des éléments du système sur lesquels on peut avoir une influence limitée à une sorte de contrat. On a cependant une connaissance limitée de leur comportement et c'est pourquoi il est plus difficile de les considérer comme connus avec certitude. D'une façon générale, on dispose de moins d'informations à leur sujet, ce qui implique que leur état devra être estimé, mais avec une incertitude limitée.

c) Les ***agents non contrôlés*** $\boldsymbol{A_{NC}}$: On place dans cette classe le reste des agents de l'environnement sur lesquels on ne peut pas, par exemple, avoir une influence directe. Il s'agit en fait d'agents "boite-noire" que l'on peut observer et avec lesquels on peut interagir, mais dont on ne connaît pas le comportement a priori. Ainsi, on ne peut pas agir directement sur eux, mais on doit interagir avec eux pour réaliser la tâche souhaitée. Par ailleurs, l'incertitude sur leur état sera bien plus forte que pour les autres classes $\boldsymbol{A_C}$ et $\boldsymbol{A_{PC}}$.

La notion de ***contrôle*** que nous venons d'introduire est donc relative au concepteur du système, mais elle peut se généraliser et refléter la capacité d'un agent à influencer facilement le comportement d'un autre agent. C'est également d'une sorte de "fiabilité subjective de l'autre", de confiance que l'agent doit prendre en compte pour interagir.

Notre classification est résumée dans le Tableau 6.

Tableau 6 - Partitions sur un ensemble d'agents hétérogènes

A	A_C	A_{PC}	A_{NC}
A_L	Programmes connus	Programmes avec lesquels il est possible de communiquer de façon privilégiée	Autres agents logiciels
A_P	Robots et autres équipements asservis	Equipements partiellement asservis et éventuellement des collaborateurs humains spécialistes	Autres intervenants humains comme des utilisateurs, le concepteur, etc.

Du point de vue de la topologie, nous voyons l'environnement comme ayant une structure de communication logicielle, appelé ***environnement logiciel***, qui se prolonge par des liens de communication terminaux dans le monde réel, dit ***environnement physique***. Le graphe Figure 16 montre un exemple illustrant la façon dont les différentes classes d'agents peuvent se répartir dans ces environnements :

- les agents logiciels de $\boldsymbol{A_L}$ perçoivent et agissent dans l'environnement logiciel et
- les agents physiques $\boldsymbol{A_P}$ perçoivent et agissent dans l'environnement physique.

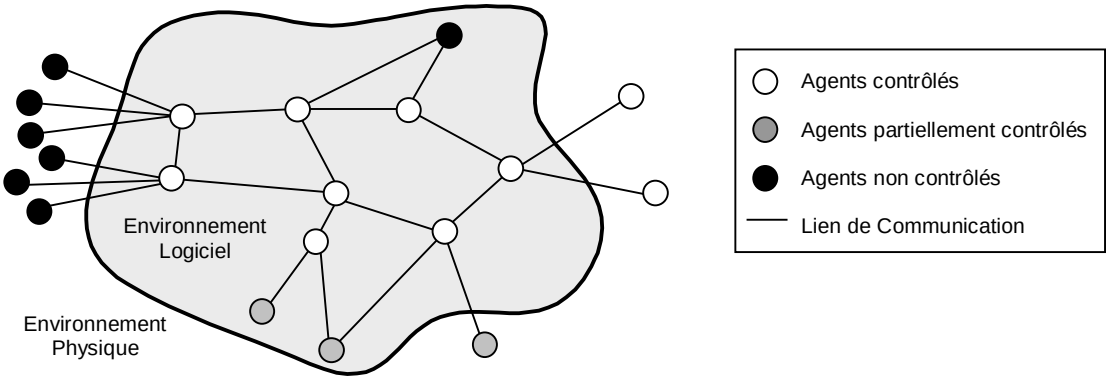


Figure 16 - Répartition des agents dans les environnements physiques et logiciels

Avec cette classification, nous ramenons le problème de l'hétérogénéité sur deux terrains :

- Le premier est celui des interfaces entre l'environnement logiciel et l'environnement physique. Dans un premier temps, afin de modéliser les services, nous nous placerons à un niveau d'abstraction suffisamment élevé pour gommer ces problèmes de liaisons.
- Le second porte sur la connaissance et l'influence que les agents ont les uns sur les autres. En effet, si l'on se place du point de vue d'un agent contrôlé physique ou que l'on cherche à concevoir un agent contrôlé logiciel, il est nécessaire de considérer les autres agents avec lesquels on doit interagir sans forcément pouvoir agir directement sur eux.

Donc, pour permettre aux agents d'un hSMA[1] de coopérer, deux points importants sont à prendre en compte : l'abstraction des différences physiques/logicielles et la considération qu'un agent peut avoir d'un autre.

2. Coopérer pour réaliser ensemble des services

Nous allons maintenant nous intéresser aux interactions entre agents hétérogènes en étudiant comment ces agents peuvent coopérer pour la réalisation de services de communication multimédia. Nous nous focalisons sur l'interaction entre des agents utilisateurs, qui ont besoin de faire réaliser des tâches et des agents fournisseurs de service, capables de réaliser ces tâches.

2.1. Un cas particulier, la coopération entre deux agents

2.1.1. Identification d'un problème de communication

Nous sommes donc amenés à chercher des schémas d'interaction pour des agents de nature très différente en ne disposant pas véritablement de langage commun au départ. Nous pouvons formuler le problème qui se pose entre deux agents hétérogènes de la façon suivante :

Etant donnés deux agents A et B qui ne soient pas contrôlables, et leurs langages de communication L_A et L_B respectifs, comment faire collaborer A et B ?

L'exemple, illustré sur la Figure 17, montre à gauche un utilisateur humain et à droite, un système capable de fournir des informations. On fait l'hypothèse que l'utilisateur a besoin d'informations et qu'il doit demander de l'aide au système qui peut le satisfaire.

[1] Dans la suite de la thèse, nous appellerons hSMA un système multi-agent qui concorde avec la classification que nous venons d'établir entre les agents : selon la nature (logicielle/physique) et selon le contrôle qu'un concepteur a sur eux ou qu'ils ont les uns sur les autres.

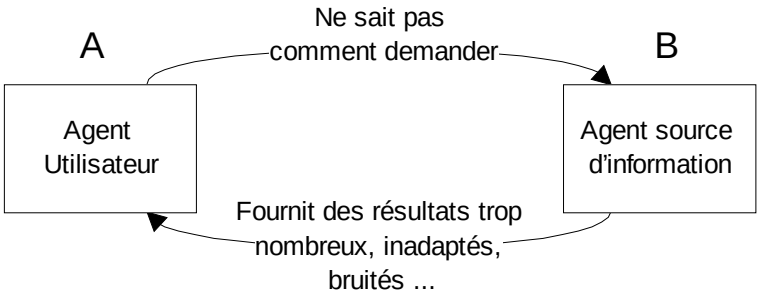


Figure 17 - Un problème de communication

On s'aperçoit que le problème vient du fait que d'un côté, l'utilisateur n'est pas capable de formuler la bonne requête et/ou a des difficultés pour interpréter le résultat correctement et que de l'autre, le fournisseur d'informations attend une requête claire et précise pour fournir des résultats bruts. Par ailleurs, la communication peut être également une source de problèmes et, par exemple, ajouter du bruit et des déformations au cours des interactions. Il y a donc là un manque que l'on doit chercher à combler. "Comment allons-nous nous y prendre ?"

Avant de détailler la solution que nous avons retenue, nous présentons quelques-uns des nombreux travaux dans lesquels nous avons identifié des approches intéressantes pour résoudre ce problème.

2.1.2. Quelques approches possibles

- Dans Open Agent Architecture (OAA) la solution proposée par *(Martin et al. 1999)* est l'introduction d'un agent intermédiaire (facilitator) qui coordonne la communauté avec un langage de communication unique appelé ICL. Dans la Figure 18, on retrouve un agent utilisateur, ici appelé demandeur, qui formule une requête correspondant à ses besoins qu'il transmet à l'agent intermédiaire. L'agent intermédiaire, qui a reçu auparavant un ensemble de descriptions des capacités des agents fournisseurs, va alors rechercher quel est le fournisseur qui est le plus à même de répondre aux besoins du demandeur. Dans OAA, on retrouve des agents d'interface pour la communication avec les humains, des agents d'applications qui fournissent des services et des méta-agents qui permettent d'effectuer des raisonnements particuliers indépendants du domaine (système de règles, planification, etc.).

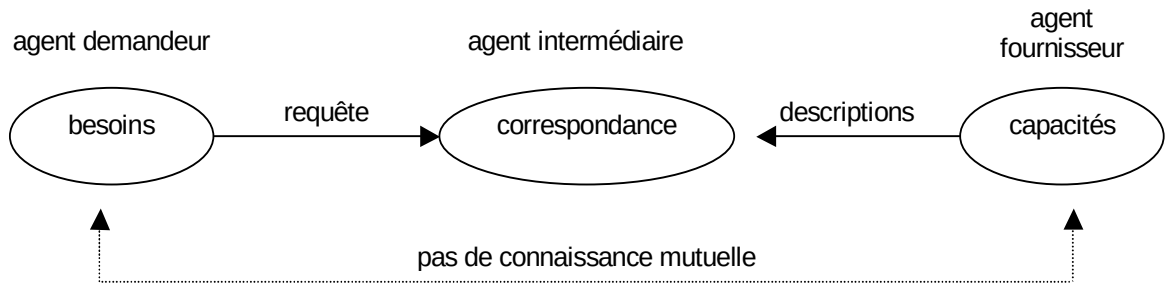


Figure 18 - L'approche de médiation dans OAA

- Dans *(Sansonnet 2001)*, on trouve la proposition d'agents dialogiques qui interagissent par le biais d'un langage de requête formel. Les agents humains peuvent intervenir dans le système grâce à une traduction de leurs requêtes en langage naturel vers le langage de requête. Sur la Figure 19, on voit comment les deux types d'agents peuvent interagir.

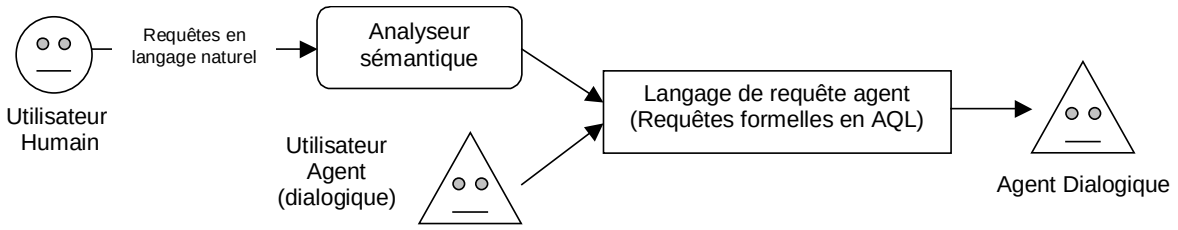


Figure 19 - Interaction avec les agents Dialogiques *(Sansonnet 2001)*

- Dans *(Vercouter 2000)*, l'auteur présente un moyen de concevoir des SMA ouverts. Il indique qu'une coopération est initiée par un agent détectant un manque dans ses compétences et par la demande d'aide aux autres agents. Il décrit la solution classique qui consiste à utiliser un courtier (broker servant d'annuaire centralisé) pour retrouver un agent capable de répondre au besoin initial. Il souligne par ailleurs les limites des courtiers et propose de rendre les agents "accueillants". Pour cela, il leur ajoute une facette qui aide les nouveaux arrivants d'un SMA à s'intégrer en remplissant les mêmes fonctions qu'un courtier.
- Avec l'approche Teamcore, *(Tambre et al. 2000)* visent la coordination d'agents hétérogènes afin que ceux-ci puissent travailler ensemble au sein d'une équipe. Cette solution est basée sur l'utilisation de KQML pour les communications d'un agent gérant les ressources de communication disponibles. Un système de "proxies" permet aux agents humains d'intervenir dans le système au même titre que les autres agents.
- Il existe également un bon nombre d'autres solutions de médiation autour de KQML, telles que UMDL *(Durfee et al. 1998)*, SHADE et COINS de *(Kuokka et Harada 1998)* ou encore NetSA de *(Côté et al. 2001)* qui ont été notamment utilisées pour des applications comme la recherche d'informations sur Internet, dans des bibliothèques électroniques, etc.

D'une façon générale, on observe que les solutions proposées tentent de combler le fossé de langage pour que l'utilisateur puisse exprimer un besoin d'une façon compréhensible par les agents logiciels, puis elles mènent à l'identification d'un agent compétent pour répondre à ce besoin et enfin à la délégation de la tâche elle-même. Cependant, on retrouve bien souvent la nécessité de modifier les agents existants dans le système, en leur ajoutant des comportements ou en les rendant capables d'utiliser un langage particulier. Ceci pourrait s'appliquer dans notre cas, mais ne serait pas adéquat pour les agents non-contrôlés ou partiellement contrôlés. Par ailleurs, rares sont les solutions qui vont aider l'utilisateur novice à formuler sa requête et à traiter facilement les résultats obtenus.

2.1.3. Notre solution : la médiation

Notre solution théorique s'appuie sur les approches de type courtier, dans lesquelles un agent particulier est capable de retrouver quel agent sait faire quoi. En effet, il est difficile d'imposer par exemple à un agent de connaître et de mettre à jour son propre réseau social et d'autant plus s'il s'agit d'un agent non-contrôlé. De la même façon, on ne peut pas imposer aux agents un langage ou un protocole commun.

Afin de faciliter l'interaction, nous proposons donc d'introduire un agent logiciel qui va être appelé un ***médiateur***. Cet agent va devoir comprendre les agents intervenants dans un service, traduire les besoins exprimés par les uns, prendre en compte les compétences déclarées par les autres, assister les agents qui en ont besoin, etc.

Dans l'exemple que nous avons donné Figure 17, comme il n'est pas possible de modifier ***A*** ou ***B***, nous ajoutons au hSMA un agent médiateur ***M***. Le médiateur va en fait être inséré pour dériver les flux de communications pouvant exister entre ***A*** et ***B***. Bien sûr, différentes combinaisons de dérivations de flux sont possibles selon le degré d'intervention de ***M*** entre ***A*** et ***B***, de plus ***M*** va pouvoir se comporter différemment selon son "attachement" à ***A*** ou à ***B*** :

- Le médiateur peut juste être un observateur du comportement exhibé par ***B*** sur un flux et donner des conseils pour ***A***, et vice-versa.

- Il peut aussi cacher complètement un agent de l'autre. Dans ce cas ***M*** peut être typiquement un assistant pour ***A***, car il indique comment répondre au comportement de ***B*** de façon adéquate ou inversement.

Dans notre approche, nous considérons que ***M*** va avoir à gérer une interaction entre ***A*** (un agent utilisateur humain) et ***B*** (un agent source d'informations logiciel), en traduisant les requêtes exprimées par ***A*** dans le langage L_A (qui peut être du langage naturel) dans le langage L_B (qui peut être du SQL) et respectivement pour les résultats. Cela mène à une nouvelle représentation de l'interaction donnée dans la Figure 20.

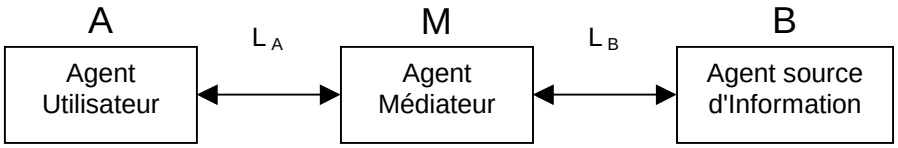


Figure 20 - Une interaction à base de médiateur

Ici, le médiateur a donc pour tâche de comprendre ce que l'utilisateur veut et il va chercher à atteindre ce but en utilisant le système d'information. Cependant, l'interaction peut fort bien se dérouler dans le sens inverse et dans l'intérêt de ***B***, qui peut devoir "pousser"des informations vers à ***A***.

2.2. Vers une médiation multi-agent

La médiation peut se généraliser et intervenir lorsqu'il y a plus de deux agents dans le système. Il est possible d'augmenter à ***n*** le nombre d'agents ***A***, ce qui conduit à une recherche d'informations multi-utilisateur. Le médiateur peut, soit se démultiplier en $M_1, ..., M_n$ et migrer vers chacun des agents utilisateurs $A_1, ..., A_n$ pour devenir leur assistant personnel, soit rester près de l'agent ***B*** et tenter d'interagir avec les agents ***A*** en exploitant davantage des phénomènes qui restent constants d'un agent utilisateur à un autre. De façon symétrique, le nombre d'agents ***B*** proposant des informations peut lui-aussi augmenter et le médiateur peut alors devoir fusionner les informations pour les utilisateurs.

De plus, la nature même de la médiation peut recouvrir de multiples applications susceptibles de se dérouler au travers des réseaux informatiques ouverts : accueil, assistance, suivi, diagnostic, conseil, traduction, recherche et diffusion d'informations, mais aussi d'effectuer une composition de ces prestations.

Dans la spécification des problèmes, cela nous conduit à identifier, tout d'abord, l'ensemble des agents non contrôlables ou partiellement contrôlables pour lesquels plus ou moins d'informations sont disponibles. Parmi ces agents, nous trouvons, par exemple, des fournisseurs d'informations, des clients, des étudiants, des experts, etc. La suite de la conception consiste à définir une tâche de coopération faisant intervenir ces agents. Il s'agira par exemple de devoir aider un utilisateur qui a un besoin donné : ce but sera délégué au médiateur qui utilisera ses propres moyens et ceux des autres pour réussir. Selon les besoins, la conception nécessitera d'ajouter, dans le système, un ou plusieurs agents contrôlés dont le comportement de médiation devra tenter de réguler le service. En plus de faire coopérer les agents, l'objectif est également de pallier les incertitudes inévitables provenant des agents non ou partiellement contrôlés et de l'environnement. Les agents contrôlés doivent donc établir des plans afin d'atteindre leurs buts, tout en restant attentifs à ce qu'ils perçoivent pour pouvoir réagir face aux évolutions du reste du SMA. Ces agents contrôlés doivent aussi être en mesure de sélectionner la meilleure action à faire dans des situations imprévues, même si elles sont peu fréquentes. Les agents doivent exhiber des capacités de communication et d'organisation pour qu'ils aient des habilités sociales envers les autres agents. Il faut en effet que ces agents apprennent à se connaître les uns et les autres. C'est à ce niveau que nous établissons le lien avec les travaux sur les concepts d'organisation et de rôle dans les systèmes multi-agents.

3. Modélisation des services

Dans nos travaux, nous considérons que la communication entre les agents s'effectue par l'échange de messages sur les médias. Cependant, le degré d'interaction est variable, car si l'on peut faire communiquer des agents logiciels avec un langage ou un protocole bien formalisé, il n'en est pas de même lorsque l'on a affaire à des agents humains non contrôlés. Nous avons choisi une modélisation à base de graphes de transitions qui est plus proche des outils de modélisation de comportements que nous utilisons par la suite. Cependant, l'utilisation des réseaux de Pétri aurait pu être tout aussi bien adaptée.

Les graphes que nous proposons de construire vont refléter les comportements typiques de haut niveau des agents pour le service considéré. Nous estimons, en effet, que les caractéristiques structurelles qui différencient les agents peuvent être négligées par une abstraction au profit des caractéristiques fonctionnelles.

On ne parlera donc plus de madame ***M*** ou de monsieur ***M'*** qui se connecte sur le site web d'un grand distributeur ***D***, pour acheter un produit que ***D*** obtiendra d'un fournisseur ***F***. Par contre, on s'intéressera au fait que ***M*** et ***M'*** jouent des rôles d'utilisateurs, que ***F*** est un fournisseur de service et que le rôle de ***D*** peut être assimilé à celui de médiateur.

Afin de modéliser les comportements typiques, nous nous sommes inspirés du modèle ALAADIN *(Gutknecht et Ferber 1998)* et *(Gutknecht 2001)* et nous avons défini l'ensemble de rôles mis en œuvre dans les services de communication multimédia en cherchant à définir ces services comme des schémas d'interaction.

3.1. Une métaphore pour le concept de rôle dans les services

Pour illustrer le concept de ***rôle*** et son application aux services de communication, nous allons utiliser une métaphore dans le domaine artistique. Une pièce de théâtre est une histoire qui se déroule au travers d'un scénario. Les différents acteurs de la pièce incarnent des personnages qui évoluent dans un décor. Chaque acteur suit les indications données par le script écrit par le scénariste et il est guidé sur la scène par le réalisateur.

Dans notre vision agent des services, nous considérons les agents comme des acteurs. Ils sont plongés dans un environnement multimédia dans lequel ils tiennent chacun un rôle défini. Les différents agents adoptent tous des comportements typiques prévus par le concepteur du service : on donc globalement un plan collectif. Le service lui-même est régi par les agents contrôlés qui agissent au niveau de la plate-forme. De la même façon que l'on classe les œuvres théâtrales avec des genres (drame, comédie, etc.), selon qu'ils respectent telle ou telle caractéristique, on définira des ***classes de service***. La correspondance que nous venons d'établir est résumée dans le Tableau 7.

Tableau 7 - Illustration du concept de rôle dans les services

Concepts agent	***Domaine du théâtre***
agents	acteurs de la pièce
classe de service	genre
comportement typique	script
concepteur de service	auteur, scénariste
environnement	acteurs, accessoires et décor
exécution du service	représentation de la pièce
le service	la pièce
plate-forme d'exécution	le théâtre
rôle	personnage
agents contrôlés	réalisateur
adaptation	improvisation

La représentation d'une pièce de théâtre ne se déroule jamais à l'identique. Le lieu, les dimensions de la scène et la position des objets interviennent, de même que les réactions du public. Parfois même, le

scénariste ne détaille pas certains passages d'une scène et/ou il laisse une liberté d'interprétation. Ainsi, l'acteur joue, non seulement en fonction du script, mais également en fonction du contexte. Ce sont ses capacités à improviser qui lui permettent de faire une bonne prestation. De la même façon, les agents intervenant dans un service peuvent avoir à s'adapter aux spécificités du contexte et aux aléas qui surviennent. Nous présenterons, dans les chapitres 4 et 5, les pistes que nous avons explorées pour concevoir un comportement adaptatif pour de tels agents.

Un aspect important dans le fonctionnement d'un système multi-agent social est l'adéquation des agents à leur rôle. Comme au théâtre, même si nous nous plaçons à un niveau d'abstraction où nous oublions qui sont les acteurs, au bénéfice du personnage qu'ils incarnent, les acteurs doivent avoir toutes les compétences physiques et intellectuelles requises pour jouer correctement. Il doit en être de même dans notre vision des services : un agent qui ***endosse*** un rôle doit satisfaire les conditions d'endossement. Ces conditions vont bien sûr porter sur les caractéristiques des agents à un niveau plus concret : capacité de traitement, connaissances, etc. D'un point de vue dynamique, si au cours d'une représentation, un problème survient, alors plusieurs possibilités s'offrent au réalisateur, il peut décider d'arrêter tout ou trouver tous les moyens pour continuer ("the show must go on"). C'est là que l'aspect de diagnostic intervient dans les services sous leur forme hSMA : afin de détecter tout glissement vers une situation anormale, tout dérapage vers un cas dangereux. Détecter ces dangers et les éviter, lors de l'exécution des services, est un thème que nous avons également prospecté et que nous aborderons dans le chapitre 5.

3.2. Rôles spécifiques dans les services de communication

Nous avons défini un ensemble ***R*** comportant quatre rôles génériques ***U***, ***S***, ***T*** et ***M*** à partir des services multimédia de Dialoca, mais aussi au cours de notre participation à d'autres projets, relevant de notre problématique, au sein de l'équipe Maia.

U

- Le rôle générique ***U*** correspond aux rôles utilisateurs d'un service. Par essence, tout service se doit de comporter au moins un rôle ***U*** pour être utile. Un rôle ***U*** peut, par exemple, avoir la charge d'une ou plusieurs fonctions parmi les suivantes :
  - ★ La consultation des informations du service,
  - ★ La fourniture d'informations au service,
  - ★ La demande d'assistance et
  - ★ L'administration du service.

  Bien souvent endossés par les agents humains, les rôles ***U*** peuvent s'instancier pour modéliser des clients, des médecins, des étudiants, mais aussi des administrateurs, des concepteurs, etc. L'endossement par un agent logiciel n'est cependant pas exclu.

S

- Le rôle générique ***S*** englobe les rôles de sources de données que l'on qualifiera d'actives. Il s'agit de rôles importants, mais non obligatoires, qui servent le plus souvent à alimenter le service en informations. Les fonctions et actions généralement attribuées aux rôles ***S*** sont :
  - ★ Le stockage d'informations à moyen-long terme,
  - ★ La spécialisation possible dans un sous-domaine de connaissances,
  - ★ Ils peuvent devoir transmettre des informations publiées et
  - ★ Ils peuvent devoir protéger des informations privées.

  Les rôles ***S*** sont par exemples tenus par des agents logiciels qui gèrent de serveurs web et des Systèmes de gestion de base de données (SGBD), plus ou moins actifs vis à vis du système. Ils peuvent aussi, plus rarement, être endossés par des agents experts humains qui sont consultés pour des besoins particuliers.

T

- Le rôle générique ***T*** recouvre les rôles de traitement et en particulier de traitement de l'information. A nouveau, ces rôles ne sont pas obligatoires, mais ils enrichissent les services par

leur présence en apportant des fonctionnalités avancées : traductions, analyse, génération d'alertes, déclenchement d'autres services, etc. Les fonctions et actions possibles des rôles ***T*** sont :

★ Des transformations de l'information à haut niveau,

★ Le raisonnement et l'inférence de nouvelles connaissances et

★ La réaction à l'information par des actions spécifiques.

Comme les rôles ***S***, les rôles ***T*** seront essentiellement joués par des agents logiciels (moteurs d'inférence, traducteurs automatiques, systèmes de diagnostic, etc.), mais on peut aussi confier des traitements spécifiques à des experts humains.

M

- Le rôle générique ***M*** désigne les rôles des médiateurs. Dans notre modèle ces médiateurs sont les clefs des services de communications et sont en conséquence incontournables. A la base, un service mettra donc en œuvre un médiateur, mais on peut fort bien concevoir une composition de services ou des interactions requérant plus d'un médiateur à l'image des groupes de coursiers (brokers) décrits dans *(Vercouter 2000)*. Les fonctions et actions que l'on peut attribuer à un rôle ***M*** sont :

  ★ La gestion des profils des utilisateurs,

  ★ La connaissance de qui est qui, qui est où et qui peut faire quoi,

  ★ L'habilité à rediriger les flux d'informations,

  ★ L'utilisation éventuelle de fonctions légères de stockage et de traitement d'information.

  Bien que le terme médiateur soit également utilisé dans les sociétés humaines et qu'il ait servi de source d'inspiration pour la définition des rôles ***M***, seule l'utilisation de médiateurs logiciels entre dans le cadre de ce modèle. Nous ne parlerons ici des médiateurs que sous la forme d'agents artificiels "intelligents" assurant la fourniture même du service.

3.3. Modélisation des services par des schémas d'interaction

Dans notre modèle, les services sont représentés, d'une part par les rôles qui sont mis en œuvre, et d'autre part par des schémas d'interactions entre ces rôles, sous la forme d'un ensemble de graphes. Nous avons préféré une vision étendue par rapport aux schémas proposés dans *(Winograd et Florès 1986)*. Nous représentons non seulement les liens d'interaction, mais également les transitions internes des comportements typiques de chaque rôle.

Nous définissons tout d'abord le comportement typique associé à un rôle ***r*** dans un service sous la forme d'un graphe ***(S, Γ)***, où

- ***S*** contient l'ensemble des états $\{s_0, \ldots, s_n\}$ possibles du comportement. Ces états correspondent à des abstractions des états réels des agents qui jouent ces rôles.
- Γ correspond à une transition du comportement entre deux états. Ces transitions font évoluer l'agent en réaction à un événement de perception ou au travers de l'accomplissement d'une action.

Un schéma d'interaction sera formé de plusieurs graphes de comportements $G_i = (S_i, \Gamma_i)$ reliés entre eux au niveau des états par des liens d'interaction où une transition $\alpha \in \Gamma_i$ peut provoquer une transition $\beta \in \Gamma_j$. Un tel lien d'interaction entre deux transitions, appartenant à des rôles différents, est une abstraction d'une séquence de communication plus complexe entre les agents qui jouent ces rôles. Par exemple, un simple lien (demander de l'aide → réception d'une requête) peut se produire par téléphone et sera beaucoup plus complexe qu'il ne paraît : il mettra en œuvre la gestion de la ligne téléphonique, l'acquisition du signal et la reconnaissance de parole, etc.

Le graphe Figure 21 montre un service conçu pour aider un touriste (à gauche) visitant une ville et faisant appel à un assistant (au milieu) qui gère la liste des questions les plus fréquemment posées (FAQ) et d'un guide touristique expert (à droite) qui peut répondre aux autres questions plus spécifiques, selon les besoins de l'assistant. Dans cet exemple, nous pouvons, de plus, associer un rôle ***U*** au touriste, un rôle ***M*** à l'assistant et un rôle ***S*** au guide touristique.

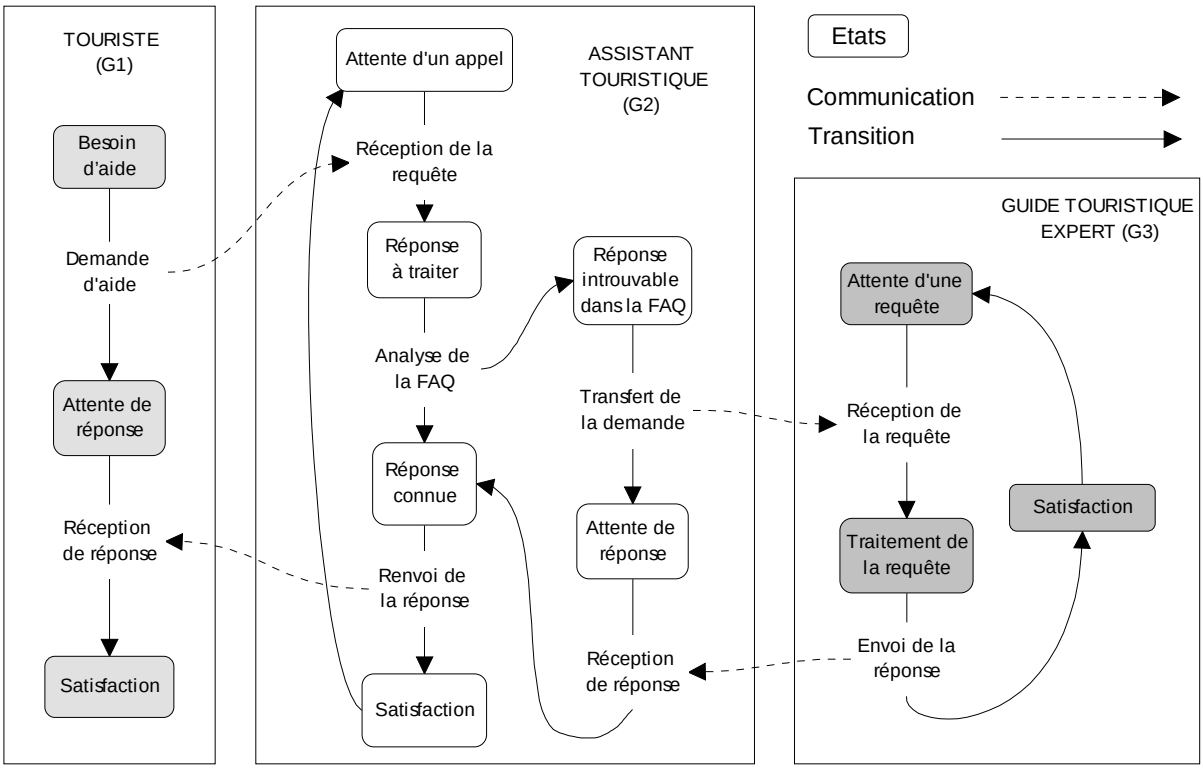


Figure 21 - Exemple de schéma d'interaction pour un service d'assistance touristique

3.4. Des services aux classes de services

Les schémas d'interactions détaillant la façon dont se déroulent les services. A plus haut niveau, nous proposons d'utiliser une modélisation organisationnelle regroupant tous les services selon un ensemble de classes données.

3.4.1. Définition

Nous définissons une **classe de service** comme un graphe ***(R, I)*** dans lequel

- ***R*** est l'ensemble des rôles génériques du service
- ***I*** est l'ensemble des liens d'interaction qui régulent les transmissions de messages entre les agents. La présence d'un arc entre deux rôles $a, b \in R$ indique que les rôles ***a*** et ***b*** interagissent dans cette classe de service.

Parmi toutes les structures d'organisation possibles pouvant être construites à partir des rôles génériques, nous présentons maintenant quatre classes de services que nous avons validées en modélisant, non seulement les services Dialoca, mais également d'autres applications comme le filtrage dans les projets européens IST Elin et Ozone ou la recherche de produits culturels dans les autres collaborations auxquelles nous avons pris part.

3.4.2. Service de pont de communication

Dans cette construction, illustrée Figure 22, le médiateur sert de pont pour permettre aux utilisateurs de dialoguer. Le service est initialement déclenché par la requête de l'un des utilisateurs au médiateur. Le médiateur essaye de contacter les autres utilisateurs pour leur délivrer des messages qui peuvent être par exemple traduits, transformés, etc. Par extension, s'il y a plus d'un médiateur, ceux-ci peuvent se peuvent retransmettre les informations. On peut alors obtenir des services permettant la communication entre les utilisateurs au travers de leurs assistants.

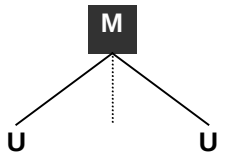


Figure 22 - Service de pont de communication

Un exemple de service pour cette classe est le service e-Nots de Dialoca. Dans l'approche proposée, les récepteurs, qui ont souscrit au service, délèguent l'usage au médiateur de la ressource de communication (ici, en donnant accès à leur boîte aux lettres électronique). Quand l'expéditeur essaye d'atteindre le receveur, la ressource déléguée transmet le message au médiateur. Ce dernier peut, par exemple, alors tenter de joindre le destinataire par un autre média pour le notifier. Dans ce service, l'expéditeur est un agent humain qui a souscrit un abonnement, mais aucune hypothèse particulière n'est faite sur le rôle de l'expéditeur. Le médiateur est bien sûr un agent interne.

3.4.3. Interface évoluée pour la navigation et la distribution

Cette construction, illustrée Figure 23, permet à un utilisateur d'accéder à une ou plusieurs sources de données, au travers d'un médiateur. Ce dernier traduit ses requêtes et les transmet à la source de données. Le médiateur peut aussi être contacté par les sources de données pour distribuer des informations aux utilisateurs. Cette construction permet ainsi de faire du push/pull d'informations, tout en cachant le nombre, la nature et donc la complexité des sources d'informations.

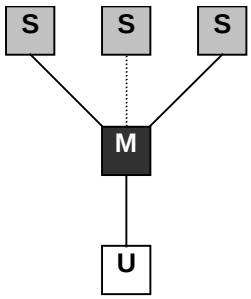


Figure 23 - Interface pour la distribution / navigation avec de multiples sources de données

Un exemple d'application est le service Dialoca TopTrades, où un agent joue le rôle d'un abonné et appelle son courtier virtuel pour obtenir des informations ou pour placer des ordres boursiers. Le courtier virtuel agit comme un médiateur et traduit l'information en provenance et à destination du site web TopTrades qui est un partenaire financier.

3.4.4. Support de meeting multi-source

Cette classe de service, schématisée Figure 24, permet à plusieurs utilisateurs d'accéder et d'utiliser diverses sources de données. Ces sources peuvent être distribuées géographiquement ou être spécialisées dans des domaines donnés. Le médiateur anime le meeting en dirigeant les flux média entre les utilisateurs ou entre les utilisateurs et les sources de données. Il peut aussi permettre une consultation partagée des sources de données, les utilisateurs effectuant des accès synchronisés. Les sources de données peuvent être communes à tous les utilisateurs, mais les utilisateurs peuvent avoir leurs propres données, gérées avec tous les aspects de sécurité requis. Les sources de données peuvent aussi être modifiées par certains utilisateurs et consultées par les autres, etc.

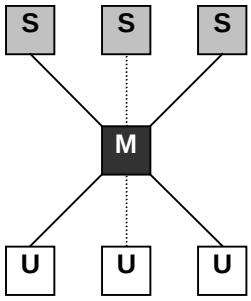


Figure 24 - Support de réunion multi-source

Deux applications illustrent cette classe de service :

- Un graphe de service limité a déjà été présenté Figure 21 pour l'assistance experte Dialoca qui a été utilisée a posteriori sur la démonstration Nancy-Tour. Dans ce service, un petit véhicule électrique, appelé Cycab, permet de découvrir le centre touristique d'une ville. En voyageant, les utilisateurs peuvent appeler un guide virtuel (le médiateur) et lui poser des questions sur divers sujets, comme l'histoire de la ville, les éléments culturels... ou l'adresse d'un bon restaurant. Le guide virtuel répond aux questions en utilisant une liste des questions les plus fréquemment posées (FAQ) et peut accéder à diverses bases de données. S'il ne trouve pas la réponse à une question, l'assistant peut faire appel à un guide touristique humain.
- Le service Dialoca RMS (Remote Meeting Service) permet à un ensemble de personnes de se rencontrer et d'échanger des informations en utilisant différents médias. L'un des participants initie le meeting, puis il demande à un animateur virtuel (un médiateur) de prendre en charge le déroulement de la réunion. Les utilisateurs transmettent les documents qu'ils souhaitent partager et celui-ci les distribue aux autres participants.

3.4.5. Agent de filtrage et de suivi

Cette dernière structure, illustrée Figure 25, regroupe les services pour lesquels l'ensemble des utilisateurs est divisé en deux. Le premier sous-ensemble d'utilisateurs fournit régulièrement une grande quantité de données aux sources de données. Les utilisateurs du second sous-ensemble sont susceptibles d'être intéressés par des parties spécifiques ou pré-traitées de cette masse de données. Ici encore, on fait l'hypothèse que le médiateur facilite l'alimentation des sources de données et la recherche d'informations. Un rôle additionnel peut aider le médiateur en réalisant des traitements spécifiques sur les informations comme un filtrage ou un diagnostic.

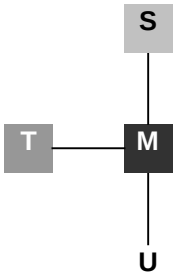


Figure 25 - Filtrage et suivi intelligent

Nous avons rencontré trois exemples d'applications qui entrent dans cette classe :

- Le projet européen IST Elin (Electronic Newspaper Initiative), qui vise à fournir un journal électronique contenant des informations adaptées à ses abonnés.

- Le projet européen IST Ozone, dont le but est de concevoir les environnements "d'intelligence ambiante" de demain.
- Diatélic *(Jeanpierre 2002)* et *(Bellot 2002)*, qui est une application de télé-médecine permettant le suivi de patients dialysés à domicile. Dans cette application, les patients remplissent des formulaires qui sont transmis aux sources de données. Un système de diagnostic aide les médecins à surveiller l'état des patients et peut générer des alarmes si existe un risque est détecté.

4. Conclusion

Dans ce chapitre, nous avons exposé notre approche théorique pour fournir des services au travers d'une coopération entre agents hétérogènes. Nous avons proposé une solution basée sur la médiation où la coordination des agents était assurée par l'ajout d'agents logiciels contrôlés appelés médiateurs.

Dans la suite de la thèse, nous allons tenter d'apporter des réponses à la question "comment conçoit-on un tel médiateur ?". Plus particulièrement, nous allons nous intéresser à deux questions :

- Quelles sont les approches pour créer un agent contrôlé qui produise un comportement de médiation ?
- Comment adapter le comportement du médiateur en fonction des agents non contrôlés avec lesquels il interagit ?

Dans les deux chapitres suivants, nous allons nous intéresser à la production du comportement des agents contrôlés par planification. Nous considérerons ensuite l'utilisation des modèles stochastiques, non seulement pour obtenir le comportement de l'agent contrôlé, mais également pour acquérir des connaissances sur les agents partiellement et non contrôlés, ce qui permettra une adaptation dynamique des services

Chapitre 4 - Vers la production de comportements des agents contrôlés

Dans ce chapitre, nous cherchons à répondre à la question "Comment définir le comportement d'un agent contrôlé ?". Dans le chapitre 1, nous avons décrit le comportement d'un agent comme une fonction qui détermine les actions que l'agent doit accomplir dans son environnement, en fonction de ses différentes perceptions.

A présent, nous avons deux besoins qui semblent s'opposer. D'une part, l'agent est ajouté dans le hSMA par le concepteur pour y tenir un rôle, avec un caractère contrôlé. D'autre part, les propriétés d'autonomie et de rationalité limitée de l'agent font que son comportement le pousse à effectuer des actions qui lui sont les plus profitables. Nous proposons de transformer cette opposition en une gradation allant du contrôle à l'autonomie. Le contrôle le plus fort impose que le concepteur spécifie le comportement à suivre, tel un automate, avec tout le travail d'horloger qui va de paire, impliquant correction, complétude, précision, etc. De façon plus souple, le concepteur peut rendre l'***agent pro-actif***, c'est-à-dire capable de poursuivre des buts qui pourront lui être demandés. L'agent doit être en mesure d'agir de façon à transformer le monde, pour que celui-ci devienne tel qu'il le souhaite, c'est-à-dire un monde où la tâche est accomplie. Pour cela, l'agent peut effectuer un raisonnement par ***planification*** qui va l'amener de ses buts à l'enchaînement des actions adéquates. Cependant, l'agent doit disposer des connaissances nécessaires sur la façon dont le monde va évoluer, selon ce qu'il va entreprendre. L'ensemble de ces connaissances que l'on désigne par le terme ***modèle***, n'est malheureusement pas toujours disponible.

Enfin, un contrôle plus léger est de motiver l'agent par un système de récompenses/pénalités. Dans ce cas de figure, l'agent se comporte de façon à optimiser un critère d'utilité. Cette optimisation est simplement un autre moyen d'envisager la réalisation d'une tâche, si ce n'est que les buts à atteindre et les dangers à éviter sont quantifiés. Avec cette approche, il est possible d'effectuer une planification du comportement si le modèle est connu, mais également d'effectuer un apprentissage s'il ne l'est pas.

Dans un premier temps, nous allons, examiner le principe du raisonnement par planification et les différentes approches existantes. Nous montrerons que les hypothèses effectuées sur l'agent et sur sa relation avec son environnement sont peu adaptées à notre problème. Dans la suite du chapitre, nous détaillerons les modèles de décision markoviens et l'apprentissage par renforcement que nous avons utilisés, car ils permettent de prendre en compte les aspects d'incertitude et de résistance aux aléas.

1. La planification

Le raisonnement par planification constitue l'un des grands thèmes de l'intelligence artificielle. Ce type de raisonnement permet de produire un comportement pour atteindre un but fixé, en utilisant des actions élémentaires comme briques de base. La planification est utilisée dans des domaines variés de résolution de problèmes, comme la recherche de chemins en robotique mobile, la génération

automatique de programmes, l'organisation de production, les jeux, la synthèse de molécules et, d'une façon générale, lors de la conception d'agents intelligents effectuant des tâches complexes.

1.1. Définition

La ***planification*** est un mécanisme de raisonnement qui consiste à élaborer des séquences ordonnées d'actions, appelées ***plans***, dont l'exécution doit permettre de faire évoluer le monde considéré, de façon à atteindre un ***but*** donné ou à réaliser une tâche complexe.

1.2. Un problème typique de planification

Un problème typique, illustré Figure 26, est celui du singe et des bananes. On place un singe dans une salle vide, excepté la présence d'une caisse et d'une banane accrochée en hauteur. Le singe veut manger la banane, et pour cela, il peut pousser ou monter sur la caisse, il peut bien sûr se déplacer et il peut enfin attraper la banane, si elle est à sa portée. A ces actions concrètes pouvant être effectuées, la planification associe le terme d'***opérateur*** qui va servir de brique de base abstraite pour créer des plans.

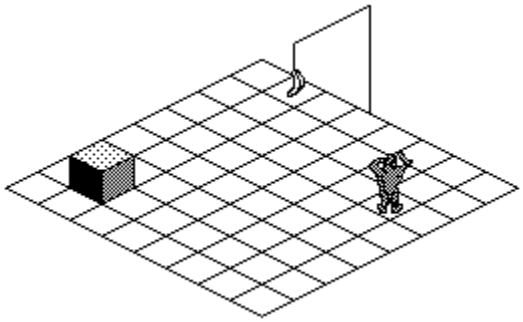


Figure 26 - Le problème du singe et des bananes

1.3. Les concepts utilisés

Nous donnons ci-dessous un ensemble de concepts qui sont utilisés dans le domaine de la planification.

Domaine : Un domaine est la description des divers objets existants, mais aussi des relations qu'ils ont entre eux et de l'ensemble des opérateurs utilisables. Il s'agit d'un cadre précis dans lequel peuvent se poser un ou plusieurs problèmes de planification.

Etat : Un état est une sorte de représentation instantanée décrivant partiellement ce qui est vrai dans le monde. Dans la description d'un état, seuls les éléments auxquels on s'intéresse et les relations qui les relient sont considérés.

Problème : Un problème est défini relativement à un domaine et spécifie l'état initial et l'état but.

But : Un but est un état du monde que l'on souhaite atteindre. Le but est donné au planificateur, contrairement aux sous-buts que celui-ci peut éventuellement se fixer pendant la résolution du problème.

Opérateur : Un opérateur est une brique de base permettant de construire des plans. Plus précisément, c'est un élément atomique de planification représentant une action qui peut être effectuée sur l'environnement. L'application d'un opérateur permet de faire évoluer le monde considéré d'un état à un autre.

Plan : Un plan est une séquence ordonnée d'opérateurs instanciés permettant de résoudre un problème. Il s'agit de la suite des actions à réaliser pour atteindre le but depuis l'état initial.

Planificateur : Un planificateur est un programme informatique qui prend en entrée un domaine de travail et un problème à résoudre dans ce domaine. Il retourne en résultat un plan, s'il existe, permettant de résoudre le problème.

Nous allons donner un aperçu des différentes méthodes de planification en débutant par la planification de type GPS/STRIPS que nous qualifierons de classique, puis nous décrirons d'autres approches plus spécifiques. Un état de l'art plus détaillé pourra être trouvé dans *(Lagoudakis 1996)*.

1.4. L'approche classique de la planification

La solution classique pour aborder la planification est d'utiliser une représentation logique à l'aide du calcul des prédicats du premier ordre. Beaucoup de planificateurs sont issus des travaux sur les systèmes de preuve de théorèmes et des recherches sur la synthèse automatique de programmes. Les premiers planificateurs sont nés avec le General Problem Solver, GPS *(Newell et Simon 1961)*, puis le STanford Research Institute Problem Solver, STRIPS *(Fikes et Nilsson 1971)*. Depuis, cette famille s'est beaucoup agrandie et ramifiée, comme on peut s'en apercevoir dans les compétitions de planification comme AIPS (Artificial Intelligence Planning and Scheduling Systems).

On peut, tout d'abord, citer la lignée de GraphPlan *(Blum et Furst 1997)*, qui effectue une planification très efficace à base de graphes et dont le principe a été réutilisé à plusieurs reprises comme dans STAN *(Long et Fox 1998)*. Cette méthode a été aussi combinée avec des approches de diagrammes de décision binaire (BDD) comme dans IPP *(Kohler et al. 1997)*, MIPS *(Edelkamp et Helmert 2000)* ou PropPlan *(Fourman 2000)*. Une seconde lignée de planificateurs a exploité l'idée de transformer un problème de planification en problème de satisfiabilité. C'est le cas de BlackBox *(Kautz et Selman 1998)*, qui se ramène à un problème de satisfiabilité booléenne (SAT), mais aussi d'autres planificateurs, comme GP-CSP *(M. Do et S. Kambhampati 2001)*, qui lui utilise une approche de satisfaction de contraintes (CSP). D'autres voies ont également été explorées avec l'utilisation d'heuristiques très efficaces pour guider la recherche, ce qui a notamment donné HSP *(Bonet et Geffner 1998)* et FF *(Hoffmann et Nebel 2001)*.

1.4.1. L'analyse des fins et des moyens

Le précurseur était GPS qui entendait résoudre n'importe quel problème, à condition que celui-ci puisse être correctement représenté. L'idée de Newel et Simon consistait à formaliser l'univers considéré en identifiant, d'une part, les objets qui le constituent et, d'autre part, des opérateurs qui transforment un objet en un autre, puis d'utiliser l'***analyse des fins et des moyens*** (Means-Ends Analysis : MEA), déjà énoncée chez Aristote :

Nous raisonnons non sur les fins elles-mêmes, mais sur les moyens de les atteindre. Un médecin, ne se demande pas s'il guérit son malade, ni un orateur s'il réussit à convaincre, ni un homme politique s'il fait régner la loi et l'ordre, et personne d'autre ne pense sur les fins à atteindre. On suppose la fin, puis on envisage comment et par quels moyens elle peut être atteinte; et si elle semble être produite par plusieurs moyens, on choisit celui par lequel elle est le plus simplement et le mieux produite, alors que si elle est réalisée par un seul moyen, on considère comment elle est atteinte par ce moyen, et ce moyen à son tour par quel moyen il peut l'être, jusqu'à ce que l'on arrive à la première cause, qui dans l'ordre de découverte est la dernière […] ce qui est dernier dans l'ordre d'analyse apparaît être le premier dans l'ordre de ce qui se produit. (Aristote -328)

Le principe de la MEA est de raisonner, en arrière depuis les buts en considérant les actions qui réduisent la différence entre la situation courante et le but. Les séquences d'actions obtenues, une fois inversées, donnent la façon d'atteindre ces buts. On commence donc par décrire le problème en spécifiant la situation initiale et la situation finale. Ensuite, on le résout en cherchant à réduire les différences détectées entre les objets considérés, en appliquant les opérateurs nécessaires.

1.4.2. Modélisation

Dans la formulation logique utilisée, les objets du domaine considéré sont représentés par des constantes. Pour le problème du singe et des bananes, on peut définir des constantes pour conceptualiser le singe, la banane, la caisse, etc. Vient ensuite la modélisation des différentes relations entre les objets à l'aide de prédicats avec, dans le cas de la banane, le fait qu'elle puisse être attrapée et mangée. La représentation d'un état se fait à l'aide d'un modèle logique pour lequel on indique, à l'aide de formules bien formées, quels sont les faits considérés comme vrais. Un opérateur de planification est composé de quatre parties :

- un profil avec son nom et une liste de paramètres permettant d'instancier l'opérateur,
- une liste de pré-conditions, qui doivent être vérifiées sur l'état courant pour que l'opérateur puisse être appliqué et
- une liste d'effets, permettant de modifier l'état courant pour obtenir le nouvel état. On y retrouve essentiellement des ajouts et des suppressions de faits.

Ces deux dernières listes sont également données à l'aide de formules logiques bien formées.

1.4.3. Les principales difficultés de la modélisation

La formulation des problèmes n'est jamais évidente et, bien souvent, il faut corriger la modélisation que l'on a construit, car dans le cas du singe et des bananes, on peut se retrouver dans une situation où le singe monterait d'abord sur la caisse puis essayerait de la pousser. On peut d'ailleurs citer trois problèmes typiques auxquels on a très vite affaire *(Bridge 2000)* :

- Le ***problème de qualification***, qui porte sur la difficulté de définir les pré-conditions d'un opérateur pour garantir le succès d'une action : pour porter un objet, il ne doit pas être trop lourd, ni être fixé par ailleurs, etc. Le risque, si on oublie un cas, est d'obtenir un échec lors de l'exécution du plan.
- Le ***problème de ramification***, qui porte sur la difficulté de définir les effets d'une action et surtout ses conséquences implicites. Un exemple classique est que le déplacement d'un conteneur déplace également son contenu.
- Le "***frame problem***", qui souligne le manque d'une spécification correcte des opérateurs. En effet, dans la planification de type STRIPS, on ne décrit que ce qui change et le reste est implicitement inchangé, alors que l'on devrait spécifier non seulement ce qui change, mais aussi ce qui ne change pas.

De plus, nous raisonnons souvent en attachant à une même action plusieurs sémantiques différentes et donc autant de façons de réaliser cette action : on peut par exemple pousser une caisse pour la faire glisser ou pour la faire tomber. Cette distinction n'est pas évidente pour le planificateur, c'est pourquoi, lorsque l'on écrit la liste des opérateurs, il faut en prévoir autant que nécessaire.

1.4.4. Déroulement de la planification

Afin de construire un plan, un planificateur de type STRIPS utilise l'approche MEA et effectue une recherche dans l'espace des états. Il utilise une pile qui lui permet de gérer les buts qu'il doit résoudre. Au début de la planification, cette pile est initialisée avec le but donné par l'utilisateur. Ensuite, le planificateur part dans une boucle de traitement de la pile où s'alternent deux types d'activités :

- ***Régression*** : Si le sommet de pile est un but simple, il va chercher à vérifier si ce but est satisfait. Pour cela, il met en œuvre un prouveur de théorèmes à base de résolution qui puise ses axiomes dans le modèle correspondant à l'état courant. Si le but est vrai, alors il est dépilé et l'opérateur correspondant est appliqué. Si le but est faux, le planificateur recherche alors un opérateur dont la liste d'ajouts spécifie des clauses qui permettraient à la preuve d'être poursuivie avec succès. En fait, il recherche un opérateur qui réduit au mieux la différence entre l'état courant du monde et l'état but. Une fois cet opérateur pertinent trouvé, le planificateur utilise ses pré-conditions pour créer de nouveaux sous-buts, puis il les ajoute dans la pile.

- ***Décomposition*** : S'il y a une conjonction en sommet de pile, alors le planificateur décompose la conjonction, puis il empile les parties fausses de la conjonction.

La sortie du planificateur est une liste d'opérateurs instanciés correspondant aux actions à effectuer pour atteindre le but.

1.4.5. Mise en oeuvre

Compte tenu du temps de calcul d'un plan, le procédé le plus classique (dite planification fermée) consiste à produire le plan lors d'une phase de conception, puis de l'utiliser dans une phase d'exécution. Cette méthode est illustrée Figure 27. A gauche, le planificateur reçoit le problème à traiter et prépare le plan. A droite, le plan est exécuté par l'agent afin de modifier son environnement. A ce niveau, une remarque importante est que la remontée des perceptions de l'environnement vers l'agent n'est pas prise en compte. Il s'agit d'une des limites de la planification classique : l'agent n'est absolument pas sensible aux changements de l'environnement. Il se contente d'exécuter un plan préétabli, sans jamais le remettre en cause.

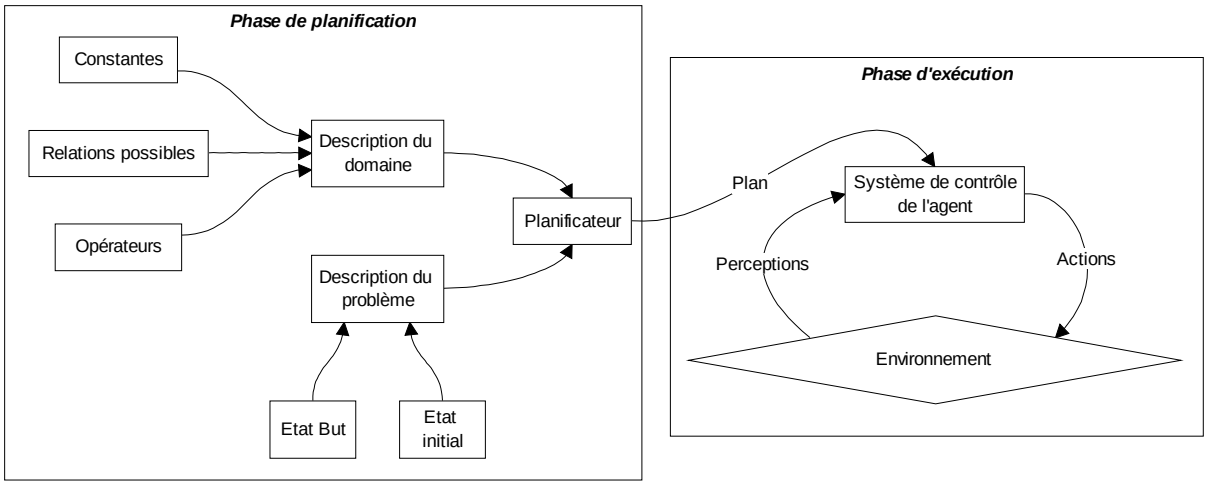


Figure 27 - Les phases de la planification classique

1.4.6. Un exemple PDDL

En pratique, afin de décrire les problèmes et les domaines, nous avons utilisé le langage PDDL (pour Planning Domain Description Language) *(AIPS 1998, Fox et Long 2001)* qui avait été défini à l'origine pour la compétition AIPS et qui est, par conséquent, supporté par un grand nombre de systèmes de planification. PDDL couvre un large spectre en terme d'expressivité (effets conditionnels, quantifications universelles, etc.), avec une conception modulaire basée sur un système de pré-requis. L'auteur d'un problème peut indiquer explicitement si une fonctionnalité donnée est estimée nécessaire. Ceci permet au planificateur utilisé de juger de sa capacité à résoudre le problème. En dehors de cette indication, une description PDDL est intentionnellement neutre, ce qui en fait un standard indépendant :

PDDL est destiné à exprimer la physique d'un domaine, c'est-à-dire quels sont les prédicats existants, quelles sont les actions possibles, quelle est la structure des compositions d'actions et quels sont les effets des actions. [...] Nous avons tenté de ne fournir aucun "conseil" dans la notation PDDL, ce qui explique la neutralité de la notation à de nombreux endroits.

Nous donnons, ci-après, une définition PDDL pour le domaine du singe et des bananes. En début de définition, on trouve la définition des prédicats qui permettent de construire les formules logiques bien formées. Ensuite, on a une liste d'opérateurs avec les pré-conditions qui doivent être vérifiées et les effets qu'ils ont sur l'état du monde lors de leur application. Ces effets sont généralement des ajouts de faits, mais si une négation "not" est utilisée, alors il s'agit de la suppression de fait.

```
(define (domain singebanane)
  (:requirements :strips)

  (:predicates     (singe ?s) (position ?p) (estEn ?obj ?pos) (objet ?obj)
                   (enHauteur ?obj) (estSur ?o ?o) (aMange ?s ?o)
                   (peutMonterSur ?s ?o) (peutManger ?s ?o) (peutPousser ?s ?o)
                   (peutPrendre ?s ?o) (transporte ?s ?o))

(:action seDeplacer  :parameters (?s ?p1 ?p2)
  :precondition (and  (singe ?s) (position ?p1) (position ?p2)
                      (not (enHauteur ?s)) (estEn ?s ?p1) )
  :effect (and (estEn ?s ?p2) (not (estEn ?s ?p1))))

(:action pousser  :parameters (?s ?o ?p1 ?p2)
  :precondition (and  (singe ?s) (position ?p1) (position ?p2) (objet ?o)
                      (estEn ?s ?p1) (estEn ?o ?p1 ) (not (enHauteur ?s))
                      (peutPousser ?s ?o))
  :effect (and (estEn ?s ?p2) (not (estEn ?s ?p1))
               (estEn ?o ?p2) (not (estEn ?o ?p1))))

(:action monterSur  :parameters (?s ?o ?p)
  :precondition (and  (singe ?s) (position ?p)  (objet ?o) (estEn ?s ?p)  (estEn ?o ?p)
                      (peutMonterSur ?s ?o) (not (estSur ?s ?o)))
  :effect (and (estSur ?s ?o) (enHauteur ?s)))

(:action descendreDe  :parameters (?s ?o ?p)
  :precondition (and  (singe ?s) (position ?p) (objet ?o) (estEn ?s ?p)  (estEn ?o ?p)
                      (peutMonterSur ?s ?o)(estSur ?s ?o))
  :effect (and (not (estSur ?s ?o)) (not (enHauteur ?s)))
)

(:action prendreEnHauteur  :parameters (?s ?o ?p)
  :precondition (and  (singe ?s) (position ?p) (objet ?o)(estEn ?s ?p)  (estEn ?o ?p)
                 (not (transporte ?s ?o)) (enHauteur ?s) (enHauteur ?o) )
  :effect (and (transporte ?s ?o)))

(:action manger  :parameters (?s ?o)
  :precondition (and  (singe ?s) (objet ?o) (transporte ?s ?o) (peutManger ?s ?o) )
  :effect (and (aMange ?s ?o))))
```

Le problème à résoudre peut être défini avec un formalisme similaire. On trouve en premier les constantes permettant de construire des faits, puis l'état initial et l'état final.

```
(define (problem simpleseb)
        (:domain singebanane)
        (:objects cheetah banane caisse pos1 pos2 pos3)
       (:init (position pos1)(position pos2)(position pos3)
              (singe cheetah) (estEn cheetah pos1)
              (objet banane) (estEn banane pos3) (enHauteur banane)
              (peutManger cheetah banane)  (peutPrendre cheetah banane)
              (objet caisse) (estEn caisse pos2)
              (peutMonterSur cheetah caisse) (peutPousser Cheetah caisse)
       )

       (:goal (and (aMange cheetah banane) ) )
 )
```

Pour effectuer la planification, nous avons choisi le planificateur IPP qui a servi de référence pour la compétition AIPS-2000. Le plan que nous obtenons prend alors la forme d'une liste d'opérateurs instanciés :

```
seDeplacer (cheetah, pos1, pos2)
pousser (cheetah, caisse, pos2, pos3)
monterSur (cheetah, caisse, pos3)
prendreEnHauteur (cheetah, banane, pos3)
manger (cheetah, banane)
```

1.4.7. Adéquation à nos besoins

D'une façon générale, la planification classique se base sur des hypothèses très restrictives :

- Le planificateur est un simple agent causal qui a un contrôle complet et exclusif de l'environnement,
- Le but est parfaitement défini et demeure constant pendant la phase de planification,
- Le planificateur a toute la connaissance requise sur l'état initial,
- Le planificateur est capable de modéliser son environnement aussi précisément que nécessaire, et
- Le planificateur dispose des ressources nécessaires (en temps, mémoire, etc.) pour utiliser ce modèle et pour raisonner sur les mondes possibles reliés aux différentes alternatives qui peuvent se produire.

Il faut également rappeler que les planificateurs classiques sont insensibles aux évolutions de leur environnement.

Dans notre modèle, utiliser la planification classique pour définir le comportement des agents contrôlés consiste à définir leurs rôles en leur donnant des buts à atteindre et à placer un module de planification dans le "cerveau" des agents pour qu'ils soient capables de construire des plans d'action. Dans l'annexe I, nous détaillons un exemple de service que nous avons modélisé avec une approche de planification classique. Les conclusions auxquelles nous aboutissons montrent que nous devons aller plus loin, car bon nombre des hypothèses énoncées ci-dessus ne peuvent être vérifiées :

- Nous considérons un système multi-agent et donc les effets des actions des autres agents doivent être pris en compte. Même s'il est possible de travailler avec un comportement typique des agents avec lesquels le médiateur interagit, leur comportement réel doit être modélisé plus finement, afin de prendre en compte toutes leurs variabilités,
- Les agents que nous considérons ont seulement des perceptions limitées et donc une connaissance partielle de leur environnement,
- Ces agents sont autonomes et donc, ils doivent gérer les ressources dont ils disposent. Ils n'ont, entre autres, pas tout leur temps pour planifier,
- Enfin, des évènements imprévus peuvent survenir et, de plus, les résultats mêmes des actions qu'un agent contrôlé peut entreprendre sont incertains.

1.5. Au-delà de la planification classique

De nombreux travaux ont été réalisés afin d'étendre les capacités des planificateurs et de dépasser les hypothèses de départ. Cependant, la planification réelle, dite "***Practical Planning***" ou "***Real Planning***" est très complexe et reste un champ de recherche très actif en intelligence artificielle. La plupart des approches ont visé l'ouverture de la planification, c'est-à-dire l'extension, voire le déplacement complet du raisonnement vers la phase d'exécution et ceci, afin d'obtenir plus de réactivité en prenant en compte les perceptions sur l'environnement. Nous allons maintenant présenter un aperçu des différentes voies qui se sont offertes à nous et, dans la seconde partie du chapitre, nous détaillerons celle que nous avons retenue : la modélisation stochastique couplée à un apprentissage par renforcement.

1.5.1. La planification non-linéaire et les plans partiellement ordonnés

Au cours de la planification, on est confronté à plusieurs types de décision :

- Quelle est la nature de la décomposition du but que l'on utilise pour résoudre le problème ?
- Comment choisit-on l'ordre de résolution des sous-problèmes que l'on a identifié ?
- Comment choisit-on les objets du domaine qui seront mis en correspondance pour l'action que l'on veut effectuer ?
- Quel est l'ordre final dans lequel les actions du plan seront effectuées ?

En ce qui concerne l'ordre des actions il faut distinguer deux phénomènes et ceci est souvent source de confusion, comme le soulignent *(Russell et Norvig 1995)*. Le premier phénomène est l'hypothèse de ***linéarité*** des plans :

Un but donné peut se décomposer en sous-buts, un sous-plan peut être construit pour chacun de ces buts et le plan final peut être formé en combinant séquentiellement chacun de ces sous-plans. (Lagoudakis 96)

Les premiers planificateurs étaient linéaires, car ils prenaient les buts dans un certain ordre, les résolvant les uns après les autres et mettaient bout à bout les sous plans obtenus. Hélas, beaucoup de problèmes ne se prêtent pas à cette décomposition et les planificateurs linéaires se heurtent à des difficultés, comme l'anomalie de Sussman *(Sussman 1975)*, où l'on peut avoir à détruire quelque-chose qui est déjà construit. La planification non-linéaire est alors devenue possible avec l'entrelacement des opérateurs appartenant à différents sous-plans et avec la protection des sous-buts.

Le second phénomène porte davantage sur l'ordre même des actions. En effet, lorsque nous mettons des chaussettes puis des chaussures, peu importe si on commence par mettre la chausette gauche ou la chaussette droite. Par contre, ce qui est important, c'est de mettre à chaque fois une chaussette avant de mettre la chaussure au pied correspondant. Le principe, expliqué dans *(Weld 94)*, consiste à ne fixer aucune action à moins d'y être forcé, car une décision prise trop tôt peut devoir être rétractée pour permettre une autre action. Cette idée de différer les choix non nécessaires le plus tard possible (***Least Commitment Planning***) est utilisée dans les planificateurs à ordre partiel (POP), comme NOAH *(Sacerdoti 1975)*, O-Plan *(Currie et Tate 1991)*, etc. Elle est également gérée dans IPP, ce qui explique que pour les plans de l'annexe I, certaines actions puissent être effectuées en parallèle. La résolution du problème implique l'utilisation de liens de dépendance causale entre des étapes du plan (quand une action rend vrai un fait nécessaire pour une autre) et de contraintes d'ordre (quand une action doit être absoluement effectuée avant une autre). Un plan est alors construit progressivement par ajout successif d'actions et lorsqu'une action risque de remettre en cause une dépendance, elle est soit rétrogradée (placée avant les deux actions liées), soit promue (placée après). La Figure 28 illustre la différence entre les ***plans partiellement ordonnés***, dans lequel certaines actions se déroulent dans un ordre indépendant l'une de l'autre, et les plans totalement ordonnés où tout est figé. Par ailleurs il faut noter qu'un plan partiellement ordonné peut être, au besoin, transformé en un ou plusieurs plans totalement ordonnés.

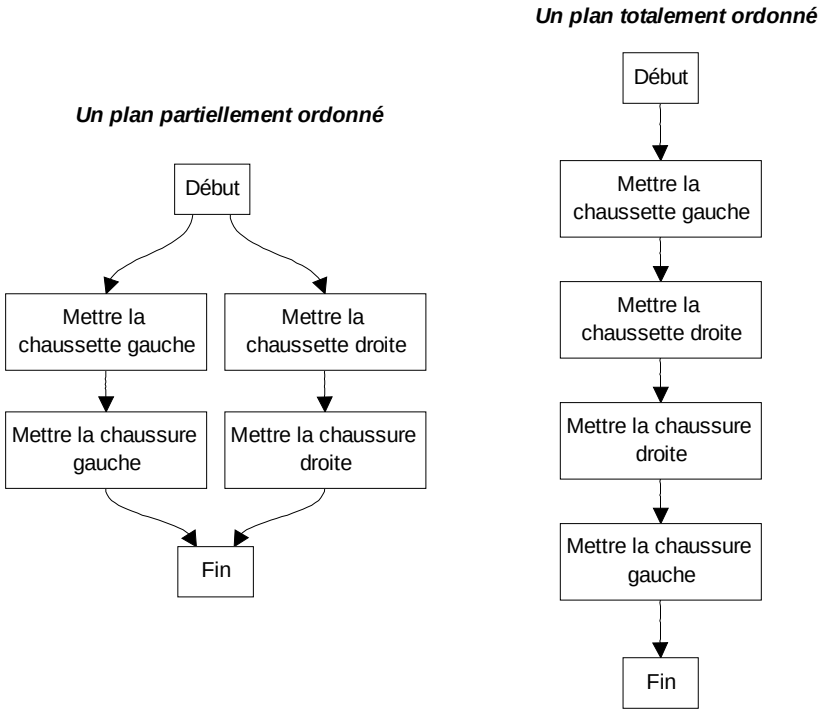


Figure 28 - Plans partiellement et totalement ordonnés

1.5.2. La planification hiérarchique

En examinant les données (domaine et problème) et les résultats d'une planification dans un cas pratique, on se rend compte que la représentation et le contrôle deviennent très vite complexes. Malgré toutes les heuristiques et les façons d'aborder la construction du plan, s'attaquer à des problèmes réels mène souvent à une explosion combinatoire. Face à ce problème, l'idée est venue aux chercheurs de différencier ce qui est important du détail, afin de simplifier les représentations et de là sont apparues les approches par niveaux d'abstraction. Si nous considérons à nouveau le problème du singe, finalement, le fait d'aller chercher la banane n'est qu'une petite partie de tout son comportement, donc un sous problème dans un comportement plus important qui a été complètement ignoré, puisqu'il ne nous intéressait pas. A l'inverse, nous n'avons pas été très précis dans la description des opérateurs, comme *manger*, car cela implique que le singe épluche la banane puis qu'il la porte à sa bouche pour la mastiquer et l'ingérer. Ceci illustre bien que pour planifier, il est possible de se placer à divers niveaux, plus ou moins précis en terme de description du monde, comme en terme de granularité des opérateurs (qui deviennent assimilables à des plans).

Dans ABSTRIPS *(Sacerdoti 1974)*, le concepteur a eu l'idée de planifier dans un haut niveau, puis de descendre avec une décomposition en sous-problèmes dont la résolution contribue à celle du problème général. Il s'est appuyé sur STRIPS et a défini une hiérarchie d'abstraction pour représenter le domaine d'un problème. Pour ce faire, il joue d'une part sur le niveau de détail des pré-conditions des opérateurs et il utilise une valeur de criticité pour déterminer à quel niveau de détail appartient une partie de la pré-condition. Ainsi, avec cette valeur, seuls les éléments critiques apparaissent dans les niveaux d'abstraction les plus élevés. Pour le contrôle, il a adopté une approche récursive avec retour arrière :

En résumé, la planification hiérarchique dans des espaces d'abstraction utilise une technique de recherche en "profondeur d'abord" et diffère l'extension de l'arbre de recherche vers les niveaux concernés par les pré-conditions détaillées d'un opérateur, jusqu'à ce que l'on sache que cela sera le plus efficace pour atteindre le but (car l'opérateur est sur un chemin menant certainement à une réussite). En évitant de travailler sur les branches infructueuses de l'arbre, la technique obtient une efficacité significative dans la formulation des plans complexes. (Sacerdoti 1974)

D'autres travaux, comme ceux de *(Wilkins 1984)* sur SIPE, ont également porté sur l'utilisation d'opérateurs avec différents niveaux de détail et on peut par ailleurs citer l'approche de la méta-planification proposée par *(Stefik 1981)* dans MOLGEN qui s'intéresse non seulement à des aspects de spécialisation, mais également à raisonner sur le processus même de planification.

1.5.3. La planification conditionnelle

Jusqu'à présent, les procédés que nous avons présentés travaillaient en deux phases (conception puis exécution) et ne prêtaient pas attention à ce qui pouvait se produire pendant l'exécution. Avec la planification conditionnelle, aussi appelée planification contingente, il est possible de construire des plans dont l'exécution varie selon des observations effectuées pendant leur exécution. Parmi les planificateurs conditionnels, on peut, entre autres, citer Warplan-C *(Warren 1976)* qui produit des plans linéaires et CNLP *(Peot et Smith 1992)* qui gère la non-linéarité.

Pendant la phase de planification, un planificateur conditionnel considère un petit nombre de sources d'incertitudes et prépare un plan qui gère les différents cas possibles. Ainsi, pendant la phase d'exécution, il peut faire face aux cas prévus à la conception. Dans *(Peot et Smith 1992)*, les auteurs illustrent cette idée en disant que le planificateur peut prévoir, par exemple, de prendre une clef alors que ce n'est pas strictement nécessaire, mais juste au cas où la porte correspondante serait fermée. Le principe de fonctionnement d'un planificateur conditionnel est de prendre en compte les combinaisons de valeurs de vérité possibles pour une formule P. Pour CNLP, trois possibilités sont envisagées : P, $\neg$P et unk(P) qui indiquent respectivement que P est vraie, fausse ou que sa valeur est inconnue. On peut alors ajouter des actions spécifiques et explicites d'observation dont le but est de déterminer quelle est la valeur d'une inconnue et ainsi de compléter le modèle du monde.

Au cœur même de la planification pour CNLP, on retrouve naturellement les liens causaux et les contraintes d'ordre, mais également des liens conditionnels entre les actions qui dépendent de la valeur observée. Ainsi, une fois prêt à l'emploi, le plan conditionnel permet une exécution efficace et plus

souple, cependant le plan lui-même devient très gros, surtout si les différentes branches correspondant à tous les cas possibles ne sont pas fusionnées par la suite.

1.5.4. La planification réactive

La planification conditionnelle n'est pas la seule possibilité pour la prise en compte des évènements survenant pendant l'exécution. La planification réactive *(Georgeff et Lansky 1987)* offre en effet un entrelacement plus fort entre la planification et l'exécution. Ceci est dû au fait qu'aucun plan n'est construit à l'avance et que l'on ne choisit que l'action suivante à exécuter. Dans le cas des plans universels *(Schoppers 1987)*, on a une phase de construction dans laquelle, comme pour les autres méthodes, on donne les conditions initiales et un but. Par contre, au lieu de produire un plan imposant avec des branches conditionnelles, on obtient un ensemble de règles condition-action. De plus, l'auteur indique que les plans sont traduits et gérés par un arbre de décision, ce qui permet d'optimiser la sélection de l'opérateur à appliquer. On peut également citer les travaux de *(Drummond 1989)* sur les règles de contrôle situées (SCR). La planification réactive est plus souple que la planification conditionnelle, car elle permet de gérer les évènements extérieurs (dits exogènes) ou l'incertitude sur les effets des actions. Il est en effet possible d'avoir une règle réactive pour chaque situation rencontrée, si tant est qu'elle puisse être envisagée. La planification réactive souffre cependant du temps mis pour évaluer les conditions à chaque cycle, afin de sélectionner la règle adéquate.

1.5.5. La planification sous contraintes de ressources

La planification est souvent utilisée pour piloter des systèmes réels, comme un procédé industriel ou un robot mobile. Dans ces applications, la notion de ressource (temps, argent, etc.) joue un rôle important et les contraintes qui sont imposées par ces ressources peuvent intervenir à différents niveaux. Le facteur temps est sûrement la ressource qui est la plus fréquemment prise en compte : on parle de temps contraint ou encore, si cela devient une ressource critique, de temps réel. Ensuite, viennent les autres facteurs comme les unités de calcul ou d'exécution, l'énergie disponible, etc.

Dans un premier temps, il faut distinguer, d'une part ce qui concerne la gestion des ressources pendant l'exécution et d'autre part, ce qui concerne les contraintes sur le processus de planification, même si, dans un second temps, on peut être amené à gérer ces aspects simultanément quand planification et exécution s'entrelacent.

En ce qui concerne la gestion des ressources utilisées pendant l'exécution, il est possible de les modéliser explicitement au niveau de la représentation du monde et dans ce cas, le principe va être le même que pour la planification classique, sauf que l'on ajoutera des calculs et des tests sur des valeurs de mesures. Ainsi, on introduira des opérateurs capables de prendre en compte ces mesures. *(Russel et Norvig 95)* donnent l'exemple d'un opérateur pour remplir un réservoir qui prend en compte le temps et la somme d'argent dépensés. En traduisant pour un plein de 55 litres et en comptant 3 secondes pour verser un litre et 3 minutes de manipulations diverses, on obtient :

```
Operateur (
  ACTION:  RemplirReservoir(QuantiteCarburant),
  EFFET:   NiveauCarburant ← Litres(55)
           Argent ← Argent - (PrixLitre(Carburant)*(Litres(55)-QuantiteCarburant))
           Temps ← Temps + Minutes(3)+(Secondes(3)/Litres(1))*(Litres(55)- QuantiteCarburant)
)
```

Pour une gestion fine de la ressource temps, il existe d'autres approches comme EXCALIBUR, décrit dans *(Nareyek 1998)*, qui s'intéresse à la planification de tâches en temps réel avec l'utilisation de valeurs issues de capteurs.

Le problème est différent lorsque c'est le processus même de planification qui dispose de ressources limitées. C'est le cas, si on considère un planificateur qui dispose d'une durée de 10 secondes pour calculer un plan. Les différents travaux sur ce domaine ont conduit les chercheurs à la conception d'algorithmes de recherche pouvant être interrompus à n'importe quel moment et fournissant la meilleure solution obtenue pour le temps imparti. Un exemple est la planification Anytime *(Charpillet et al. 1998)* que l'on retrouve dans le modèle PROGRESS *(Charpillet et Boyer 1997)*.

1.5.6. La planification multi-agent

La planification multi-agent s'intéresse à produire les comportements de plusieurs agents qui soient coordonnés de façon à résoudre ensemble un problème. De nombreux systèmes de raisonnement distribué ont été conçus, comme ATOME *(Laasri et al. 1988)*, PHOENIX *(Cohen et al. 1989)*, PGP *(Durfee et Lesser 1991)*, etc. Dans le Tableau 8, on retrouve les différents champs de recherche selon qu'il s'agit d'une distribution de la planification et/ou de l'exécution.

Tableau 8 - Types de planification multi-agent

	Plans et exécution centralisés	Plans et exécution distribués
Planification centralisée	Ce cas correspond à la planification classique mono-agent de type STRIPS que nous avons présentée précédemment.	Dans ce cas, il s'agit d'une planification multi-agent centralisée. Les actions des agents sont coordonnées en un point unique, ce qui facilite la résolution d'éventuels conflits. Un agent coordinateur crée des sous-plans avec des actions de synchronisation et les distribue aux autres agents, puis il assure le suivi de l'exécution.
Planification distribuée	Dans ce cas, c'est le processus de planification qui est distribué. Cet aspect relève donc plutôt de la parallélisation de l'IA. Par ailleurs, un certain nombre de problèmes surviennent comme le coût des communications et la garantie de convergence.	Ce cas, parmi les plus difficiles, correspond à une planification multi-agent distribuée dans laquelle la résolution du problème est complètement distribuée (DPS). Chaque agent construit son propre plan et une coordination inter-agent est nécessaire, ce qui fait souvent appel à la négociation et à la résolution de conflits.

Dans *(Durfee 1999)*, l'auteur illustre le problème de la résolution distribuée de problème (DPS) en considérant une équipe d'agents qui doit, par exemple, concevoir les plans d'une voiture en répondant à des contraintes globales : la voiture doit avoir quatre roues, le moteur doit rentrer sous le capot et avoir une puissance suffisante pour déplacer la voiture, etc. Il indique que les agents doivent se coordonner :

- pour décomposer le problème (partage en tâches),
- pour allouer les sous-problèmes (répartition des tâches), puis
- pour échanger les solutions des sous-problèmes, et enfin
- pour combiner les solutions (synthèse du résultat).

Une distinction doit être faite entre la planification dans les SMA où chaque agent est capable de réaliser indifféremment les taches données et la planification dans des SMA où certains agents sont experts pour la réalisation de tâches particulières. Le premier cas se limite à la théorie, car il implique que les agents soient des clones parfaits avec les mêmes capacités, ce qui est peu réaliste. A l'inverse, le second est plus riche, mais implique d'identifier quel agent est capable de faire quoi.

Afin de permettre une distribution des tâches selon les capacités des agents, on a vu apparaître des approches comme Contract-Net *(Smith 1980)*. Il s'agit d'une approche distribuée et récursive dans laquelle l'allocation de tâches est négociée. Des agents managers se voient alloués une tâche qu'ils décomposent. Ils font un appel d'offre pour savoir qui va réaliser les sous-tâches. Les agents contractants (subordonnés) qui sont qualifiés et qui souhaitent réaliser la tâche vont faire des offres au manager. Le manager décide quel contractant il choisit et lui alloue une sous-tâche. Le manager surveille l'avancement des sous-tâches et il est responsable de l'intégration des résultats. Si un contractant échoue dans la réalisation d'une tâche, alors la tâche est réallouée.

On peut également citer DVMT *(Lesser et Corkill 1984)* où des agents à base de tableau noir échangent des solutions partielles qui convergent en solution finale. Il existe enfin des travaux qui ont

cherché à concevoir des architectures intégrant diverses technologies de planification multi-agent comme MPA de *(Wilkins et Myers 1998).*

1.5.7. La planification probabiliste

La planification conditionnelle était un premier pas vers la prise en compte des évènements survenant pendant l'exécution. Le second était l'extension des conditions afin de retranscrire, par l'ajout de probabilités, la possibilité de tomber dans un cas ou dans un autre. Des planificateurs conditionnels probabilistes comme C-BURIDAN (*Draper et al. 1994)* sont alors apparus. La prise en compte de l'incertitude a rejoint les travaux sur les Processus de Décision Markoviens (tels que les MDP) dont nous allons parler dans la partie suivante de ce chapitre. On trouve d'ailleurs des travaux *(Blum et Langford 1999)*, dans lequel les auteurs utilisent une approche à base de Processus de Décision Markoviens pour étendre le planificateur GraphPlan, afin qu'il devienne capable de gérer l'incertitude sur le résultat des actions effectuées. Ils expliquent que l'on peut avoir un opérateur "ouvrir porte" avec 88% de chances de réussite, 10% de cas dans lesquels la porte ne s'ouvre pas et 2% de cas dans lesquels la poignée reste dans la main de celui qui l'ouvre.

1.5.8. Comparaison des approches

Après avoir effectué ce tour d'horizon des méthodes de planification avancée, nous proposons de résumer les apports et les limites de chacune dans le Tableau 9.

Tableau 9 - Comparatif des méthodes de planification avancées

Type de planification avancée	***Apports***	***Limites***
Non linéaire / Ordre partiel	Les plans construits sont plus souples et peuvent permettre une parallélisation des tâches. Davantage de problèmes, comme ceux menant à une anomalie de Sussman *(Sussman 1975)*, peuvent être résolus.	Les hypothèses posées et la nature des problèmes abordés restent sensiblement les mêmes que dans la planification classique.
Hiérarchique	La capacité de se placer à différents niveaux d'abstraction rend possible la résolution de problèmes complexes par raffinement. De plus, les plans obtenus étant hiérarchiques, ils sont plus faciles à gérer et à comprendre.	Ce type de planification nécessite toutefois que l'espace d'état se prête facilement à une décomposition hiérarchique ce qui n'est pas toujours évident.
Conditionnelle	La force de la planification conditionnelle est de pouvoir prendre en compte des phénomènes qui ne sont pas connus lors de la conception du plan. Elle lève donc en partie l'hypothèse donnant au planificateur la connaissance complète sur l'environnement.	Les plans produits sont plus complexes que pour la planification classique, car ils prennent en compte tous les cas envisagés à la conception. Ces plans sont plus coûteux à générer et à manipuler, sans pour autant être sensibles à toutes les évolutions de l'environnement.
Probabiliste	Cette planification permet la gestion explicite des environnements incertains. Elle lève davantage l'hypothèse donnant au planificateur la connaissance complète sur l'environnement pendant la planification.	La planification probabiliste part cependant sur l'idée que l'agent n'obtient plus d'informations sur l'état du monde pendant l'exécution. L'agent suit donc son plan sans réactivité vis-à-vis de l'état courant.
Sous contrainte de ressources	L'avantage de ce type de planification est qu'elle permet d'aborder des problèmes plus réalistes où les actions peuvent générer ou consommer des ressources. Elle permet en particulier la gestion du temps qui est un critère déterminant avec, pour la planification Anytime, une optimisation du rapport temps/efficacité.	L'une des difficultés est de trouver la bonne représentation, mais aussi la quantité d'information nécessaire et suffisant pour gérer la ressource de façon efficace. Le risque est de voir la complexité exploser pour certains domaines, surtout avec des ressources multiples
Réactive	Contrairement aux autres types de planifications, la planification réactive offre une forte capacité de réponse aux changements du monde pendant l'exécution.	Cette planification génère une action pour chaque situation, ce qui limite son usage à des domaines de taille restreinte. De plus, la réévaluation de l'état requiert un usage intensif des données issues des capteurs qui peuvent mettre du temps à être traitées.
Multi-Agent	L'aspect distribué des approches multi-agent permet de s'affranchir de l'hypothèse d'un contrôle exclusif. Le problème à résoudre se retrouve découpé et résolu plus simplement par la collaboration de chaque agent.	La difficulté et les coûts de l'approche se reportent dans la coordination des actions, ce qui implique de mettre en œuvre des protocoles d'interaction efficaces.

D'une façon générale, nous nous apercevons que, pour chaque méthode, certaines hypothèses restrictives de la planification classique peuvent être levées, alors que d'autres persistent toujours.

1.6. Adéquation à nos objectifs

A partir de ce tour d'horizon des méthodes de planification, nous allons expliquer l'approche que nous avons retenue afin de produire le comportement des agents contrôlés.

En nous replaçant dans notre problématique, nous devons tout d'abord rappeler que l'incertitude et la dynamique sont des aspects très importants. L'incertitude provient essentiellement de la nature des hSMA qui, rappelons-le, font le plus souvent intervenir des agents partiellement ou non contrôlés, comme les êtres humains. De plus, comme la communication utilisée pour interagir n'est pas forcément fiable, des imprévus sont susceptibles de se produire, ce qui fait que les perceptions et les effets des actions entreprises par les agents contrôlés sont incertains. D'un autre côté, l'environnement du hSMA est dynamique, car l'agent est loin d'être seul à agir. En conséquence, il est difficile pour le concepteur de prévoir tous les cas possibles. Ce sont ces considérations qui expliquent notre intérêt pour les planifications probabilistes et réactives.

La planification multi-agent, quand-à-elle, paraîtrait, de prime abord, appropriée pour nos besoins. Elle nécessite cependant que l'ensemble des agents sur lesquels le problème se distribue soit homogène et contribue à la solution globale. Ce besoin d'homogénéité n'est hélas pas satisfait pour les hSMA.

Dans le chapitre précédent, nous avons proposé de modéliser les services en nous basant sur la médiation : on peut considérer que ce sont les utilisateurs qui délèguent des buts aux médiateurs chargés de la fourniture du service. Comme il s'agit de satisfaire des utilisateurs qui ne sont pas nécessairement des experts, il est parfois difficile d'obtenir des buts définis précisément et avec certitude. Ainsi, l'une des difficultés que nous rencontrons se situe au niveau de la définition même du but d'un service. Pour contrer cela, nous avons suivi le raisonnement suivant : plutôt que d'imposer un but déterminé lors de la conception, ce sont finalement ces agents, avec lesquels le médiateur considéré doit interagir, qui vont lui déléguer des buts et qui vont le contrôler. Comme, bien souvent, le but est seulement connu par l'utilisateur de façon imparfaite, nous avons cherché à faire évoluer la notion de contrôle. Notre choix s'est porté sur une solution plus souple qui est l'attribution de récompenses et de pénalités avec une découverte du but par le médiateur.

Tout ceci nous a conduit à nous intéresser à la théorie de la décision, et plus particulièrement aux ***Processus de Décision Markoviens*** que nous allons présenter en détail dans la seconde partie de ce chapitre. L'avantage de ces modèles est qu'ils permettent de construire des agents rationnels capables de travailler dans des environnements dynamiques et incertains, en cherchant, non plus à suivre une séquence d'action fixée, mais plutôt à maximiser une utilité espérée.

2. Les processus de décision markoviens

Dans cette partie de chapitre, nous présentons l'outil de modélisation que nous avons choisi afin de construire des agents médiateurs. Celui-ci est issu de la théorie de la décision et de la théorie des probabilités. Il permet de répondre à la question "Comment agir sur un système qui évolue de façon incertaine afin d'obtenir ce que l'on veut ?". Nous nous intéresserons aux différents problèmes qu'il permet de résoudre, puis nous montrons qu'il est adapté pour ceux que nous nous posons.

2.1. Une définition intuitive

Un ***processus de décision markovien*** permet de modéliser un problème de décision dans lequel deux éléments entrent en jeu. On suppose que l'on a d'une part, un système qui évolue dans le temps comme un automate probabiliste et d'autre part, une sorte de contrôleur extérieur qui examine l'état actuel du système. Ce contrôleur dispose d'un certain nombre d'actions qu'il peut utiliser pour obtenir ce qu'il veut. L'objectif, pour lui, est de construire un plan optimal qui maximise la récompense qu'il peut obtenir.

L'évolution du système est vue comme un ***processus markovien***, c'est-à-dire une suite temporelle d'états distincts qui évolue suivant des probabilités de transition, en ne dépendant que des phénomènes de l'état précédent. *(Puterman 1994)* présente en détails ces modèles qui se situent à la confluence de la théorie de la décision et de celle des probabilités. Les processus de décision markoviens permettent de prendre des décisions séquentielles en présence d'incertitude. Dans la littérature scientifique en anglais, cette classe de modèles stochastiques est appelée "Markov Decision Process" et l'abréviation courante est MDP, bien que PDM soit parfois utilisé en langue française.

Un MDP est composé de deux éléments distincts : d'une part, un processus markovien et de l'autre un élément contrôleur. A chaque pas de temps, le contrôleur du système observe le processus et possède la capacité à agir dessus, en effectuant une action éventuellement nulle. Un raisonnement à base de MDP peut amener au discours suivant : "Je sais que je suis dans telle situation et si je fais telle action il y a tant de chances pour que je me retrouve dans cette nouvelle situation en obtenant tel profit."

Nous allons maintenant présenter les quatre composants d'un MDP, à savoir les états, les actions, les transitions et les récompenses.

2.1.1. Etats

Un ***état*** représente la façon dont le système existe actuellement. Idéalement, il s'agit d'une représentation abstraite et compacte qui ne prend en compte que les éléments nécessaires pour résoudre le problème que l'on se pose, tout le reste étant négligé dans la modélisation. Il peut s'agir de la position d'un robot mobile dans son environnement ou de la valeur d'un portefeuille de valeurs boursières... Si l'on imagine toutes les combinaisons de valeurs possibles des éléments modélisés, alors on obtient un ensemble d'états appelé ***espace d'état***. Dans ce qui va suivre, on suppose que l'ensemble des états est discret, c'est-à-dire qu'il est fini et que l'on peut compter ses éléments.

2.1.2. Actions

Une ***action*** est un moyen dont on dispose et dont l'effet permet de modifier l'état du processus. En reprenant les exemples précédents, le robot mobile peut par exemple avancer ou tourner, un courtier peut vendre ou acheter et des parts de marché, etc. On suppose donc l'existence d'un ensemble fini et discret d'actions que l'on peut effectuer (contenant éventuellement une action dite nulle qui consiste à ne rien faire). Un problème que nous allons aborder sera alors de savoir laquelle de ces actions choisir dans un état particulier du monde.

2.1.3. Transitions

Les ***transitions*** permettent de dire comment évolue l'état du système dans le temps et en particulier sous l'effet d'une action. Les MDP sont des outils puissants, car ils prennent en compte le fait que les effets des actions puissent être probabilistes et qu'ils ne sont donc pas garantis. Nous allons illustrer le concept de transition probabiliste sur l'exemple du robot, illustré sur la Figure 29 : si le sol est glissant et que le robot essaye d'avancer depuis l'état s_0, il peut effectivement avancer tout droit vers s_2 dans 90 % des cas mais il peut aussi dévier de sa trajectoire à gauche vers s_1 (5% de chances) ou à droite vers s_3 (5% de chances). On peut aussi dire que, sur l'exemple boursier, un ordre d'achat donné peut être ou non satisfait.

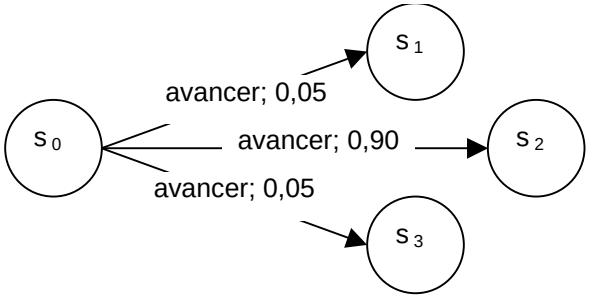


Figure 29 - Exemple d'incertitude sur le résultat d'une action

2.1.4. Récompenses

Les MDP visent à automatiser le processus de décision, aussi, on doit être capable d'avoir une mesure sur l'utilité des différentes actions possibles. C'est pourquoi on spécifie des valeurs gagnées ou des pénalités correspondant au choix de telle action dans tel état. Une ***récompense***, dite immédiate, est perçue pour chaque transition, mais elle peut tout à fait être nulle. On peut alors chercher à agir de façon à maximiser la récompense totale à court, moyen ou long terme.

2.2. Une définition formelle

Un Processus de Décision Markovien est un quadruplet **<*S, A, T, R*>,** où

- $S = \{s_0, \ldots, s_N\}$ est un ensemble fini d'états du système considéré.
- $A = \{a_0, \ldots, a_K\}$ est un ensemble fini d'actions possibles sur le système. On peut également faire dépendre l'ensemble des actions de l'état courant. En effet, toutes les actions ne sont pas toujours réalisables à partir de n'importe quel état. Dans ce cas on a une fonction du type $A : S \rightarrow P(A)$.
- $T : S \times A \times S \rightarrow [0 ; 1]$ est une fonction de transition modélisant le passage du processus d'états en états. Elle représente l'incertitude que l'on a sur les effets d'une action. On peut interpréter ***T(s, a, s')*** comme la probabilité, lorsque le système est dans l'état ***s***, qu'il puisse se retrouver dans l'état ***s'*** après avoir subi une certaine action ***a***. Cette probabilité qui s'écrit ***P(s' | s, a)*** et se traduit par une distribution de probabilité sur l'état obtenu.
- ***R*** est une fonction de récompense permettant de caractériser les buts à atteindre, mais aussi les états dangereux à éviter. Elle correspond soit simplement à la valeur obtenue dans un état donné (***R : S*** $\rightarrow$ ***Réel***), soit à la valeur obtenue dans un état, en ayant effectué une certaine transition auparavant (***R : S*** $\times$ ***A*** $\times$ ***S*** $\rightarrow$ ***Réel***).

On appelle paramètres du MDP les données correspondant à la fonction de transition et à la fonction de récompense. Si on note ***d***, la fonction de décision du contrôleur du MDP, avec ***d : S, R*** $\rightarrow$ ***A***, on obtient alors la représentation de la Figure 30.

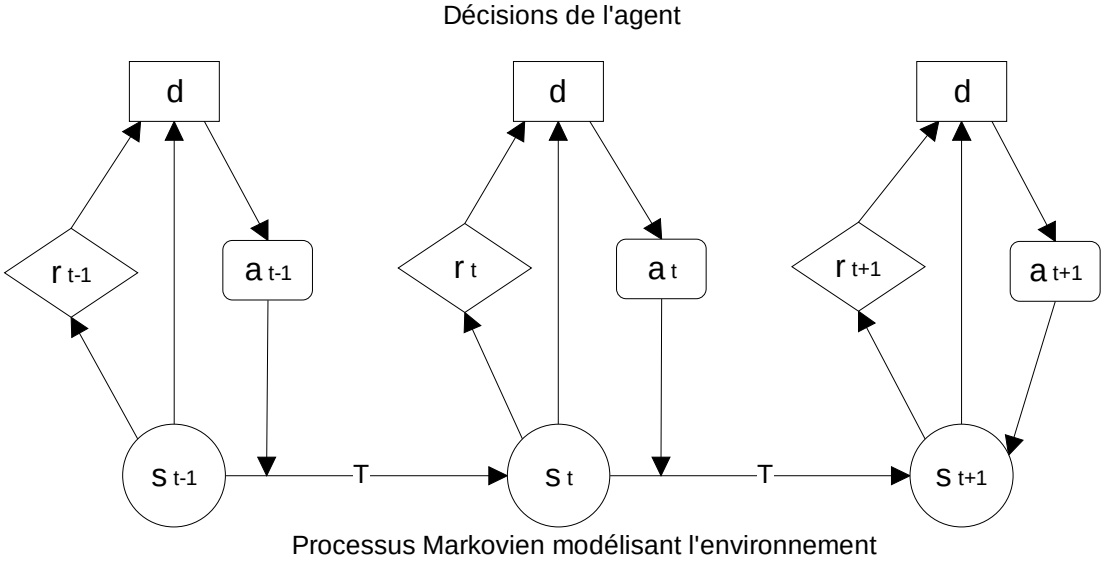


Figure 30 - Evolution d'un MDP selon les décisions de l'agent

2.3. Hypothèse de Markov

Les processus de décision Markovien respectent l'***hypothèse de Markov*** : l'état d'un processus Markovien d'ordre ***n*** à un instant donné ne dépend que des ***n*** instants précédents. Ainsi on a simplement besoin de spécifier l'état suivant résultant pour chaque état de départ et pour chaque action. Plus particulièrement, un processus markovien d'ordre 1 ne nécessite pas la mise en oeuvre d'un historique du passé, puisque seules les informations du temps précédent suffisent : *l'état suivant dépend seulement de l'état courant et de l'action choisie*. Se placer dans le cadre de l'hypothèse de Markov permet donc de simplifier le choix des actions, en ne nécessitant pas l'utilisation d'une mémoire à partir de l'état courant.

2.4. Exemple de MDP

Afin d'illustrer le principe des MDP, nous donnons ici un exemple utilisé par *(Sutton et Barto 1998)*:

Un robot mobile a pour tâche de collecter des boites de soda vides dispersées dans un bâtiment. Il peut les détecter au moyen de capteurs, les ramasser avec une pince pour les placer dans un container qu'il transporte. Il fonctionne à l'aide de batteries qu'il peut aller recharger à sa base. On suppose qu'il dispose des systèmes de contrôle nécessaires à sa navigation dans le bâtiment, à l'interprétation des données perçues et au contrôle du bras articulé. Dans la modélisation on considère que tout ce qui ne relève pas du module prenant cette décision constitue l'environnement. Le problème consiste à décider, seulement en fonction de la charge actuelle des batteries, quelle action le robot doit effectuer :

a) aller à la recherche de boites vides,

b) attendre sur place que d'éventuelles boites soient déposées directement dans son container ou

c) retourner à sa base pour se recharger.

On suppose que la prise de décision s'effectue au bout d'un temps donné ou à l'arrivée d'un événement comme la collecte d'une boite. Le robot reçoit, par ailleurs, une récompense de +1 lorsqu'il collecte une boite. La meilleure façon de collecter des boîtes est de partir en exploration, mais cela consomme beaucoup d'énergie, alors qu'attendre ne consomme rien. Quand le robot recherche, il existe une probabilité que le robot tombe en panne d'énergie. Le robot distingue deux états pour le niveau de sa batterie : ***S = {haut, bas}***. Selon son état, il ne dispose pas des mêmes ensembles d'actions, car aller se recharger alors que les batteries sont pleines n'a pas de sens. On distingue donc :

$$A_{haut} = \{chercher, attendre\}$$

$$A_{bas} = \{chercher, attendre, recharger\}$$

Effectuer une recherche de boites avec un haut niveau d'énergie ne modifie pas l'état de la batterie avec une probabilité de α et fait passer la batterie à l'état bas avec une probabilité de $(1-\alpha)$. Effectuer une recherche de boites avec un bas niveau d'énergie ne modifie pas l'état de la batterie avec une probabilité de β et épuise complètement la batterie avec une probabilité de $(1-\beta)$. Si le robot tombe en panne de batterie, alors il est dépanné : il est rechargé à bloc, mais reçoit également une récompense valant -3. On note $R^{chercher}$ et $R^{attendre}$ le nombre de boites que l'on estime pouvoir collecter respectivement en cherchant et en attendant, avec $R^{chercher} > R^{attendre}$.

Dans le Tableau 10, on retrouve le modèle, c'est-à-dire les probabilités de transition et la valeur de la récompense pour chaque passage d'un état courant ***s*** à un nouvel état ***s'*** en effectuant une action de A_s.

Il est également possible d'illustrer la façon dont le système évolue à l'aide d'un graphe, donné Figure 31, contenant des nœuds état (en blanc) et des nœuds action (en gris). Les arcs entre nœuds état et les nœuds action portent le nom de l'action effectuée, alors que les arcs entre les nœuds action et états portent les probabilités de transitions et les récompenses estimées correspondantes.

Tableau 10 - Probabilités de transition et récompenses du robot collecteur *(Sutton et Barto 1998)*

$s=s_t$	$s'=s_{t+1}$	$a = a_t$	$P(s' \mid s, a)$	$R(s, a, s')$
haut	haut	chercher	α	$R^{chercher}$
haut	bas	chercher	$1-\alpha$	$R^{chercher}$
bas	haut	chercher	$1-\beta$	-3
bas	bas	chercher	β	$R^{chercher}$
haut	haut	attendre	1	$R^{attendre}$
haut	bas	attendre	0	$R^{attendre}$
bas	haut	attendre	0	$R^{attendre}$
bas	bas	attendre	1	$R^{attendre}$
bas	haut	recharger	1	0
bas	bas	recharger	0	0

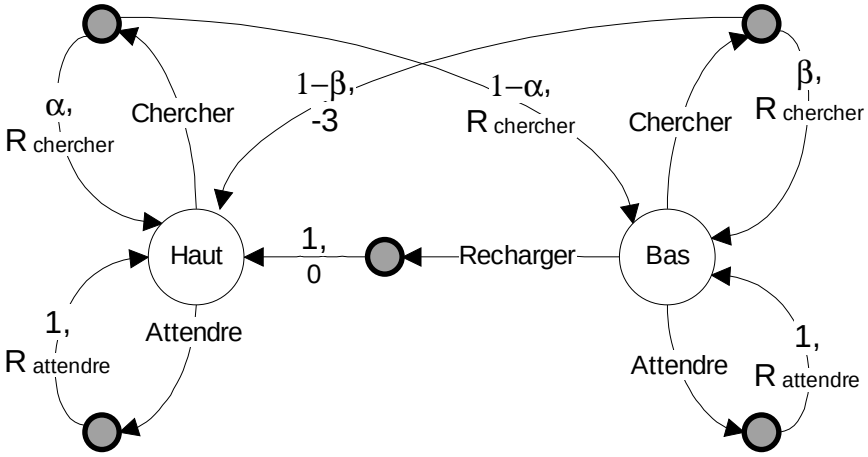


Figure 31 - Graphe de transition du MDP pour le robot collecteur *(Sutton et Barto 1998)*

2.5. Récompenses attendues et horizons

On cherche à obtenir le meilleur gain que l'on puisse espérer à long terme. Si on se place à l'instant ***t***, il est possible d'obtenir la séquence de récompenses $r_{t+1}, r_{t+2}, r_{t+3}, ..., r_T$ où T est l'instant pour lequel se termine toute activité. Comme on veut maximiser la récompense globale à laquelle on peut s'attendre, une idée simple est de chercher à maximiser l'ensemble de toutes les récompenses perçues.

On définit la notion d'***horizon*** comme le nombre de périodes pour lesquelles on doit effectuer une décision, en considérant pratiquement que le processus meurt au bout de cet horizon. On peut parfois se limiter à rechercher la meilleure politique, en supposant que l'on a un horizon fini. Par exemple, supposons que chaque jour, on se lève et on décide à propos de quelque chose. On veut décider en supposant que l'on a à prendre une décision seulement pour un nombre de jours fixé. La longueur de l'horizon est le nombre de jours pour lesquels on décide. Si on considère une infinité de périodes de temps, on parle alors d'horizon infini.

La récompense attendue pour un horizon fini donne : $R_t = \sum_{i=t+1}^{T} r_i$

Et dans le cadre d'un horizon infini, on obtient : $R_t = \sum_{i=t+1}^{+\infty} r_i$.

En horizon infini, on a cependant un problème dans les calculs : la somme peut tendre facilement vers l'infini. La solution, si les récompenses sont bornés, consiste à introduire un coefficient d'escompte $\gamma \in [0;1[$ pour ramener R_t à une somme finie.

$$R_t = \sum_{i=0}^{+\infty} \gamma^i r_{t+i+1}$$

En fait γ correspond à l'intérêt porté à un instant donné aux récompenses que l'on obtiendra par la suite. Ainsi, avec $\gamma = 0$, on ne porte aucune attention aux récompenses futures et on se contente de travailler sur les récompenses immédiates.

2.6. Décision et politiques dans les MDP

Décider, c'est sélectionner une action dans l'ensemble des actions possibles à des instants discrets. La solution d'un MDP est appelée une ***politique*** (en anglais policy). Souvent notée π, elle spécifie comment effectuer cette décision en sélectionnant la meilleure action pour chaque état afin d'optimiser un critère donné. Dans le cas le plus général, on définit une politique comme une fonction qui retourne la probabilité $\pi(s,a) = P(a \mid s)$ de choisir l'action a étant dans l'état ***s***.

$$\pi : S \times A \rightarrow [0;1]$$

Il s'agit là d'une ***politique stochastique*** : quand le contrôleur considéré reçoit l'état du système et la récompense éventuelle, tout se passe, comme s'il tirait au hasard l'action qu'il va effectuer avec un dé qui respecte les probabilités. Si nous comparons la planification stochastique avec les approches que nous avons présentées dans la première partie, nous pouvons dire qu'une politique est une sorte de plan probabiliste. Dans le cas particulier où à chaque état ***s*** est associé une et une seule action ***a***, on parle de ***politique déterministe*** :

$$\pi : S \rightarrow A$$

Par la suite, on notera Π, l'ensemble des politiques déterministes.

Le choix de la politique à adopter, doit prendre en compte d'une part les effets immédiats, mais aussi les effets à plus long terme. Lorsque le critère d'optimisation porte sur la récompense, il s'agit donc de choisir l'action qui offre un meilleur rendement entre les gains immédiats et futurs. Pour cela, on calcule une fonction de valeur, à partir de laquelle on obtiendra une politique.

2.6.1. Fonction de valeur

La ***fonction de valeur*** $V^{\pi}(s)$ d'un état, au sens d'une politique π, est la récompense à laquelle on peut s'attendre lorsque à partir de l'état ***s***, on suit cette politique en sélectionnant l'action $\pi(s)$ On parle également de fonction d'utilité d'un état.

$$V^{\pi}(s) = E_{\pi}\left\{ R_t \middle| s_t = s \right\} = E_{\pi}\left\{ \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \middle| s_t = s \right\}$$

On rappelle que l'espérance d'une variable aléatoire ***X*** pouvant prendre être réalisée par $\{x_1 ; \ldots ; x_n\}$ avec les probabilités $p(x_1) \ldots p(x_n)$ s'écrit $E(X) = \sum_{i=1}^{n} x_i p(x_i)$.

De la même façon, on définit la fonction $Q^{\pi}(s,a)$ par la valeur liée à la sélection de l'action ***a*** dans l'état ***s*** suivant une politique π.

$$Q^{\pi}(s,a) = E_{\pi}\left\{ R_t \middle| s_t = s, a_t = a \right\} = E_{\pi}\left\{ \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \middle| s_t = s, a_t = a \right\}$$

On appellera ***Q-Valeur*** le résultat de cette fonction pour le couple ***(s,a)***. La Figure 32 illustre le fonctionnement d'une transition en prenant en compte la valeur de l'état d'origine, celles des états d'arrivée et les Q-Valeurs associées aux paires état-action.

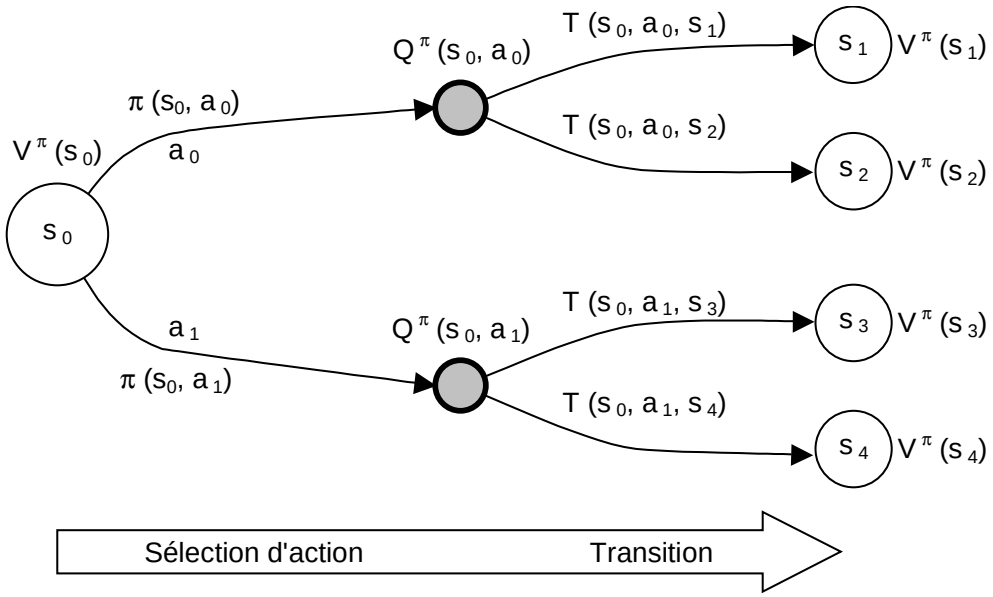


Figure 32 - Calcul de la fonction de valeur pour les états et des Q-Valeurs pour les paires état-action

Dans, *(Sutton et Barto 1998)* les auteurs montrent que la fonction de valeur vérifie l'équation suivante pour toute politique π et tout état s :

$$V^{\pi}(s) = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} T(s,a,s') \left[R(s,a,s') + \gamma V^{\pi}(s') \right]$$

Il s'agit d'une équation de Bellman pour V^{π} qui montre la relation entre la valeur d'un état et celle de ses successeurs.

2.6.2. Politique optimale

Le ***Principe d'optimalité*** de Bellman de la programmation dynamique indique :

*Une **politique optimale** a la propriété que, quelque-soit l'état initial et la décision initiale, les décisions restantes doivent constituer une politique optimale par rapport à l'état résultant de la première décision. (Bellman 1957)*

Donc, pour une π^* politique optimale et π_s^* , une sous-trajectoire de π^* , ayant ***s*** pour origine. Alors π_s^* est optimale pour la fonction de valeur restreinte aux trajectoires ayant pour origine ***s***. La valeur V^{π} de π^* pour chaque état est obtenue en résolvant l'équation de Bellman.

Une politique π est dite optimale, et notée π^* si, pour tout état, la valeur attendue en suivant cette politique est plus grande ou égale à celle attendue pour toute autre politique.

$$\forall s \in S, \forall \pi' \in \Pi, \qquad V^{*}(s) \geq V^{\pi'}(s)$$

V^* est la fonction de valeur optimale : il s'agit de l'unique solution de l'équation de Bellman pour π^*. Elle correspond pour chaque état à la récompense maximale que l'on peut espérer obtenir (en suivant une politique optimale π^*).

$$V^{*}(s) = \mathrm{Max}_{\pi \in \Pi} V^{\pi}(s) \qquad \forall s \in S$$

Ainsi, si on connaît V^* pour tout état ***s*** et un modèle de l'environnement ***T*** et ***R***, alors on obtient une stratégie optimale π^* en choisissant toujours l'action qui amène vers l'état suivant ayant la plus grande utilité.

$$\pi^*(s) = \text{argmax}_a \, (R(s,a) + \gamma V^* (T(s,a))$$

On a de façon similaire ***Q*(s,a)***, qui donne la récompense maximale que l'on peut attendre en sélectionnant l'action ***a*** dans l'état ***s*** (en suivant une politique optimale π^*).

$$Q^*(s,a) = \text{Max}_{\pi \in \Pi} Q^{\pi}(s,a) \qquad \forall s \in S$$

Or, on peut écrire ***Q**** comme l'espérance de gain en fonction de ***V**** :

$$\begin{aligned} Q^*(s,a) &= E \{ r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s ; a_t = a \} \\ &= \sum_{s' \in S} T(s,a,s') (R(s,a,s') + \gamma V^*(s')) \end{aligned}$$

Par ailleurs, ***V*(s)*** est finalement la valeur obtenue en choisissant l'action ***a*** qui amène à la plus grande valeur ***Q*(s,a)***.

$$V^*(s) = \text{Max}_{a \in A} Q^*(s,a)$$

On obtient alors l'équation de Bellman pour calculer ***Q****.

$$Q^*(s,a) = \sum_{s' \in S} T(s,a,s') (R(s,a,s') + \gamma \text{Max}_{a' \in A} Q^*(s',a'))$$

2.7. Les problèmes MDP

Nous allons maintenant présenter les grandes classes de problèmes que les MDP permettent de résoudre. En effet, selon la connaissance que l'on a des paramètres (***T*** et ***R***) du modèle et selon que l'on dispose ou non d'une politique, on peut utiliser ces connaissances pour contrôler le système ou chercher à améliorer la connaissance que l'on en a. Dans chaque cas, nous illustrerons les problèmes sur l'exemple du robot collecteur, puis nous indiquerons quels sont les algorithmes classiques des MDP qui permettent de les résoudre.

2.7.1. Contrôler une exécution

Lorsque l'on dispose des paramètres (***T*** et ***R***) du modèle et d'une politique π, on peut essayer de contrôler le système réel à partir du modèle. Le principe consiste à observer le système afin de déterminer quel est son état courant s_t. On sélectionne alors l'action a_t à réaliser en utilisant la politique π que l'on connaît : $a_t = \pi (s_t , a_t)$. On effectue l'action sur le système, puis on recommence le cycle pour le pas de temps suivant. Sur l'exemple de robot collecteur, cela se traduit simplement par le fait de laisser le robot choisir ce qu'il doit faire en suivant la politique donnée.

2.7.2. Chercher à apprendre une meilleure politique

Il est toujours possible de se donner une première politique π_0. Cette politique peut être donnée a priori, avoir été déjà apprise ou être totalement arbitraire, mais donner des résultats insuffisants. On peut alors chercher à l'améliorer en utilisant par exemple l'algorithme Policy Iteration *(Howard 1960)*. Pour le robot collecteur, on peut s'apercevoir qu'une première politique permet au robot de collecter un bon nombre de boites, mais qu'elle le conduit trop souvent dans une situation de panne de batterie. On peut alors chercher une politique tout aussi performante, mais plus économe en d'énergie.

2.7.3. Apprendre une politique

Dans ce cas de figure, le contrôleur du système ne sait pas quelles actions il doit effectuer pour atteindre son but. Il s'agit alors d'un problème typique de planification. Si on connaît les paramètres du modèle, on peut les utiliser pour apprendre une politique, comme le fait l'algorithme Value Iteration *(Bellman 1957)*. Par contre, si les paramètres ne sont pas connus, on peut toujours essayer de procéder par essai-erreur et on parle alors d'apprentissage par renforcement. On peut alors essayer utiliser des algorithmes comme le Q-Learning qui sera présenté plus loin. Pour notre robot, apprendre une politique peut l'amener à partir à la recherche de boites, alors qu'il a un niveau de batterie faible. S'il tombe en panne, il pourra mémoriser que partir à l'aventure n'est peut-être pas le plus raisonnable quand on ne dispose pas de ressources suffisantes. Une autre possibilité est d'améliorer les politiques en apprenant sur des traces, utilisant des approches du type Monte-Carlo ou Différences Temporelles (TD) avec, également ce même avantage de ne pas nécessiter une connaissance complète de l'espace d'état.

3. Apprentissage par renforcement

Parmi la multitude d'algorithmes existants, nous nous sommes plus particulièrement intéressés à l'approche par renforcement *(Kaelbling et al. 1996, Sutton et Barto 1998)*, car elle a pour avantage de ne pas nécessiter de connaissance du modèle sous-jacent. La fonction de transition ***T*** et la fonction de récompense ***R*** sont effectivement apprises au fur et à mesure.

3.1. Principe général

L'***apprentissage par renforcement*** (ou RL pour Reinforcement Learning) consiste à laisser un comportement s'optimiser par essai-erreur.

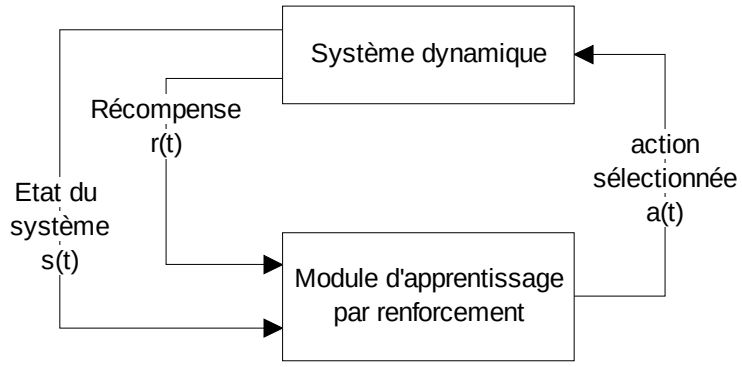


Figure 33 - La boucle d'un système d'apprentissage par renforcement *(Sutton et Barto 1998)*

Pour réaliser un apprentissage par renforcement sur un système dynamique à nombre d'états fini, dans le cadre des MDP, on commence tout d'abord par considérer un module d'apprentissage. Pour l'instant, on va regarder ce module comme une boîte noire. On lui fournit à sa création un ensemble d'états et un ensemble d'actions. On lui donne en entrée l'état du système ainsi qu'une récompense éventuelle et on obtient en sortie l'action que le module a sélectionnée. On place la boite noire en boucle avec le système dynamique, comme le montre le schéma Figure 33, et on la laisse évoluer.

Si ce n'est le fait que l'on utilise des récompenses, l'apprentissage par renforcement pourrait presque être rattaché à la classe des apprentissages non supervisés. En effet, on ne fournit pas d'échantillons contenant à la fois des entrées et les sorties attendues correspondantes. Les seules indications qu'il reçoit pour savoir s'il agit correctement sont des récompenses. Elles sont positives pour favoriser les succès et négatives pour pénaliser les échecs et les comportement coûteux. Il faut noter que l'obtention d'une récompense non nulle n'est pas systématique pour chaque cycle temporel.

Le module se donne une politique au départ et son comportement va osciller entre l'exploration de l'espace d'état-actions $S \times A$ et l'utilisation de cette politique. Il va donc choisir des actions et il sera récompensé ou non. En fonction de cela, il va mettre à jour son comportement avec une atténuation progressive.

3.2. Caractéristiques des problèmes d'apprentissage par renforcement

Dans le cadre des processus de décision markoviens, on peut caractériser le problème d'apprentissage par renforcement par le manque de connaissance concernant

- la fonction de transition $T: S \times A \times S \rightarrow [0;1]$
- la fonction de récompense $R : S \times A \rightarrow Réel$ ou $R : S \times A \times S \rightarrow Réel$

3.3. Le Q-Learning

Le ***Q-Learning*** est un algorithme d'apprentissage par renforcement, introduit par *(Watkins 1989)*, qui recherche une politique optimale déterministe $\pi^* : S \rightarrow A$ qui maximise la récompense accumulée. Un grand avantage est que l'on peut exploiter ce qui a déjà été appris, tout en poursuivant l'apprentissage. Plutôt que de travailler sur l'utilité ***V(s)*** des différents états, le Q-Learning apprend donc les valeurs ***Q(s,a)*** des paires (état, action). On reprend l'équation de Bellman permettant de calculer ***Q****.

$$Q^*(s,a) = \sum_{s' \in S} T(s,a,s') \left(R(s,a,s') + \gamma \, Max_{a' \in A} Q^*(s',a') \right)$$

On va chercher à calculer la fonction ***Q*(s,a)***. Si on ne connaît pas ***T(s, a, s')***, mais que dans un déroulement, on est passé de l'état $s = s_t$ à l'état $s' = s_{t+1}$ en sélectionnant l'action $a = a_t$, alors tout se passe comme si on avait eu à cet instant :

$T(s, a, s') = 1$ si $s = s_t$, $a = a_t$ et $s' = s_{t+1}$
$= 0$ sinon

La fonction va devoir vérifier l'équation suivante :

$$Q^*(s_t, a_t) = R(s_t, a_t, s_{t+1}) + \gamma \, Max_{a' \in A} Q^*(s_{t+1}, a')$$

Supposons maintenant que l'on connaisse, au temps ***t***, la fonction $Q_t(s,a)$ et que l'on s'appuie sur cette connaissance pour les valeurs des états suivants s_{t+1} pour toute action $a' \in A$, alors l'équation précédente nous permet de calculer une nouvelle valeur $Q_{t+1}(s_t, a_t)$ pour le temps ***t+1*** afin d'approcher ***Q****.

$$Q_{t+1}(s_t, a_t) = R(s_t, a_t, s_{t+1}) + \gamma \, Max_{a' \in A} Q_t(s_{t+1}, a')$$

En pratique, on ajoute un taux d'apprentissage α permettant de prendre plus ou moins en compte cette nouvelle valeur tout en gardant la connaissance passée Q_t. On obtient alors une formule permettant de mettre à jour des Q-valeurs :

$$Q_{t+1}(s,a) = (1- \alpha) \, Q_t(s_t, a_t) + \alpha \left[R_t(s_t, a_t) + \gamma \max_{a' \in A} Q_t(s_{t+1}, a') \right]$$

Une fois les Q-valeurs apprises, l'action optimale pour chaque état est celle avec la plus grande Q-valeur. Après avoir été initialisé de façon arbitraire, les Q-valeurs sont estimées sur la base d'expérience dans une boucle infinie. Dans *(Kaelbling 1996)*, on retrouve que si les paires action-état sont tous visités suffisamment souvent, le Q-Learning finit par converger vers une politique optimale π^* avec une probabilité 1 à partir d'un certain temps.

Si chaque action est exécutée dans chaque état un nombre infini de fois lors d'une exécution infinie et que α décroît de façon appropriée, alors les Q-Valeurs convergent vers Q avec une probabilité de 1.*

Davantage d'informations sur le Q-Learning et sa convergence peuvent être trouvées dans *(Watkins 1989), (Tsitsiklis 1994)* et *(Jaakkola et al. 1994).*

3.3.1. Algorithme général

Pour illustrer le fonctionnement général du Q-Learning, une étape du cycle d'apprentissage est illustrée sur la Figure 34 :

1. Depuis l'état courant s_t, on sélectionne une action a_t
2. On reçoit une récompense r_t pour cette action et le système évolue vers l'état suivant s_{t+1}
3. On met à jour l'utilité du choix que l'on a effectué avec la formule suivante

$$Q_{t+1}(s_t,a_t) = (1-\alpha)\, Q_t(s_t,a_t) + \alpha \left[r_t + \gamma \max_{a' \in A} Q_t(s_{t+1},a') \right]$$

4. On recommence au premier point.

L'algorithme conserve donc en mémoire la table de Q-Valeurs ***Q(s,a)***, le dernier état s_t du système et la dernière action effectuée a_t, car il les réutilise dans les étapes (1) et (3).

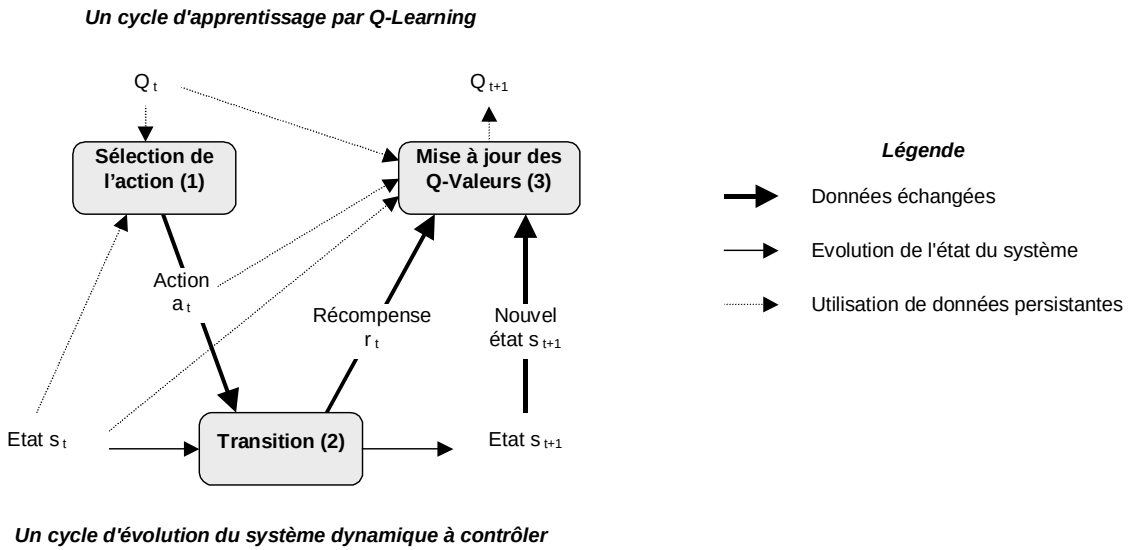


Figure 34 - Détail d'une étape pour l'algorithme du Q-Learning

3.3.2. Remarques

Il existe diverses stratégies pour sélectionner l'action à effectuer dans la première étape de l'algorithme. Le tout est que la fonction de sélection permette à chaque paire état-action d'être visitée suffisamment souvent, ce qui implique un compromis entre l'exploration des possibilités et l'exploitation du comportement acquis. En pratique, une possibilité est de tirer une action au hasard en respectant une distribution de probabilité de Boltzmann. La fonction de Boltzmann donne, pour une action donnée, la probabilité ***p(a)*** que l'on choisisse cette action.

$$P(a) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{b \in A} e^{Q(s,b)}}$$

Dans l'expression, on voit apparaître un paramètre ***T*** qui est la ***température de Boltzmann***. Lorsqu'elle est élevée, la température assure une exploration suffisante puis, quand on la fait décroître progressivement en tendant vers ***0***, elle réduit la marge de manœuvre, car les probabilités finissent par converger vers ***1*** pour la meilleure action et vers ***0*** pour les autres. La Figure 35 montre un exemple d'évolution des probabilités selon la température pour les Q-Valeurs suivantes : $Q(a_0) = 5$, $Q(a_1) = -1$, $Q(a_2) = -4$ et $Q(a_3) = 3$. A haute température, les actions sont équiprobables, mais à plus faible température, les probabilités sont réparties selon les Q-Valeurs. En approchant ***0***, l'action a_0 devient pratiquement la seule à avoir une chance d'être sélectionnée.

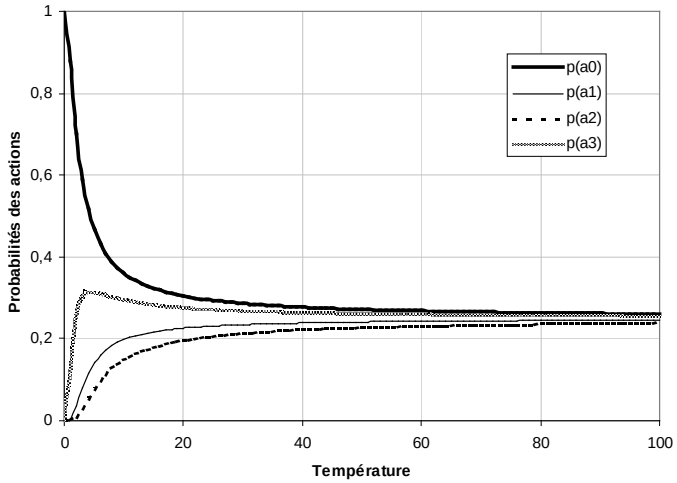


Figure 35 - Exemple d'évolution de probabilités d'actions selon la température de Boltzmann

Quand le taux d'apprentissage décroît dans le temps avec un séquencement approprié, les Q-valeurs convergent vers des valeurs correctes pour des environnements stationnaires. En pratique, on peut prendre par exemple un α par état ou par paire état-action et faire par exemple dépendre α du nombre ***nbv*** fois que l'on a "visité" cette combinaison. En prenant, par exemple α comme l'inverse du nombre de visites, cela permet d'apprendre davantage pour un nombre faible de visites alors qu'un passage fréquent modifiera peu les Q-Valeurs.

Pour davantage de détails, le pseudo-code correspondant à notre implantation de l'algorithme du Q-Learning est donné en annexe II.

3.4. Avantages, limites et problèmes

Les processus de décision markoviens permettent de déterminer des plans d'actions plutôt souples et efficaces, alors que l'on a une incertitude sur l'effet des actions. De plus, leur utilisation couplée à un apprentissage par renforcement peut combler le manque de connaissance sur les probabilités de transition ou sur la fonction de récompense. Cependant une limite apparaît lorsque l'on veut prendre en compte les incertitudes concernant les perceptions. En effet, l'approche MDP suppose que l'on ait accès, pour chaque étape de décision, à l'état complet du système considéré. On dit dans ce cas que le système est totalement ***observable***, car on en a une vision complète et précise, ce qui fait qu'il n'est pas possible de confondre deux états. Cette supposition d'observabilité complète est hélas rarement vérifiée dans les cas réels. Le plus souvent, on peut seulement obtenir un ensemble d'observations qui reflètent partiellement l'état du système, d'où la qualification d'observabilité partielle.

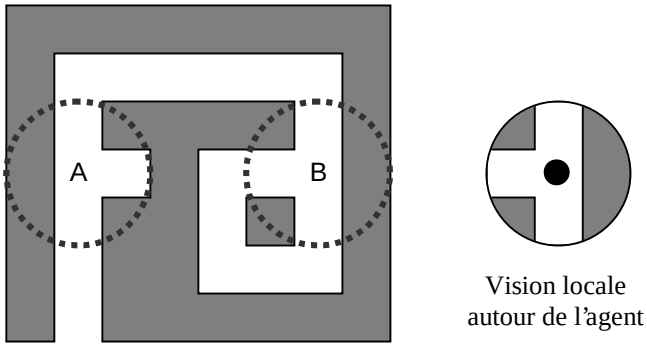


Figure 36 - Vision locale versus vision globale

Afin d'illustrer la différence entre observabilité totale et observabilité partielle, on peut imaginer une personne qui se promène dans le labyrinthe de la Figure 36. Dans un premier temps (illustré à gauche), on suppose que le promeneur est doté d'organes perceptifs parfaits, de portée infinie, qui lui permettent de savoir exactement où il se trouve. A priori, notre promeneur ne peut pas se perdre, car il n'y a pas de confusion possible et qu'à la limite, il perçoit même la sortie depuis le point de départ. Le manque de réalisme de cette observabilité totale nous amène dans un second temps à limiter la portée des organes perceptifs de notre promeneur (illustré à droite). Cette personne va toujours pouvoir appréhender son environnement, mais ceci de façon locale. L'observabilité étant devenue partielle, le promeneur va pouvoir percevoir deux carrefours similaires comme étant les mêmes, alors qu'ils sont en réalité à des endroits différents (A et B sur le schéma).

Pour traiter le cas de l'observabilité partielle, il est possible d'étendre le formalisme des MDP et d'aboutir aux ***Processus de Décision Markoviens Partiellement Observables*** (POMDP). Dans ce modèle, l'état du système considéré n'est plus directement accessible à l'agent. Seuls les éléments d'un ensemble de symboles Ω sont alors observables pour l'ensemble de l'espace d'état. A partir de l'état courant et éventuellement selon l'action effectuée, une fonction d'observation $O : S \times \Omega \rightarrow [0 ; 1]$ ou $O : S \times A \times \Omega \rightarrow [0 ; 1]$ donne alors la probabilité d'observer un symbole donné. Les méthodes des MDP ne sont hélas pas transposables directement pour les POMDP. Plus particulièrement, dans *(Dutech 1999)*, l'auteur explique que les algorithmes de renforcement classiques ne sont plus applicables, car ils ne convergent plus forcément vers une politique optimale déterministe. Les solutions consistent soit à se contenter d'une politique POMDP sous-optimale, soit d'utiliser des états de croyance avec des problèmes de continuité de l'espace d'état ou encore d'utiliser des politiques non-markoviennes où une mémoire du passé est utilisée. Dans notre modélisation, nous nous sommes ramenés à des états totalement observables, mais son extension à l'observabilité partielle est tout à fait possible et fait, par ailleurs, partie des perspectives de nos travaux.

Une autre limite au modèle que nous avons présenté est que les phénomènes pris en compte doivent être discrétisés en termes de temps, d'états et d'actions et c'est l'hypothèse que nous utilisons dans notre modèle. Dans *(Ten Hagen 2001)*, l'auteur expose plusieurs solutions pour utiliser l'apprentissage par renforcement lorsqu'il y a un problème de continuité. Il est par exemple possible d'utiliser une discrétisation en utilisant des algorithmes comme Parti-Game *(Moore 1994)* ou d'utiliser des approximateurs de fonctions avec une approche de type Actor-Critic.

4. Notre utilisation des MDP

En examinant le domaine de la planification dans le but de trouver un moyen de concevoir le comportement des agents contrôlés des hSMA, nous avons découvert toutes ces pistes, allant de la planification classique aux approches réactives, hiérarchiques ou multi-agent. Dans les applications auxquelles nous avons affaire, les effets des actions entreprises par les agents sont incertains et nous souhaitions trouver un modèle qui prenne en compte cette incertitude. De plus, nous souhaitions

définir le comportement des agents contrôlés, de façon à ce qu'ils soient capables d'adapter leurs actions en fonction de l'état de l'environnement.

Le but était donc de trouver un modèle offrant des capacités d'adaptation pendant son utilisation ce que ne permettait pas de faire la planification classique. Nous devions donc trouver des outils adaptés à nos besoins dans toutes les extensions considérées, avec une préférence pour les approches probabilistes. Idéalement nous aurions souhaité trouver un planificateur hiérarchique (pour le gain en abstraction), non linéaire et partiellement ordonné (pour permettre d'effectuer des traitements en parallèle lorsque c'est possible), distribué (pour pouvoir mettre en œuvre plusieurs agents contrôlés) et bien sûr avec une gestion probabiliste (tant sur les incertitudes des perceptions, que sur les incertitudes des actions), etc.

En considérant la possibilité d'apprendre le comportement par renforcement et la grande richesse qu'ils offrent pour modéliser l'incertain, nous avons opté pour les processus de décision markoviens MDP, et leurs descendants prenant en compte l'observabilité partielle, avec les POMDP, ou les hiérarchies d'abstraction, avec les HMDP *(Dietterich 2000)* et les HPOMDP *(Pineau et al. 2001)*. L'intérêt de ces approches s'est de plus confirmé au regard des scripts des services Dialoca que nous présenterons dans le chapitre 7 et qui peuvent se ramener à un automate, donc à un plan réactif. A ce niveau, il est intéressant de voir que les travaux sur la planification réactive se ramènent, selon *(Russel et Norvig 1995)*, aux politiques des MDP :

Un système qui contient une fonction d'agent représentée explicitement, qu'elle soit implantée par une table ou par un ensemble de règles condition-action, ne nécessite pas de se soucier des évolutions inattendues de l'environnement. Tout ce qu'il doit faire est d'exécuter n'importe quelle action qui lui est recommandée par sa fonction pour l'état dans lequel il se trouve (ou, dans le cas d'environnements inaccessibles, la séquence de perceptions jusqu'à l'instant considéré). Le domaine de la planification réactive vise à exploiter ce fait, en évitant de la sorte la complexité d'une planification dans un environnement dynamique et inaccessible. Les Plans Universels (Shoppers 1987) ont été développés comme un cadre pour la planification réactive, mais se sont retrouvés être une redécouverte de l'idée des politiques dans les processus de décision Markoviens.

Ainsi, nous utilisons les processus de décision markoviens dans nos travaux afin de concevoir le comportement des agents contrôlés au travers des politiques et, comme le montre le schéma Figure 34, tout l'environnement de l'agent forme un système avec une dynamique markovienne que l'agent observe et sur lequel il agit. Nous avons, par ailleurs, conçu nos agents contrôlés de façon à ce qu'ils puissent calculer leur récompense, en fonction de ce qu'ils reçoivent de leur environnement. Une modélisation d'agent contrôlé utilisant un MDP et basé sur le Q-Learning sera présentée dans le chapitre 6.

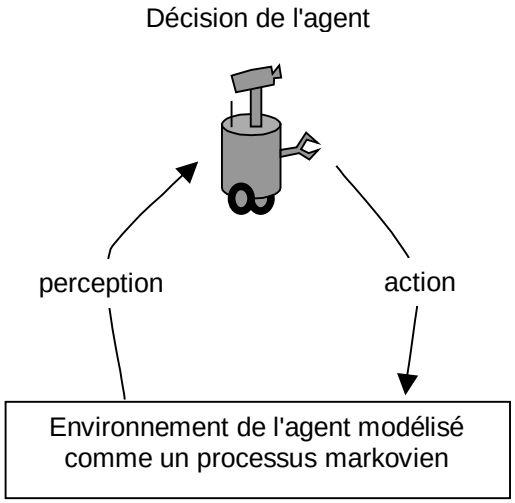


Figure 37 - Boucle perception-action d'un agent placé dans un environnement incertain

5. Conclusion

Savoir comment agir de façon adéquate dans un monde complexe n'est jamais une chose aisée et, en particulier, lorsque nous ne sommes pas certains de nos perceptions de l'environnement ou de l'effet de nos propres actions. Dans ce chapitre, nous avons cherché à donner un aperçu des différentes possibilités que nous avons considérées pour la production du comportement des agents contrôlés. Nous avons ensuite détaillé l'approche de la planification stochastique, et notamment sur l'apprentissage par renforcement dans les processus de décision markoviens. Dans le chapitre suivant, nous allons nous pencher sur les considérations d'adaptation du comportement aux autres agents. Nous discuterons également de l'approche par diagnostic qui est également possible pour le suivi et la reprise du déroulement des services dans le cadre des hSMA.

Chapitre 5 - Modélisation et diagnostic pour rendre les comportements adaptatifs

Nous avons évoqué à plusieurs reprises, le besoin d'adaptation des services à divers phénomènes pour obtenir plus de souplesse pendant leur déroulement. Dans ce chapitre, nous nous intéressons, tout d'abord, à l'étude de méthodes visant à adapter le comportement des agents contrôlés à celui des agents non ou partiellement contrôlés. Nous présenterons à cet effet la palette des différents outils qui sont utilisables pour modéliser les caractéristiques et les préférences des utilisateurs. Dans la seconde partie, nous nous pencherons sur les aspects de robustesse dans les services, afin de tenir compte des phénomènes pouvant altérer leur déroulement. Nous nous intéresserons à la caractérisation des états des systèmes informatiques, aux moyens de détection d'anomalies et aux possibilités de faire face à de telles situations. Nous montrerons ensuite comment une approche de modélisation de normalité peut être utilisée pour la détection des problèmes qui peuvent survenir pendant l'interaction dans un hSMA.

1. L'adaptation de services

Lorsque l'on conçoit un système informatique, il est bien souvent difficile de prendre en compte tous ses cas d'utilisation. En effet, même en ayant effectué un recensement avancé des besoins dans le cahier des charges du système, il arrive parfois qu'un cas imprévu se présente. Pour répondre à ce nouveau besoin, il faut alors essayer d'adapter le système. Bien entendu, ces adaptations sont possibles jusqu'à un certain degré, au-delà duquel, tout le système est remis en cause.

L'étude des systèmes adaptatifs est une branche très importante de la recherche en intelligence artificielle. En effet, l'adaptation reste un problème complexe et toujours ouvert. D'un point de vue fonctionnel, nous pouvons définir l'***adaptation*** d'un système comme étant une modification dans le traitement réalisé, induite par un changement de propriétés des données fournies, des résultats attendus et/ou des conditions de fonctionnement et ceci, de sorte à atteindre un objectif donné.

Dans cette définition, nous sous-entendons que l'adaptation et la réutilisation de systèmes sont fortement liées, car il s'agit bien d'assurer une certaine pérennité. Dans le cas des agents, et donc pour des entités autonomes, la modification du traitement se traduit par un changement motivé de son comportement : une ***auto-adaptation***. Afin de réaliser un système adaptatif, il faut déterminer quels sont les points sur lesquels on souhaite avoir une adaptation, puis choisir quel type de méthode utiliser. Dans le cas de systèmes interagissant avec des humains, l'adaptation peut être vue comme *la modification du comportement en considérant à la fois les besoins individuels des utilisateurs humains et les conditions spécifiques à l'environnement de l'application.*

Dans notre approche, les services font interagir des agents contrôlés avec d'autres agents pour lesquels nous avons une connaissance a priori limitée : il s'agit des agents partiellement contrôlés et non contrôlés.

Nous nous sommes plus particulièrement intéressés à deux problèmes d'adaptation :

- Comment adapter le comportement des agents contrôlés en fonction de celui des autres agents et de leurs caractéristiques ?
- Comment surveiller le bon déroulement d'un service pour y détecter d'éventuels problèmes et tenter d'y remédier ?

2. L'adaptation aux autres agents

L'aspect social du comportement d'un agent contrôlé nous amène à la prise en compte des autres agents. Dans cette partie, nous allons nous pencher sur ce problème qui concerne la représentation qu'un agent a des autres entités avec lesquelles il doit interagir.

2.1. Motivations pour la modélisation des agents

Par définition, le comportement des agents partiellement et non-contrôlés n'est pas connu, tout du moins avec une certitude suffisante. Or, si nous examinons les objectifs à atteindre dans les services de communication multimédia, il s'agit d'assister les utilisateurs et de les guider dans leurs tâches, voire de réaliser, pour eux, ces mêmes tâches. Il est donc nécessaire de savoir comment interagir au mieux avec eux. Il faut, dans certains cas, prévoir leurs réactions, pouvoir se rappeler leurs préférences, mais aussi, savoir comment présenter les informations en utilisant leurs références. Dans le cadre d'applications commerciales, on souhaite également faire aux clients des propositions les plus proches possibles de ce qu'ils souhaitent et qui répondent au mieux à leurs besoins. Hormis les utilisateurs, il est aussi possible de considérer un humain jouant le rôle d'un guide touristique expert, comme dans le service Nancy-Tour présenté au chapitre 3. Cet agent partiellement contrôlé, bien qu'étant supposé avoir un comportement plus fiable, peut, lui aussi, avoir besoin d'une interaction adaptée, voire personnalisée.

Adapter un système à ses utilisateurs nécessite de disposer des connaissances adéquates sur ces derniers. L'ensemble des connaissances relatives à un utilisateur donné est appelé un ***modèle utilisateur*** ou encore un ***profil utilisateur***.

2.2. Modélisation des utilisateurs

De nombreux travaux de recherche ont attrait à la modélisation des utilisateurs et pour avoir un aperçu du domaine, on peut se reporter à *(Jameson 1999)*. La modélisation des utilisateurs est utilisée dans de nombreuses applications, comme la recherche d'informations ou les systèmes de dialogue, mais aussi pour des applications commerciales. Les systèmes d'enseignement assisté par ordinateur ont également contribué au domaine, au travers de la modélisation d'étudiants, comme on le retrouve dans *(Auberger 1998)*.

2.2.1. Définition

Dans *(Wahlster et Kobsa 1986)*, les auteurs ont proposé une définition de la modélisation des utilisateurs pour les systèmes de dialogue qui a été généralisée par *(Errico 1997)* :

Un modèle utilisateur est une source de connaissance qui contient des acquisitions sur tous les aspects de l'utilisateur qui peuvent être utiles pour le comportement du système.

Selon cette définition, le contenu du modèle utilisateur dépend des besoins du système et donc des objectifs de l'adaptation. Nous pouvons, par exemple, retrouver les objectifs suivants :

- La modélisation des utilisateurs peut être utilisée pour adapter la forme de l'interaction. C'est, entre autres, le cas lorsque l'on souhaite permettre l'accès à l'information pour les personnes nomades ou handicapées. La modélisation portera alors sur les caractéristiques et les préférences des utilisateurs.

- Si on souhaite adapter le déroulement de l'interaction pour un utilisateur donné et, par exemple, tenir compte de ses réactions face au système, on peut tenter de modéliser son comportement.
- Dans d'autres cas, on peut devoir proposer des offres ou des réponses ciblées. Pour atteindre cet objectif, on peut essayer de reconnaître les buts ou de mémoriser les sujets d'intérêt de l'utilisateur.
- Dans les systèmes de tutorat ou d'assistance, le système peut analyser la façon de procéder de l'utilisateur et lui proposer, par exemple, l'usage d'une autre méthode. Dans ce cas, nous avons plutôt affaire à une reconnaissance de plan.
- Le système peut enfin avoir à collaborer avec l'utilisateur pour accomplir une tâche. Dans ce cas, une modélisation est également nécessaire pour ne pas se gêner mutuellement et travailler de concert, tant au niveau des objectifs, que pour les plans d'actions.

Sur le schéma présenté Figure 38, nous avons représenté les différents stades de la modélisation d'un utilisateur. Tout d'abord, il faut déterminer quels sont les phénomènes auxquels on souhaite s'adapter. Il faut ensuite choisir une technique d'acquisition et une façon de représenter la connaissance. On dispose alors de différentes méthodes d'inférence pour réaliser une adaptation. Nous allons tenter de dresser un rapide aperçu de ces éléments.

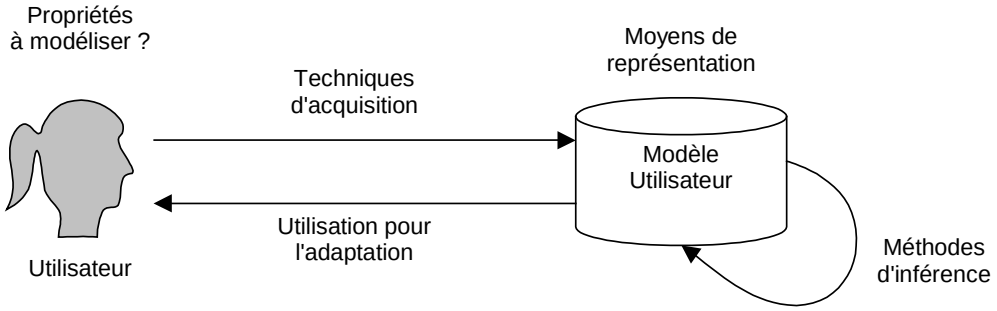


Figure 38 - Le cycle de la modélisation utilisateur

2.2.2. Aspects légaux et éthiques

De par sa nature, la modélisation des utilisateurs est soumise à des aspects légaux. En effet, il est important de rappeler les dispositions de la loi "Informatique et Libertés" *(JORF 1978)* qui stipulent notamment que tous les fichiers et traitements automatisés d'informations nominatives doivent être déclarés à la Commission Nationale de l'Informatique et des Libertés (CNIL). Lors de la collecte d'informations nominatives, il est, de plus, obligatoire d'informer les usagers de leurs droits, et entre autres, de permettre l'accès et la modification des données personnelles, etc. Il est donc primordial de prendre connaissance des différents textes de loi, en particulier si les modèles utilisateurs sont nominatifs, incluent des informations personnelles et/ou conservent des données dans un historique.

Par ailleurs, au-delà de l'aspect légal, il est important d'assumer une certaine déontologie pour que la relation entre l'utilisateur et le système puisse se dérouler le mieux possible. Comme le souligne *(Kobsa 1990)*, il est, non seulement important d'informer l'utilisateur de la présence d'un système de modélisation, mais aussi de l'avertir des risques d'erreurs induits par son utilisation. Il conseille d'en permettre la désactivation, même si les performances peuvent s'en trouver dégradées. Il explique enfin que la conservation et l'utilisation des données sur du long terme n'ont de sens que pour des tâches comparables et qu'elle peut apporter des problèmes de péremption.

2.2.3. Propriétés modélisées

Dans *(Jameson 1999)*, l'auteur liste un ensemble de propriétés qui peuvent être utilisées pour la modélisation des utilisateurs :

- les caractéristiques personnelles qui peuvent influencer fortement l'interaction (âge, sexe, etc.),

- les intérêts et préférences générales liées à la tâche à accomplir, qui vont permettre une adaptation assez fine aux attentes de l'utilisateur,
- les compétences ou le niveau d'expertise dans un domaine donné, qui seront essentiellement utilisés pour déterminer les besoins de formation,
- les capacités non cognitives (comme les handicaps moteurs ou visuels), qui sont à prendre en compte au niveau des interfaces de communication homme-machine,
- le but courant, dont l'importance est forte mais l'inférence souvent difficile,
- les croyances, d'une importance moindre et qu'il est tout aussi difficile d'appréhender,
- les habitudes comportementales, qui concernent des cas particuliers mais qui peuvent être apprises, à condition de disposer de suffisamment de données observées, et
- les états psychologiques (comme le stress) et physiologiques (comme la fatigue), sur lesquels il reste encore difficile de se baser pour agir.

Le ***contexte*** de l'interaction est également cité par Jameson. Il s'agit là, à notre sens, d'une propriété constituant une extension du modèle utilisateur qui peut, selon le cas, nécessiter un module à part entière dans un système adaptatif. En effet, bon nombre d'éléments, comme la localisation spatio-temporelle, peuvent être pris en compte à ce niveau, surtout si on s'intéresse à la mobilité.

Le contexte est toute l'information qui peut être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un endroit ou un objet qui est considéré comme pertinent pour l'interaction entre un utilisateur et une application, en incluant l'utilisateur et l'application (Dey et Abowd 2000).

Selon leur importance vis-à-vis de l'application cible, les informations sur le contexte peuvent également entrer dans la composition des états du système modélisé. Les travaux relatifs à ce sujet parlent alors de context-awareness et c'est notamment l'objectif du projet IST Ozone auquel nous avons participé.

Un système est dit "context-aware" s'il utilise le contexte pour fournir des informations pertinentes et/ou des services à l'utilisateur, où la pertinence dépend de la tâche de l'utilisateur (Dey et Abowd 2000).

2.2.4. Les différents types d'acquisition utilisés

La nature de l'interaction entre l'utilisateur et le système détermine fortement quelles approches il est possible de mettre en œuvre : c'est en effet la clef du succès de l'adaptation. En particulier, la capacité de perception du système est un facteur déterminant quant aux informations obtenues par observation. Ensuite, selon la méthode utilisée, l'acquisition et l'exploitation de connaissances sur un utilisateur peuvent être plus ou moins complexes. Classiquement, on retrouve les méthodes suivantes :

- Sélection d'un profil statique : Dans cette approche, il est demandé à des experts de définir des catégories d'utilisateurs qui, une fois constituées, sont figées pour tout le temps d'utilisation du système. Lorsqu'il commence à interagir avec le système, l'utilisateur doit choisir l'un des profils selon ses préférences. Il s'agit de l'approche la plus simple qui donne des informations typiques mais qui souffre de nombreux inconvénients, car elle ne permet pas une adaptation souple. De plus, la constitution d'un ensemble consistant de profils n'est pas toujours chose facile. Les profils statiques sont souvent utilisés dans les logiciels pour déterminer un niveau d'aptitude : novice, débutant, initié, etc.
- Constitution d'un profil par réflexivité : Ici, l'acquisition des connaissances se place généralement en début d'interaction. Cette approche est dite réflexive, car c'est l'utilisateur qui doit configurer son profil en se décrivant. Le plus souvent, il devra remplir un formulaire ou répondre à des questions. Le système enregistre alors le profil dans une base de connaissances. La combinaison des réponses possibles permet d'établir des profils plus complexes. En pratique, l'obtention d'informations par ce moyen est plutôt aisée, cependant, elle dépend de la capacité, et naturellement du bon vouloir de l'utilisateur, à donner un ensemble cohérent de réponses. En effet, certains champs peuvent être difficiles à remplir avec précision, surtout s'il y a un risque de mal

comprendre l'énoncé. Il s'agit d'une méthode d'acquisition classique que l'on retrouve très souvent sur les sites web adaptatifs.

- Constitution progressive par évaluation : Dans ce type d'acquisition d'informations, il est demandé à l'utilisateur d'effectuer l'évaluation des objets spécifiques de façon répétitive. La méthode a l'avantage d'être peu coûteuse pour l'utilisateur, par rapport au remplissage de formulaire, car il s'agit de réactions spontanées. Par exemple, dans *(Rusmevichentong et Van Roy 2001)*, les auteurs proposent de trouver quelle est la meilleure séquence de produits à proposer. Cette méthode est typiquement utilisée pour la proposition de produits commerciaux comme les livres, la musique, les programmes télévisés, etc.
- Constitution du profil par observation des actions : Afin de concevoir des systèmes d'aide aux utilisateurs, qui ne sont pas forcément experts dans le domaine de l'application, on peut tenter d'apprendre un modèle d'un utilisateur en observant fréquemment son comportement. Par la suite, on peut proposer à l'utilisateur de retenir les informations dans son modèle. Dans certains cas, la mise à jour peut être automatisée, afin d'éviter de solliciter l'utilisateur trop souvent, ce qui rend la modélisation plus transparente. Par exemple, si on s'aperçoit qu'il effectue systématiquement une action β après avoir effectué une action α, alors on peut proposer d'effectuer β automatiquement après avoir observé α. Cependant, mis à part pour des domaines rigoureusement restreints et pour un faible nombre de buts possibles, il est quasiment impossible d'apprendre les intérêts de l'utilisateur seulement à partir d'observations, sans information a priori et sans interagir avec lui. On retrouvera dans cette catégorie les systèmes éducatifs et d'assistance en ce qui concerne les aides méthodologiques, comme *(Horvitz et al. 1998)*.
- Constitution de profil par approche communautaire : Dans certains cas, il est difficile d'acquérir des informations sur un individu particulier, surtout lorsque les interactions que l'on a avec lui sont rares ou que le nombre d'utilisateurs à gérer devient important. Une possibilité est alors de tenter de constituer des profils moyens correspondant à des groupes d'individus. L'avantage est qu'une information acquise sur le groupe peut être répercutée (de façon plus ou moins importante) sur l'individu et vice-versa. C'est ce principe qui est appliqué dans le filtrage collaboratif *(Shardanand et Maes 1995)*.
- Il est également possible d'utiliser un certain nombre de mesures physiques issues de capteurs. On obtient des résultats très variables selon la nature, la précision et la technologie du capteur employé. A titre d'exemple, on peut mentionner l'utilisation de capteurs GPS pour détecter la position de l'utilisateur et adapter une présentation, comme dans l'application Nancy-Tour. Un autre exemple est l'utilisation de mesures quotidiennes (poids, tension, etc.) pour le suivi de patients à domicile, comme dans Diatélic *(Jeanpierre 2002)* et *(Bellot 2002)*. A nouveau, on conviendra de déterminer dans les données obtenues ce qui dépend de l'utilisateur de ce qui dépend du contexte d'utilisation du système.

2.2.5. Inférences sur les données acquises

Une fois les données obtenues, il est possible d'utiliser différentes méthodes d'inférence, comme le montre la classification de *(Jameson 1999)* :

- Actuellement, les modèles graphiques, issus de la théorie de la décision, comme les Réseaux Bayésiens *(Pearl 1988)* et les Diagrammes d'Influence *(Howard et Matheson 1981)* connaissent un succès croissant. Ces modèles mixtes (mêlant le numérique et le symbolique) sont tout à fait appropriés pour des raisonnements causaux et probabilistes et ont l'avantage, du fait de leur nature, de gérer l'incertitude sur les données. Ils souffrent cependant d'une complexité pouvant être NP-Difficile lors d'inférences exactes et nécessitent l'établissement de la structure avec des connaissances expertes. De plus, l'apprentissage du modèle doit être effectué auparavant et nécessite une quantité importante de données. Ces approches ont été utilisées dans de nombreux travaux, notamment pour la reconnaissance de but *(Gonzales et Wuillemin 1998)*, *(Horvitz et al. 1998)* et pour la reconnaissance de plans *(Huber et al. 1994)*, *(Albrecht et al. 1998)*.
- On retrouve ensuite les méthodes d'apprentissage numérique. Dans ces méthodes, une grande quantité de données étiquetées issues des sessions (appelée corpus) est utilisée pour effectuer

l'apprentissage hors ligne à l'aide d'un étiquetage ou d'un critère de performance. Parmi les plus classiques, on peut citer :

- ★ l'apprentissage de règles à base d'arbres de décision *(Quinlan 1986)*, qui est utilisé pour prédire les décisions de l'utilisateur et pour définir le comportement de certains assistants personnels;
- ★ l'apprentissage à base de réseaux de neurones de type perceptron *(Rosenblatt 1959)*, qui permet, par exemple d'évaluer comment l'utilisateur considère l'importance d'une caractéristique du domaine et
- ★ l'apprentissage probabiliste, qui est utilisé pour effectuer le filtrage de documents auxquels l'utilisateur donne une notation.

- Enfin, les méthodes les plus classiques sont les méthodes symboliques, avec entre autres celles à base de logique, qui permettent, à partir de faits observés de poser et de vérifier des hypothèses sur l'utilisateur avec des méthodes d'inférence valides. Ces dernières sont utilisées par exemple dans des interpréteurs de commandes adaptatifs *(Pohl 1997)*. Elles impliquent cependant un compromis entre expressivité et complexité, selon la puissance du langage logique utilisé et ne sont pas initialement prévues pour prendre en compte toute l'incertitude inhérente à la modélisation des utilisateurs. Parmi les autres méthodes symboliques, on trouve aussi les stéréotypes *(Rich 1979)*, qui sont des ensembles de caractéristiques avec des déclencheurs qui amènent à d'autres stéréotypes et qui sont utilisés pour effectuer de l'adaptation hypertexte.

2.2.6. Représentation des connaissances

Les connaissances acquises sur un utilisateur sont généralement stockées dans une structure appelée modèle ou profil utilisateur. L'une des représentations la plus simple des données personnelles prend la forme d'un ensemble de couples (attributs, valeur). A titre d'illustration, on trouve ce type d'informations au niveau des logiciels de messagerie Internet, comme la carte de visite informatique vCard *(IMC 1996*) dont un exemple est donné ci-dessous :

```
BEGIN:VCARD
VERSION:2.1
N:MANVUSSA;Gérard;;Mr
FN:Gérard MANVUSSA
TEL;HOME;VOICE:01.98.76.54.32
TEL;CELL;VOICE:09.51.35.72.46
TEL;HOME;FAX:01.23.45.67.89
ADR;HOME:;;42, Rue de l'Arche;PIERREVILLE-SUR-ARCHES;75;75999;FRANCE
LABEL;HOME:42, Rue de l'Arche PIERREVILLE-SUR-ARCHES, 75 75999  FRANCE
URL;HOME:http://manvussa.mapageperso.fr
EMAIL;PREF;INTERNET:Gerard.Manvussa@loa.fr
REV:20011116U232826Z
END:VCARD
```

Dans les systèmes adaptatifs, on trouve aussi des structures plus complexes, organisées de façon hiérarchique, ou des ensemble de faits dans un système logique, etc. Pour modéliser l'incertitude, on peut également stocker une distribution de probabilité sur les valeurs possibles de la structure.

2.2.7. Passage à l'échelle et aspect collectif

Il est rare qu'un système informatique soit destiné à un seul utilisateur. Aussi, en se plaçant dans une optique de passage à l'échelle, il faut considérer le cas où le système doit gérer plus d'un utilisateur. Des points communs et des différences apparaissent alors entre les différents profils. De plus, les problèmes de représentation et de stockage des connaissances s'amplifient. Il devient difficile d'envisager une gestion centralisée de profils individuels complets. Il est alors possible, soit de distribuer les profils, ce qui est la politique suivie par de nombreux sites web qui utilisent les "cookies", ou d'envisager une généralisation des connaissances. On peut se demander ce qu'une session a de spécifique et si cette spécificité est due à l'utilisateur courant ou à un ensemble d'utilisateurs. Une idée est d'effectuer une classification des utilisateurs, ce qui permet de travailler sur

un profil typique moyen au lieu d'une multitude de profils individuels. Une exploitation typique de l'aspect communautaire pour l'adaptation est le filtrage collaboratif *(Shardanand et Maes 1995)*, où la similarité des intérêts d'un utilisateur avec ceux des autres est exploitée pour recommander des documents.

2.3. Modélisation du comportement et reconnaissance de plans

Dans le chapitre 4, nous avons décrit comment un agent pouvait produire un comportement pour atteindre un but donné par planification. Le procédé complémentaire existe et consiste, à partir de l'observation du comportement d'un agent à reconnaître le plan qu'il suit ou le but qu'il cherche à atteindre. Ces deux champs de recherche sont très proches et la différence est que la ***reconnaissance de plans*** cherche à découvrir comment les agents se comportent pour atteindre leurs objectifs, alors que la ***reconnaissance de buts*** tente de comprendre quels sont les objectifs, en accordant moins d'attention à la façon dont ils sont atteints. Il s'agit de problèmes complexes, surtout pour des agents utilisateurs humains, car leur comportement est difficile à appréhender et à anticiper. Le principe de la reconnaissance de plan peut être illustré comme le montre la Figure 39.

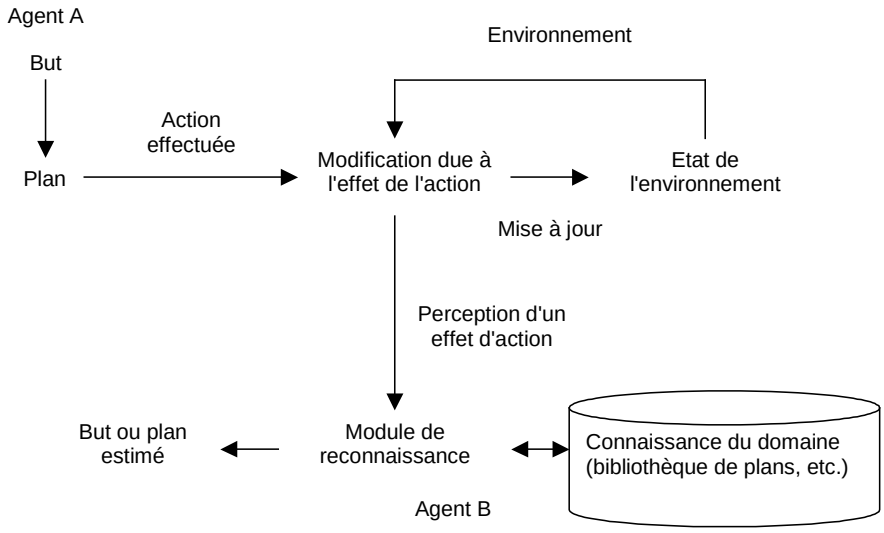


Figure 39 - Reconnaissance de plans par observation d'actions

Lorsqu'un agent ***A*** exécute un plan pour atteindre un but donné, il effectue des actions sur l'environnement. Cet environnement évolue et les effets des actions peuvent laisser des traces perceptibles. Un autre agent ***B*** peut alors observer ces traces, tenter d'inférer quelles ont été les actions effectuées par ***A*** et en déduire le plan suivi ou le but le plus probable à partir d'une connaissance du domaine. L'une des principales difficultés est qu'une même observation peut être compatible avec des actions différentes, donc à plus forte raison, à des buts ou des plans différents.

Dans *(Kautz 1987)*, l'auteur indique qu'il existe deux façons d'aborder la reconnaissance de plans :

- l'approche intentionnelle, où l'agent observé participe et essaye de rendre ses actions les plus claires possibles, et
- l'approche dite du "trou de serrure", où les actions de l'agent sont observées sans possibilité d'interaction, comme si on regardait ce qui se passe dans une pièce par le trou d'une serrure.

Les systèmes de reconnaissance de plans se distinguent ensuite selon que la connaissance du système est complète ou non sur le domaine et selon que les plans sont exécutés par l'agent observé avec ou sans erreurs. Dans le cas d'interactions avec des humains, la reconnaissance de plans humains est très difficile à réaliser, à moins de se placer dans des conditions très particulières où, par exemple le nombre de possibilités est très limité. Tout d'abord, *(Wærn 1996)* préconise l'usage conjoint de l'approche intentionnelle et de l'approche "trou de serrure". Ensuite, *(Lesh et al. 1999)* proposent de

rendre possible la reconnaissance de plan, en utilisant la notion de focus d'attention et la possibilité de demander des explications.

La connaissance du domaine de l'application prend généralement la forme d'une bibliothèque de plans qui énumère tous les plans possibles. Cette bibliothèque est soit constituée par des experts, soit acquise automatiquement à partir d'un corpus de traces d'actions, comme le propose *(Bauer 1998)*. Une fois la connaissance des buts/plans possibles acquise, on fait le plus souvent appel à des méthodes d'appariement de graphe pour essayer de faire correspondre la suite des actions observées aux plans de la bibliothèque. Les réseaux bayésiens sont également utilisés pour inférer les plans des utilisateurs et leurs buts. Dans ces approches, reconnaître un but nécessite de trouver une distribution de probabilité sur les buts possibles. Il faut rechercher quel but justifie le plus l'observation, en fonction du contexte de l'interaction.

2.4. Impact sur la conception de services adaptatifs

Dans nos travaux, nous nous sommes principalement intéressés à permettre aux agents contrôlés d'effectuer la modélisation des préférences des autres agents et en particulier lorsqu'il s'agit d'agents humains. Cependant, il est tout à fait envisageable de modéliser plus généralement les préférences et les comportements de chacun des agents avec lesquels les agents contrôlés interagissent.

Pour prendre en compte les variabilités dans le comportement des autres agents, nous avons choisi une autre approche que la modélisation explicite, en effet, comme nous l'expliquerons dans le chapitre 6, nous avons choisi un apprentissage par renforcement de stratégies de médiation, ce qui permet d'apprendre comment réagir face au comportement d'un utilisateur.

3. Rendre un système robuste aux aléas

Un système informatique est rarement à l'abri des divers aléas qui peuvent survenir pendant sa mise en œuvre. Les services auxquels participent les agents d'un hSMA ne font pas exception. Certains agents peuvent soudainement avoir des réactions tout à fait imprévues, de la même façon que certains évènements inattendus peuvent survenir. Si certains phénomènes sont anodins, d'autres en revanche, peuvent parfois avoir des conséquences graves. Nous avons donc examiné les moyens de détecter et de corriger les situations anormales dans le déroulement d'un service.

3.1. Suivi et diagnostic dans les systèmes dynamiques

Les travaux sur la surveillance de systèmes dynamiques de *(Basseville et Cordier 1996)* nous renseignent sur la nature et sur l'enchaînement des éléments qui peuvent être utilisés. Tout d'abord, il faut se doter d'un ensemble de capteurs capables d'extraire les informations caractéristiques des éléments à surveiller. Il faut analyser les données et détecter les perturbations et les anomalies. Toutes les données disponibles sont ensuite fusionnées, puis en fonction de ces informations, le système doit décider s'il y a lieu de générer une alarme ou un avertissement. En cas de déclenchement d'une alarme, il est alors nécessaire d'interpréter la situation, de localiser le problème et de tenter d'y remédier.

Comme on peut le constater, la chaîne de détection peut devenir complexe et un élément clef est la fonction de décision : c'est elle qui fait la part entre les situations normales et les situations anormales. Mettre au point cette fonction nécessite à nouveau des connaissances expertes dans le domaine, mais de nombreux travaux en intelligence artificielle ont cherché à la déterminer automatiquement. A titre d'exemple, nous pouvons citer les travaux sur le système d'aide à la décision Diatélic dans le domaine de la télé-médecine : *(Jeanpierre 2002)* et *(Bellot 2002)* utilisent respectivement les POMDP et les réseaux bayésiens. Le principe du diagnostic est d'apprendre un modèle à partir d'un corpus d'observations puis d'effectuer la classification d'une situation nouvelle en utilisant ce modèle. Si la situation a une forte probabilité d'appartenir à une classe d'état dangereuse, alors une alarme est déclenchée.

3.2. Inspiration des systèmes de détection d'intrusion

Un service peut être vu comme le cadre des interactions normales pour une application donnée. Tout comportement d'un agent du hSMA sortant de ce cadre peut porter atteinte à l'intégrité du système ou des données manipulées, mais peut aussi avoir d'autres conséquences, parfois graves, pour les autres agents. La détection des activités anormales d'un système informatique s'apparente fortement aux travaux autour de la détection d'intrusion. La principale différence étant que dans le cas d'une intrusion, il y a une intention délibérée de nuire au bon fonctionnement du système et que les actions sont souvent structurées avec éventuellement un effacement des traces.

Les travaux de *(Dasgupta et González 2002)* s'inspirent du fonctionnement du système immunitaire humain. Ils définissent l'ensemble des activités normales et autorisées comme étant le "self", le reste pouvant contenir aussi bien des attaques virales, des tentatives de piratage ou d'autres comportements malins. Ils proposent de représenter l'espace des états possibles du système comme un vecteur où chaque composante correspond à une caractéristique du système. Sur cet espace multidimensionnel, ils définissent une fonction d'appartenance floue au self, puis ils proposent deux techniques. La première, appelée "caractérisation positive" est relativement classique, car elle travaille sur des cas positifs et cherche à apprendre un modèle des situations normales du self. En revanche, la seconde, appelée "caractérisation négative", est plus originale, car elle cherche à doter le système de détecteurs de cas anormaux, d'une façon similaire aux cellules T du système immunitaire humain qui sont libérées par le thymus pour traquer les antigènes (corps étrangers attaquant le corps humain). Les détecteurs se présentent sous la forme de règles permettant de décider si la situation est normale et sont obtenus par une évolution à base d'algorithmes génétiques. Le principe, illustré dans la Figure 40 sur un espace à deux dimensions, est de couvrir l'ensemble de l'espace d'états par des hyper-rectangles correspondant chacun à un détecteur de situation anormale. Les détecteurs peuvent également être organisés en niveaux de gravité de situation autour du self, ce qui apporte une souplesse au niveau des alarmes.

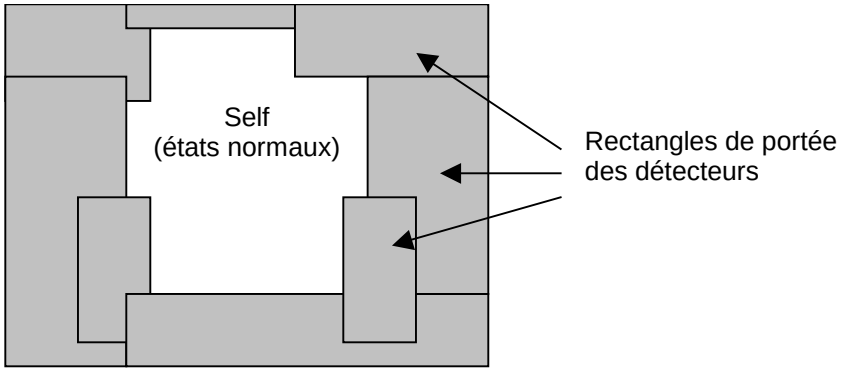


Figure 40 - Couverture par des détecteurs de situations anormales *(Dasgupta et González 2002)*

Dans *(Crosbie et Spafford 1996)*, nous retrouvons le même principe consistant à faire évoluer une population de détecteurs, à la différence que les auteurs travaillent sur des combinaisons d'évènements observés et que les détecteurs sont des Réseaux de Pétri Colorés *(Jensen 1994)* dont les parties action peuvent être différentes d'un détecteur à l'autre. Ainsi, lorsqu'une combinaison d'évènements anormale se produit, la réponse du système peut être adaptée au problème plutôt que de déclencher une alarme générique.

La difficulté est cependant que bon nombre des méthodes que nous avons évoquées nécessitent un apprentissage sur un corpus étiqueté, ce qui limite quelque peu l'adaptation du système de façon dynamique. En revanche, l'idée de caractériser le niveau de normalité et de se doter de détecteurs autonomes et évolutifs reste séduisante.

3.3. Suivi des interactions d'un hSMA

En nous replaçant dans le problème de l'adaptation aux aléas dans le cadre des hSMA, nous pouvons tenter de surveiller les comportements exhibés par les agents. L'ensemble des actions que ceux-ci effectuent provoquent des effets dans l'environnement comparables à des traces de cheminement en direction de leurs buts : les agents vont chercher à remplir leur rôle, mais ils vont se comporter de façon plus ou moins indépendante et chaotique, selon qu'ils sont contrôlés, partiellement contrôlés ou non contrôlés. A l'instar des travaux que nous venons de présenter, nous pouvons tenter d'utiliser ces traces pour analyser l'exécution des services et décider si une interaction se trouve dans un état normal. Le diagramme de la Figure 41 montre un processus de suivi possible.

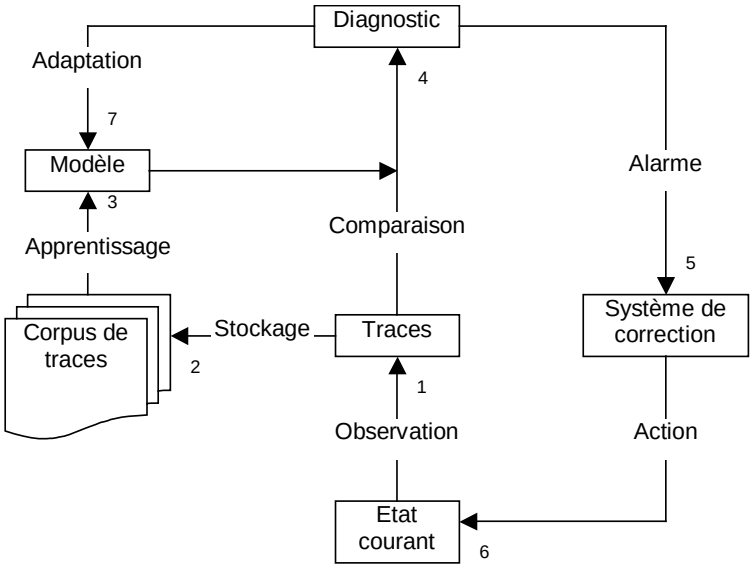


Figure 41 - Suivi et correction d'un service

La première étape (1) consiste à observer les évènements qui se produisent à chaque instant du déroulement de l'interaction. Ces flux d'évènements fournissent les traces comportementales sur les chemins que les agents ont parcourus. Les chemins sont alors stockés (2) de sorte à acquérir un corpus. Le corpus est utilisé pour apprendre (3) un modèle de normalité du service. Après avoir obtenu ce modèle, on peut alors l'utiliser pour le comparer aux traces observées pour l'état courant lors d'une exécution particulière. Le résultat de cette comparaison (4) est un diagnostic de ce qui se passe dans l'état courant. On peut détecter si tout se passe normalement ou s'il y a une différence. A partir de ce point, s'il y a une différence significative et donc un risque, le système de suivi doit déclencher une alarme (5) et tenter de corriger l'état courant avec une ou plusieurs actions spécifiques (6). Le système de diagnostic peut aussi réaliser que l'état courant n'est pas pris en compte par le modèle de normalité alors qu'il est correct, c'est pourquoi il doit être en mesure (7) d'adapter le modèle courant.

3.4. Suivi dynamique du comportement d'un agent

En première approche, on peut faire l'hypothèse qu'il existe un seul agent dont l'état interne est totalement observable. En reprenant l'idée des systèmes de détection d'intrusion, on peut utiliser un espace d'état multidimensionnel pour représenter l'état interne de l'agent. Afin d'avoir une vision plus intelligible et de se rapprocher de problèmes connus, nous allons utiliser l'image métaphorique d'un robot qui se déplace dans un environnement physique incertain en tentant d'atteindre un but, tout en cherchant à éviter les obstacles qu'il peut rencontrer.

Dans la Figure 42, le robot est parti de l'état initial et peut suivre divers chemins, dessinés en gris, qui peuvent être qualifiés de normaux, pour atteindre l'état but. Tout au long de ces chemins, les petits cercles représentent des points de reprise pour lesquels l'on estime que la situation est normale et sous contrôle. Les états dangereux sont marqués par les zones grisées et l'agent doit les éviter à tout prix. Dans l'exemple, le chemin noir est la trajectoire actuellement suivie par le robot. Sur l'exemple, on

peut observer qu'il se dirige vers un état dangereux, aussi, il est nécessaire de corriger sa direction pour le ramener vers un point de reprise sauf. Une correction possible est représentée par une flèche en pointillés.

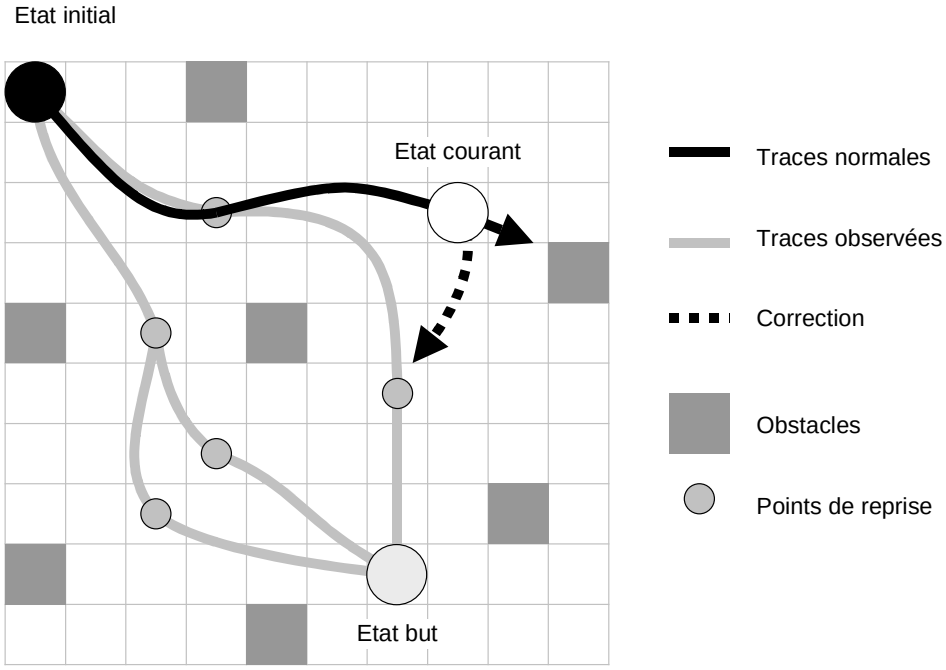


Figure 42 - Suivi de la trace d'un agent robotique dans un environnement

Nous nous retrouvons dans le domaine des problèmes typiques de navigation en robotique qui ont été largement étudiés et pour lesquels les approches par modélisation stochastique donnent de très bons résultats. En se dotant d'une fonction de récompense sur l'espace d'état donnant des valeurs positives pour les états buts et négatives pour les obstacles, nous obtenons une caractérisation des états selon leur utilité. La Figure 43 est une capture d'écran donnant une politique stochastique pour chaque état de la carte précédente : dans chaque case, la longueur des bras de la croix indique la probabilité de choisir l'action de déplacement dans la direction correspondante. Cette politique est obtenue à l'issue d'un apprentissage par renforcement de type Q-Learning. Afin de simuler les incertitudes dans l'exécution des actions de déplacements, nous avons utilisé le modèle de glissement de la Figure 44. On s'aperçoit qu'avec le comportement appris, l'agent est capable d'atteindre le but et d'éviter les obstacles sur son chemin, tout en contrecarrant l'incertitude sur le résultat des actions. Bien entendu, pour des états visités peu fréquemment, la politique est toujours relativement risquée.

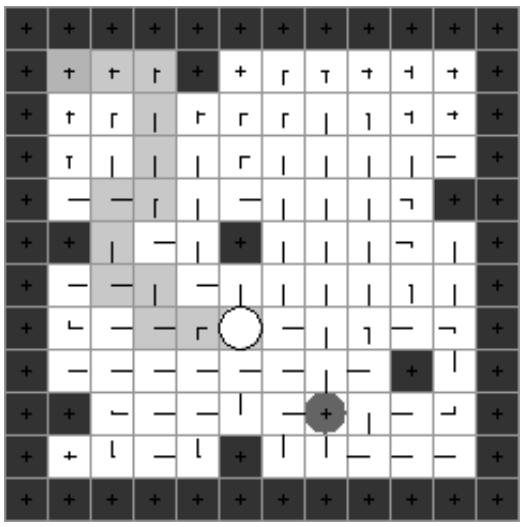


Figure 43 - Un exemple de politique stochastique dans un monde 2D

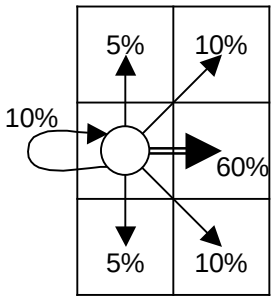


Figure 44 - Erreur sur le résultat d'une action de déplacement

Ces techniques donnent de bons résultats sur des problèmes jouets, mais pour les utiliser dans la modélisation de services, il faut couper fortement dans la complexité. En particulier, nous ne retrouvons pas forcément les propriétés de la topologie de l'environnement, ce qui multiplie les effets possibles d'une action. Comme le coût des calculs croît de façon exponentielle, il faut utiliser un haut niveau d'abstraction pour limiter l'ensemble des états et des actions et ainsi rendre les problèmes plus accessibles.

3.5. Suivi d'une interaction Multi-Agent

Dans le cas d'un hSMA, les services font coopérer des agents qui doivent atteindre un état global pour lequel un sous-ensemble d'agents est satisfait par l'achèvement commun d'un but applicatif. Le suivi du bon déroulement d'un service consiste à observer l'état de chaque agent vis-à-vis de son rôle respectif.

Pour mieux comprendre, nous allons compliquer le problème du paragraphe précédent en considérant l'interaction de deux agents ***A*** et ***B***, montrée Figure 45, où ***A*** est non-contrôlé et où ***B*** est contrôlé. Ces agents évoluent en parallèle dans des sous-espaces non disjoints de l'environnement et peuvent se synchroniser au travers de leurs interactions qui se déroulent dans les parties de l'environnement qu'ils ont en commun : une action produite par un agent peut influencer le comportement d'un autre agent. Supposons que l'agent ***A*** soit un humain jouant un rôle d'utilisateur et que l'agent ***B*** soit en charge de la fourniture du service et de son bon déroulement. Si le comportement de ***A*** sort de la normalité, ***B*** doit faire son possible pour corriger la situation. Cependant, ***B*** ne peut pas toujours avoir une influence directe sur l'état de ***A***, mais il peut, en revanche, tenter d'amener ***A*** vers son état de satisfaction en l'influençant au travers des interactions qu'il a avec lui. C'est sur ce problème d'interaction adaptative entre un agent contrôlé médiateur et un agent utilisateur que nous allons proposer une approche à base de MDP dans le chapitre suivant.

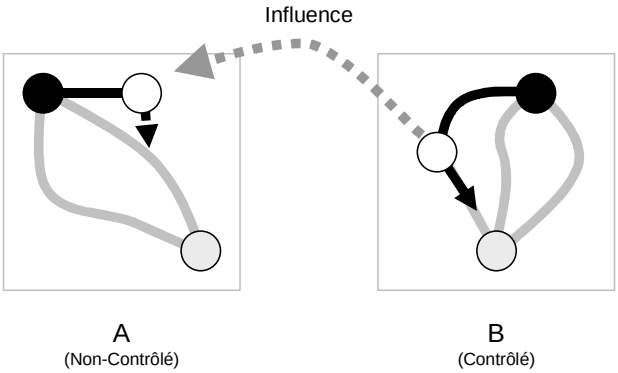


Figure 45 - Influence dans une interaction multi-agent

Lorsqu'il y a davantage d'agents dans le système, nous retrouvons des concepts comparables dans la classe des problèmes multi-agents classiques *(Stone et Veloso 2000)*. Par exemple, dans le domaine "proie-prédateur", quatre agents jouent le rôle de prédateurs qui tentent de capturer un agent qui joue le rôle de la proie en l'encerclant. En comparant avec notre problème, nous pouvons voir l'agent utilisateur ***A*** comme étant la proie, car il n'est pas contrôlable. La proie a un comportement typiquement aléatoire, mais peut tenter d'éviter sa capture. Les prédateurs sont des agents contrôlés, comme ***B***, qui doivent se synchroniser pour capturer la proie. Des approches, comme les processus de décision markoviens décentralisés, ont été proposées pour résoudre ce genre de problème *(Chadès et al. 2002)*, mais les problèmes deviennent très vite complexes, surtout dans les systèmes hétérogènes et on peut dire que l'apprentissage de comportements multi-agent qui se coordonnent reste encore un problème ouvert.

4. Conclusion

Dans ce chapitre, nous avons effectué un tour d'horizon des méthodes d'adaptation qu'il était possible d'utiliser pour prendre en compte les entités avec lesquelles les agents contrôlés d'un hSMA pouvaient être amenés à interagir. Nous avons abordé ce problème en nous ramenant aux travaux sur la modélisation des utilisateurs et de leurs comportements. Nous avons ensuite considéré le problème de l'adaptation vis-à-vis des risques pouvant survenir pendant l'exécution des services. Il y a de nombreuses façons d'aborder l'adaptation. Pour nos travaux, nous avons choisi d'utiliser un apprentissage par renforcement, car il s'agit d'un outil de production de comportement qui a l'avantage de continuer à s'adapter tout au long de leur utilisation, même si des périodes d'apprentissage préalables restent inévitables. Dans le chapitre suivant nous allons détailler comment nous avons utilisé une approche MDP et mis en œuvre nos idées pour construire un agent médiateur.

Chapitre 6 - La médiation pour les services de recherche d'informations

Dans le chapitre 3, nous avons décrit de façon générale notre solution théorique consistant à introduire un médiateur pour résoudre le problème de la coopération dans les hSMA. Nous allons maintenant examiner comment ce problème s'instancie dans la classe des services de recherche d'informations avec un agent humain, jouant un rôle d'utilisateur et un agent logiciel, jouant celui de la source d'informations.

Nous allons examiner en détails comment produire le comportement d'un agent médiateur en nous focalisant sur un type de service particulier : l'assistance à la recherche d'informations. Nous détaillons tout d'abord comment les systèmes de dialogue et plus précisément l'apprentissage de stratégies à base de modèles stochastiques nous ont inspirés pour la conception d'un agent médiateur. Dans la suite du chapitre, nous décrivons une application de réservation de vol d'avion sur laquelle nous avons travaillé, puis nous détaillons une spécification permettant de concevoir un médiateur à base de MDP.

1. La médiation dans les services de recherche d'informations

1.1. Principes de la recherche d'informations

Parmi les classes de service que nous avons définies, nous avons particulièrement étudié celle concernant la ***recherche d'informations***. L'exemple le plus courant est celui des moteurs de recherches sur Internet. Dans un système de recherche d'informations classique, illustré Figure 46, un programme d'accès à une base de données fait remplir un formulaire à l'utilisateur. L'utilisateur remplit les différents champs et demande la recherche. Le programme (souvent un script CGI, PHP, etc.) construit la requête à partir du contenu des champs, puis il se connecte à la base de données et exécute la recherche. Les résultats retournés par la base de données sont récupérés par le programme d'accès qui les affiche ensuite pour l'utilisateur (en générant, par exemple, une page web à la volée).

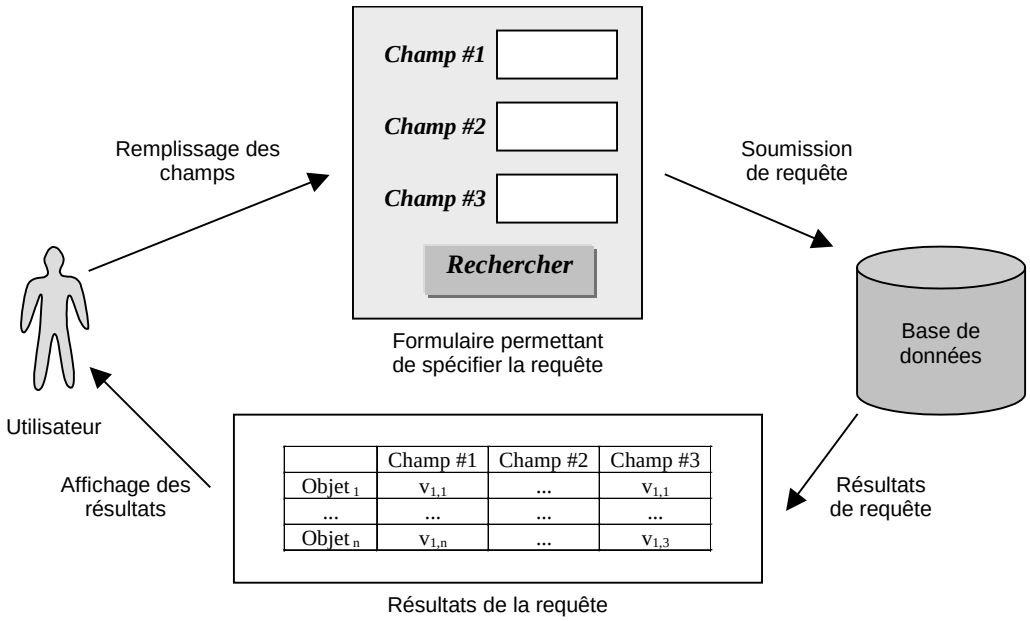


Figure 46 - Un système de recherche d'informations à base de formulaire

A l'origine, l'utilisateur a un besoin que le système d'informations peut satisfaire, mais que se passe-t-il si, comme sur la Figure 47, il n'arrive pas à formuler précisément sa requête ou à comprendre les résultats ?

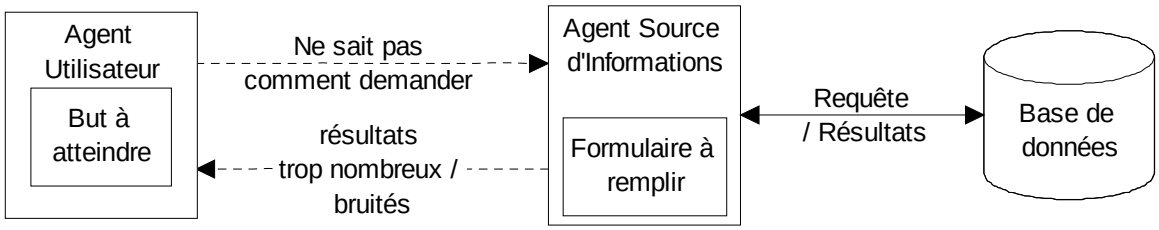


Figure 47 - Un problème de communication dans le cas d'une recherche d'informations

Les rôles qui interviennent sont les suivants : un rôle d'utilisateur et un ou plusieurs rôles de source d'informations. La solution que nous avons proposée dans le chapitre 3 est d'introduire un médiateur qui aide l'utilisateur à accéder aux sources d'informations. Ce médiateur amène les rôles à interagir selon le schéma Figure 48.

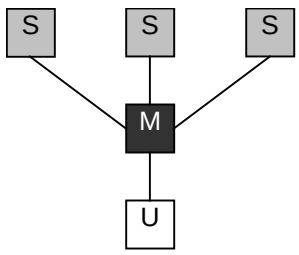


Figure 48 - Schéma d'interaction pour la recherche d'informations

Dans cette classe de service, le rôle du médiateur est de permettre à l'utilisateur de trouver des informations depuis l'une ou l'autre des sources, de façon totalement transparente, car il comble la différence de langage qui existe, en essayant d'établir un référentiel commun. Le médiateur passe, en effet, d'une communication de haut niveau, avec un humain, à un protocole de communication classique, quand il échange des messages avec les sources.

1.2. Hypothèses de travail

Dans la partie qui suit, nous nous intéresserons principalement aux interactions entre l'utilisateur et le médiateur, en faisant l'hypothèse que l'utilisateur est un agent humain et qu'une seule source de données est mise en œuvre. La ***médiation*** consiste à fournir et à adapter le service, c'est-à-dire aider l'utilisateur à trouver ce qu'il cherche, alors que ce dernier ne possède qu'une idée imprécise de son but ou rencontre des difficultés de formulation. Nous allons présenter une vision du service dans laquelle le médiateur tente d'établir une suite d'interactions du type question-réponse avec l'utilisateur, afin de surmonter la difficulté que celui-ci a pour exprimer ses buts. Le problème consiste donc à déterminer quelle est la meilleure séquence de questions à poser pour découvrir les buts. Les rôles des agents U, S et M sont les suivants :

- L'utilisateur (U), qui entre en contact avec le médiateur, a un besoin informationnel à satisfaire, mais il ne sait pas comment formuler sa requête, ni comment la source d'informations fonctionne. Il peut cependant répondre à quelques questions sur son but, mais pas à toutes, de plus, ses réponses peuvent être incertaines. En revanche, il peut évaluer les résultats qui lui sont présentés : cette évaluation est explicitement donnée au médiateur quand il sélectionne l'une des propositions.
- Le médiateur (M) doit découvrir comment la requête doit être formulée vers la source d'informations afin d'obtenir les données adéquates. Pour cela, il cherche à se positionner dans l'espace de recherche en posant des questions à l'utilisateur. Il est capable de proposer à l'utilisateur des sélections de données obtenues depuis la source d'informations.
- L'agent (S) qui gère la source d'informations est supposé avoir un comportement d'interface simple à une base de donnée. Nous supposerons que le médiateur et la source communiquent par échanges de requêtes et de résultats avec un langage de communication agent ou à l'aide d'un protocole plus classique.

Les applications cibles Dialoca étant axées en grande partie autour d'interactions vocales, ceci nous a amenés à nous placer dans le cadre d'un système de dialogue oral géré par le médiateur et dirigé par le but. Nous rejoignons en ce sens les travaux de *(Allen et al. 2000)*. Cependant, l'approche de médiation que nous proposons est généralisable à d'autres types d'interactions, car finalement le dialogue peut être considéré comme un cas particulier d'interaction.

2. Systèmes de dialogue pragmatiques

Dans cette partie, nous présentons le principe des systèmes de dialogues dirigés par le but. Cependant, comme il s'agit à nouveau d'un vaste champ de recherche, nous limitons notre description aux travaux relatifs à la prise de décision dans la gestion des interactions.

2.1. Définition

(Allen et al. 2000) indiquent que le terme de ***système de dialogue*** est utilisé de façon différente dans plusieurs travaux dont le point commun est d'interagir avec des êtres humains. En fait, le principal objectif est de permettre une interaction de haut niveau avec les utilisateurs, en utilisant par exemple le langage naturel, afin de réaliser diverses tâches. Les systèmes de dialogue auxquels nous nous intéressons gèrent des cycles de question-réponse pour identifier un but applicatif en optimisant des critères donnés, comme la longueur du dialogue ou l'usage des ressources, etc. Un exemple de but applicatif est le remplissage d'un formulaire fourni par un autre agent *(Goddeau et al. 1996)*. Les dialogues que nous considérons ont un aspect pragmatique (ce terme étant la traduction que nous estimons la plus fidèle de l'expression anglaise "practical dialogue") et donc relativement plus simples à traiter :

*La compétence conversationnelle requise pour les **dialogues pragmatiques**, bien que toujours complexe, est significativement plus simple à atteindre que celle d'une conversation humaine au sens général. (Allen et al. 2000)*

Dans les systèmes de dialogue pragmatiques, on peut également faire l'hypothèse d'une certaine indépendance du domaine, cependant, la réalisation de systèmes de dialogue génériques reste encore un problème ouvert dans le domaine.

Dans les dialogues pragmatiques, une grande partie de la complexité se situe dans l'interprétation du langage et la gestion du dialogue est indépendante de la tâche à accomplir. (Allen et al. 2000).

2.2. Approche Intuitive

Le système doit atteindre le but de l'utilisateur qui lui délègue en quelque sorte la tâche à réaliser. Tout d'abord, il faut que le système décode et analyse les informations transmises par l'utilisateur. Une fois ces informations obtenues, il faut déterminer ce qu'elles impliquent : le système doit les interpréter et maintenir à jour la connaissance qu'il a de la tâche à réaliser. En fonction de son estimation de la tâche et de l'achèvement de celle-ci, le système décide des actions à faire : il peut chercher à obtenir des informations complémentaires, tenter de réaliser la tâche avec celles dont il dispose ou encore proposer des résultats à l'utilisateur. Nous avons donc affaire à une reconnaissance de but couplée à une planification des actions requises pour atteindre ce but. Par ailleurs, le système doit également tenir compte de tout le contexte (espace-temps) dans lequel l'interaction se déroule.

2.3. Composants d'un système de dialogue

La Figure 49, représente la boucle générique qui se retrouve dans une majorité de systèmes de dialogue et qui n'est pas sans rappeler la boucle sensori-motrice que nous avons présentée dans le chapitre 1.

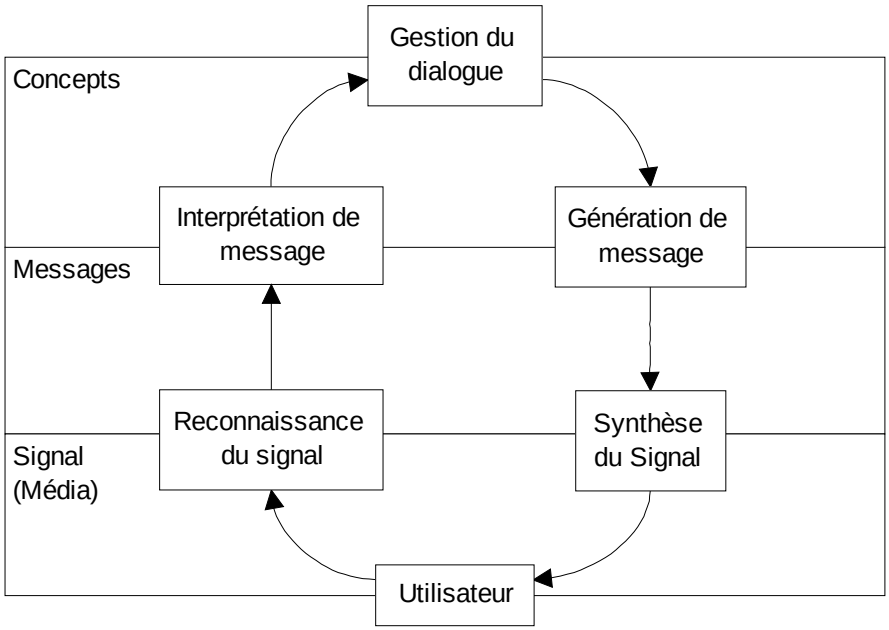


Figure 49 - Le cycle des systèmes de dialogue

Les systèmes de dialogue sont souvent "orientés parole" et font appel à la reconnaissance et à la synthèse automatique de la parole, mais ils peuvent être généralisés à d'autres médias. On peut distinguer deux parties dans la boucle. A gauche, un flux montant est constitué des signaux émis par l'utilisateur, auxquels viennent bien souvent se corréler divers types de bruits. Ces signaux sont reconnus et interprétés pour être transmis au gestionnaire du dialogue. A droite, l'autre partie de la boucle consiste à transmettre des messages à l'utilisateur en utilisant une synthèse de signal. Nous retrouvons donc les éléments suivants : une partie compréhension, qui interprète ce que dit l'utilisateur, une partie raisonnement, qui va prendre en compte les informations reçues et décider des actions à faire, enfin une partie pour la génération de réponse et une pour la production de signal.

Malheureusement, comme le souligne *(Heeman 2002)*, chaque partie constitue toujours un challenge et aucun de ces problèmes n'est complètement résolu.

2.4. Le contrôle dans les systèmes de dialogue

Afin de simplifier le traitement du langage, il est nécessaire d'utiliser un mécanisme pour contraindre l'interaction en restreignant les options possibles. C'est pourquoi, beaucoup de systèmes fonctionnent à l'aide de scripts et donnent le contrôle du dialogue au système avec des stratégies à base de prompts. Ces stratégies peuvent être représentées sous la forme d'un arbre, comme celui présenté Figure 50, dans lequel les nœuds sont des questions et les branches correspondent aux différentes réponses possibles.

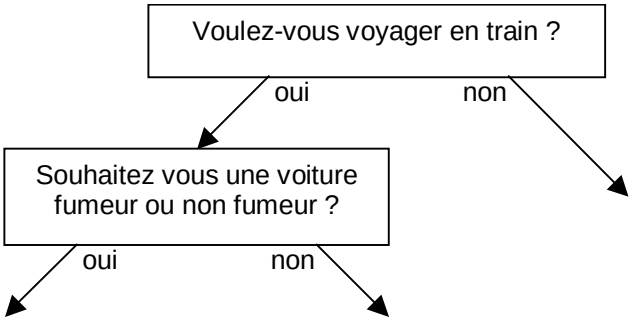


Figure 50 - Un dialogue arborescent à base de prompt

Une telle structure permet d'imbriquer récursivement des conditions du style *si-alors-sinon*, mais limite tout de même les possibilités de dialogue. Par la suite est apparu le principe d'initiative mixte, dans lequel l'utilisateur n'est plus aussi contraint et peut donner des informations sans qu'elles lui soient explicitement demandées. A tout instant, il peut fournir une infinité d'entrées possibles au système et, comme l'indiquent *(Levin et al. 1999)* cela devient très vite ingérable. Une solution est de procéder séquentiellement et de déterminer quelle est l'action suivante à effectuer, tout en fixant l'ordre de remplissage des champs par le script de stratégie. Cette solution est utilisée dans la plupart des systèmes mixtes comme Galaxy II *(Seneff et al. 1998)*, Amica *(Pieraccini et al. 1997)*, Railtel *(Bennacef et al. 1996)*, etc.

2.5. Approches pour construire des systèmes de dialogue

Une approche basique pour construire un système de dialogue est de définir le comportement en terme de machine à état fini. Dans ce cas, toutes les étapes sont complètement spécifiées par des états et des règles de transitions. On peut remarquer que c'est la solution qui est également utilisée pour spécifier le comportement d'agents simples. Cette approche fonctionne bien pour des tâches statiques mais manque d'adaptation et n'est pas adéquate pour traiter l'incertitude inhérente à la communication avec les humains.

Les arbres de décision *(Quinlan 1986)* ont également été utilisés. Il s'agit de modèles qui déterminent les questions les plus utiles à poser. Ils placent en premier celles qui discriminent au mieux l'ensemble des buts possibles par apprentissage sur un corpus étiqueté. Le problème est que les arbres de décision nécessitent que l'utilisateur connaisse la réponse à chaque question qui lui est posée et que l'ordre des questions est déterminé à l'avance. Nous pensons qu'une approche plus souple devrait permettre à l'utilisateur de donner des réponses du style "Je ne sais pas". C'est pourquoi le système doit savoir ce qu'il peut faire si une réponse est invalide, incomplète ou incertaine.

De nombreuses voies ont été explorées, et dans la dernière décennie, les travaux sur les méthodes probabilistes sont apparus et connaissent toujours un succès croissant. Actuellement, une voie prometteuse pour gérer les systèmes de dialogues est basée sur les modèles stochastiques *(Levin et Pieraccini 1997, Levin et al. 1998, Young 1999 et Young 2000)* comme les Processus de Décision Markoviens (MDP) et les Processus de Décision Markoviens Partiellement Observables

(POMDP). L'approche est particulièrement intéressante, car ces modèles sont bien adaptés pour prendre des décisions en présence d'incertitude sur les perceptions et sur le résultat des actions. De plus, dans *(Levin et al. 1998)* les auteurs ont montré que la conception des systèmes de dialogue est équivalente à un problème de MDP et que les solutions des MDP, comme l'apprentissage par renforcement, peuvent être appliquées pour apprendre des stratégies de dialogue. Ainsi, on dépasse les automates finis classiques, dont la stratégie n'évolue plus après la phase de conception.

2.6. Modélisation de dialogues avec des MDP

Dans *(Young 1999)*, l'auteur donne une représentation fonctionnelle d'un système de dialogue parlé, illustrée Figure 51.

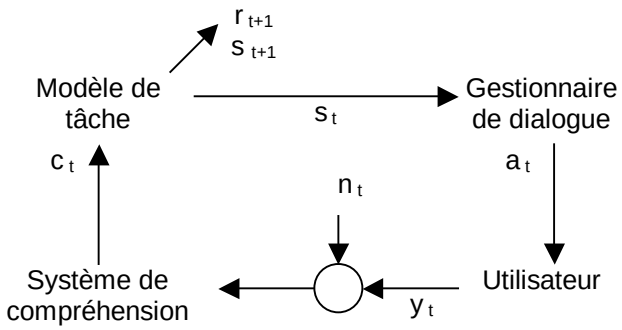


Figure 51 - Vision fonctionnelle d'un système de dialogue *(Young 1999)*

En commençant en haut à gauche au temps ***t***, le gestionnaire de tâche transmet son état d'avancement s_t au gestionnaire de dialogue. Le gestionnaire de dialogue décide en fonction de cet état quelle doit être l'action a_t à réaliser. Il s'agit le plus souvent d'envoyer une question à l'utilisateur à l'aide d'un prompt vocal, mais il peut également s'agir d'accéder à une base de données, etc. Lorsque l'utilisateur répond, il produit un signal vocal y_t. Ce message n'arrive généralement pas intact au système, mais subit le plus souvent une altération avec un bruit blanc ambiant $\mathbf{n}_t$. Lorsqu'il arrive dans le système de dialogue, le signal est traité par un système de compréhension qui l'analyse et transmet le ou les concepts reconnus c_t au gestionnaire de tâche. L'arrivée de concepts dans le gestionnaire de tâche déclenche une mise à jour des connaissances, ce qui donne lieu à un nouvel état s_{t+1} et à la génération d'une récompense r_{t+1}, prise en compte à l'étape suivante.

Dans *(Levin et Pieraccini 1997)*, les auteurs avaient déjà proposé l'utilisation des MDP pour gérer les interactions homme-machine et l'utilisation de l'apprentissage par renforcement pour apprendre des stratégies de dialogue dans leur système AMICA *(Pieraccini et al. 1997)*. Par la suite, *(Young 1999)*, a cherché à apprendre automatiquement des stratégies de dialogue avec des méthodes de programmation dynamique et des méthodes d'échantillonnage comme Monte-Carlo et les Différences Temporelles (TD) que l'on retrouve dans *(Sutton et Barto 1998)* où elles sont adaptées à l'apprentissage par renforcement.

2.7. Les champs d'application

Les systèmes de dialogue portent sur divers domaines concernés typiquement par les applications Dialoca, comme la recherche d'informations, le commerce électronique, les systèmes d'aide et l'assistance personnelle. Parmi les nombreuses applications et projets déjà réalisés, nous pouvons citer :

- le projet "How may I help you" de AT&T Research Labs, qui permettait d'aiguiller les appels téléphoniques *(Gorin et al. 1997)*, et
- le système d'information pour le trafic aérien "Air Travel Information Service" *(Levin et al. 2000)*, également chez AT&T.

Dans les projets plus récents, nous pouvons citer :

- les travaux du Robotics Institute de Canergie Mellon sur Florence Nightingale *(Roy et al. 2000)* qui est un robot mobile d'assistance aux personnes âgées, et
- le système NJFun de *(Singh et al. 2002)* qui est un service téléphonique d'informations sur le New Jersey.

2.8. Inspiration des systèmes de dialogue

Tous ces travaux sur les systèmes de dialogue, et en particulier l'approche MDP couplée à l'apprentissage par renforcement, nous ont fourni une source d'inspiration pour faire coopérer des agents hétérogènes. Cela nous a amené à baser le médiateur autour d'un gestionnaire de médiation avec, non seulement l'idée que le médiateur pouvait apprendre à interagir avec les agents humains, mais aussi qu'il pouvait, en partie, apprendre à remplir son rôle dans un hSMA.

Nous allons maintenant présenter une des applications sur laquelle nous avons travaillé, puis nous détaillerons comment nous avons conçu un agent médiateur à base de MDP et de Q-Learning.

3. Une application typique de réservation de vol

Notre approche est illustrée sur un problème classique de réservation de vols aériens. Dans cette application, une agence de voyage propose à ses clients un ensemble de vols entre différents aéroports internationaux. Les clients interagissent avec le système à l'aide de bornes multimédia avec une interface vocale. Le système d'information, appelé agent de service, accède à une base de données où les vols sont décrits par leurs aéroports de départ et d'arrivée, la classe de voyage, la compagnie aérienne et la tranche horaire.

L'objectif du médiateur est d'aider les clients à formuler des requêtes précises pour trouver le vol qui correspond à leurs besoins et de le réserver. Le médiateur interagit avec les clients et l'agent de service : il construit une requête en posant des questions aux clients sur leur vol et demande les résultats à l'agent de service. Quand des réponses sont disponibles, elles peuvent être transmises au client qui va sélectionner ou rejeter les résultats. Une suite typique d'interactions, où un usager (U) qui veut voyager de Paris à Moscou communique avec le médiateur (M), est donné Figure 52.

M :	*Quelle est votre ville de départ ?*
U :	*Paris*
M :	*Quelle est votre ville de destination?*
U :	*Moscou*
M :	*En quelle classe souhaitez-vous voyager ?*
U :	*Je ne sais pas*
M :	*Voulez-vous voyager en classe économique ?*
U :	*oui*
M :	*Il y a deux réponses à votre demande...*

Figure 52 - Un exemple de suite d'interactions

4. Spécification du médiateur

Dans cette partie, nous proposons une structure d'agent médiateur et la généralisation des stratégies de médiation inter-agent. La Figure 53 présente un diagramme décrivant les modules qui composent le médiateur avec leurs dépendances :

- Le ***gestionnaire d'interactions*** est le module central qui envoie et reçoit les messages vers et depuis les autres agents. Il supervise les cycles d'interactions et transmet les mises à jour nécessaires au gestionnaire de tâche.

- Le ***gestionnaire de tâche*** est responsable de la représentation de l'état de la médiation et de sa mise à jour. Il est paramétré selon l'application et modélise la tâche à accomplir.
- Le ***module de décision***, basé sur l'algorithme du Q-Learning *(Watkins 1989)*, gère la sélection de la prochaine action à accomplir selon l'état courant abstrait, transmis par le gestionnaire de tâche (voir § 4.2.2). Afin d'améliorer sa politique, et donc la stratégie de médiation, ce module reçoit également un signal de récompense du gestionnaire d'interactions.
- Le ***module de profil*** permet de gérer des connaissances probabilistes sur les différents agents avec lesquels le médiateur interagit.

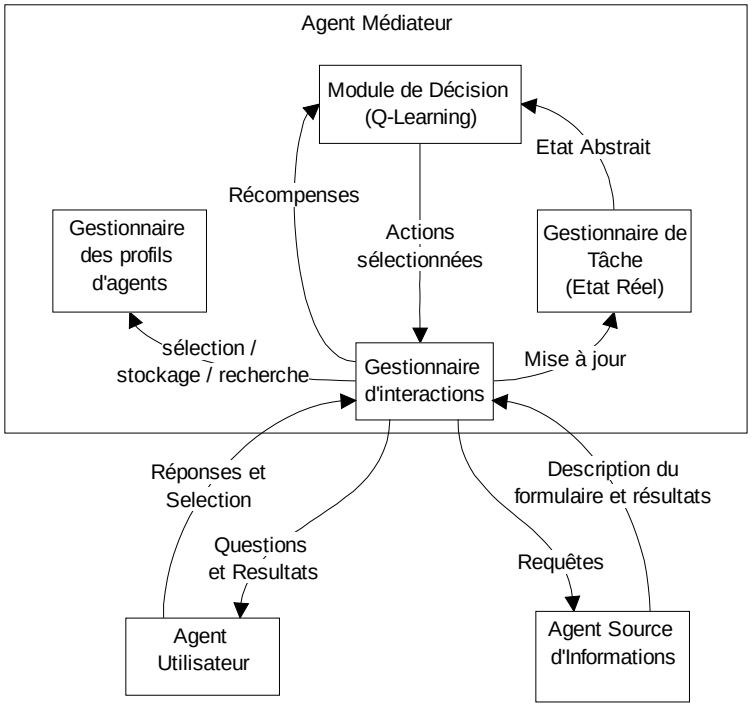


Figure 53 - Composition de l'agent médiateur

4.1. Modélisation de la tâche

Nous rappelons que le service se déroule en sessions au cours desquelles l'utilisateur contacte le médiateur et communique avec lui afin que celui-ci lui trouve une information spécifique, appelée un ***but*** Pour cela, le médiateur doit déterminer comment formuler sa requête auprès de la source d'informations pour trouver les données correspondant à ce but. Le médiateur construit un modèle de la tâche en utilisant une description de la source d'informations. Nous supposons que la spécification de l'application fournit au médiateur un formulaire de requête type pour la source d'informations ou que le médiateur est capable de l'obtenir en interrogeant cette même source. Ce formulaire est ensuite décrit comme un espace multidimensionnel, avec approche similaire aux E-Forms de *(Goddeau et al. 1996)* : les différents champs sont utilisés pour fournir un ***référentiel*** à cet espace de recherche sous la forme d'un ensemble d'***attributs***. Chaque attribut est associé à un domaine de valeurs possibles et à un ensemble de questions typiques. Ainsi, valuer les attributs revient à localiser un point dans l'espace généré par le référentiel, ce qui permet de cibler le but de l'utilisateur.

Les données de la source d'informations sont supposées être distribuées sur l'ensemble des combinaisons d'attributs possibles. Le médiateur va devoir interagir avec l'utilisateur en lui posant une suite de questions et chaque réponse va lui permettre de construire progressivement une requête en déterminant les valeurs des attributs. Comme nous le décrivons sur la Figure 54, tout se passe comme si le médiateur se déplaçait dans l'espace de recherche jusqu'à ce qu'il ait réussi à localiser le but. Comme à chaque position de l'espace de recherche correspond potentiellement un ensemble de

données, le médiateur peut sonder la source d'informations pour déterminer s'il y a ou non des données à cet endroit de l'espace de recherche. Cependant, il ne s'agit pas de maximiser la quantité des données retrouvée, mais de ne retourner à l'utilisateur qu'une quantité limitée de données pertinentes.

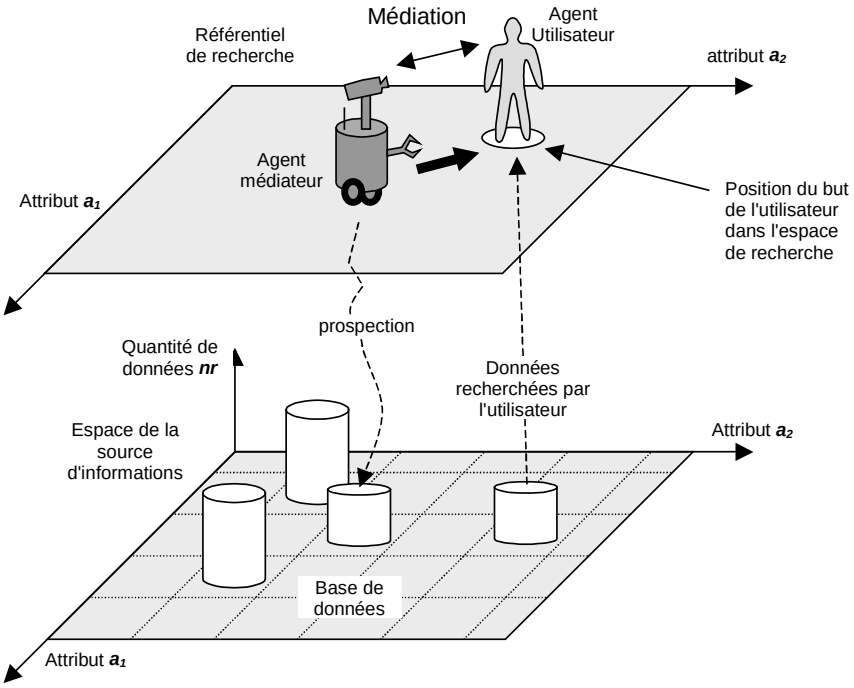


Figure 54 - Positionnement dans le référentiel d'attributs

4.1.1. Définition du Référentiel

Le référentiel permet de positionner dans l'espace multidimensionnel les différents objets manipulés, comme les requêtes du médiateur, les informations de la source ou encore le but de l'utilisateur. Il s'agit d'un système d'axes $Attrib = \{a_1, ..., a_m\}$ où chaque a_i est un attribut du problème. Dans l'exemple de réservation, un vol est caractérisé par une ville de départ, une ville d'arrivée, une classe de voyage, une compagnie et une plage horaire, ce qui donne le référentiel suivant :

Attrib = {Départ, Arrivée, Classe, Compagnie, Horaire}

4.1.2. Attributs

Un attribut défini par un tuple **<*N, D, Q*>,** où

- ***N*** est le nom de l'attribut,
- ***D*** est le domaine de l'attribut sous la forme d'un ensemble fini de valeurs possibles $\{v_1 ; \ldots ; v_k\}$ et
- ***Q*** contient un ensemble de questions pouvant être transmises à l'utilisateur.

Pour faciliter l'accès aux différents composants d'un attribut, on définit les fonctions ***nom***, ***domaine*** et ***requêtes*** qui retournent respectivement ***N***, ***D*** et ***Q***. L'ensemble des attributs correspondant à l'exemple est donné en Tableau 11. Afin de constituer l'ensemble des questions à poser, on pourra par exemple utiliser le label du champ correspondant et l'on distinguera plusieurs types de questions :

- des questions de valuation, qui demandent quelle est la valeur de l'attribut (sans suggérer de valeur),
- des questions de proposition, dans lesquelles une valeur est proposée pour l'attribut,
- des questions de confirmation, qui demandent la confirmation d'une valeur d'attribut.

Certaines questions peuvent être paramétrées dynamiquement par une valeur du domaine de l'attribut placé entre les crochets. On pourra, par exemple, utiliser la valeur la plus probable selon le modèle de l'utilisateur ou alors, la valeur qui a été affectée à cet attribut dans une session précédente.

Tableau 11 - Exemple d'ensemble d'attributs pour l'application de vol aérien

Nom d'attribut	Départ	
Domaine	{Paris, Bruxelles, Londres, Berlin, Madrid, Rome, Genève, Luxembourg}	
Questions	Valuation	Quelle est votre ville de départ ?
	Proposition	Souhaitez vous partir de [ville] ?
	Confirmation	Etes vous sûr de vouloir partir de [ville] ?

Nom d'attribut	Arrivée	
Domaine	{New York, Moscou, Tokyo, Sydney, Istanbul, Dakar}	
Questions	Valuation	Quelle est votre ville de destination ?
	Proposition	Souhaitez vous vous rendre à [ville] ?
	Confirmation	Etes vous sûr de vouloir vous rendre à [ville] ?

Nom d'attribut	Classe	
Domaine	{affaire, normale, économique}	
Questions	Valuation	En quelle classe souhaitez-vous voyager ?
	Proposition	Voulez-vous voyager en classe [classe] ?
	Confirmation	Etes vous sûr de vouloir voyager en classe [classe]?

Nom d'attribut	Compagnie	
Domaine	{American Airlines, Air France, British Airways, Delta Air Lines, Bankair, Luxair, Lufthansa, United Airlines}	
Questions	Valuation	Avec quelle compagnie souhaitez-vous voyager ?
	Proposition	Voulez-vous voyager avec [compagnie] ?
	Confirmation	Etes vous sur de vouloir voyager avec [compagnie] ?

Nom d'attribut	Horaire	
Domaine	{le matin, l'après-midi, le soir, de nuit}	
Questions	Valuation	Quelle période de la journée voulez vous partir?
	Proposition	"Voulez-vous partir[horaire] ?
	Confirmation	Etes vous sur de vouloir partir[horaire] ?

4.1.3. Représentation des buts

Soit $Goal = \{g_1, \ldots, g_n\}$, l'ensemble des buts possibles. Ces buts sont positionnés dans le référentiel à l'aide d'une fonction de position : chaque but $g_i \in Goal$, peut être décrit par un vecteur V de coordonnées $[v_{i,1}, \ldots, v_{i,m}]$ où $v_{i,j} \in Domaine(a_j)$, $\forall j \in [1;m]$ correspond à la valeur prise pour l'attribut a_i.

On a alors :

$$position(g_i) = [v_{i,1}, \ldots, v_{i,m}]$$

Nous utiliserons la notation suivante :

$$g_i = \{ nom(a_1) = v_{i,1} ; \ldots ; nom(a_m) = v_{i,m} \}$$

Ce qui donne, par exemple :

$$g_i = \{ \quad départ = "Paris"; arrivée = "Tokyo"; classe = "affaire"; \\ compagnie = "Air France"; horaire = "matin" \}$$

Les buts se retrouvent indexés par des points dans le référentiel construit sur l'ensemble des attributs. Un utilisateur peut avoir une idée approximative de la position du but qu'il cherche à atteindre. Aussi, la tâche du médiateur est d'interagir avec l'utilisateur pour déterminer la meilleure position ou le meilleur sous-espace possible contenant le point but. En cas d'échec, il doit indiquer que le but ne peut pas être identifié ou atteint.

4.1.4. Modélisation des états de la tâche

Comme indiqué dans *(Levin et al. 1998)*, l'état de la tâche doit capturer toute la connaissance que le médiateur peut avoir sur les entités avec lesquelles il interagit. Dans notre modèle, l'état de la tâche utilise une indexation partielle pour représenter la connaissance que le médiateur a du but utilisateur. Il reflète aussi la connaissance sur la capacité de réponse à court terme de la source pour cette requête. Pour chaque dimension ***j*** du référentiel $\{a_1; ...; a_m\}$, nous utilisons un couple (ea_j , val_j). La variable d'état ea_j prend ses valeurs dans l'ensemble ***Ea = {Ouvert, Affecté, Fermé}*** et la seconde variable $val_j \in Domaine\ (a_i)$ stocke la valeur correspondante, si elle existe (quand ea_j est affecté).

Dans le Tableau 12, on retrouve les différents cas pouvant être rencontrés.

Tableau 12 - Description des états possibles pour un attribut

Valeur	Etat de l'attribut
Ouvert	L'information sur cet attribut n'a pas encore été demandée à l'utilisateur et la dimension correspondante n'est pas contrainte. Cet état sera marqué par un point d'interrogation "?" pour signifier que la valeur val_i est encore indéterminée.
Affecté	Une question a déjà été posée à l'utilisateur qui a répondu par une valeur val_j. On marquera cet état par un "***A***" indiquant que l'attribut est instancié.
Fermé	Une question a déjà été posée, mais le médiateur n'a pas reçu de réponse utilisable. Par exemple, l'utilisateur a dit "*je ne sais pas*". Cet état sera noté par un "***F***" qui signifie que la valeur val_j restera inconnue, mais aussi qu'une question a déjà été posée.

L'état complet de la tâche va mémoriser :

- pour chaque attribut, les couples (ea_i, val_j) contenant l'***état d'affectation***, ainsi que la valeur qui est affectée (ou nul s'il n'y en a pas).
- l'ensemble ***Rep*** des résultats de la dernière réponse de la source d'informations, s'il y en avait, afin de pouvoir les transmettre à l'utilisateur.

L'état ***s*** de la tâche prendra la forme suivante :

$$s = \{ (ea_1 , val_1) ; \ldots ; (ea_m , val_m) ; Rep\}$$

On constate que la taille de l'espace d'état ***S*** de la tâche est très grande, car il faut non seulement considérer toutes les combinaisons de valeurs d'attributs possibles, mais aussi les résultats qui dépendent de la source d'informations. L'ensemble ***Rep*** sera limité aux nr_{max} premières réponses, où nr_{max} est le nombre maximum de réponses permises. Comme nous l'expliquons plus loin, seule une abstraction de l'état de la tâche sera utilisée par le module de décision.

4.2. Module de décision à base de MDP

Formuler le comportement de l'agent médiateur en terme de Processus de Décision Markovien requiert de définir un espace d'état ainsi que les actions possibles. Comme elles dépendent de la réponse de l'utilisateur, les transitions du modèle ne peuvent pas être données, mais sont apprises indirectement par renforcement. La solution est obtenue sous la forme d'une politique qui indique pour chaque état, quelle action sélectionner. Dans les systèmes de dialogue, les politiques MDP peuvent être interprétées comme des ***stratégies de dialogue*** *(Levin et al. 1998)* et elles dictent quelles sont les meilleures questions à poser à l'utilisateur avec un coût minimal. Dans notre approche, comme la stratégie sera utilisée pour définir le comportement de l'agent médiateur, nous parlerons plutôt de ***stratégie de médiation***. Résoudre le problème revient à atteindre un état où le but de l'utilisateur est suffisamment reconnu et à recevoir de l'utilisateur une sélection valide dans une liste de suggestions.

(Singh et al. 2002) indiquent que pour appliquer l'apprentissage par renforcement pour la conception de stratégies de dialogue, il est nécessaire de définir une représentation basée sur l'état de l'interaction. D'une façon générale, la représentation idéale d'un état serait une trace intégrale de tous les échanges qui se sont déroulés, mais ce n'est pas du tout réalisable. Afin de pouvoir respecter au maximum l'hypothèse de Markov, nous avons constitué l'espace d'état, de sorte à ce que toute l'information nécessaire pour décider quelle action le médiateur doit effectuer soit résumée dans les états.

Ainsi, l'état du processus de décision sera une abstraction de l'état de la tâche et les actions sont définies à partir des questions attachées à un attribut donné, sous la forme de requêtes transmises à l'utilisateur. Des actions, dirigées vers un autre agent du système sont également possibles, comme par exemple une requête transmise à la source d'informations. Afin de quantifier l'efficacité et la qualité du comportement de l'assistant et de comparer différentes politiques, il est également nécessaire de définir des récompenses données au médiateur. Les récompenses peuvent être des valeurs positives ou négatives selon la satisfaction finale de l'utilisateur, la longueur de la médiation, le coût d'utilisation des ressources, etc. Pour mesurer la satisfaction de l'utilisateur, divers indices peuvent être exploités. Le plus simple est d'utiliser une notation explicite à la fin de la médiation, mais cela peut être aussi le résultat d'un calcul sur l'état final (prenant par exemple en compte le nombre d'objets retournés par une requête). Nous avons choisi de donner une récompense positive quand l'utilisateur sélectionne une réponse et une récompense négative quand il ne le fait pas ou que la médiation est trop longue.

Le modèle que nous proposons est la combinaison d'une partie ***U***, liée à la construction de la requête avec l'utilisateur et d'une partie ***I***, rattachée aux informations retournées par la source :

- L'espace d'états ***S*** peut se décomposer en $\boldsymbol{S} = \boldsymbol{S}_U \times \boldsymbol{S}_I$ avec un sous-espace $\boldsymbol{S}_U$ où le médiateur positionne le but de l'utilisateur et un sous-espace $\boldsymbol{S}_I$, où le médiateur quantifie la capacité de réponse de la source d'informations. La Figure 55 montre comment cette décomposition intervient dans le schéma de la classe de service.

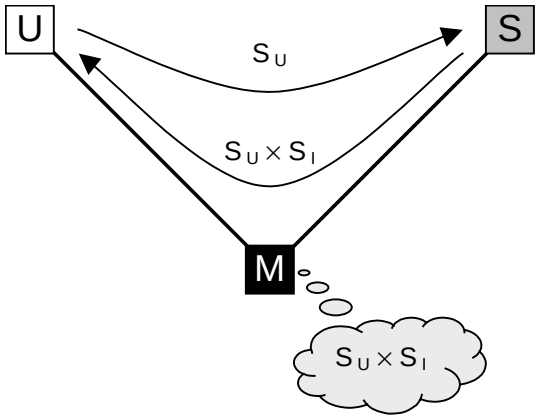


Figure 55 - Décomposition de l'espace d'états abstrait

- L'ensemble des actions possibles peut également se décomposer, comme le montre la Figure 56, en $A = A_U \cup A_I$, où A_U est l'ensemble des actions au travers desquelles le médiateur peut solliciter l'utilisateur et où A_I est l'ensemble des actions dont le médiateur dispose pour interroger la source d'informations.

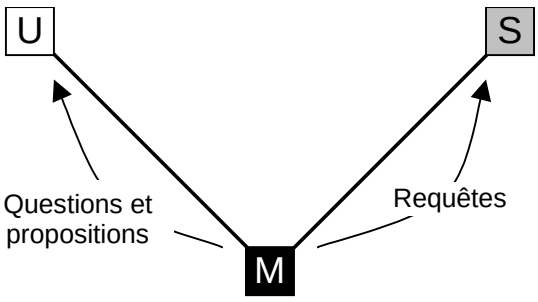


Figure 56 - Décomposition de l'ensemble des actions du médiateur

- Il en est de même pour la fonction de récompense $r = f(r_U, r_I)$. Comme le montre la Figure 57, elle est composée d'une partie mesurant la satisfaction de l'utilisateur sur le service rendu par le médiateur et d'une autre partie reflétant l'utilité de l'interaction avec la source d'informations en termes de quantité de données retrouvées.

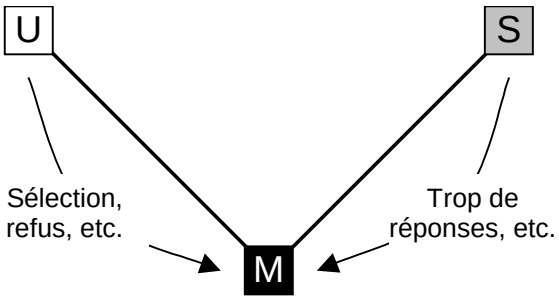


Figure 57 - Décomposition de la fonction de récompense

4.2.1. Etude de la taille du modèle

L'espace S_U correspond à l'ensemble des requêtes partielles possibles. Chaque attribut de la requête peut prendre son état dans l'ensemble ***{'?' ; 'F' ; 'A'}***. Quand un attribut est affecté (***A***), on lui associe une valeur et considérant que l'on a en moyenne ***i*** valeurs possibles pour un attribut, cela donne ***2+i*** états possibles.

Soit ***m*** le nombre d'attributs, la taille de l'espace S_U est donc $Card(S_U) = (2+i)^m$.

L'espace des résultats S_I représente l'ensemble des réponses possibles de la source d'informations à une requête. Il s'agit donc de l'ensemble des parties de l'ensemble de toutes les entrées de la source d'informations, soit $\sum_{i=0}^{n} C_n^i = 2^n$. On soit ajouter à S_I le cas pour lequel la réponse est indéterminée

La taille de l'espace d'état est donc :

$$Card(S) = Card(S_U) \times Card(S_I) = (2+i)^m (2^n+1)$$

où ***i*** est le nombre moyen de valeurs par attribut,

m est le nombre d'attributs et

n est le nombre d'éléments de la base.

Sur l'application de voyage aérien, avec ***8*** villes de départ, ***6*** villes d'arrivée, ***3*** classes de voyage, ***8*** compagnies aériennes et ***4*** créneaux horaires, il y a potentiellement $8 \times 6 \times 3 \times 8 \times 4 = 4608$ vols possibles. Même si la base ne contient que ***10%*** de ces vols, soit ***461***, l'ensemble des réponses possibles à une requête atteint tout de même 2^{461}. Par ailleurs, en considérant en moyenne ***6*** valeurs par attribut, le nombre de requêtes partielles possible est $(2+6)^5 = 8^5 = 32768$.

La taille de l'espace d'état complet est donc de $8^5 \times (2^{461}+1)$, soit $1{,}9.10^{143}$ états.

4.2.2. Abstraction des états

Le processus de décision doit travailler sur une représentation réduite de l'espace d'état. En effet, les états de la tâche ne peuvent pas être donnés tels quels au processus de décision, à cause de leur surnombre. La solution la plus simple pour réduire la complexité est de faire abstraction de la valeur de l'attribut. Ainsi, l'état ***(ea, val)*** de chaque attribut sera simplifié en ne gardant que l'état d'affectation ***ea***, ce qui donne, par exemple, ***A*** au lieu de ***(A ; valeur)***.

Les résultats de la requête passée sont également abstraits en appliquant deux seuils au nombre ***nr*** de réponses : un seuil minimal fixé à une réponse et un seuil maximal nr_{max}. Ainsi, seule un indicateur de quantité de résultats ***qr*** sera utilisé, comme l'indique le Tableau 13.

Tableau 13 - Qualité de la réponse

Nombre de réponses	Quantité de réponses	Description du cas
inconnu	**qr = '?'**	La requête n'a pas encore été effectuée
$nr = 0$	**qr = ' 0'**	Aucune réponse n'a été retournée par la requête
$nr \in [1 ; nr_{max}]$	**qr = ' +'**	Il y a eu un nombre suffisant de réponses à la requête
$nr \in] nr_{max} ; + \infty [$	**qr = ' *'**	Il y a eu trop de réponses à la requête

Les états abstraits transmis au module de décision seront notés avec la forme suivante :

$$< ea_1 \ ; \ldots ; ea_m \mid qr >$$

Comme chaque état d'attribut ea_i prend une valeur dans ***('?', 'A' , 'F')*** et que la quantité de réponses ***qr*** prend une valeur dans ***('?', '0', '+', '*')***, on obtient une taille de 4×3^m, soit ***108*** états pour ***3*** attributs et ***972*** pour ***5*** attributs.

4.2.3. Actions

La liste des actions possibles du médiateur est donnée Tableau 14. Il y a trois actions par attribut (valuer, proposer et confirmer), mais aussi deux autres actions (une pour la requête vers la source d'informations et une autre pour demander à l'utilisateur d'effectuer une sélection). Le nombre d'actions est donc $3 \times m+2$. Notons que dans le cadre des MDP, il est tout à fait possible de modifier l'ensemble des actions réalisables selon l'état, c'est pourquoi, la colonne "conditions" indique à quel moment une action peut être choisie.

Tableau 14 - Actions possibles pour le médiateur

Code	***Conditions***	***Description de l'action***
value	ea_s = **'?'**	Demander à l'utilisateur la valeur de l'attribut a_s.
propose	$ea_s \neq$ **'A'**	Proposer à l'utilisateur une valeur pour l'attribut a_s.
confirm	ea_s = **'A'**	Demander à l'utilisateur de confirmer la valeur de l'attribut a_s.
rqte	***qr*** = **'?'**	Envoyer une requête vers la source d'informations.
select	***qr*** = **'+'** ou **'*'** si la tâche est pleinement contrainte	Demander à l'utilisateur la sélection d'une entrée dans la liste des résultats.

4.2.4. Récompenses

Le gestionnaire d'interactions transmet les récompenses au module de décision, donc au MDP. Ces récompenses proviennent de chacune des interactions. Elles peuvent venir de l'utilisateur, quand il sélectionne ou qu'il refuse un résultat, s'il abandonne la session en cours ou quand la médiation est trop longue (t_{max} interactions). Une récompense peut aussi être obtenue lorsque le médiateur interagit avec la source de données et qu'il y a trop de réponses.

Soit $\tilde{g} \in Goal$, le but estimé de l'utilisateur. On dira qu'un objet ***o*** "satisfait" le but estimé de l'utilisateur s'il appartient au sous-espace engendré par ce but, c'est-à-dire que l'objet vérifie toutes les contraintes imposées sur le but. Pour toute dimension ***j*** du référentiel d'attribut $\{ a_1 , ... , a_j \}$, toute contrainte d'affectation d'état d'attribut du but $ea_j(\tilde{g})$ à une valeur donnée est vérifiée sur l'objet par une affectation $ea_j(o)$ à la même valeur.

$$satisfait(o, \tilde{g}) \Leftrightarrow (\forall j \in [1 ; m], (ea_j(\tilde{g}) = A) \Rightarrow ((ea_j(o) = A) \wedge (val_j(o) = val_j(\tilde{g}))))$$

Le problème est que seul l'utilisateur connaît le véritable but et c'est donc à lui que revient la vérification de la satisfaction globale de la tâche.

La fonction de récompense r_t obtenue au temps ***t*** dans l'état s_t est initialisée à zéro, puis elle est modifiée au fur et à mesure des entrées perçues. Les valeurs données Tableau 15 peuvent venir s'accumuler quand arrive une réponse utilisateur ou une réponse de la source d'informations. Nous avons ajouté d'autres récompenses, mais elles ne sont pas obligatoires. En effet, une requête qui échoue, à cause d'un surnombre de réponses ou d'une absence de réponse voire d'un refus de sélection pénalise la séquence d'actions en terme de "temps" dépensé alors qu'il aurait été préférable de sélectionner une autre action.

Tableau 15 - Récompenses obtenues par le médiateur

Récompense	*Valeur*	*Condition d'attribution*
$R_{selection}$	+5	Le module de décision a choisi une action ***select*** et l'utilisateur a sélectionné dans l'ensemble des réponses $Rep(s_t)$ un élément ***r*** qui satisfait son but.
R_{absrep}	1	Il n'y a pas de réponses alors que la tâche est totalement contrainte. La médiation se termine car le système a rempli son rôle, mais c'est la source qui n'a pas donné de résultat.
$R_{surnombre}$	-1	La source d'informations a retourné trop de réponses (***qr = '*'***).
$R_{troplong}$	-4	Le nombre d'interactions a dépassé la limite fixée.

4.2.5. Apprentissage par renforcement

Le modèle auquel est confronté le médiateur est composé de la fonction de transition ***T*** et de la fonction de récompense ***R***. Le problème est que chacune d'elles dépend de l'interaction avec l'utilisateur et de l'interaction avec la source d'informations :

- $T = f(T_U, T_I)$
- $R = f(R_U, R_I)$

C'est principalement ce qui motive le choix d'un apprentissage par renforcement et, en particulier, celui de l'algorithme du Q-Learning. Comme nous l'avons indiqué dans le chapitre 4, l'avantage de cette approche est de ne pas nécessiter une connaissance complète du modèle sous-jacent. De plus, ce choix est motivé par la possibilité d'utiliser l'algorithme du Q-Learning en ligne, ce qui permet au gestionnaire d'interactions de continuer son apprentissage durant les différentes sessions.

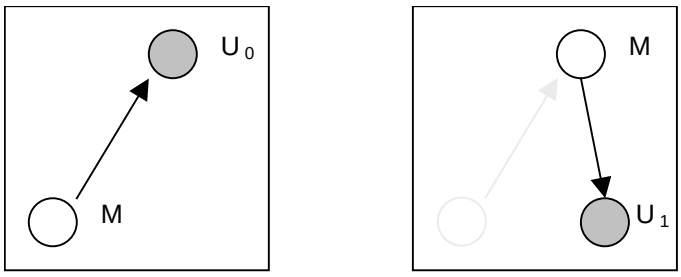
4.3. Gestion des interactions

Le rôle du gestionnaire d'interactions est de gérer les cycles d'interactions et d'envoyer les mises à jour au gestionnaire de tâche en fonction des entrées. Nous allons examiner l'état de départ, puis nous considérerons comment l'état évolue au cours des interactions de médiation.

4.3.1. Etat de départ

L'état de départ va dépendre de la connaissance que le médiateur a de l'utilisateur. Si l'on ne connaît rien de lui a priori, il va partir d'une position où le but est ouvert. Il s'agira typiquement de l'état $<?, ..., ? | ?>$. Par contre, s'il y a déjà eu des sessions passées, il peut être intéressant de considérer les états finaux de celles-ci, afin de trouver un point plus astucieux à partir duquel il est possible de redémarrer directement.

Examinons de plus près ce problème de l'état de départ sur l'exemple de la Figure 58 : un nouvel utilisateur se présente au temps $t = 0$ et commence une première session. Le médiateur tente de se positionner en fonction des besoins de cet utilisateur dans l'espace de requête en $U_0 = < u_{0,0}, ..., u_{n,0} >$. Le système va poser une série de questions afin d'atteindre ce point. A la fin du service, l'utilisateur n'interagit plus avec le médiateur et celui-ci peut mémoriser ce dernier emplacement avant de passer à une autre session. Lorsque le même utilisateur revient plus tard dans une nouvelle session, ses besoins se sont déplacés au point $U_1 = < u_{0,1}, ..., u_{n,1} >$ et le médiateur doit alors se rendre maintenant à ce point U_1, et ainsi de suite...



Espace de recherche au temps t = 0

Espace de recherche au temps t = 1

Figure 58 - Suivi de l'évolution du but utilisateur

Sur l'exemple de voyage aérien, un usager peut demander de Paris à Tokyo en classe économique, puis revenir ensuite dans une autre session et partir de Tokyo vers Istanbul en classe économique. Avec un tel historique, on peut se dire que cet usager voyage souvent en classe économique et lui proposer directement cette classe lors d'une session ultérieure... on peut aussi se dire qu'il repart de là où il est arrivé la dernière fois et lui suggérer un vol au départ d'Istanbul.

Nous disposons donc là d'une piste pour modéliser la façon dont évoluent les préférences de l'utilisateur : une loi qui relie la requête U_t à la ou les précédentes $U_0, ..., U_{t-1}$. Dans l'absolu, connaître cette loi permettrait de s'adapter en se préparant aux requêtes futures et ainsi d'optimiser la recherche du but. La loi d'évolution des requêtes dépend de l'utilisateur, mais elle peut également dépendre de l'application. Une perspective intéressante de nos travaux serait de chercher à apprendre cette loi, en s'appuyant éventuellement sur des connaissances additionnelles sur le comportement de l'utilisateur.

4.3.2. Evolution de la tâche

Le gestionnaire d'interactions met à jour la tâche qui évolue en fonction des actions du médiateur et de leurs effets sur l'environnement. Deux types d'évolutions sont possibles : d'une part, le système peut s'adresser à l'utilisateur et recevoir la réponse (en posant une question sur une valeur d'attribut ou en fournissant une sélection de résultats) et d'autre part, il peut envoyer une requête vers la source

d'informations et obtenir les résultats. Par ailleurs, diverses sources d'incertitude peuvent apparaître à ce point. Une première cause, comme nous l'avons indiqué dans le chapitre 2, vient de la fiabilité des médias. Une autre source importante d'incertitude est l'utilisateur, car il peut ne pas comprendre une question et donner une réponse inadaptée.

Dans le Tableau 16, nous donnons un ensemble de perceptions possibles du médiateur en indiquant pour quelle action elles peuvent se produire. On laisse à l'utilisateur la possibilité d'indiquer qu'il ne connaît pas de réponse à la question posée (on notera cette perception par 'nsp').

Tableau 16 - Ensemble des observations possibles du médiateur

Code	***Action contexte***	***Description de la perception***
val	value	Valeur d'attribut valide donnée par l'utilisateur
oui	propose, confirm	L'utilisateur a accepté ou confirmé une valeur d'attribut
non	propose, confirm	L'utilisateur a refusé ou remis en cause la valeur de l'attribut
table	rqte	Résultats donnés par la source d'informations
numéro	select	L'utilisateur a sélectionné une proposition
refus	select	L'utilisateur a refusé une proposition
stop	value, propose, confirm, select	Arrêt par l'utilisateur
nsp	value, propose, confirm, select	L'utilisateur indique qu'il ne sait pas répondre
aberrant	value, propose, confirm, select	Perception aberrante ou inattendue

Les cas concrets dans lesquels les états des attributs évoluent sont détaillés dans le Tableau 17, où les entrées aberrantes sont ignorées.

Tableau 17 - Mise à jour de l'état réel des attributs

$(ea, val)_t$	Act_t	$Percept_t$	$(ea, val)_{t+1}$
?	propose [v]	nsp	F
?	propose [v]	non	?
?	propose [v]	oui	A; v
?	value	nsp	F
?	value	v	A; v
A, v	confirm	nsp	F
A, v	confirm	non	?
A, v	confirm	oui	A; v
F	propose [v]	non/nsp	F
F	propose [v]	oui	A; v
F	value	nsp	F
F	value	v	A; v

En ce qui concerne l'action de requête ***rqte***, elle peut être sélectionnée à partir de n'importe quel état et la table des résultats est mise à jour en fonction des réponses. Le résultat modifie l'état courant au niveau de la quantité de réponses. Ainsi, on passera par exemple d'un état **<*?, ?, ? | ?*>** à l'état **<*?, ?, ? | **>** si on fait une requête alors que l'on ne connaît rien et que la base de données renvoie logiquement une grande quantité de résultats. A l'inverse, une requête fortement contrainte effectuée depuis un état **<*A, A, A | ?*>** peut ne pas donner de résultats. Dans ce dernier cas l'état devient alors **<*A, A, A | 0*>**. Il faut noter par ailleurs que toute transition modifiant l'état des attributs fait repasser la quantité de résultats dans l'état inconnu (***?***).

A chaque fois que l'état comporte une quantité de réponses positive (+), il est possible de demander la sélection dans la liste à l'utilisateur grâce à l'action ***select***. Dans cette situation, deux cas de figure peuvent se présenter. Dans le premier cas, on reçoit un numéro, ce qui correspond à la sélection d'un résultat. On considère alors que la tâche est accomplie et la médiation s'arrête en stockant l'état courant pour l'utilisateur. Dans le second cas, on reçoit un refus de sélection. On vide alors la zone de réponses, ce qui a pour effet de faire repasser la quantité de résultats à ***0***.

Il existe un dernier type de transition : lorsque l'utilisateur a demandé l'arrêt du service. Si cela se produit, le service s'arrête après avoir mémorisé l'état courant pour cet utilisateur.

4.4. Gestion des profils utilisateurs

Dans le chapitre 5, nous avons abordé le problème de l'adaptation aux utilisateurs. De sorte à prendre en compte les préférences des utilisateurs, le médiateur contient également un module qui gère les profils utilisateurs. Chaque utilisateur est représenté avec un ensemble de données et un profil par défaut peut être utilisé pour les nouveaux utilisateurs.

Les profils utilisateurs ont été conçus pour stocker les préférences comme des distributions de probabilités $p(v \mid u, a)$ sur les valeurs v de chaque attribut a et chaque utilisateur u. La distribution de probabilité est initialisée avec une loi uniforme et mise à jour quand l'utilisateur sélectionne une valeur. Le module de préférence utilisateur permet de donner une valeur dans le cas où l'on formulerait une question de proposition. Par exemple, un utilisateur qui a voyagé deux fois en classe affaire, une fois en classe normale et sept fois en classe économique va plus probablement voyager en classe économique la prochaine fois. La sélection d'une question de proposition sur l'attribut de classe donnera donc : "*Voulez-vous voyager en classe économique* ?".

5. Limites de l'approche

Dans nos travaux, nous retrouvons des problèmes similaires à ceux qui ont été rencontrés dans la conception de systèmes de dialogue. Tout d'abord, l'apprentissage d'une politique requiert une large quantité d'expériences. Pour résoudre ce problème, *(Young 1999)* propose de construire un automate probabiliste à état fini qui reproduit le comportement d'utilisateurs réels observés pendant des expériences de type Magicien d'Oz. Une fois cet automate obtenu, il propose de le placer dans une boucle fermée avec le système de dialogue pour aider ce dernier à apprendre une politique. Un second problème est que nous sommes limités par le caractère partiellement observable de l'utilisateur. Ces limites de l'approche MDP à prendre en compte l'incertitude sur l'état du dialogue sont décrites dans *(Roy et al. 2000)*. En se recentrant sur l'utilisateur et en le considérant comme un processus markovien partiellement observable, ils ont défini des systèmes de dialogues en termes de POMDP où les "états utilisateurs" sont seulement perçus au travers d'observations incertaines. Comme la complexité des POMDP rend la résolution très difficile, ils ont proposé d'abstraire la distribution sur les états possibles, appelée "belief state", en un couple contenant d'une part l'état estimé le plus probable et d'autre part l'entropie de la distribution. Cette factorisation d'état leur a permis de formuler et de résoudre le problème en revenant dans une vision MDP.

6. Conclusion

Dans ce chapitre, nous avons proposé la spécification d'un agent logiciel dans le cadre particulier de la médiation entre un agent humain et un service de recherche d'informations. Nous avons également établi un parallèle entre la médiation dans les hSMA et les systèmes de dialogue. Ceci nous a permis de transposer les approches à base de MDP existantes et les solutions qui sont utilisées dans ce domaine, comme l'apprentissage par renforcement pour modéliser le médiateur. L'avantage de cette solution est d'être applicable, même si les autres agents ne sont pas contrôlables, puisque le médiateur apprend à interagir avec eux et non l'inverse. Dans le chapitre suivant, nous allons revenir sur les services Dialoca et proposer une architecture pour la fourniture de services adaptatifs à base de médiateurs.

Partie III. La solution applicative

Chapitre 7 - Validation de l'approche au travers des services Unimédia

Dans le chapitre 2, nous avons présenté en détail la plate-forme Unimédia de la société Dialoca, ainsi que les services de communication qu'elle propose. Au cours de nos travaux, nous avons réalisé une étude de chacun de ces services et nous allons présenter les grandes lignes de cette analyse dans la première partie de ce chapitre. Nous proposons ensuite une vision abstraite au travers de squelettes de services dans lesquels l'impact de l'utilisation des médias est gommé pour ne garder que ce qui concerne la gestion de service et le traitement de l'information. Au chapitre 3, nous avons présenté notre solution théorique de collaboration dans un hSMA dans laquelle nous avons défini un ensemble de classes de service. Dans cette présentation, nous avons cherché à associer à chaque classe les services Dialoca qu'elle permettait de modéliser. La suite du chapitre détaille, en illustrant sur le service E-Nots de notification de messages électroniques *(Maillet 2000)*, comment ces classes de services peuvent être intégrées dans une architecture conceptuelle adaptée aux services de communication multimédia. Nous décrivons cette architecture destinée à faire communiquer des agents hétérogènes, en détaillant chacun des niveaux d'abstraction qui la constituent. A la fin du chapitre, nous présentons trois principales façons de concevoir des services en utilisant les approches que nous avons présentées jusqu'alors.

1. Analyse des services Unimédia

Lorsque nous avons décrit les services Unimédia au chapitre 2, nous les avons présentés comme l'interprétation, par l'Application Générique Unimédia, de scripts qui décrivent exhaustivement leur déroulement. Nous proposons maintenant d'analyser la façon dont ils sont structurés afin d'en obtenir une représentation abstraite.

1.1. Les scripts de service et leur interprétation

Les ***scripts*** sont écrits par les concepteurs de service dans le langage spécifique, décrit par *(Assadourian et Maillet 2000)*, puis stockés sous forme de fichiers texte. Ils sont formés d'un ensemble d'étapes qui sont exécutées séquentiellement. Chaque étape porte un nom et se compose de plusieurs champs, dont certains sont optionnels :

- Le champ ***action*** d'une étape indique à l'interpréteur, quelle action doit être effectuée dans cette étape. Si ce champ est *NULL*, alors l'étape permet d'effectuer des traitements internes à l'interpréteur, tels que des calculs sur ses variables. Pour chaque étape, le script donne précisément quelle action effectuer, mais comme le résultat de l'action n'est pas garanti, le passage à l'étape suivante peut être conditionné selon les valeurs de retours.

- Le champ ***destination*** permet d'identifier la ressource Unimédia qui va être concernée par l'action. Si ce champ vaut *NULL*, et que le service a été déclenché par l'arrivée d'un événement depuis une ressource, alors l'action concerne cette ressource.
- Le champ ***modèle*** permet d'indiquer à la ressource concernée, quel est le fichier de modèle qu'elle doit utiliser pour savoir précisément comment dérouler l'action.
- Le champ **params** décrit, d'une part, les traitements internes sur les variables de l'interpréteur à l'aide de macros et, d'autre part, les paramètres d'action qu'il faut transmettre à la ressource.
- Le champ **wait** indique si l'interpréteur doit attendre ou non le résultat de l'exécution de l'action ou l'arrivée d'un événement Unimédia.
- Le champ ***branch*** décrit une arborescence de branchements conditionnels vers d'autres étapes
- Le champ ***for*** permet d'effectuer des boucles de traitement et il faut alors placer un branchement ***next*** correspondant.

A ce niveau, il est important de préciser que l'exécution d'un script peut gérer simultanément

- différents usagers sur un même média,
- le même usager sur plusieurs médias, ou encore
- plusieurs usagers sur plusieurs médias.

C'est pourquoi l'interpréteur gère ses variables par contextes. Pour chaque application, on retrouve :

- Un contexte d'application contenant les variables globales,
- Des contextes utilisateur contenant les variables spécifiques à la connexion d'un usager
- Des contextes de ressources contenant les variables pour la gestion d'un usager sur un média.

Dans Unimédia, l'état courant d'un service est donc déterminé par l'étape courante pour chaque connexion et par le contenu de chacun des contextes (application, utilisateur et ressource). L'évolution de l'état, quant à elle, est soumise à la réception des évènements, à l'observation du succès ou de l'échec des actions entreprises et aux entrées d'informations associées.

Le schéma Entité-Association de la Figure 59, que nous avons mis au point pour la réalisation de notre logiciel de chargement des scripts en base de données, révèle la complexité intrinsèque des scripts de service.

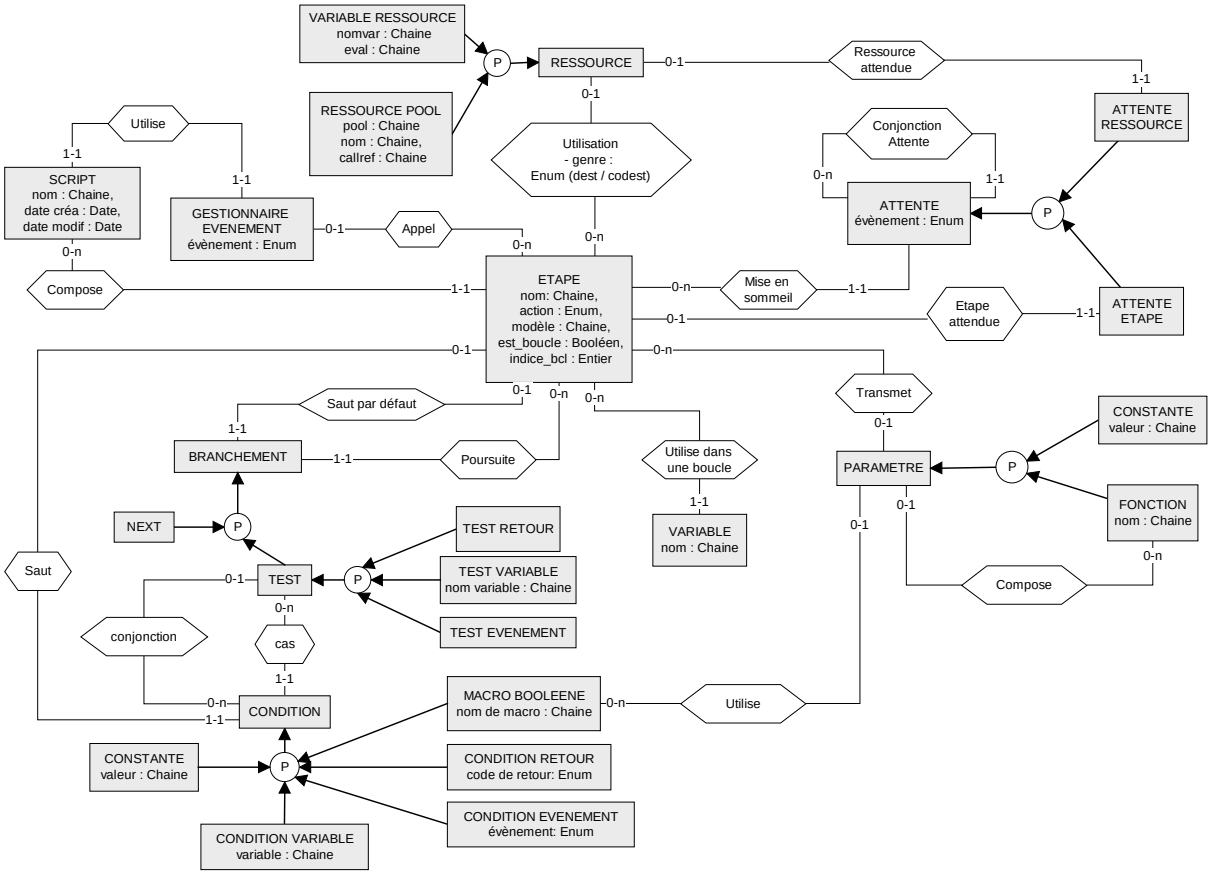


Figure 59 - Schéma Entité-Association représentant les scripts Unimédia

1.2. Graphes Etapes/Branchement

La description détaillée du script de notification de messages électroniques qui a servi de base pour le service E-nots actuel est donnée en annexe III. Nous nous sommes aperçus que les services pouvaient être représentés par des automates à états avec mémoire, dans lesquels un état correspond à une étape et une transition correspond à un branchement ou à l'arrivée d'un évènement. Nous avons implanté un analyseur de scripts Unimédia qui permet de représenter cet automate sous la forme d'un ***graphe étape/branchement*** orienté et nous allons maintenant examiner les différents graphes que nous avons obtenus.

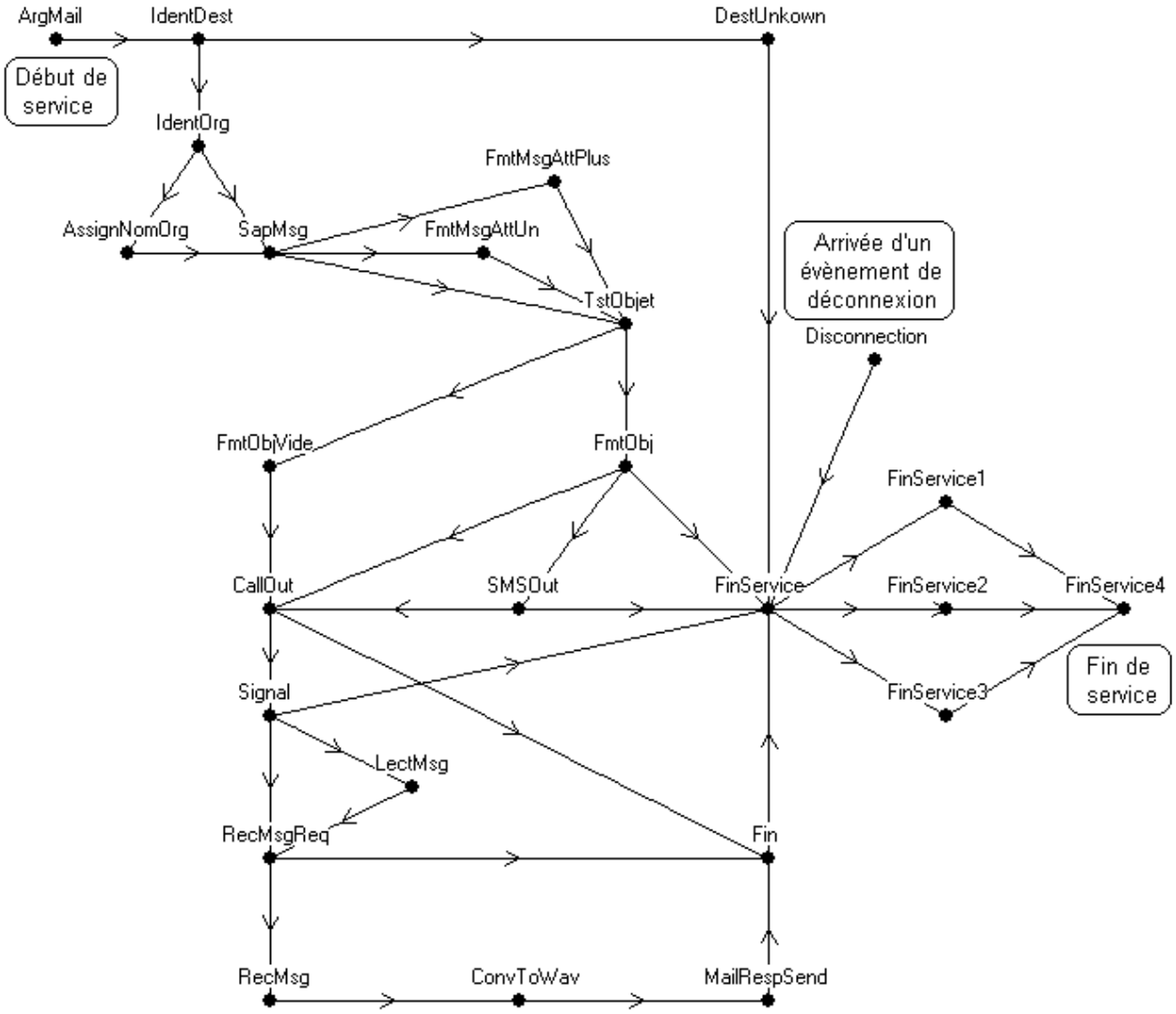
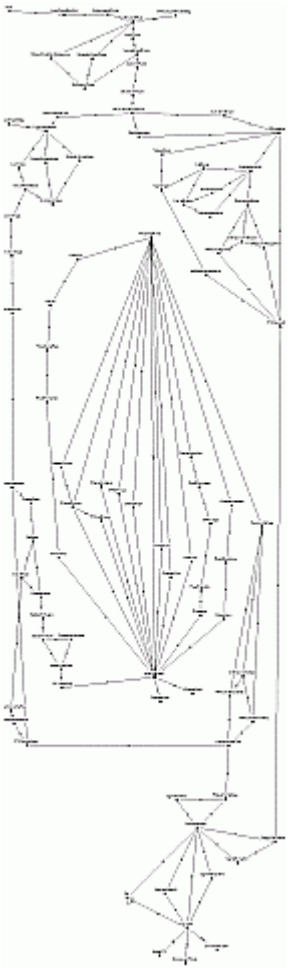


Figure 60 - Graphe pour le service E-Nots simplifié

La Figure 60 montre le graphe correspondant au script du service E-Nots. Dans ce graphe, nous retrouvons 26 étapes (nœuds) et les 41 branchements entre les étapes (arcs). Pour plus de lisibilité, nous avons mis en évidence les nœuds de départ et de fin de service et omis les conditions des branchements. Nous avons également indiqué la présence de l'étape spécifique qui gère la déconnexion de l'utilisateur. Cette étape est, en quelque sorte, un gestionnaire d'événements vers lequel l'application peut être déroutée, à partir du moment où la communication téléphonique est établie. A nouveau, afin d'obtenir un graphe plus compréhensible, nous n'avons pas représenté tous les arcs qui y conduisent.



a. Graphe d'une instance de service MAS

96 nœuds et 140 arcs

b. Graphe d'une instance de service VOS

55 nœuds et 100 arcs

c. Graphe d'une instance de service RMS

78 nœuds et 170 arcs

Figure 61 - Exemples de graphes pour des services Unimédia réels

Les graphes présentés Figure 61 ont été générés à partir de scripts écrits pour des clients de la société Dialoca. Ils illustrent le fait qu'une grande complexité apparaît lorsque l'on passe à des services réels. Les services les plus simples sont séquentiels et leurs graphes prennent la forme de chaînes. La prise en compte de la diversité, tant au niveau des choix entre différentes alternatives, qu'au niveau des cas possibles à prendre en compte, conduit à des graphes partant sous forme arborescente. Ces différentes branches alternatives se referment ensuite, bien souvent, à des points de jonction, ce qui donne des aspects de sous-graphes en faisceaux. Le service MAS en est d'ailleurs un bon exemple, au niveau de la partie centrale du graphe [Figure 61 - a] qui gère, de façon cyclique, toutes les actions possibles, à partir de l'interface du poste du télé-conseiller. Il faut, par ailleurs, noter que ce script MAS regroupe à la fois à la gestion web+téléphone du poste de télé-conseiller (à gauche et au centre) et celle de la borne (à droite). L'aiguillage vers les sous-graphes s'effectue selon le test du type de client. Cependant, à l'exception des cas pour lesquels il est possible d'isoler quelques motifs, les graphes deviennent très vite enchevêtrés comme on le voit sur la [Figure 61 - b et c]. En examinant ces graphes, il est donc possible de mesurer à quel point la conception d'un service relève d'une expertise très méthodique.

1.3. Echanges entre l'application et les ressources

Les graphes étapes/branchement que nous venons de présenter sont utiles, car ils permettent de suivre le déroulement de l'application. Il faut également noter qu'à chaque étape, des actions sont envoyées aux différentes ressources qui produisent des résultats et des évènements. Nous avons indiqué dans le chapitre 2 que le service E-Nots, par exemple, devait non seulement gérer des variables de l'interpréteur, mais pas moins de six ressources : le pilote d'accès base de données (BDD), le pilote de messages électroniques (E-Mail), le pilote de téléphonie (Vocal), le pilote de mini-messages (SMS), le moteur de synthèse de parole (SAP) et le moteur de reconnaissance de la parole (RAP). Le schéma Figure 62 montre comment l'application E-Nots travaille de concert avec ces différentes ressources au cours du déroulement du service. La partie gauche correspond à un graphe étape/branchement simplifié.

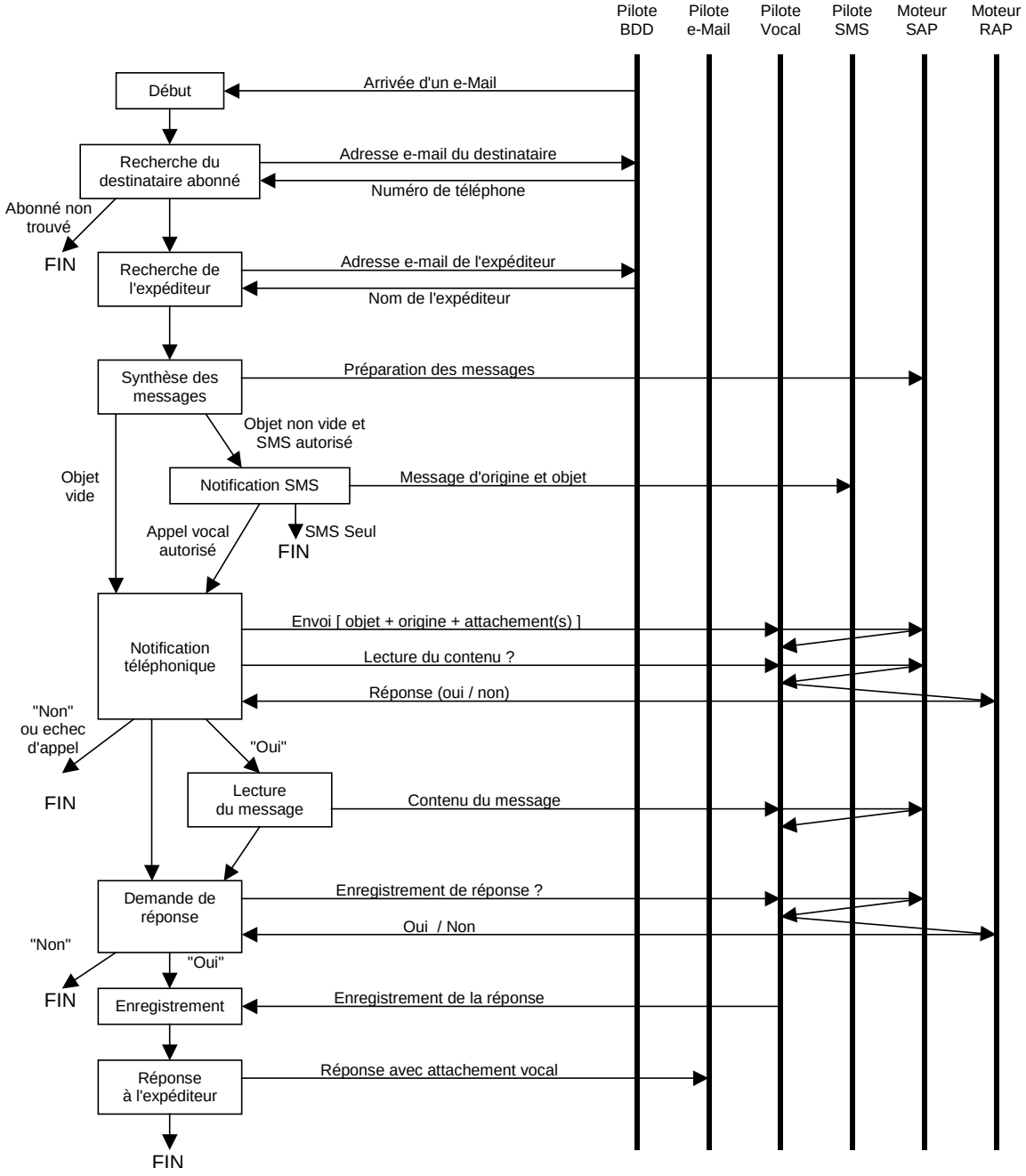


Figure 62 - Echanges entre l'application et les ressources pour le service E-Nots

1.4. Observations des ensembles d'étapes

Lors de notre analyse des scripts Unimédia et de l'Application Générique nous avons constaté qu'il est possible d'identifier, dans un script, des ensembles d'étapes qui concourent à des activités particulières du service. Ainsi, un groupe d'étapes peut, par exemple, servir à la formulation d'une requête et une autre à son exploitation. Dans le service E-Nots, on retrouve les activités suivantes :

- La prise en compte de l'arrivée d'un message électronique avec la récupération des informations sur l'émetteur et le destinataire, qui sont nécessaires pour savoir quel traitement effectuer,
- La préparation et l'envoi d'une notification par SMS,
- La préparation et la notification par téléphone avec deux sous-activités qui sont d'une part la diffusion du message et d'autre part l'enregistrement et l'envoi de la réponse à l'expéditeur.

De la même façon, il est bien souvent possible de retrouver des fonctionnalités similaires entre les différents services :

- La gestion des sessions au niveau des services déclenchés sur les appels entrants d'usagers. Ceci implique l'identification de l'utilisateur à partir d'une base de données et l'obtention de diverses informations utiles comme les droits et les options applicables.
- La recherche d'informations dans une source de données en utilisant un média particulier.
- Les conversions entre différentes représentations à l'aide d'un moteur pour le passage de l'information d'un média à un autre.
- La génération et l'envoi de messages pré-formatés qui sont complétés dynamiquement.
- Le contact d'un autre usager et la mise en correspondance, principalement en ce qui concerne les appels téléphoniques. Ceci sous-entend la gestion de la connexion, du synchronisme et des contraintes temps réel.
- La gestion d'interfaces de navigation dans l'espace d'information pour des médias particuliers, telle que la présentation sur une borne hypertexte ou le dialogue vocal au téléphone.

Ceci nous a conduit dans un premier temps, comme nous montrons plus loin, à nous intéresser à des groupes d'étapes plutôt que les étapes elles-mêmes.

1.5. Expertise des spécialistes Dialoca pour la conception de services

La conception d'un nouveau service Unimédia nécessite tout d'abord une première phase, dans laquelle la spécification de scénario est réalisée conjointement avec le client. Le concepteur doit ensuite déterminer quelles sont les ressources Unimédia (moteurs et pilotes) qui entrent en jeu et prévoir toutes les données que l'application doit manipuler. Il faut notamment prévoir l'accès aux bases de clients et de produits, aux échantillons audio statiques à diffuser, aux modèles des pages web, etc. Il faut ensuite écrire le script de l'application, les modèles indiquant comment exécuter chaque action au niveau des pilotes médias ainsi que les configurations qui seront utilisées par les moteurs. En effet, utiliser un moteur comme celui de la reconnaissance automatique de la parole nécessite la fourniture de modèles de langage et de modèles phonétiques. Afin d'utiliser le service, il faut ensuite lui créer une configuration pour l'administration Unimédia. Il est alors nécessaire de vérifier son bon fonctionnement en effectuant de nombreux tests et de le faire valider par le client.

Comme nous l'avons précisé plus haut, notre étude des services nous a amené à voir les services Unimédia comme des automates qui spécifient entièrement les actions effectuées sur les médias. Leur déroulement est préétabli de façon statique et exhaustive, leur conception devient vite très complexe et requiert l'intervention d'un expert. Par ailleurs, cela conduit à une rigidité pendant l'exécution et donc à des manques de souplesse et d'adaptation.

2. Abstraction des services Unimédia

Nous avons défini un formalisme intermédiaire, appelé ***squelette de service***, qui permet de représenter les comportements des applications en se détachant des médias et d'obtenir, en ce sens, des modèles plus généralisés.

L'exemple de squelette de service présenté dans la Figure 63 propose une généralisation du service E-Nots. L'application est tout d'abord en attente d'un événement ***E*** entrant (1) . Lorsqu'elle reçoit un évènement ***E*** de type message, elle en extrait le destinataire $Dest_E$, l'expéditeur Exp_E et le sujet Msg_E. L'application tente alors d'identifier (2) le service ***Srv*** à réaliser en fonction de l'événement. Si le service existe (3), alors elle va interroger (4) une base de profils des utilisateurs abonnés au service ***BD_Profil*** pour retrouver les préférences $Profil_D$ du destinataire $Dest_E$, sinon l'application termine (18). Si le profil existe (5), alors l'application va filtrer (6) le message Msg_E en fonction des règles décrites dans le profil du destinataire $Profil_D$. Selon l'action spécifiée (7), l'application pourra éventuellement se contenter de stocker le message (8) et se terminer (18). Si l'action à réaliser est une notification du destinataire, alors le service va transmettre (9) le message Msg_E sur le média spécifié dans le profil $Profil_D$, puis elle se mettra en attente (10) d'un événement correspondant à une réponse de cet utilisateur. Lorsqu'un tel événement E_2 est reçu et que le message associé Msg_{E2} est valide (11), alors il est utilisé (12) pour former un accusé de réception ***MsgRep***. L'accusé de réception peut également être spécifié à l'avance dans le profil de l'abonné. Dans ce cas, le service utilise le message prévu (13). Le service peut ensuite rechercher s'il trouve (14 & 15) le profil $Profil_E$ de l'expéditeur du message ***E*** pour renvoyer (16) l'accusé de réception sur média préféré. Si ce profil est introuvable, alors le service utilise le média de départ (qui a véhiculé ***E***) pour retourner l'accusé de réception (17).

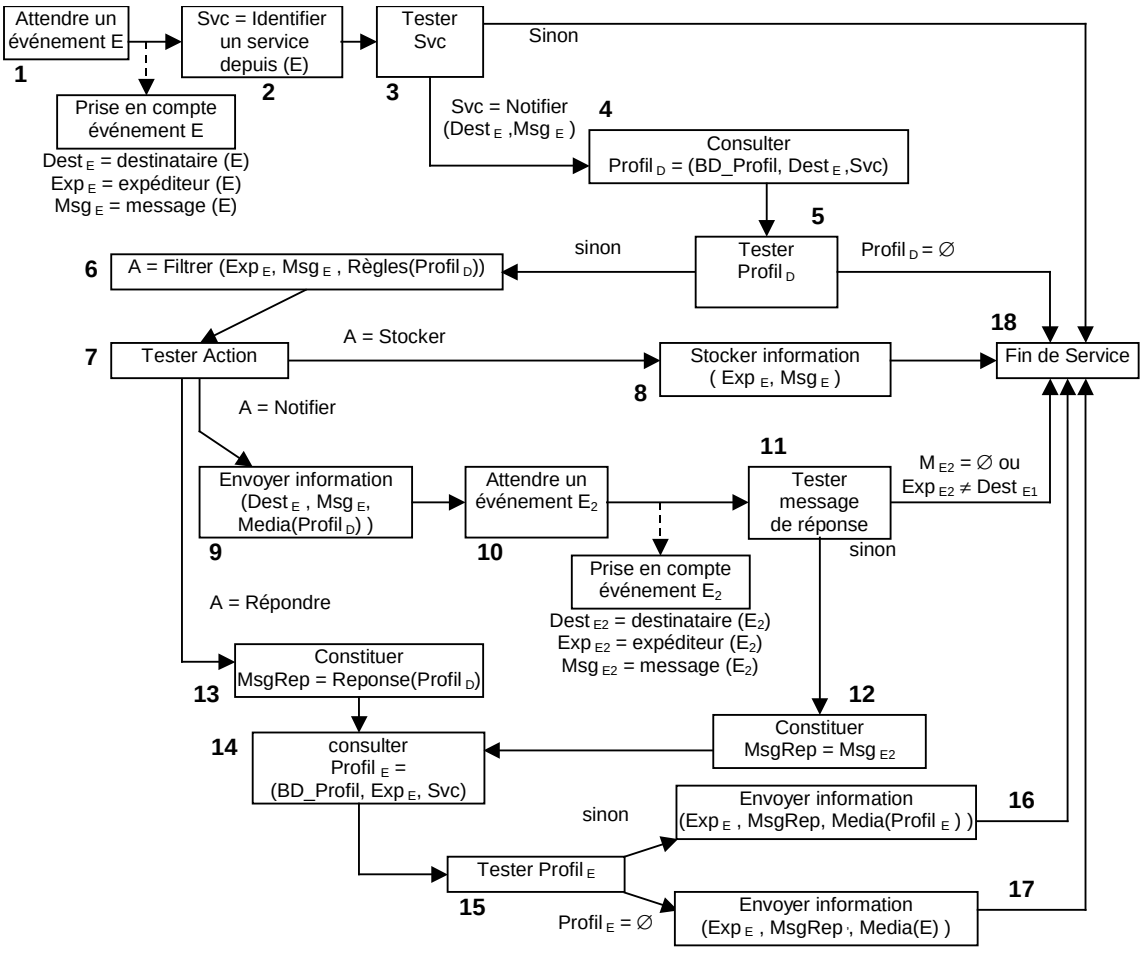


Figure 63 - Squelette de service pour le service E-Nots

Si nous regardons maintenant les actions utilisées, nous obtenons la liste du Tableau 18. Pour chaque action, nous avons donné une description accompagnée de la cible concernée par l'action et le domaine de traitement auquel elle se rattache (gestion de l'information, gestion du service).

Tableau 18 - Actions abstraites du squelette de service

Action	***Description***	***Cible***	***Domaine***
Constituer	Cette action permet de constituer de façon interne une requête/ou un message à partir d'un ensemble d'informations	Interne	information
Consulter	Cette action envoie une requête à un système d'information ou à un gestionnaire de profils pour retrouver des données	Système d'informations	information
Envoyer	Cette action envoie une information à un destinataire sur un média donné	média	information
Stocker	Cette action permet de mémoriser une information de façon interne	Variables Contextes	information
Attendre un évènement	Cette action permet l'attente d'un événement entrant en provenance d'un pilote média	Pilote Média	service
Filtrer	Cette action permet de décider quel traitement appliquer à une information selon un profil utilisateur	Interne	service
Fin de service	Cette action permet de terminer le déroulement du service	Interne	service
Identifier un service	Cette action permet de retrouver le service associé à la gestion d'un type d'événement	Interne	service
Tester	Cette action permet d'effectuer un test conditionnel et de choisir la suite du séquencement selon le résultat	Interne	service

Bien entendu, cette liste peut être complétée au vu des autres services et l'on peut rajouter des actions comme la mise à jour de données (par exemple, pour l'enregistrement de préférences), la recherche des informations stockées ou encore la mise en relation d'utilisateurs sur un pilote média. En terme d'entrées, nous avons considéré que tout arrivait à l'application sous la forme d'évènements transportant des messages sur un média support quelconque avec un expéditeur, un destinataire et un contenu.

3. Des scripts aux interactions multi-agent

A la lumière de l'étude des scripts et de leur abstraction en squelette de services, il est possible de voir un script comme la description d'un comportement exhibé par l'agent Unimédia vis-à-vis des autres agents lorsqu'il doit fournir le service associé. Selon cette optique, c'est le script Unimédia qui structure les interactions.

La question théorique que nous nous sommes posée dans le cadre de la coopération dans un hSMA était : "Comment concevoir un médiateur ?". Par ailleurs, dans le chapitre 2, nous avons présenté notre vision d'Unimédia comme un agent logiciel A_L communiquant avec les utilisateurs. Unimédia est, de plus, un agent contrôlé A_C puisqu'en se plaçant du point de vue du concepteur de service, nous pouvons spécifier l'intégralité de son comportement. Nous pouvons donc instancier la question théorique dans le cadre de la communication multimédia de la façon suivante : "Comment rendre Unimédia capable de remplir un rôle de médiation ?".

En reprenant le cas du service E-Nots, nous nous apercevons que trois agents $\{a_1, a_2, a_3\}$ entrent en interaction : deux agents humains $\{a_1, a_2\} \subset (A_P \cap A_{NC})$ et un agent logiciel Unimédia $a_3 \in (A_L \cap A_C)$.

Nous pouvons également affecter un rôle à chacun de ces agents dans l'interaction :

- le premier agent humain a_1 possède un rôle d'utilisateur en tant qu'expéditeur de message,
- le second agent humain a_2 joue un rôle d'utilisateur en tant que destinataire du message et
- l'agent logiciel Unimédia a_3 a en charge la médiation entre l'expéditeur et le destinataire.

Nous avons représenté sur la Figure 64, comment les trois agents interagissent au travers d'un ensemble d'éléments de communication.

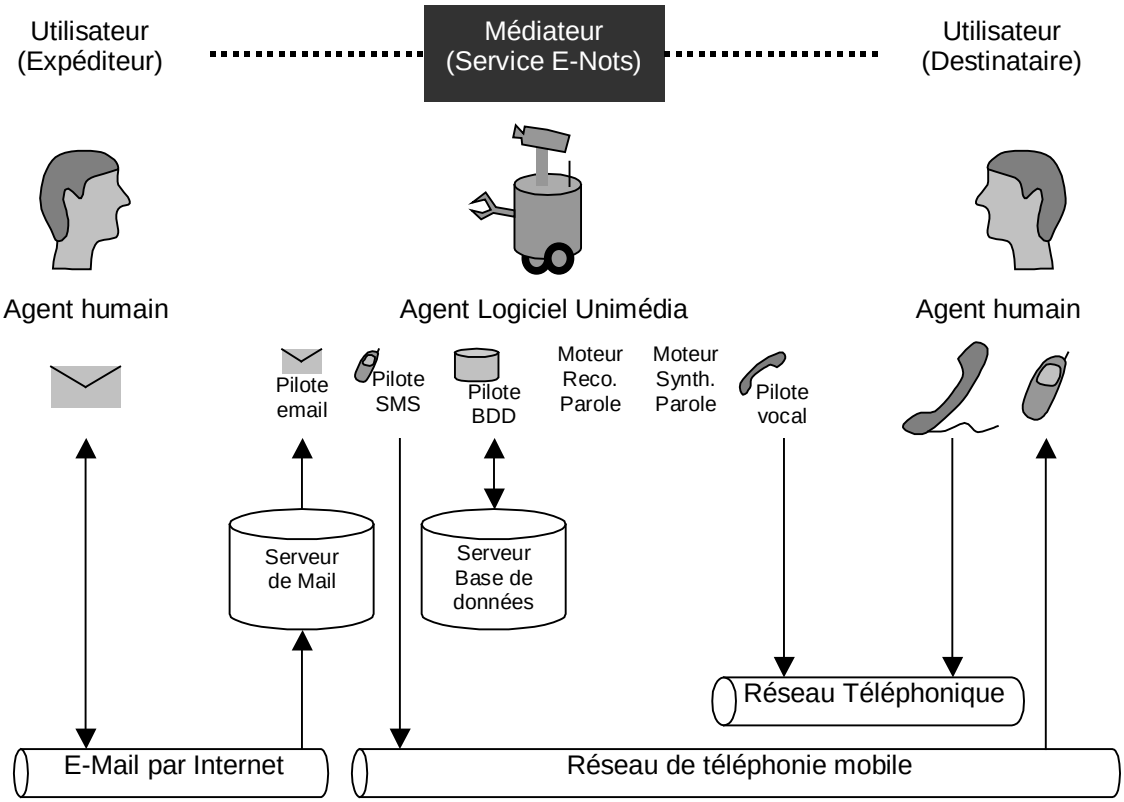


Figure 64 - Vue agent du service E-Nots

Dans la partie suivante, nous allons présenter l'architecture que nous avons définie afin d'abstraire le déroulement des interactions dans la plate-forme Unimédia.

4. Une architecture agent pour la communication multimédia

Nous avons défini une architecture avec quatre niveaux d'abstraction qui permettent d'effacer graduellement la spécificité des médias, les contraintes d'exécution et de se concentrer sur le cœur des services, en éliminant ainsi une part de la complexité. Une vue globale et générique de l'architecture est présentée dans la Figure 65. Les différents niveaux sont des vues abstraites progressives des quatre éléments des services multimédia : médias (M), ressources (R), agents (A) et services (S). L'abstraction entre les niveaux M et R permet de gommer la spécificité des médias, l'abstraction entre R et A, celle des types de ressources et enfin, on efface l'hétérogénéité des agents du hSMA avec l'abstraction entre A et S.

A la base, le niveau média gère les signaux de l'environnement de communication réel comme des flux d'informations dans des canaux média. Au-dessus de ces flux, le niveau des ressources décrit les éléments fonctionnels qui sont, en quelque sorte, branchés sur les médias. Quand une ressource est utilisée comme un effecteur, elle envoie des messages vers le média sur lequel elle est branchée. A l'inverse, une ressource peut aussi être utilisée pour recevoir des messages entrants ou comme un organe permettant un traitement spécifique de l'information. Les deux niveaux les plus bas (média et

ressource) ont pour but la gestion des aléas de l'environnement. Ils fournissent aux agents des primitives d'assez haut niveau, sous la forme d'opérateurs fonctionnels qui peuvent être combinés pour obtenir des plans d'action. C'est au-dessus, au niveau agent, que se réalise cette planification de comportements. Le niveau agent décrit les entités du hSMA qui raisonnent sur l'information et utilisent les ressources sous-jacentes pour accomplir leurs buts, dirigées par le rôle qu'elles obtiennent du niveau supérieur. Tout au-dessus, le niveau de service gère la combinaison des rôles comme une interaction de collaboration multi-agent. Cette fois, c'est la nature des agents qui est abstraite. Il n'y a finalement plus que des acteurs d'un service qui jouent chacun un rôle. A ce niveau, nous retrouvons les schémas d'interaction que nous avons définis dans le chapitre 3 : il s'agit de comportements de haut niveau associés à chaque rôle, sans détails de réalisation (contexte, contraintes de ressources, etc.). Les services sont contrôlés au niveau le plus haut en observant les évènements qui se déroulent sur les médias de sorte à maximiser la satisfaction d'un sous-ensemble donné d'agents.

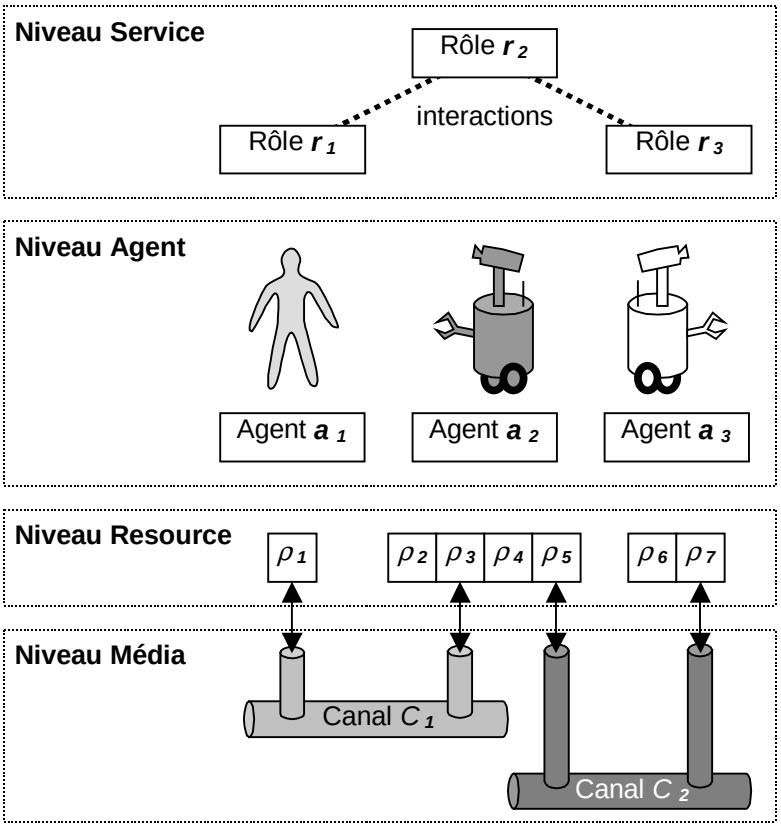


Figure 65 - Une architecture pour la communication multimédia

Le Tableau 19 décrit niveau par niveau les éléments et les fonctions que l'on retrouve dans l'architecture.

Tableau 19 - Description des niveaux de l'architecture

Niveaux	*Entités et Concepts*	*Fonctions*
Service	C'est dans ce niveau que sont considérées les sociétés d'agents : l'individu est gommé et on ne considère plus que les rôles dans les classes de services ainsi que les schémas d'interaction.	Le niveau service a la charge de la coordination des rôles pour fournir les services applicatifs.
Agent	Ici, nous retrouvons les agents produisant des actions en fonction de leurs perceptions.	Ce niveau gère les comportements de haut niveau des agents (tant pour la modélisation que pour la production).

Ressource	Nous regroupons ici l'ensemble des ressources de perception, d'action et de traitement.	Ce niveau fournit aux agents les moyens de traitement de l'information sous forme d'opérateurs de comportement élémentaires.
Média	Nous retrouvons ici les canaux médias qui servent d'environnement concret aux agents. C'est ici qu'interviennent les messages physiques et les interfaces de communication.	Ce niveau assure le transport physique de l'information avec ce qui relève de la gestion des connexions et de l'acheminement des messages.

Nous allons maintenant expliquer plus en détails chacun des niveaux de l'architecture.

4.1. Le niveau Media

Le ***niveau média*** décrit la façon dont les données sont échangées entre les agents. Il gère les signaux de l'environnement de communication réel. C'est l'endroit où le service se déroule concrètement et cela constitue une grande part de l'environnement des agents, car c'est là où ils perçoivent et agissent réellement. Ce niveau est composé de canaux média qui peuvent être vus comme des vecteurs pour transporter les flux d'informations.

Chaque médium a ses propres caractéristiques : mode de transmission (peer to peer, diffusion), adressage, topologie, délai de transmission, bruit, etc. On considère que les canaux média fournissent des ***prises média*** qui se comportent comme des interfaces pour les nivaux supérieurs. Pour être capable d'utiliser un média (afin d'envoyer ou de recevoir des données) une ressource doit supporter le type de prise qu'il fournit.

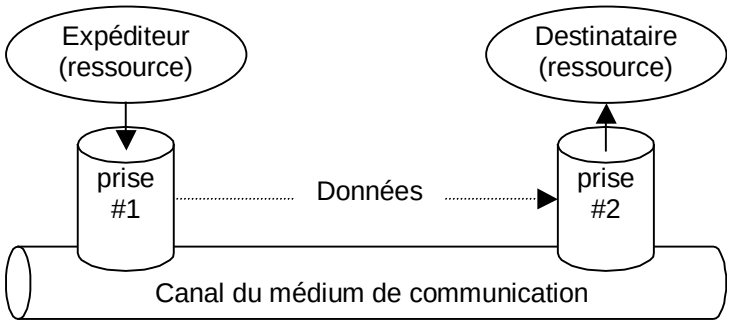


Figure 66 - Transmission sur un flux de communication

Afin d'illustrer ce que nous retrouvons dans ce niveau, nous pouvons prendre l'exemple d'un réseau fonctionnant avec le protocole TCP/IP. Le média englobe tous les niveaux inférieurs du modèle OSI *(ISO 1994)* depuis le médium physique jusqu'aux sockets TCP. Les ports TCP sont alors considérés comme des prises sur lesquelles les ressourcee d'accès au réseau peuvent recevoir ou envoyer des paquets de données.

4.2. Le niveau Ressource

Le ***niveau ressource*** décrit les éléments fonctionnels de l'environnement. Une ***ressource*** est un dispositif matériel, une partie logicielle ou corporelle utilisée par les agents pour réaliser leur tâche de façon interne ou externe dans l'environnement multimédia (au travers des prises et des canaux). Les ressources effectrices, comme les muscles, donnent aux agents des moyens de s'exprimer sur les médias alors que les capteurs, comme les yeux, leur permettent de percevoir les évènements qui se

déroulent sur les médias. Un troisième type de ressource fournit aussi un ensemble de fonctions, comme des boîtes à outils de traitement de l'information.

La granularité et la complexité des ressources sont variées : elles peuvent aller de bas niveau à haut niveau et peuvent être composées récursivement d'autres ressources plus petites. Par exemple, on peut considérer comme une ressource composée la chaîne de la Figure 67.

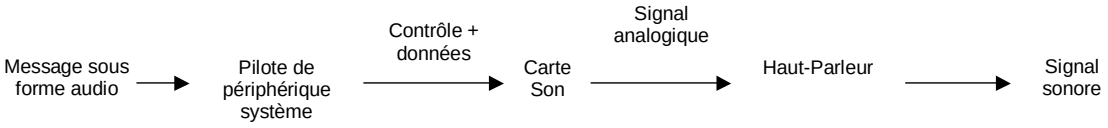


Figure 67 - Un exemple de ressource composée

La Figure 68 montre qu'il est possible de considérer une ressource comme une "boîte noire" qui offre deux types d'interfaces :

- une interface supérieure, qui permet des échanges de messages de haut niveau avec le "corps" des agents. La ressource reçoit les actions à effectuer et elle est capable de renvoyer ses messages en retour (perceptions, erreurs, résultats de traitements, etc.)
- une interface inférieure, composée de prises qui peuvent être utilisées pour accéder aux médias sur lesquels la ressource envoie et reçoit des données.

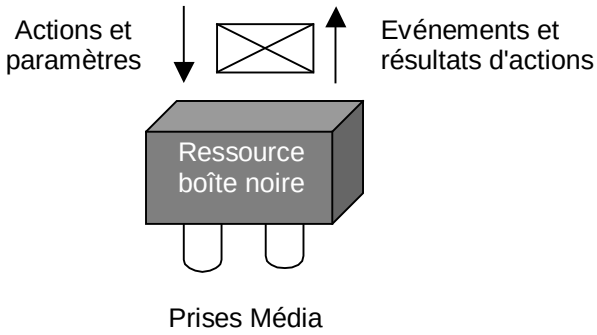


Figure 68 - Vision des ressources comme des boîtes-noires

Le Tableau 20 présente une liste, non exhaustive, des fonctionnalités de l'interface supérieure des ressources que l'on retrouve dans les pilotes Unimédia.

Tableau 20 - Exemple d'actions possibles pour les ressources Unimédia

Ressource	***Action***	***Paramètres***	***Résultat***
Base de données	connecter	• adresse du serveur • identité (+ mot de passe)	• connexion Base de Données
Base de données	déconnecter	• connexion Base de Données	
Base de données	requérir	• commande SQL • connexion Base de Données	• table résultat
Fax	envoyer	• numéro de téléphone • document	
e-Mail	envoyer	• adresse e-mail • sujet • contenu • attachement(s)	
Reconnaissance de parole	traduire	• voix	• message

SMS	envoyer	• numéro de téléphone • message	
Synthèse de parole	traduire	• message	• voix
Téléphone	appeler	• numéro de téléphone • interface	• connexion téléphonique
Téléphone	attacher	• ***n*** connexions téléphoniques	• pont téléphonique
Téléphone	déconnecter	• connexion téléphonique	
Téléphone	détacher	• pont téléphonique	• ***n*** connexions téléphoniques
Téléphone	acquisition	• connexion téléphonique	• message vocal
Téléphone	envoyer	• connexion téléphonique • message vocal	
Web	déconnecter	• numéro de connexion TCP/IP	
Web	envoyer	• numéro de connexion TCP/IP • page • fichiers utilisés	
Web	initialiser	• numéro de connexion TCP/IP	

De la même façon, le Tableau 21 décrit un ensemble de perceptions possibles depuis ces mêmes ressources Unimédia qui se placent en dehors des retours d'actions prévus.

Tableau 21 - Perceptions possibles depuis les ressources Unimédia

Source	***Perception***	***Résultat***
Fax	réception d'un fax	• numéro de téléphone expéditeur • document
Mail	arrivée d'un e-mail sur le serveur	• adresse destination • sujet • contenu • attachement(s)
Web	requête http	• requête HTTP demandée
SMS	réception d'un SMS sur le serveur	• numéro de téléphone • message
Téléphone	notification de déconnexion	• perte de connexion téléphonique
Téléphone	appel entrant	• connexion téléphonique

4.2.1. Besoin d'une description des ressources

Le fait que les ressources soient utilisées par les agents logiciels contrôlés pour effectuer des actions planifiées soulève un certain nombre de problèmes :

- Comment un agent peut-il connaître les fonctionnalités proposées par une ressource donnée ?
- Comment un agent peut-il décider quelle ressource il va utiliser pour accomplir une action donnée ?

Dans le cas d'agents physiques, comme les agents humains, la connaissance des membres de leur corps est résolue par l'apprentissage : les enfants apprennent à utiliser leurs mains pour attraper des objets et leurs yeux pour regarder les choses qui les entourent... En revanche, l'usage des objets complexes nécessite cependant d'avoir une formation ou de lire un manuel d'utilisation.

Dans le cas particulier des agents contrôlés, savoir gérer ses ressources (logicielles et matérielles) implique d'utiliser des descriptions de ressources en termes de méta-données. Une approche possible est d'imposer aux ressources de fournir une description de leurs caractéristiques, leurs fonctionnalités et leurs capacités. Typiquement, si l'on considère une ressource comme un objet (au sens de la programmation objet), le descripteur est la partie visible (donc l'interface) de l'objet : les méthodes et attributs publics.

Une solution est la description des ressources selon une formulation uniforme et structurée. Avec cette solution, la correspondance peut être établie sur les critères et donc, des ressources équivalentes peuvent être comparées et inter-changées. Pour un service dégradé, les aspects de fiabilité pourraient par exemple permettre de choisir des ressources offrant une performance moindre mais une plus grande fiabilité pour fournir les meilleurs résultats possibles. Pour illustrer cela, on fait l'hypothèse que dans une application de téléphonie, les utilisateurs aient à sélectionner un élément dans une liste. L'agent jouant le rôle d'un médiateur a le choix pour interagir avec l'utilisateur entre la reconnaissance de la parole et la DTMF (Dual Tone Multiple Frequency, comme les touches du téléphone). La DTMF peut être considérée comme plus robuste, mais l'ergonomie est lourdement dégradée par rapport à la reconnaissance de parole. Si la ligne téléphonique paraît bruitée, l'assistant virtuel pourrait "préférer" l'interface DTMF, s'il estime que les mots prononcés risquent de ne pas être reconnus avec suffisamment de certitude.

Nous proposons de faire correspondre à chaque ressource, logicielle ou matérielle, un ***descripteur***, qui sert de manuel d'utilisation pour les agents logiciels contrôlés. Au-delà du fait d'être capable d'utiliser la ressource, le but est de fournir aux agents suffisamment d'informations pour estimer le coût global d'utilisation d'une fonctionnalité donnée sur une ressource donnée (par exemple au travers d'une somme pondérée de coûts).

4.2.2. Le contenu des descripteurs

Le descripteur d'une ressource doit contenir certaines informations sur la façon d'accéder à la ressource, mais peut aussi fournir bon nombre d'autres propriétés. Le Tableau 22 décrit un ensemble de caractéristiques susceptibles d'apparaître dans le descripteur d'une ressource :

Tableau 22 - Caractéristiques permettant de décrire les ressources pour un agent logiciel contrôlé

Caractéristiques des ressources	***Description***	***Importance***	***Exemple(s)***
Accès à la ressource	Cette propriété définit une façon d'identifier la ressource.	Très grande	• Nom • Adresse...
Langage / protocole d'interaction et propriétés de codage	Cette propriété décrit les langages et les protocoles utilisés pour communiquer avec la ressource, pour réaliser des actions et interpréter des messages entrants. Elle inclut la description des divers codages utilisés.	Très grande	• Langages: KQML, SQL, POP3... • Encodages : ASCII, UTF-8...
Fonctionnalités publiées	Cette propriété liste quelles sont les fonctions qui sont fournies par la ressource en terme d'actions de perceptions et d'évènements. Chaque objet de la liste inclut également une description des paramètres et quels sont les résultats. Depuis cette liste, les agents peuvent extraire des opérateurs de planification comme décrit plus loin.	Très grande	• Consulter les messages électroniques • Générer un signal vocal • Retrouver une identité depuis une adresse e-mail ...
Type	Cette propriété décrit à quelle classe la ressource appartient. Cette propriété est statique car on fait l'hypothèse que le type de la ressource n'évolue pas.	Très grande	• Boite aux lettres • Serveur de base de données • Serveur Web • Appareil téléphonique
Paramètres de configuration	Cette propriété concerne les paramètres requis par les ressources et ainsi que les paramètres pour le branchement des prises médias.	Variable	• Ressource : activation, délai... • Prise : numéro de téléphone, Adresse IP, Port TCP, adresse e-mail...
Accès, confidentialité	Cette propriété décrit les dispositifs de sécurité relatifs à la ressource, comme l'authentification des usagers.	Grande	• A-t-on besoin d'un mot de passe et d'un nom d'utilisateur ? • Restrictions • Cryptage • Niveau de certification du système
Coûts d'utilisation	Il est nécessaire d'informer l'agent le plus possible sur les coûts d'utilisation d'une ressource relativement aux actions qu'il peut demander.	Grande	• Coûts financiers : en € par minute • Coûts en fonction de la taille des données...
Capacité / Fiabilité / robustesse	Cette propriété décrit la charge que la ressource peut supporter et le niveau de qualité que l'on peut en attendre.	Utile	• Nombre de connexions simultanées • Vitesse de débit
Périodes de disponibilité	Cette propriété informe les agents des plages pendant lesquelles la ressource est disponible.	Utile	• plages horaires d'accessibilité

Le descripteur peut par exemple prendre la forme d'un document que l'agent reçoit quant il acquiert la connaissance d'une ressource donnée. Pour cela, diverses alternatives sont possibles :

- La ressource peut publier elle-même son propre descripteur,
- Un sous-système de découverte du gestionnaire de ressources peut accéder à la ressource au travers de son interface, l'analyser et construire le descripteur, ou encore,
- Les agents peuvent essayer de déterminer d'eux-mêmes les caractéristiques des ressources à l'aide par exemple d'un processus réflexif.

4.3. Le niveau Agent

Le ***niveau Agent*** contient l'ensemble des "acteurs" qui vont "jouer un rôle" dans le service que nous considérons comme un hSMA. Nous reprenons les classes d'agents qui ont été définies dans le chapitre 3 et nous examinons plus précisément comment, selon le contrôle, chacune des partitions peut se retrouver dans le niveau agent.

4.3.1. Classe des agents contrôlés A_C

La classe des agents contrôlés est composée par le sous-ensemble des agents qui est sous l'influence explicite des services de la plate-forme. Dans les services Dialoca, ces agents sont des agents logiciels Unimédia. Le comportement des agents contrôlés peut être défini par planification ou par apprentissage en fonction :

- de leurs buts et des contraintes qu'ils ont reçus, comme des limites temporelles,
- des opérateurs issus des ressources dont ils disposent,
- de leurs perceptions,
- de leur état estimé, ou encore
- du coût qu'ils doivent minimiser.

On peut considérer que les agents de cette classe sont les plus "responsables" quant à l'adéquation et à l'accomplissement de leur rôle dans le service. Dans le cas d'un comportement appris par renforcement, on retrouvera, par exemple, une récompense qui dépendra de la bonne exécution du service. Par ailleurs, ce sont eux qui doivent adapter leur comportement à celui des autres agents et non l'inverse. Pour cela, ils doivent maintenir un modèle du comportement des agents partiellement contrôlés qui dépend de la description du service et qui est enrichi de ce qu'ils observent d'eux. Maintenir à jour un tel modèle peut permettre d'éviter certains risques de malentendu et favorise la coopération entre les agents.

4.3.2. Classe des agents partiellement contrôlés A_{PC}

Cette classe contient les agents "tiers" qui interviennent dans les services. Il s'agit par exemple des agents partenaires, considérés au travers de leur serveur web, mais aussi des humains (comme des experts ou des conseillers). Ils ont également à suivre le rôle qu'ils ont contracté dans le service. Il peut leur être demandé de réaliser des actions, mais de façon indirecte, au travers des agents contrôlés.

4.3.3. Classe des agents non contrôlés A_{NC}

Tous les autres agents (comme les clients, les utilisateurs, etc.) se retrouvent dans la classe des agents non contrôlés. En effet, leur comportement est inconnu au départ. En revanche, ils sont tout de même supposés jouer un rôle dans le service, c'est pourquoi un comportement "typique" peut être utilisé par les agents contrôlés pour interagir avec eux. Ici, il est possible d'utiliser des techniques de modélisation de comportement et de reconnaissance de plans telles que celles présentées au chapitre 5 pour savoir ce qui les motive et pour essayer de prédire leurs buts et leurs actions.

4.3.4. Enregistrement des opérateurs fonctionnels

Les agents agissent au travers des ressources et, dans le cas d'agents contrôlés, ils s'adressent à la ressource de leur choix pour effectuer des actions en utilisant les opérateurs adéquats. Lorsque nous avons décrit les ressources, nous avons évoqué la nécessité que les agents obtiennent les descriptions fonctionnelles des capacités pour les ressources. Nous proposons alors que les descripteurs de ressources soient traités par les agents pour en extraire les opérateurs qui seront enregistrés dans une structure de treillis, comme le montre la Figure 69.

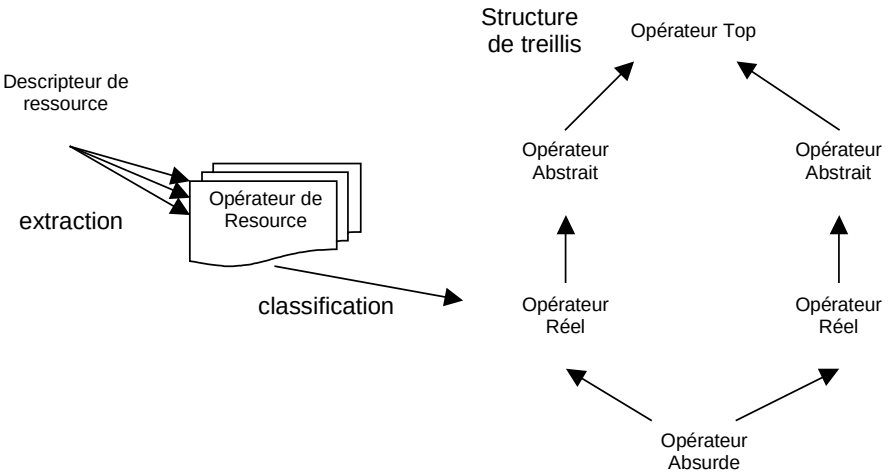


Figure 69 - Enregistrement d'un descripteur dans un treillis d'opérateurs fonctionnels

Dans ce treillis d'opérateurs ***(Op, ≤, T, ⊥)*** :

- Nous proposons d'admettre dans le treillis différents types d'opérateurs :
  - ★ les opérateurs abstraits Op_A
  - ★ les opérateurs réels atomiques Op_R , qui sont issus des descripteurs de ressource
  - ★ les macro-opérateurs composés Op_C , qui sont des combinaisons d'opérateurs

  L'ensemble des opérateurs ***Op*** du treillis est donc défini par : $Op = Op_A \cup Op_R \cup Op_C$
- l'élément Top (***T***) est l'opérateur universel : il représente l'action la plus générale qui ne fait rien, mais qui peut théoriquement être utilisée partout.
- En bas du treillis, nous retrouvons un opérateur théorique dit absurde (⊥) qui peut être vu comme le plus spécifique.
- L'ordre partiel ≤ du treillis est utilisé pour comparer les opérateurs. Pour deux opérateurs ***A*** et ***B*** du treillis, la sémantique de $A \leq B$ indique que ***B*** est plus général que ***A*** (qui est plus spécifique) ou, en d'autres termes que ***B*** est une abstraction de ***A***. Bien sur, sans opérateur réel dans le treillis, les agents ne seraient pas capables d'effectuer des actions réelles. Pour construire des plans réels, les agents peuvent commencer par planifier hiérarchiquement en utilisant un opérateur abstrait ***B*** et ensuite choisir quel opérateur ***A*** conviendrait le mieux en fonction du contexte à l'exécution. Un exemple typique d'opérateur abstrait est l'envoi d'une information à un destinataire de façon asynchrone. Deux opérateurs réels peuvent être utilisés pour le réaliser : l'envoi d'un message e-mail ou l'envoi d'un message SMS.

Avant l'enregistrement de toute ressource, le treillis ne contient que des opérateurs abstraits et quelques opérateurs concrets de base permettant de stocker des informations et les retrouver. Les schémas des opérateurs concrets sont extraits des descripteurs de ressource et on utilise un procédé de classification pour les positionner dans le treillis. Un agent apprenant peut aussi se remémorer les sous-plans qu'il estime utiles et réutilisables. Ces sous-plans ainsi que d'autres combinaisons de ressources seraient stockées dans le treillis comme des macro-opérateurs composés pour être réutilisés ultérieurement.

4.3.5. Structure des agents contrôlés

Nous proposons de structurer les agents contrôlés autour d'un noyau qui peut recevoir les ressources comme des composants périphériques. Le cœur de l'agent obtient les évènements entrants à partir des ressources de perception et les transmet à une ressource spécifique qui est utilisée pour déterminer les actions à effectuer : il s'agit, en quelque sorte, du "cerveau" de l'agent qui permet, par exemple, d'utiliser un système à base de règles, un apprentissage par renforcement, etc. Une fois que les actions à effectuer ont été déterminées, l'agent utilise le treillis d'opérateurs pour les décliner de façon adéquate et demande l'exécution aux ressources associées en transmettant les paramètres nécessaires.

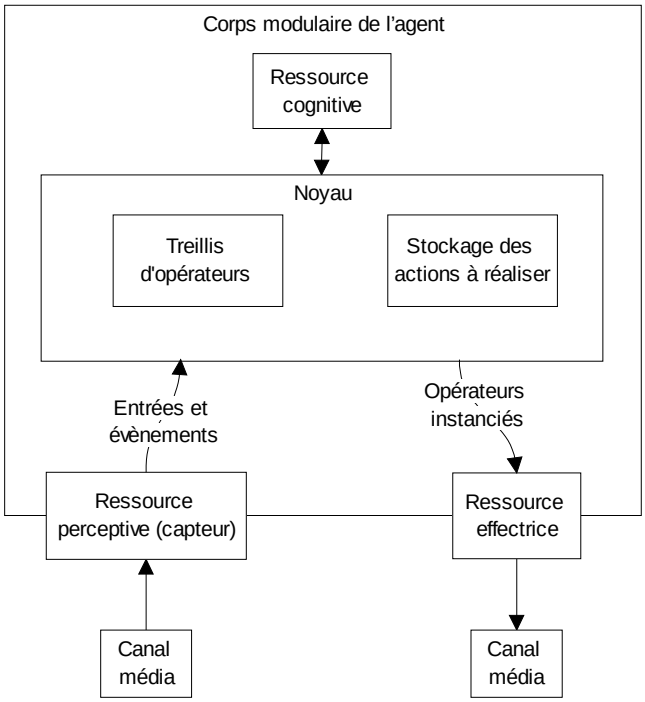


Figure 70 - Un corps pour les agents contrôlés

4.4. Le niveau Service

Dans le ***niveau service***, nous retrouvons les différents rôles qui interagissent dans le service considéré. A chaque agent du niveau inférieur, nous associons un rôle distinct dans le service. C'est également dans cette partie de l'architecture qu'interviennent les schémas d'interaction que nous avons définis dans le chapitre 3. Nous avons notamment indiqué que, dans un graphe de comportement associé à un rôle, les arcs correspondaient à une transition du comportement entre deux états, soit en réaction à une perception, soit au travers d'une action.

La Figure 71 présente le déroulement d'une transition dans le cas d'une action effectuée par un agent logiciel contrôlé $\boldsymbol{a_1} \in \boldsymbol{A_{LC}}$ ayant endossé un comportement de graphe $\boldsymbol{G = (S, \Gamma)}$. La transition $\gamma \in \Gamma$ se traduit par l'utilisation d'un opérateur abstrait $\boldsymbol{o_1} \in \boldsymbol{Op_A}$ puisé dans le treillis d'opérateurs que nous avons présenté. Lorsqu'il réalise effectivement l'action, l'agent utilise l'opérateur concret approprié $\boldsymbol{o_2} \in \boldsymbol{Op_C}$ avec $\boldsymbol{o_2} \preceq \boldsymbol{o_1}$ et agit ainsi au travers d'une ressource effectrice ρ_1. Le canal média $\boldsymbol{C}$ sous-jacent est alors modifié et produit un flux d'informations. Lorsque ce flux arrive à une ressource ρ_2 d'un autre agent $\boldsymbol{a_2}$ qui peut le percevoir, les informations remontent jusqu'à celui-ci sous la forme d'un évènement, ce qui produit une transition γ' dans son graphe $\boldsymbol{G' = (S', \Gamma')}$.

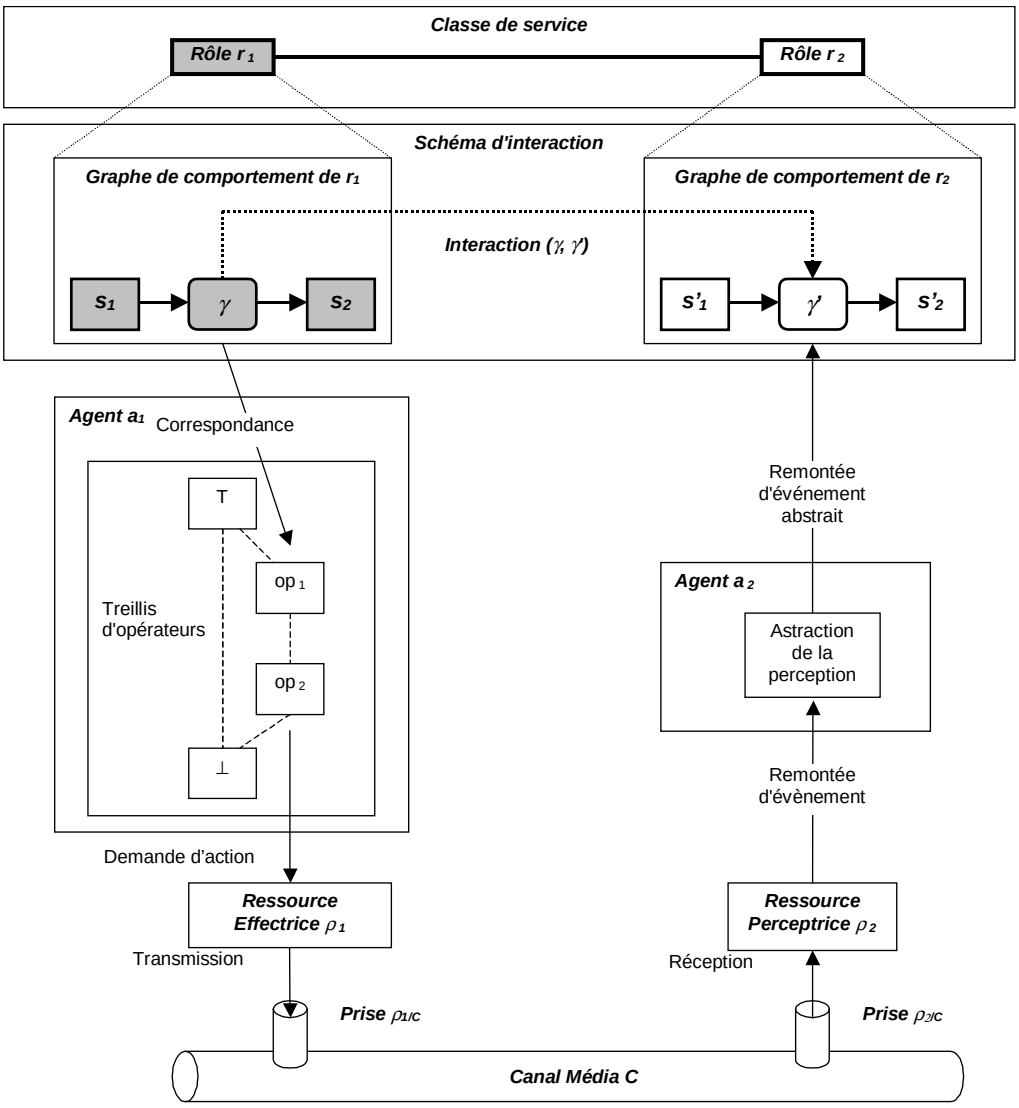


Figure 71 - Déroulement dans l'architecture d'une transition du graphe de comportement

5. Les approches de conception des services

L'architecture que nous venons de décrire permet, de suivre plusieurs approches de conception de services, qui ont chacune leurs avantages et leurs inconvénients : l'approche par décomposition, la planification de service et l'apprentissage de service par renforcement.

5.1. L'approche par décomposition

Ce processus de conception est basé sur la construction progressive d'un squelette de service par le concepteur à l'aide d'un arbre de décomposition. En racine de l'arbre, nous retrouvons la description du service la plus générale possible, puis les tâches sont décomposées à l'aide de nœuds (et, ou, itération, choix, etc.) et au niveau des feuilles, nous retrouvons des actions abstraites, telles que celles que nous avons présentées dans le Tableau 18. A titre illustratif, la Figure 72 donne l'arbre de décomposition simplifié pour un service de notification comme E-Nots.

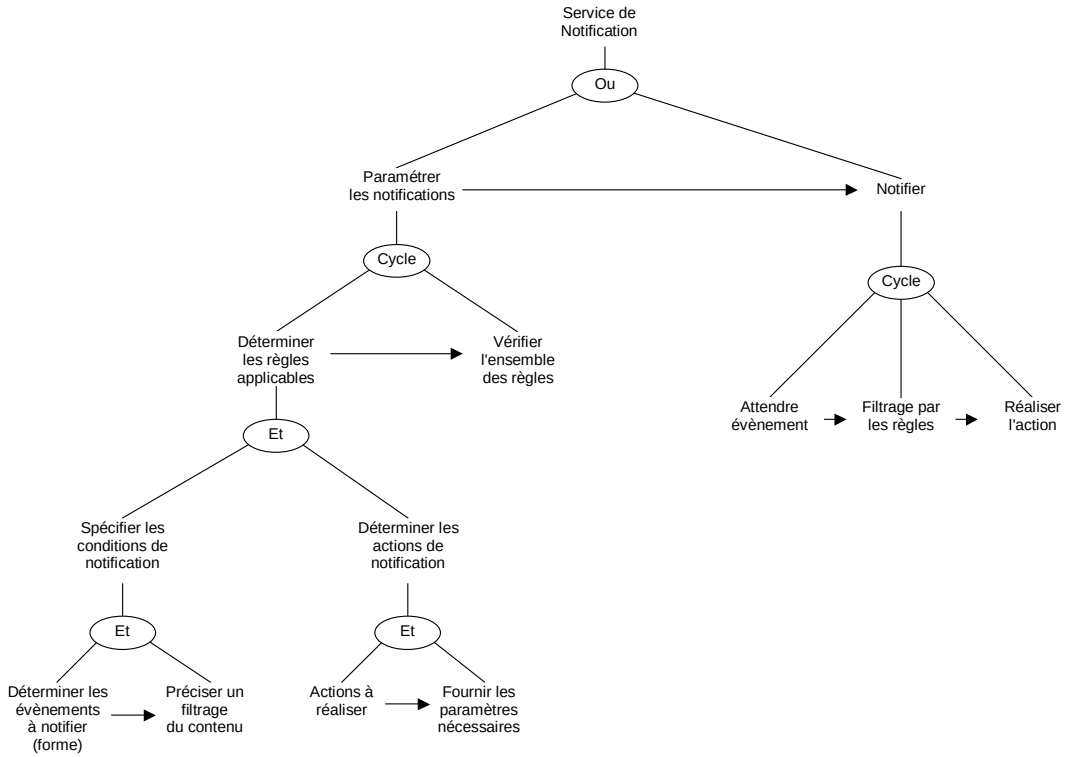


Figure 72 - Exemple de décomposition pour un service de notification

Cette approche permet en fait de guider la conception, mais ne permet pas d'obtenir des comportements adaptatifs aux aléas. Son étude, dans le cadre de nos travaux a été relativement restreinte, car elle a laissé la place aux approches par planification et par la suite à l'apprentissage de comportements par renforcement.

5.2. L'approche par planification classique

Dans cette approche, le concepteur spécifie l'événement susceptible de déclencher le service ainsi que le but à atteindre. L'agent logiciel peut alors planifier un comportement à partir de l'ensemble des opérateurs dont il dispose. En effet, les fonctionnalités publiées par les ressources peuvent être utilisées par les agents contrôlés comme des opérateurs de planification qui font évoluer le monde d'un état à un autre. A titre d'exemple, la Figure 73 montre un plan qui pourrait être construit à partir des différents opérateurs de ressources pour le service E-Nots. Nous avons représenté les opérateurs de planification par des boîtes grisées. Les labels arrondis au-dessus et en dessous représentent respectivement les faits qui sont vrais avant et après l'application.

L'approche par planification offre l'avantage d'éviter au concepteur de créer le plan dans le détail, cependant, comme nous l'avons précisé dans le chapitre 4, lorsque nous avons présenté cette approche, ainsi que dans l'annexe I pour laquelle nous avons donné un autre exemple de service planifié, il est difficile de dépasser les comportements séquentiels et de prendre en compte l'incertitude.

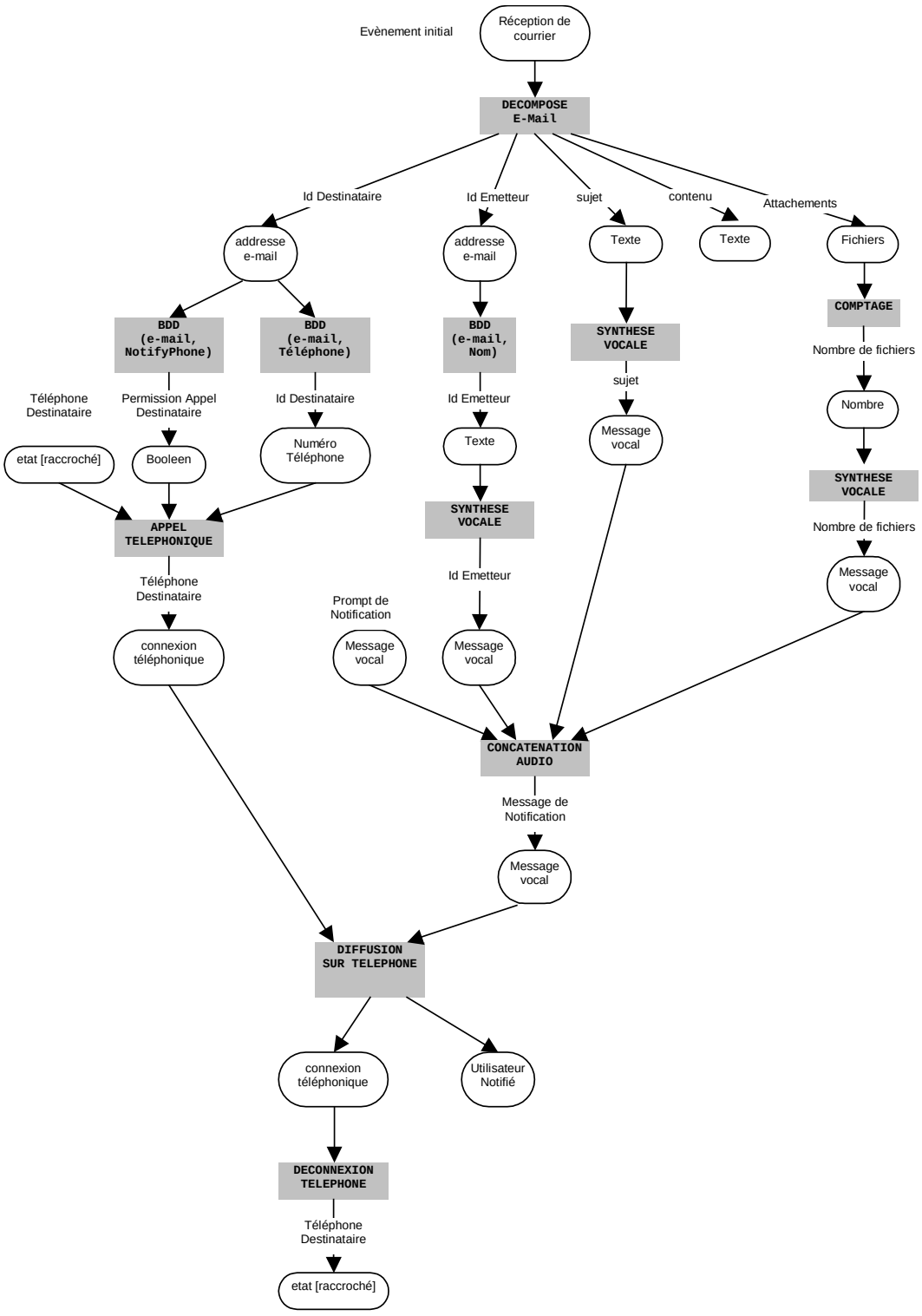


Figure 73 - Un exemple de plan pour la notification téléphonique

5.3. L'approche de l'apprentissage par renforcement

Il s'agit de la méthode que nous avons suivie pour la conception du médiateur dans un service de recherche d'informations dans le chapitre 6. Le principe consiste à définir le service comme un MDP en précisant :

- l'ensemble des états ***S***,
- l'ensemble des actions ***A*** abstraites que l'agent de service peut utiliser, ainsi que

- l'ensemble des récompenses ***R*** de renforcement qu'il peut recevoir.

Cette approche offre à nouveau l'avantage d'éviter au concepteur de décrire le comportement de l'agent, mais de plus, elle est adaptative quant aux divers aléas qui peuvent se produire. En revanche, elle nécessite un apprentissage dont la durée dépend de la complexité de la tâche à réaliser.

6. Conclusion

Dans ce chapitre, nous sommes partis de l'étude des scripts de services et nous avons proposé de les généraliser, notamment au travers de squelettes de services. Nous avons ensuite présenté une architecture conceptuelle permettant d'abstraire la complexité inhérente aux différents médias et ressources qui entrent en jeu dans un service de communication multimédia. Dans la fin du chapitre, nous avons résumé les différentes voies que nous avons considérées pour concevoir un service : les schémas de décomposition, la planification classique et l'apprentissage par renforcement.

En fait, le choix de la méthode de conception, et donc du type de "sous-cerveau" à placer dans l'agent est plutôt dépendante du type de service à réaliser. Pour un service conséquent, nous conseillerions une approche par décomposition, car les autres méthodes conduiraient à une explosion en terme de complexité. En revanche, si le nombre d'états possible reste raisonnable, nous conseillerions une modélisation à base de MDP qui offre l'avantage de l'adaptation pendant l'exécution du service.

Dans le chapitre suivant, nous allons décrire le prototype d'agent contrôlé que nous avons réalisé et son utilisation dans cadre d'une application pour laquelle nous avons effectué l'apprentissage d'un comportement par renforcement.

Chapitre 8 - Mise en œuvre expérimentale

Afin de pouvoir valider notre approche expérimentalement, nous avons réalisé diverses applications. Dans un premier temps, nous allons présenter une bibliothèque de programmation, appelée SmallMu, permettant d'obtenir un corps d'agent contrôlé qui utilise des ressources média. Nous décrirons ensuite notre implantation d'un prototype de médiateur, correspondant à la spécification que nous avons présentée dans le chapitre 6, et son utilisation pour la réservation d'un voyage aérien.

1. SmallMu, un corps pour les agents contrôlés

Pour des raisons pratiques, nous avons eu besoin de nous doter d'une plate-forme légère sur laquelle nous pouvions utiliser un certain nombre de médias, sans requérir à l'utilisation du système Unimédia complet. C'est à cette occasion que nous avons progressivement, et au fil des besoins, constitué la bibliothèque expérimentale SmallMu (qui signifie littéralement "petite unification de multiples médias"). La bibliothèque, écrite en langage Java, permet de réaliser le corps d'un agent contrôlé et offre l'accès à divers canaux de communication multimédia au travers de ses ressources. Ces ressources encapsulent aussi bien l'accès aux serveurs de messagerie électronique Internet, l'envoi de messages SMS ou les communications avec une base de données.

Le choix du langage Java se justifie, tant pour des raisons conceptuelles que pratiques. Tout d'abord, il s'agit d'un langage objet, ce qui permet un développement structuré et une grande modularité. Ensuite, le choix de Java nous a semblé judicieux, car l'ensemble des classes standard est déjà très fourni et uniforme, notamment pour tout ce qui concerne la communication et les interfaces. Par ailleurs, nous avons pu ainsi bénéficier des nombreuses standardisations de ce langage, en ce qui concerne les fonctionnalités évoluées, telles que la gestion de la parole avec la Java Speech API évoquée plus loin.

1.1. Composants de l'architecture de base (Noyau)

La bibliothèque SmallMu est axée autour d'un noyau dont l'essentiel des classes de l'implantation se retrouve dans la Figure 74.

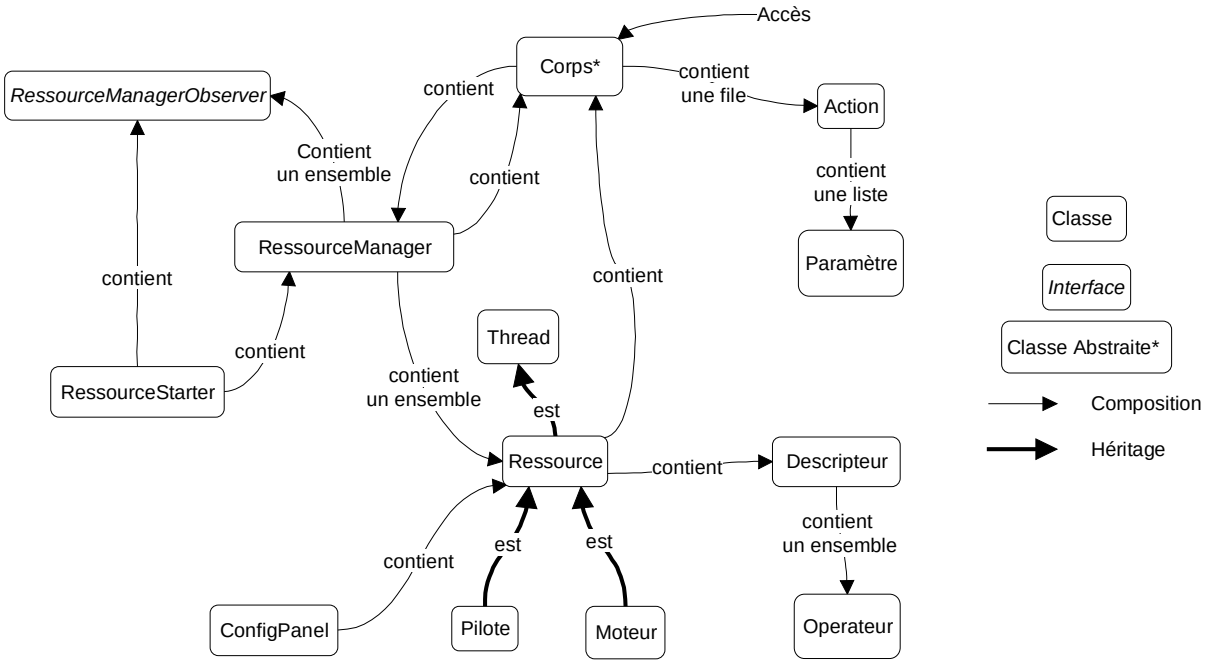


Figure 74 - Les classes du noyau SmallMu

1.1.1. Le corps

Avec une inspiration anatomique, le ***corps*** (schématisé Figure 75) est la classe au travers de laquelle les entrées en provenance des ressources périphériques remontent à la ressource cerveau, soit sous la forme d'une simple entrée textuelle, soit sous forme de structure récursive à attributs typés. Le corps gère également une file d'actions à réaliser, en provenance de la partie cerveau. Le corps d'agent utilise un gestionnaire auquel se rattachent les différentes ressources qui sont actuellement lancées pour l'agent. Dans la pratique, nous avons abstrait cette classe afin de pouvoir utiliser différents types de corps selon nos besoins.

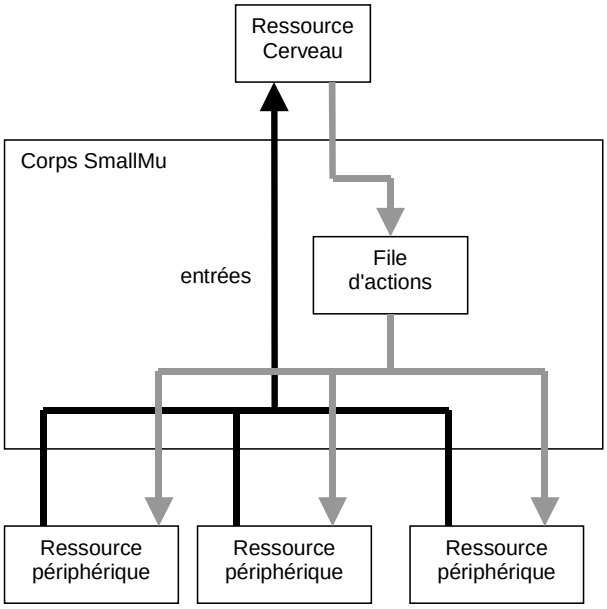


Figure 75 - Flux perceptions/actions dans le corps de l'agent

1.1.2. Les actions

Les actions sont constituées d'une référence d'identification, d'un nom d'opérateur à appliquer, d'un type de ressource destinataire et d'un ensemble de paramètres formés de couples attribut-valeur. Le destinataire d'une action peut éventuellement être "interne" (par exemple, pour demander au gestionnaire de ressources le démarrage d'une ressource d'un type donné), au quel cas, l'action est gérée au niveau du corps de l'agent.

1.1.3. Ressources

Les ressources sont conçues comme des membres que l'on peut attacher et détacher du corps de l'agent. L'attachement d'une ressource se traduit par une instanciation de la classe correspondante et au démarrage de son thread. A l'inverse, le détachement est réalisé par un arrêt et une libération de la ressource, si nécessaire.

Les ressources de la plate-forme se répartissent en plusieurs types. Nous distinguons d'une part des ressources dites "***cerveaux***" comme le gestionnaire d'interactions que nous présenterons plus loin ou encore un petit module, appelé JessAgent, qui encapsule l'utilisation d'un moteur à base de règles Jess *(Friedman-Hill 2003)*, pour la notification orale de courriers électroniques. D'autres ressources sont spécialisées en ***pilotes*** qui, à l'instar des pilotes Unimédia, gèrent l'interface avec un média de communication particulier.

Pour chaque ressource, nous avons associé un descripteur XML décrivant quelques informations accompagnées de la liste des opérateurs utilisables. L'exploitation du descripteur au niveau du prototype reste cependant limitée, car nous ne l'avons pas mis en œuvre sur l'approche de planification hiérarchique à opérateurs abstraits du Chapitre 7. Un exemple de descripteur est donné ci-dessous :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ressource SYSTEM "ressource.dtd">
<RESSOURCE>
  <MEDIA>
    <TYPE>TTS</TYPE>
    <PROTOCOL>API javax.speech</PROTOCOL>
  </MEDIA>
  <OPLIST>
    <OPERATOR>
      <OPNAME>setPrompts</OPNAME>
      <PRECOND>name</PRECOND>
    </OPERATOR>
    <OPERATOR>
      <OPNAME>say</OPNAME>
      <PRECOND>sentence</PRECOND>
    </OPERATOR>
    <OPERATOR>
      <OPNAME>res_say</OPNAME>
      <PRECOND>prompt</PRECOND>
    </OPERATOR>
  </OPLIST>
</RESSOURCE>
```

Nous avons par ailleurs gardé la possibilité que nous avions utilisée pour accéder à un paramètre ***x*** donné d'une ressource par reflexivité, au travers de méthodes de consultation ***getX*** et de modification ***setX***. C'est la classe ConfigPanel qui exploite notamment cette possibilité en offrant une interface graphique générique pour le paramétrage d'une ressource.

1.2. Un ensemble de ressources

Afin de permettre l'accès à divers média, nous avons cherché à constituer un ensemble de ressources pilotes de communication.

1.2.1. Pilote Console

Le pilote console constitue l'interface de communication la plus simple avec l'utilisateur se trouvant sur la machine hôte. Il permet d'envoyer des messages sur la fenêtre console de l'utilisateur et de saisir des informations textuelles à partir d'entrées au clavier.

1.2.2. Pilote SMS

Afin de réaliser le pilote gérant les messages SMS, nous avons utilisé une interface, développée par Eric Lucchese de l'équipe MAIA, qui se connecte via une liaison série RS-232 sur un modem GSM. Le modem est piloté par un ensemble de commandes AT permettant le stockage et l'envoi de SMS. En pratique, afin de mutualiser l'accès à cette interface un pont HTTP/CGI a été installé. En revanche, ce pont et divers autres problèmes restreignant la possibilité de réception de SMS, le pilote a été limité à l'envoi de messages.

1.2.3. Pilote BDD

Le pilote de base de données permet d'envoyer des requêtes en langage SQL à une base de données. Il s'agit d'une implantation limitée à quelques requêtes SQL simples. Pour construire ce pilote, nous avons utilisé un pont JDBC (Java Data Base Connectivity) / ODBC (Open Database Connectivity).

1.2.4. Pilote Mail

Le pilote mail permet l'accès à une boîte aux lettres électronique, avec réception et envoi de courrier.

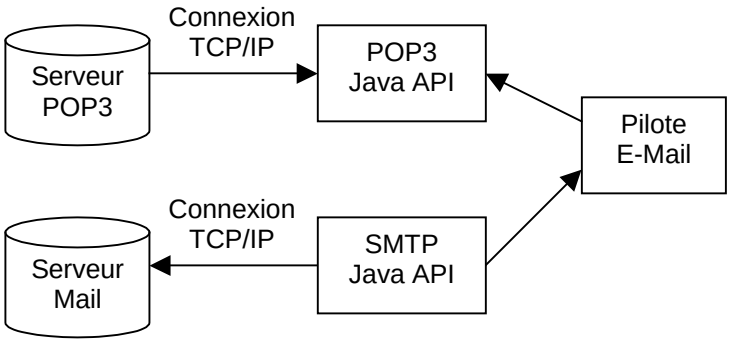


Figure 76 - Le pilote E-Mail

Comme l'illustre la Figure 76, le pilote E-Mail est composé de deux parties :

- La partie de réception des messages s'active à intervalles réguliers. Elle utilise le protocole POP3 pour se connecter, via TCP/IP, sur le serveur de courrier électronique spécifié. Elle récupère les messages et analyse notamment les lignes d'en-tête à la recherche de l'expéditeur, du destinataire et du sujet.
- La partie d'envoi de messages utilise le protocole SMTP pour se connecter, à nouveau par TCP/IP, sur le serveur indiqué. Elle demande alors l'envoi d'un message électronique au serveur.

En pratique, nous nous sommes appuyés sur les librairies POP3 de John Thomas (http://www.cruzio.com/~jthomas/) et JFA org.net.jfa de Jason Adams et Daniel Frohlich (http://www.frohlich.net/java).

1.2.5. Pilote Reco

Cette ressource gère une reconnaissance automatique de la parole en français sur le moteur IBM Viavoice 8 à l'aide de l'interface Javax.Speech de SUN *(JSAPI 1998)* de son implantation IBM Java Speech API. Nous avons choisi d'utiliser une reconnaissance à base de règles écrites dans le langage Java Speech Grammar Format *(JSGF 1998)*. Ce choix, à l'inverse des grammaires de dictée, permet de

limiter les entrées possibles à une combinaison de règles syntaxiques prévues. A titre d'exemple, nous donnons ci-dessous la règle que nous avons défini pour gérer le cas où la personne qui parle indique qu'elle ne sait ou ne veut pas donner une information en réponse à une question.

```
public <JNSP> = [je [ne]] sais pas {clore}
           | (j'en | je n'en) sais rien  {clore}
           | inconnu {clore}
           | peu [m'] importe {clore}
           | [(j'en | je n'en) ai] aucune idée {clore}
           ;
```

Sur chaque ligne, on retrouve les différentes formes acceptées par le système. L'utilisateur peut donc prononcer "Je ne sais pas" ou "Je n'en ai aucune idée". Les groupes de mots entre crochets sont optionnels et la barre verticale indique les alternatives possibles. A la fin des règles, on retrouve les étiquettes retournées par le moteur de reconnaissance lorsque l'intégralité de la phrase a été reconnue. Une difficulté que nous avons rencontrée à ce niveau pour la prise en compte de l'incertitude des entrées est qu'il n'est pas possible d'obtenir la liste des ***n*** meilleures entrées avec leurs probabilités associées (dites "n-bests"). Le pilote de reconnaissance que nous avons implanté est capable de charger un ensemble de règles et de les activer, ou de les désactiver, au besoin pendant la médiation. Cette contextualisation permet de limiter les risques de mauvaise reconnaissance des phrases prononcées par l'utilisateur.

1.2.6. Pilote TTS

Le moteur de synthèse de parole (dit TTS pour Text To Speech) est, lui aussi, basé sur l'interface Javax.Speech et utilise le moteur IBM True Voice. Le moteur permet de faire prononcer un ensemble de phrases prédéfinies. Il est également possible de demander à la ressource de prononcer une phrase qui lui est transmise dynamiquement. Dans certains cas, comme pour notre prototype de médiateur, nous utilisons une méthode intermédiaire mêlant des intitulés statiques dans lesquels certaines parties sont paramétrées et remplacées dynamiquement, en fonction des paramètres de l'action demandée.

2. Description du prototype de médiateur

La spécification du médiateur, présentée au chapitre 6, a servi de base pour l'implantation d'un prototype en Java qui a été testé sur l'application de réservation de vol aérien. Cette partie donne un aperçu de son implantation en termes d'architecture et de fonctionnalités.

2.1. Architecture du prototype

Comme le montre la Figure 77, le prototype est construit autour du corps générique d'agent SmallMu que nous venons de présenter. Il utilise les ressources génériques de reconnaissance et de synthèse de parole (*Pilote Reco* et *Pilote TTS*), ainsi que la ressource d'accès à une base de données (*Pilote BDD*).

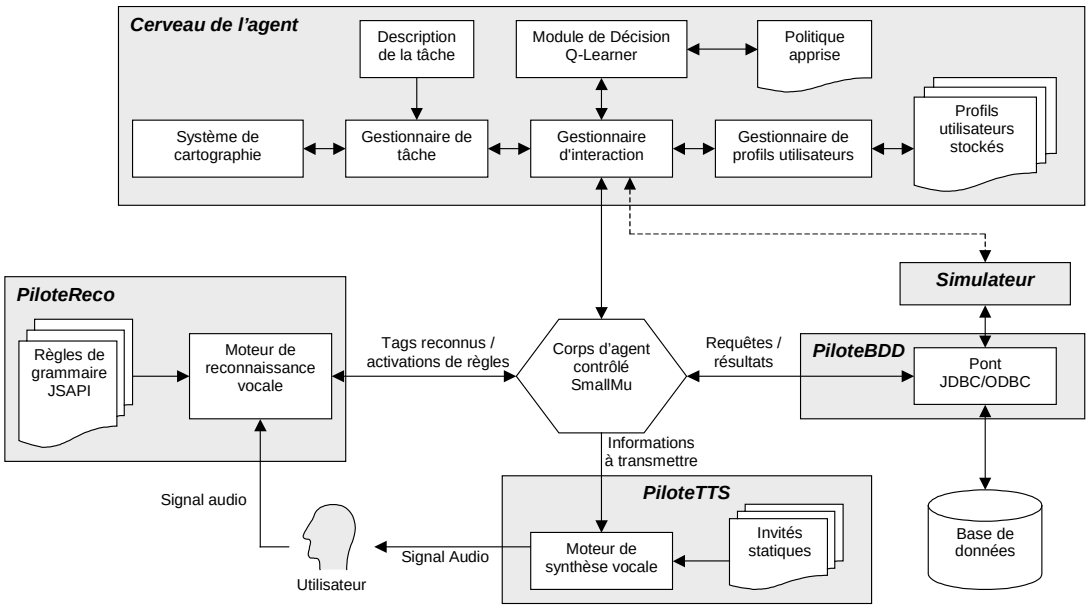


Figure 77 - Architecture du prototype d'agent médiateur

Le "cerveau" ressource du médiateur est composé d'un gestionnaire d'interactions lié à un gestionnaire de tâche et à un gestionnaire de profils utilisateurs. Le comportement global de l'agent médiateur est contrôlé par un module de décision qui utilise l'algorithme du Q-Learning.

2.1.1. Le gestionnaire d'interactions

Ce module est à la base de la ressource "cerveau". A l'initialisation, il reçoit la description de l'application qu'il utilise pour paramétrer le gestionnaire de tâche et le module de décision. Il gère ensuite les cycles d'interactions avec l'utilisateur, en traitant les entrées obtenues depuis le corps de l'agent. Lorsqu'il reçoit une entrée qui permet de faire évoluer la tâche, il transmet cette information au gestionnaire de tâche. Il obtient alors le nouvel état abstrait de la tâche et demande au module de décision quelle est l'action à réaliser. Les actions sont ensuite dirigées vers le corps de l'agent avec les paramètres nécessaires. Le gestionnaire d'interactions maintient à jour le compte du nombre d'interactions passées, avec une limite fixée à ***maxInteractions***, de sorte à restreindre la longueur de la médiation.

2.1.2. Le module de décision

Le module de décision est une implantation de l'algorithme du Q-Learning dont notre pseudo-code est donné en annexe II. Lors de son initialisation, il reçoit du gestionnaire d'interactions l'ensemble des états abstraits possibles, l'ensemble des actions qu'il peut sélectionner et l'état abstrait de départ. Il est ensuite appelé lors d'un cycle d'interactions pour déterminer l'action suivante.

L'utilisation du Q-Learning implique la définition d'un certain nombre de paramètres. Tout d'abord, il faut lui préciser le coefficient γ d'escompte du gain qui est utilisé dans la mise à jour des Q-Valeurs. Ensuite, comme nous utilisons une sélection d'action avec une distribution de probabilités de Boltzmann, le module a besoin d'un paramètre T appelé température. Nous avons choisi de limiter la température de Boltzmann dans un intervalle $[t_{Min} ; t_{Max}]$. Pour plus de souplesse dans le paramétrage du modèle, nous utilisons une décroissance combinant une partie géométrique (avec un coefficient ***tFactor***) et une partie linéaire Δ_T, ce qui donne la formule suivante :

$$T_{n+1} = tFactor \ . \ T_n + \Delta_T$$

La mise à jour des Q-Valeurs nécessite également l'utilisation d'un taux d'apprentissage $\alpha \in [0;1]$. Ce paramètre détermine comment la valeur d'une action dans un état donné doit être prise en compte par rapport à ce qui a déjà été appris. Les états ne sont pas tous visités avec la même fréquence, nous

avons choisi de faire décroître le taux α de façon inversement proportionnelle au nombre de visites des états v_s depuis un taux de départ α_0. Comme il est parfois utile de différer cette décroissance en laissant un fort apprentissage au départ, nous avons ajouté un seuil $\textbf{\textit{Seuil}}_\alpha$ sur le nombre de visites en dessous duquel la décroissance est désactivée, car à ce moment, nous estimons que l'agent n'a pas acquis suffisamment de connaissances sur l'état courant. Nous obtenons ainsi les formules suivantes :

$$\alpha_s = \alpha_0 \quad \text{si} \quad v_s < \text{Seuil}_\alpha$$

$$\frac{1}{v_s - \text{seuil}_\alpha + \frac{1}{\alpha_0}} \quad \text{si} \quad v_s \geq \text{Seuil}_\alpha$$

La Figure 78 montre un exemple de décroissance de α pour α_0 = 0,5 et $\textbf{\textit{seuil}}_\alpha$ = 10.

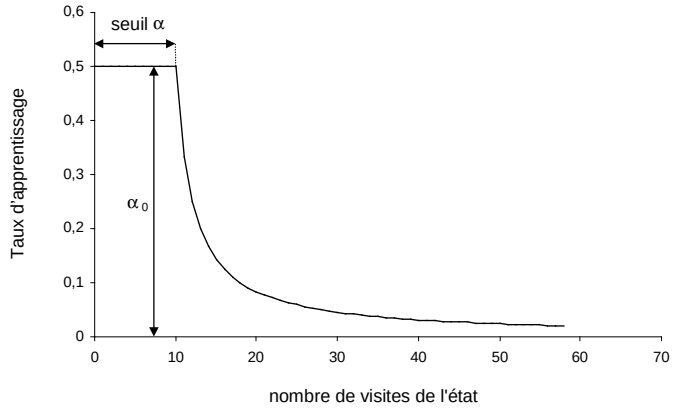


Figure 78 - Exemple d'évolution du coefficient d'apprentissage selon le nombre de visites d'un état

Enfin, les différentes récompenses transmises au module d'apprentissage correspondent à celles données dans le chapitre 6. En pratique, le réglage de tous ces paramètres est réalisé empiriquement, essai après essai.

2.1.3. Le gestionnaire de tâche

Il s'initialise en chargeant la description de la tâche qui décrit les différents champs du formulaire de requête. Pour représenter la tâche en cours, nous avons conçu un système de cartographie d'objets. Celui-ci permet la construction du référentiel d'attributs à partir de la description de la tâche. Il est ensuite utilisé pour placer les objets dans ce référentiel et obtenir les descriptions concrètes des états $\textbf{\textit{(ea}}_1\textbf{\textit{, val}}_1\textbf{\textit{) ; … ; (ea}}_m\textbf{\textit{, val}}_m\textbf{\textit{)}}$, ainsi que les descriptions abstraites < $\textbf{\textit{ea}}_1$; … ; $\textbf{\textit{ea}}_m$ >.

2.1.4. Le gestionnaire de profils utilisateur

Ce module permet de stocker les préférences des utilisateurs en ce qui concerne les valeurs des attributs. Il est en effet possible d'indiquer quel est l'utilisateur courant et de maintenir à jour le nombre de fois qu'il a choisi une valeur donnée pour un attribut donné. En effectuant ce comptage, nous pouvons estimer une distribution de probabilité reflétant les préférences de l'utilisateur. Lorsque le module de décision sélectionne une action de proposition de valeur, celle-ci est alors obtenue par tirage aléatoire selon les probabilités pour l'attribut correspondant.

2.2. Description du fonctionnement

Le prototype est relativement générique, car il peut être paramétré en fonction du problème donné : cela inplique de lui transmettre le formulaire décrivant la tâche à accomplir, ainsi que les grammaires JGSF et les invités correspondants. Une fois la frame analysée, le gestionnaire de tâche construit le référentiel d'attributs, utilisant une bibliothèque spécialisée dans la cartographie d'objets, puis il initialise l'état de la tâche.

Le module de décision sélectionne l'action à accomplir. Lorsqu'une question doit être posée, les actions sont envoyées dans le corps de l'agent où elles sont transmises aux moteurs vocaux : le moteur de synthèse (*Pilote TTS*) reçoit les prompts à générer et respectivement, le moteur de reconnaissance (*Pilote Reco*) reçoit les règles de grammaire à activer.

Quand une action mettant en œuvre la source d'informations est sélectionnée, alors la requête est formée et envoyée à la ressource de gestion de données (*Pilote BDD*) qui la transmet à la source de données. Quand une réponse parlée est obtenue ou quand des résultats sont disponibles de la source, alors l'entrée est retournée vers le gestionnaire de tâche qui met à jour son état.

2.3. Le simulateur

De sorte à entraîner le module de décision du médiateur en mode non-interactif, un simulateur a été construit avec un comportement de réponse naïf aux questions. Afin de permettre un apprentissage de la stratégie de médiation, le simulateur a été mis en boucle directement avec la ressource cerveau, ce qui apparaît en pointillés sur la Figure 77. Le simulateur sélectionne un but et répond aux questions du médiateur. Pour vérifier si notre prototype était résistant aux aléas, nous avons introduit diverses perturbations aléatoires. Il arrive donc que dans 20% des cas, notre comportement simulé ne sache délibérément pas répondre aux questions sur un attribut donné et, de plus, à tout moment de la médiation, il y a 1% de chances que le but choisi soit altéré au niveau d'un attribut.

3. Expériences

Nous allons maintenant présenter les expériences réalisées sur le prototype de médiateur, accompagnés des résultats obtenus en simulation sur l'application de vol aérien. Les expériences que nous décrivons dans cette partie s'intéressent à l'influence du nombre attributs de la tâche sur l'apprentissage d'une politique de médiation. Nous avons fait ce choix, car comme nous l'avons indiqué Chapitre 6 - 4.2, c'est le nombre d'attributs m qui détermine la complexité de l'espace d'état abstrait et le nombre d'actions possibles. Etant donnée la croissance exponentielle de la complexité, nous avons travaillé sur des tâches de 3, 4 et 5 attributs. Dans le Tableau 23, nous donnons le nombre d'états abstraits et d'actions et de Q-Valeurs qui sont manipulés par le module de décision.

Tableau 23 - Nombre d'états abstraits, d'actions et de Q-Valeurs utilisés pour les expériences

Nombre d'attributs m	Nombre d'états abstraits 4×3^m	Nombre d'actions possibles $3 \times m + 2$	Nombre de Q-Valeurs
3	108	11	1188
4	324	14	4536
5	972	17	16 524

3.1. Paramètres utilisés

Au cours de nos expérimentations, décrites plus loin, nous avons utilisé les paramètres suivants :

- Température de Boltzmann ***T*** :

t_{Min}	= 0,01	Température minimale
t_{Max}	= 100	Température maximale
Δ_t	= 0	Décroissance linéaire de la température
tFactor	= 0,999	Décroissance géométrique de la température

  Nous avons choisi de faire varier la température dans un intervalle relativement restreint pour éviter les débordements dans la représentation flottante double, au niveau du calcul des exponentielles sur les Q-Valeurs. Nous avons, par ailleurs, obtenu les meilleurs résultats avec une faible décroissance géométrique et une décroissance linéaire nulle, ce qui évite aux Q-Valeurs de se figer trop vite sur de mauvaises valeurs.

- Le coefficient d'escompte du gain γ:

  Ce paramètre a été fixé empiriquement à une valeur de 0,9 car elle permet d'obtenir une stratégie prenant en compte les récompenses avec une vision à plus long terme.

- Le taux d'apprentissage α :

  Le décalage du taux d'apprentissage n'ayant pas apporté de différence significative sur les résultats, nous avons fixé la valeur de départ α_0 à 1 et le seuil du nombre de visites avant décroissance $\boldsymbol{Seuil}_{\alpha}$ à 0. Dans les expériences, la valeur du taux d'apprentissage d'un état est donc simplement l'inverse du nombre de visites de cet état.

- Le nombre maximum d'interactions :

  Agir sur la valeur du paramètre ***maxInteractions*** permet de laisser une certaine souplesse pour la recherche de la meilleure politique, tout en pénalisant les échanges interminables. Une valeur plus grande que nécessaire n'est pas gênante, d'autant plus que le Q-Learning optimise la longueur de la politique qu'il apprend. En revanche, le fait de choisir un maximum trop petit élimine toute chance de convergence. En pratique, la valeur de ***maxInteractions*** peut être fixée selon le nombre d'attributs. Cependant, pour permettre la comparaison des résultats obtenus au cours des différentes expériences, nous avons fixé ce paramètre à 50 interactions, car cette valeur permet d'atteindre plus souvent le but au début de l'apprentissage pour 5 attributs ou moins.

- Les récompenses :

  A nouveau, les dépassements sur les calculs des Q-Valeurs nous ont conduits à choisir des valeurs assez peu élevées pour les récompenses, comme le montre le Tableau 24. Nous avons choisi de pénaliser davantage les médiations trop longues ($\boldsymbol{R}_{troplong}$). Le coût de l'accès aux données est modélisé par une petite pénalité sur la manipulation de nombreux résultats ($\boldsymbol{R}_{surnombre}$). En pratique, il peut par exemple s'agir d'un coût financier sur la consultation de la source d'informations. Nous accordons la plus forte récompense pour la sélection d'une proposition par l'utilisateur ($\boldsymbol{R}_{selection}$). Une faible récompense ($\boldsymbol{R}_{absrep}$) est enfin accordée lorsque la requête totalement spécifiée, mais qu'il n'y a pas de réponses.

Tableau 24 - Les valeurs des récompenses utilisées dans les expériences

Récompense	*Variable correspondante*	*Valeur*
$\boldsymbol{R}_{troplong}$	*recTimeout*	-4
$\boldsymbol{R}_{absrep}$	*recSorryNoAnswer*	1
$\boldsymbol{R}_{selection}$	*recUserSelection*	5
$\boldsymbol{R}_{surnombre}$	*recHeavySourceResponse*	-1

3.2. Tâche à trois attributs

3.2.1. Description de la tâche

Nous donnons ci-dessous, la description de l'application de réservation de vol avec trois attributs : la classe de voyage, la ville de départ et la ville d'arrivée.

```
{form
  nom : "vol",
  champs : (
    {champ
      nom :      "classe",
      type :     ("affaire","normale","economique"),
      open :     "En quelle classe souhaitez-vous voyager ?",
      close :    "Voulez-vous voyager en classe #max ?",
      confirm :  "Etes vous sur de vouloir voyager en classe #aff ?"
    },

    {champ
      nom :     "depart",
      type :   ("Paris", "Berlin", "Londres", "Madrid", "Bruxelles","Rome","Luxembourg",
                "Geneve"),
      open :    "Quelle est votre ville de depart ?",
      close :   "Voulez-vous partir de #max ?",
      confirm : "Etes vous sur de vouloir partir de #aff ?"

    },

    {champ
      nom :     "arrivee",
      type :   ("Tokyo", "Chicago", "Moscou", "Sydney", "Pekin", "Istanbul", "Dakar"),
      open :    "Quelle est votre ville de destination ?",
      close :   "Voulez-vous aller a #max ?",
      confirm : "Etes vous sur de vouloir aller a #aff ?"

    }
  ),

  reponse : "Depart de #depart#, arrivee a #arrivee# et voyage en classe #classe#.",
  nombreReponseMax : 5
}
```

3.2.2. Critères d'évaluation

L'avantage du Q-Learning est qu'il permet une évaluation au cours de l'apprentissage. Nous avons utilisé une évaluation qui se déroule de la façon suivante :

- Le simulateur compte une itération (notée ***i***) à chaque fois que le module de décision est interrogé.
- Toutes les mille itérations, le simulateur inscrit :
  - ★ le nombre total de médiations n_i et
  - ★ le nombre de médiations réussies $nSucc_i$, terminées par une sélection ou par une absence de résultat pour une requête pleinement spécifiée.
- Il est ensuite possible de calculer, par milliers d'itérations :
  - ★ le pourcentage $pSucc_i$ de médiations réussies et
  - ★ le nombre moyen d'itérations par médiation, autrement dit, la longueur moyenne d'une médiation $lMoy_i$.

$$pSucc_i = \frac{nSucc_i}{n_i} \times 100 \qquad\qquad lMoy_i = \frac{1000}{n_i}$$

Ces deux quantités sont les critères que nous avons retenus pour l'évaluation des résultats. Ainsi, plus ***pSucc*** est grand et plus ***lMoy*** est petite, meilleure est la politique de médiation apprise.

3.2.3. Résultats expérimentaux

Les graphes présentés Figure 79 et Figure 80 ont été obtenus pour l'apprentissage d'une médiation à 3 attributs. La Figure 79 montre l'évolution du pourcentage de médiations réussies selon le nombre d'itérations (***pSucc***).

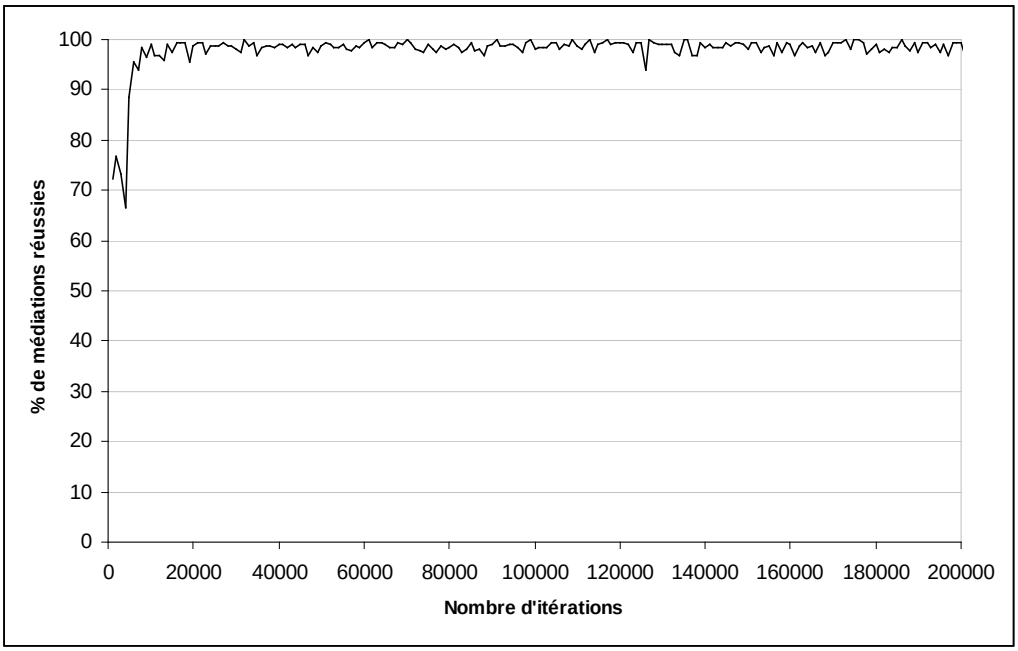


Figure 79 - Taux de réussite pour une médiation à 3 attributs

Nous constatons qu'un nombre important d'interactions est nécessaire pour obtenir une politique acceptable : choisir un attribut et le faire valuer. Après une partie d'exploration due à une grande température de Boltzmann, la politique amène vers 10 000 itérations aux 100 % espérés avec des variations normales dues à la diversité des requêtes et aux perturbations aléatoires. Nous pouvons noter que le pourcentage de départ est assez élevé, ce qui s'explique par une limite maximale de 50 interactions relativement laxiste vis-à-vis du nombre d'attributs utilisés.

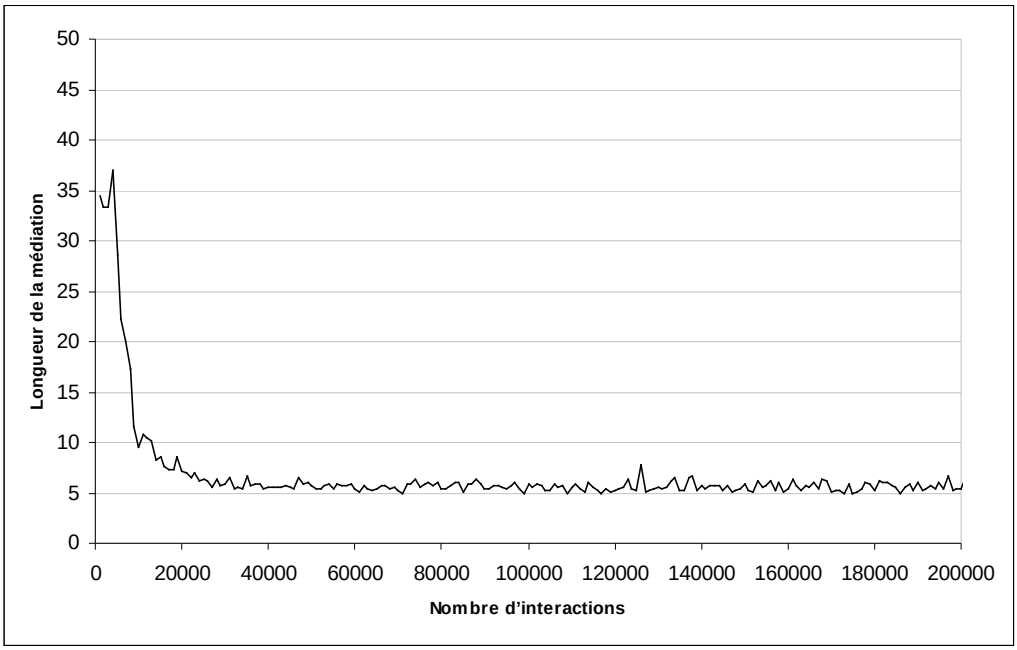


Figure 80 - Longueur moyenne de la médiation pour 3 attributs

Il faut toutefois considérer ces résultats relativement à la Figure 80 qui, quant à elle, montre l'évolution de la longueur moyenne d'une médiation selon le nombre d'itérations (***lMoy***). Nous observons également que la longueur moyenne des médiations décroît et arrive autour de 5 interactions, ce à quoi nous pouvions nous attendre. En effet, il faut poser une question pour valuer chaque attribut, puis envoyer la requête vers la source d'informations et enfin demander la sélection. Il faut cependant attendre 30 000 itérations pour atteindre cette politique optimale vis à vis du simulateur. Il y a donc une première phase plus exploratoire pendant laquelle le médiateur apprend une politique pour atteindre son but, puis une seconde dans laquelle il cherche à améliorer cette politique.

3.3. Tâche à quatre attributs

3.3.1. Description de la tâche

Nous donnons, ci-dessous, l'extension de la tâche précédente de façon à permettre le choix d'une période de départ.

```
{form
  nom : "vol4",

    [...]

    {champ
      nom :     "horaire",
      type :   ("le matin",  "l'apres-midi",  "le soir",  "de nuit"),
      open :    "Quelle periode de la journee voulez vous partir?",
      close :   "Voulez-vous partir #max ?",
      confirm : "Etes vous sur de vouloir partir #aff ?"
    }
  ),

  reponse : "Depart #horaire# de #depart#, arrivee a #arrivee# et voyage en classe #classe#.",
  nombreReponseMax : 5
}
```

La tâche utilise donc quatre attributs : la classe de voyage, la ville de départ, la ville d'arrivée et le choix d'une période départ dans la journée.

3.3.2. Résultats expérimentaux

Les graphes présentés Figure 81 et Figure 82 représentent l'évolution des même critères ***pSucc*** et ***lMoy*** pour la tâche à quatre attributs.

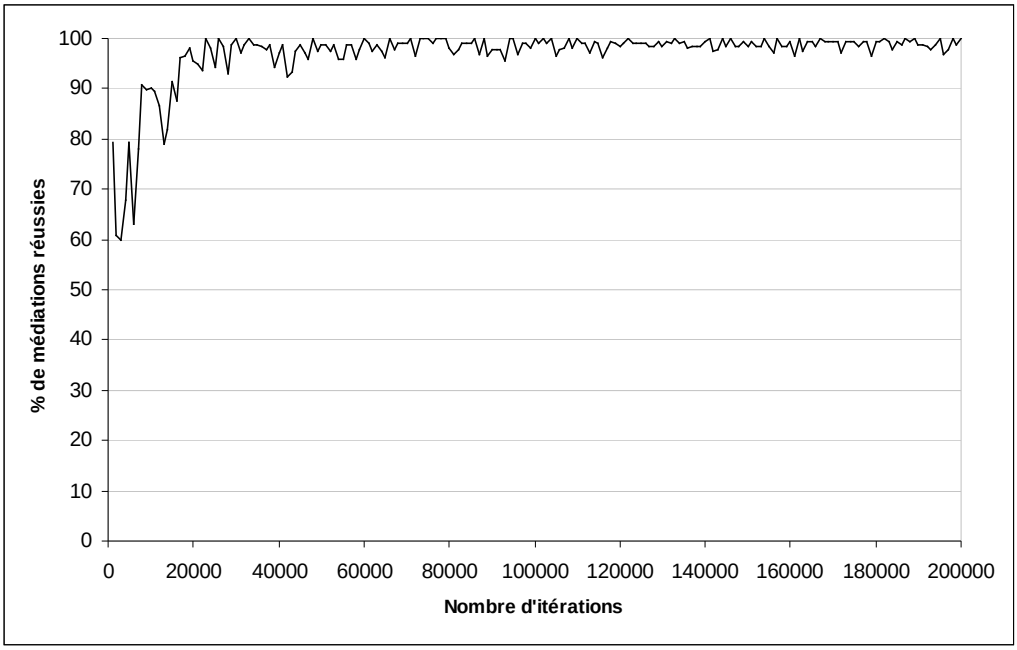


Figure 81 - Taux de réussite pour une médiation à 4 attributs

Nous pouvons observer que la limite des 100 % de succès est à nouveau atteinte, mais au bout d'un apprentissage plus long (20 000 itérations environs) et il faut attendre 200 000 itérations pour obtenir une politique qui soit optimale avec 6 interactions. Nous retrouvons donc les deux phases observées précédemment, mais sur une période d'apprentissage plus longue.

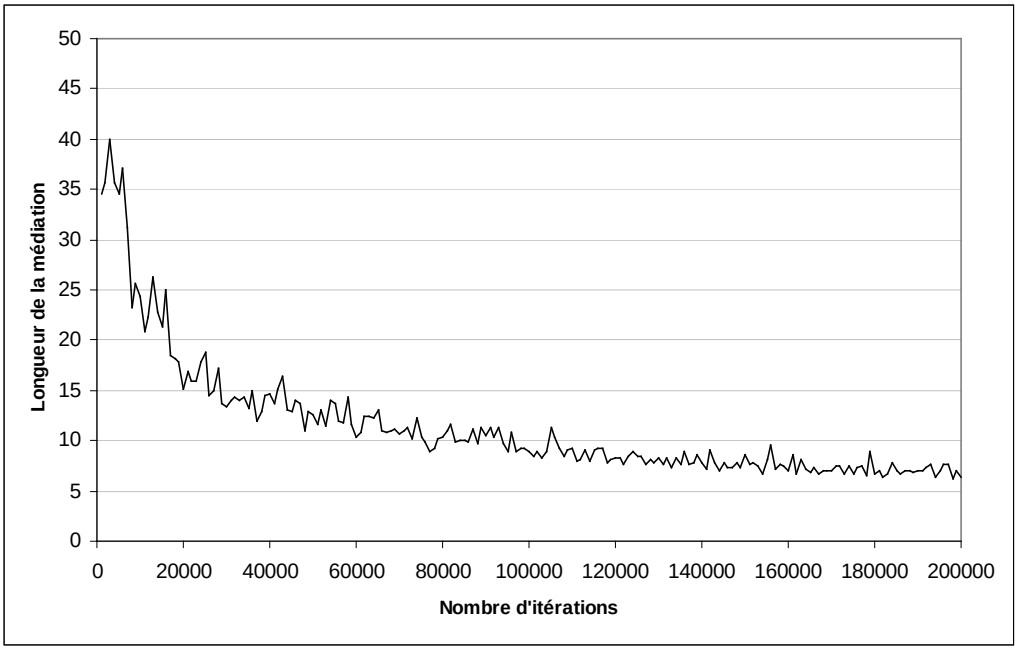


Figure 82 - Longueur moyenne de la médiation pour 4 attributs

3.4. Tâche à cinq attributs

3.4.1. Description de la tâche

Nous donnons, ci-dessous, une nouvelle extension de la description de l'application de réservation de vol utilisée dans les expériences précédentes avec le choix de la compagnie aérienne.

```
{form
  nom : "vol5",

    [...]

    {champ
      nom :     "compagnie",
      type :   ("American Airlines",  "Air France",  "British Airways",  "Delta Air Lines",
                "Bankair",  "Luxair",  "Lufthansa",  "United Airlines"),
      open :    "Avec quelle compagnie souhaitez-vous voyager ?",
      close :   "Voulez-vous voyager avec #max ?",
      confirm : "Etes vous sur de vouloir voyager avec #aff ?"
    },

  reponse : "Vol de la compagnie #compagnie#, depart #horaire# de #depart#, arrivee a
#arrivee# et voyage en classe #classe#.",
  nombreReponseMax : 5
}
```

Nous obtenons une tâche à cinq attributs permettant d'indiquer : la classe de voyage, la ville de départ, la ville d'arrivée, le choix d'une compagnie aérienne et d'une période de voyage dans la journée.

3.4.2. Résultats expérimentaux

Comme précédemment, les deux graphes Figure 83 et Figure 84, correspondent aux valeurs de ***pSucc*** et ***lMoy***, mais cette fois pour une médiation à 5 attributs.

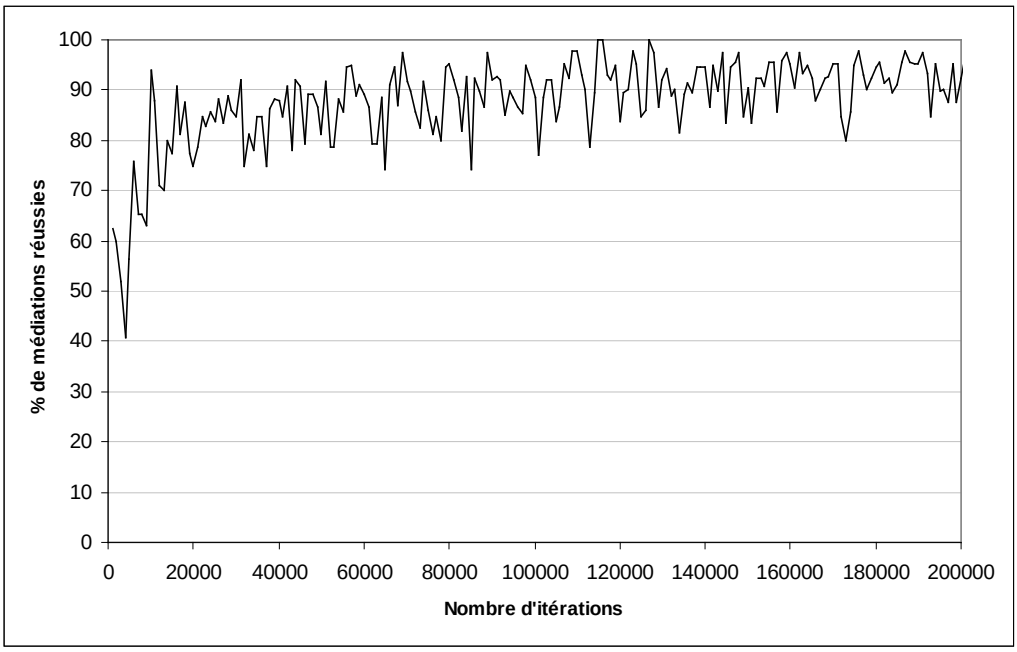


Figure 83 - Taux de réussite pour une médiation à 5 attributs

Nous constatons, sur le même intervalle, que la politique mène bien plus lentement à 100% de succès par milliers d'itérations. La plus forte amplitude des variations indique également une moins bonne résistance aux perturbations aléatoires.

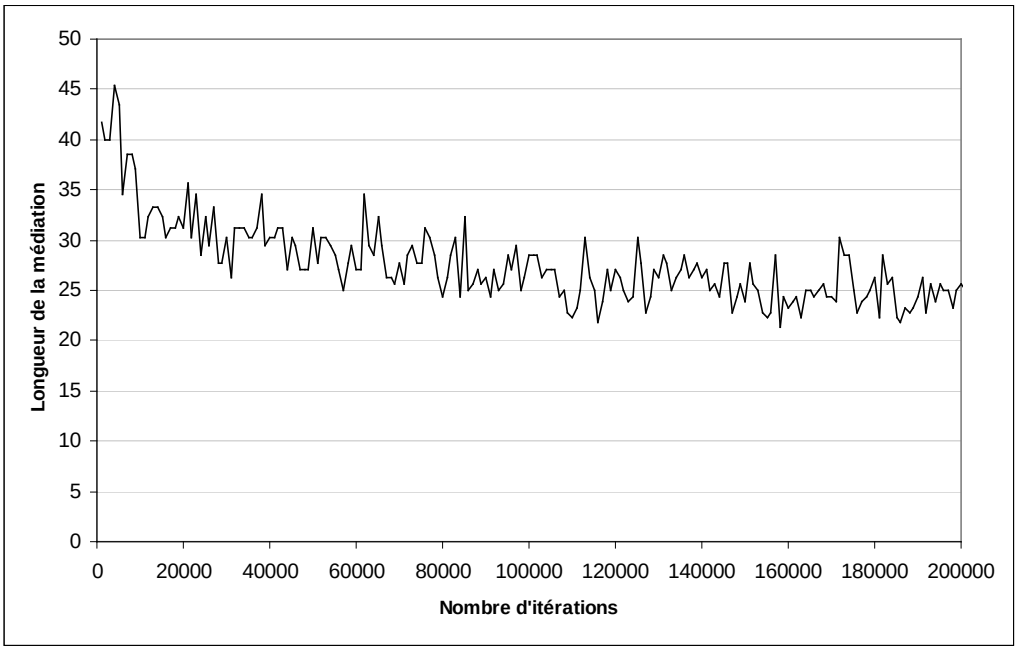


Figure 84 - Longueur moyenne de la médiation pour 5 attributs

La politique du médiateur est donc bien plus longue à obtenir en terme d'itérations et elle reste encore plus longtemps sous-optimale vis à vis du simulateur, car la longueur de la médiation ne décroît que jusqu'à 25 interactions pour 200 000 itérations, soit 50% de la limite puis continue à décroître très légèrement. Nous avons poursuivi l'expérience et la valeur de ***lMoy*** finit par atteindre 8 interactions au bout de 25 000 000 d'itérations. Par extrapolation, la limite basse des 7 interactions est obtenue vers 40 000 000 itérations.

3.5. Comparaison des résultats

Sur les graphes Figure 85 et Figure 86, nous avons superposé les différentes courbes du pourcentage de médiation réussies et la longueur moyenne de médiation pour les tâches à 3, 4 et 5 attributs.

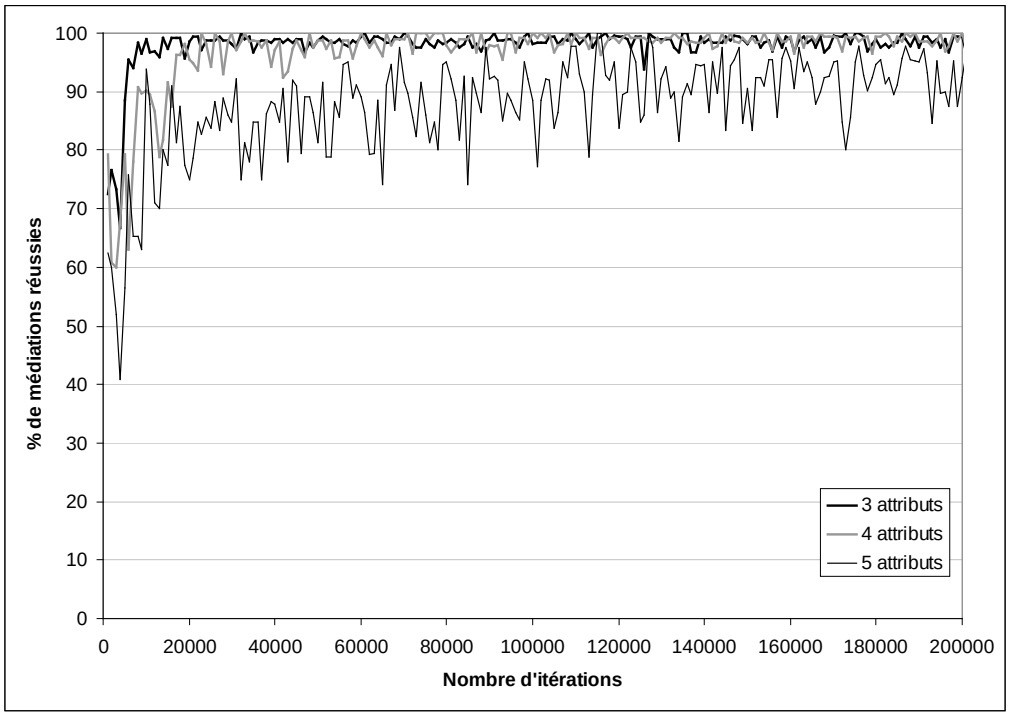


Figure 85 - Comparaison des taux de réussite des médiations

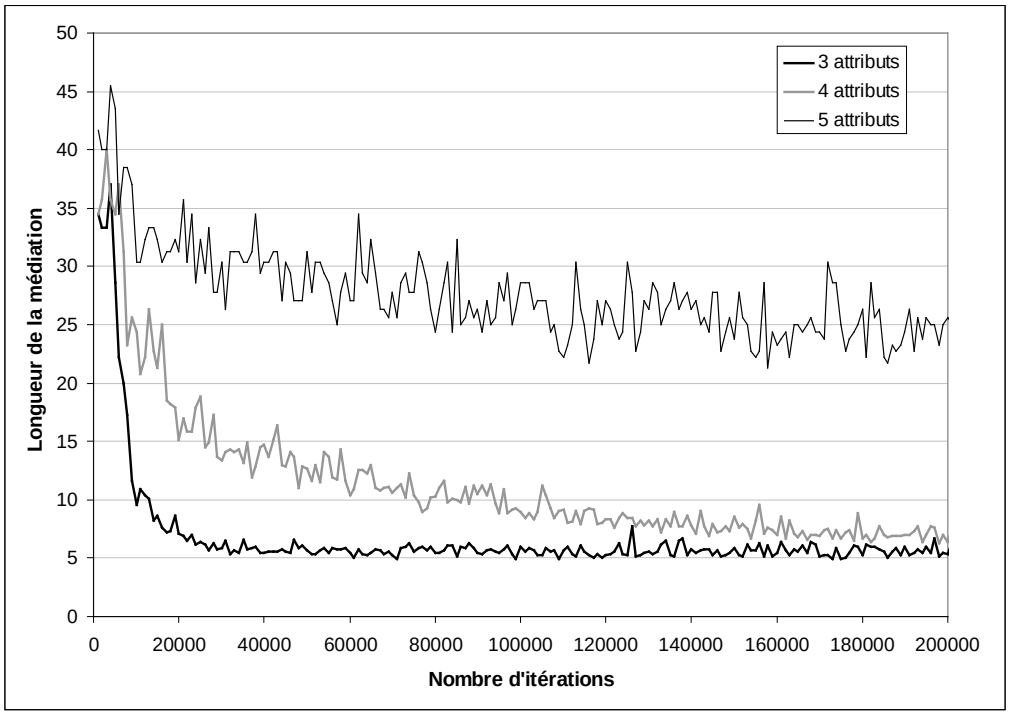


Figure 86 - Comparaison des longueurs moyennes des médiations

Nous constatons plus nettement que l'optimalité et la robustesse de la politique de médiation apprise décroissent avec le nombre d'attributs. Les courbes mettent également en évidence l'allongement du délai d'obtention de la politique de médiation.

4. Analyse des résultats

L'objectif des expériences était de montrer la possibilité d'apprendre une politique de médiation par renforcement entre un utilisateur simulé et une source d'informations. Les résultats des différentes expériences que nous avons réalisées nous ont amenés à plusieurs constats. Tout d'abord le nombre d'interactions nécessaires pour l'obtention d'une bonne politique rend l'apprentissage et l'utilisation du système irréaliste en dehors d'une période de simulation préalable. Ensuite, la limite basse du nombre d'interactions à laquelle nous arrivons est déterminée par le nombre d'attributs avec des résultats satisfaisants pour un petit nombre. Le fait est que la complexité croissante du problème de 3 à 5 attributs peut rendre cette limite très difficile à atteindre. Il est important de rappeler que non seulement le nombre d'états augmente, mais aussi le nombre d'actions possibles à partir de chaque état. Nous avons plusieurs idées pour résoudre ce problème de complexité. Une première serait d'exploiter les dépendances entre les attributs. Sur l'application de réservation de vol, la ville de départ restreint par exemple le nombre de compagnies possibles. Une seconde possibilité serait de chercher une autre représentation de l'espace d'état pour guider le médiateur qui soit plus compacte, avec une croissance non exponentielle et qui permette de définir des niveaux de décomposition hiérarchiques.

En pratique, utiliser un comportement appris avec cette simulation permet bien à un utilisateur d'interagir avec le système pour obtenir ce qu'il veut. Cependant, le problème auquel nous avons été confrontés est que le simulateur que nous avons construit est basé sur un comportement naïf par rapport à la diversité d'un comportement humain. L'interaction paraît alors assez rigide, car le système apprend une politique au plus près du comportement du simulateur et essaye de s'y ramener le plus possible. En revanche, nous pensons qu'un apprentissage avec des utilisateurs humains exploiterait davantage le potentiel du système. En ce sens, nous retrouvons les limites identifiées au chapitre 6 et dont une solution est le type de simulation proposé par *(Young 1999)*. De sorte à produire un comportement plus intéressant, le médiateur doit être entraîné avec un simulateur construit à partir de traces d'interactions réelles. Nous n'avons hélas pas eu la possibilité de faire l'acquisition d'un tel corpus d'interactions.

Les résultats expérimentaux montrent donc que le médiateur est capable d'apprendre à optimiser les interactions. Il cherche à se restreindre aux questions, aux requêtes et aux propositions nécessaires et trouve à quel moment il est opportun de les poser.

5. Conclusion

Dans ce chapitre, nous avons décrit notre implantation de corps d'agent contrôlé pour la communication multimédia. Nous avons ensuite présenté comment nous avons réalisé un agent médiateur, en illustrant son utilisation sur l'application de transport aérien. Nos expérimentations ont montré que les méthodes d'apprentissage par renforcement pouvaient être appliquées avec succès pour apprendre une politique de médiation. Nous avons cependant identifié un certain nombre de limites, notamment sur la complexité des tâches et sur le temps d'apprentissage nécessaire. Par ailleurs, notre système étant construit avec l'hypothèse que la perception reçue correspond à la phrase qui a été effectivement transmise, il ne permet pas la prise en compte de l'incertitude sur les perceptions. Pour résoudre ce problème la solution serait de passer à une approche partiellement observable comme celle de *(Roy et al. 2000)*. Comme nous l'indiquions dans la fin du chapitre 4, nous disposons à nouveau de plusieurs pistes pour la prise en compte de l'observabilité partielle dans nos travaux futurs. Nous envisageons donc diverses évolutions de notre prototype de sorte à permettre, entre autres, le traitement de problèmes plus complexes et la gestion de l'incertitude sur les perceptions.

Conclusion et Perspectives

Les travaux que nous avons réalisés au cours de cette thèse se placent dans le domaine de l'apprentissage de comportements adaptatifs pour favoriser la collaboration dans un système multi-agent hétérogène.

L'aspect industriel s'est axé autour de notre collaboration avec la société Dialoca et de sa plate-forme Unimédia. Nous nous sommes intéressés à faciliter la conception et la fourniture de services multimédia pour la recherche intelligente d'informations, la notification de messages ou encore pour la communication entre des utilisateurs et des experts. Nous avons notamment été plongés au cœur d'un projet de recherche d'informations intelligente avec un grand groupe de distribution de biens culturels. Nous avons pu présenter un prototype en comité de direction, mais ce projet n'a hélas pas pu continuer pour des questions économiques.

Pendant notre thèse, nous avons également participé au démarrage de plusieurs projets européens IST, ce qui nous a permis d'élargir le champ d'application de nos travaux. Tout d'abord, le projet Elin dont l'objectif est d'étudier ce que pourrait être le journal du futur, mêlant à la fois des informations hypertexte, la diffusion de clips vidéo, etc. Ensuite, le projet Ozone qui s'intéresse à définir une architecture d'intelligence ambiante, où des services multimédia sont fournis à des utilisateurs en passant d'une ressource à une autre sans interruption, tout en s'adaptant non seulement à l'utilisateur, mais aussi au contexte.

Toutes les études bibliographiques que nous avons effectuées, toutes les réflexions et les échanges que nous avons pu avoir, tant avec les membres de Maia, de Dialoca, lors des réunions de projets ou au cours des conférences auxquelles nous avons pu participer ont été très enrichissantes à la fois sur le plan théorique que sur le plan applicatif. Au cours de notre thèse, nous avons cependant été limités par la nécessité de travailler sur des données concrètes pour obtenir un système avec un minimum de réalisme. En effet, nous avons pu par exemple constater combien il était difficile, et pourtant nécessaire, d'avoir une indexation détaillée et fiable des produits commercialisés pour proposer un système d'aide à la vente. De la même façon, il est difficile de proposer un système adaptatif sans disposer de traces d'interactions. Nous avons donc été quelque peu déçus de n'avoir pu mener à bien certaines expériences sur des cas réels.

1. Notre démarche

1.1. Etudes préliminaires

Afin de comprendre comment se déroulaient les services Dialoca existants, nous avons été amenés à les étudier en détail. Nous avons pu décrire les services proposés par des automates déterministes à mémoire et à états finis. Chaque service produit un comportement qui utilise un ensemble de ressources pour gérer une médiation avec un nombre déterminé d'interlocuteurs. Il est ressorti de l'étude que si les services existants sont performants, c'est qu'ils requièrent une conception experte couvrant exhaustivement les cas possibles. Nous avons par ailleurs identifié un manque de dynamique : une fois le service conçu, prendre en compte un cas imprévu nécessite une nouvelle phase de conception. Ceci a motivé notre recherche des possibilités de rendre les services adaptatifs. Par cette voie, nous souhaitions aller plus loin que les services statiques et prendre en compte les spécificités des utilisateurs mais aussi, de façon plus générale, les variabilités de l'environnement tout en libérant le concepteur de l'exhaustivité des cas possibles. De cette démarche préliminaire, nous

avons conclut qu'il fallait trouver comment aller vers une collaboration entre les différents intervenants avec l'idée que le service est finalement le produit de cette collaboration.

1.2. Les services vus comme des schémas d'interaction

Un service est la réponse à un besoin qu'un intervenant ne peut pas satisfaire à lui seul. Le plus souvent, il s'agit d'un utilisateur qui va, en quelque sorte, déléguer la réalisation d'une tâche en s'adressant à un intervenant compétent, mais ce n'est cependant pas la seule origine possible des services. En effet, un service peut tout aussi bien provenir d'un fournisseur de données, qui souhaite diffuser ses informations et dans ce cas, le but est d'acheminer ces informations vers les consommateurs intéressés.

Les intervenants que nous avons considéré peuvent être de nature très différente. Le plus souvent, il s'agissait d'êtres humains et de systèmes logiciels, mais il est tout à fait concevable de prendre en compte d'autres intervenants, comme par exemple, des robots. Les idées que nous défendons sont d'une part que chacun des intervenants, pris indépendamment des autres, peut être vu comme un agent et d'autre part, qu'il est possible de voir un service comme si chacun des agents incarnait un rôle défini et adoptait un comportement correspondant : les uns devenant des clients ou des utilisateurs, les autres des experts d'un domaine ou encore des spécialistes capables par exemple de traduire l'information...

Ces considérations nous ont amenés à proposer la modélisation des intervenants par un ensemble d'agents hétérogènes et à avancer que la coopération de ces agents avait pour cadre un service qu'elle permettait de réaliser.

Nous avons défini une architecture conceptuelle, à quatre niveaux, avec une abstraction progressive : média, ressource, agent et enfin service. Nous avons également distingué les agents selon l'influence que nous pouvons avoir sur eux. Nous avons distingué les agents contrôlés pour désigner les agents logiciels dont nous ne pouvons pas influencer directement le comportement. Nous avons indiqué que ceux-ci ont à planifier leurs actions de sorte à remplir des buts et à atteindre un état de satisfaction dicté par leur rôle. A l'inverse, lorsque nous avons considéré, par exemple, des êtres humains jouant des rôles comme celui d'un utilisateur de service nous les avons définis comme des agents non-contrôlés.

En étudiant le niveau le plus haut, où se déroulent les services, nous avons cherché à extraire les caractéristiques communes entre les diverses applications considérées. Nous nous sommes aperçus que nous retrouvions plusieurs rôles principaux : un rôle d'utilisateur, un rôle de source d'informations et un rôle de traitement d'information. Nous avons proposé de modéliser les différents services sous forme de schémas de communication. Dans ces schémas, nous avons ajouté un rôle particulier, le médiateur, qui doit gérer une médiation intelligente entre les différents interlocuteurs, les comprendre, et faire passer les messages en demandant à l'un des données complémentaires et une traduction à l'autre... Nous nous sommes focalisés sur des interactions riches, mais complexes : le cas où un utilisateur veut mettre en oeuvre un service au travers d'un médiateur, celui-ci l'aidant à obtenir ce qu'il veut. Le lien entre un utilisateur humain et un médiateur nécessite un véritable dialogue et il en est de même pour la communication avec les autres agents logiciels ou matériels du système. Par contre, si les agents logiciels peuvent facilement échanger des informations en suivant des protocoles de communication bien formalisés comme ACL *(FIPA 2002)*, il n'en est pas de même lorsque des êtres humains interviennent. Le médiateur, qui peut fort bien être un agent logiciel d'un site commercial, doit pouvoir déterminer les besoins d'un agent humain client en le guidant et en allant lui présenter les produits vendus en détail, si nécessaire.

Afin de rendre les services robustes aux problèmes pouvant survenir au cours du déroulement, il est également important de pouvoir détecter les problèmes et de préparer des actions de réparation appropriées. Ce contrôle du service est nécessaire et doit interpréter le comportement des agents avec des fonctions de diagnostic. Si une situation est évaluée dangereuse, il faut alors trouver un moyen susceptible de ramener le système dans un état souhaité.

1.3. Des services à une communication adaptative

Inspirés par les systèmes permettant la gestion d'un dialogue entre un utilisateur et une application donnée, nous avons émis l'hypothèse qu'un service peut se traduire en une médiation entre les différents agents.

Au travers d'une étude des systèmes de dialogue, nous avons constaté des similitudes avec la gestion des services. En effet, les automates finis déterministes ont été utilisés pour gérer des dialogues mais les approches récentes ont laissé la place à l'utilisation de modèles stochastiques. Dans *(Young 1999)*, Young propose en effet d'utiliser ces modèles pour optimiser les stratégies de dialogue alors qu'ils avaient déjà montré de très bonnes performances sur des niveaux inférieurs de la communication. Nous nous sommes donc intéressés à des applications utilisant des processus de décision markoviens (MDP) comme *(Levin et al. 1998)* et des MDP partiellement observables (POMDP) comme *(Roy et al. 2000)*. La force de l'approche stochastique est que les solutions obtenues sont résistantes aux incertitudes liées aux résultats des actions (pour les MDP et les POMDP) mais aussi à celles liées aux entrées du système (pour les POMDP). L'expérience de notre équipe *(Maia 2002)* dans ce domaine a par ailleurs montré qu'il s'agit d'outils adéquats pour produire des comportements par planification dans des environnements incertains.

Les modèles stochastiques permettent de répondre à nos problèmes et nous avons par exemple réussi à utiliser un MDP afin de contrôler le déroulement des interactions entre un médiateur logiciel et un utilisateur humain et de découvrir le but de ce dernier au moyen d'une série de questions. Le médiateur apprend quand poser telle ou telle question, à quel moment il doit mettre en œuvre une source de données ou un traitement. Il réussit à apprendre son comportement et à s'adapter, car il prend en compte le coût de chaque action afin d'optimiser le coût total à plus ou moins long terme.

Sur l'architecture que nous avons définie, nous avons retrouvé différents points où il est possible d'avoir encore une meilleure adaptation des services. Nous avons notamment mis en évidence le besoin d'adaptation aux ressources qui sont utilisées pour communiquer, mais surtout le besoin d'adaptation aux autres agents, comme les utilisateurs.

1.4. S'adapter, c'est aussi apprendre des autres

L'adaptation est loin d'être un problème trivial, mais elle ouvre de nombreuses perspectives. Afin d'obtenir des services réellement adaptatifs, une étape nécessaire est de considérer le comportement des agents non contrôlés, ce qui pose divers problèmes. La difficulté vient du fait qu'une bonne partie des connaissances que nous ayons à leur sujet viennent de ce que nous pouvons observer d'eux alors que nous en avons une faible connaissance a priori. Tout se passe comme si nous regardions l'autre par le "trou d'une serrure" *(Albercht et al. 1998)* : nous n'en avons qu'une vue partielle. Il n'est pas possible de dire avec précision quel comportement un agent externe va adopter dans un contexte donné mais grâce à l'utilisation des modèles stochastiques, nous pouvons tenter d'en obtenir une approximation.

Ces réflexions nous ont conduit à nous intéresser à deux domaines connexes : celui de la modélisation des utilisateurs, que nous étendons aux agents non contrôlés et celui de la reconnaissance de leurs buts. Après une étude des travaux existants, notre choix s'est porté sur la modélisation stochastique, car elle est à même de gérer le caractère tout à fait incertain des observations réalisées. Dans le cadre de nos travaux sur les interactions utilisateur-médiateur, ces aspects se traduisent par la constitution de profils d'utilisateurs par le médiateur. Ces profils regroupent différentes données sur l'utilisateur, comme ses caractéristiques, son niveau d'expertise ou ses préférences... Ces données peuvent lui être demandées ou être apprises au cours des interactions. Avec de tels profils, il est par exemple possible de retenir les réponses les plus fréquentes afin d'optimiser la médiation. Les médiateurs sont également en mesure de leur fournir des services qui répondent davantage aux besoins identifiés, tout en traitant l'incertitude et l'incomplétude des requêtes. Ainsi, il est possible, pour une application commerciale de filtrage, de conseiller les clients en constituant des sélections personnalisées qui répondent aux besoins qui auront été identifiés comme étant les plus probables.

Construire un modèle des autres permet de favoriser la coopération : un agent contrôlé est alors capable de déterminer à qui il va s'adresser pour demander de réaliser une tâche donnée. Cela lui

permet notamment de prévoir les réactions de l'autre et d'agir en conséquence en effectuant, par exemple, les traductions appropriées. Nous considérons de plus que la façon dont un agent apprend sur un utilisateur peut se généraliser en un apprentissage sur les autres agents avec lesquels il interagit. C'est par ailleurs le principe du filtrage collaboratif qui est mis en oeuvre pour la fourniture d'informations pertinentes aux utilisateurs dans certaines applications auxquelles nous nous sommes intéressés.

2. Nos apports

Dans notre thèse, nous espérons tout d'abord avoir apporté au lecteur une vue d'ensemble de la problématique complexe à laquelle nous nous sommes attaqués. Ce que nous estimons être l'un des principaux apports est d'envisager les services comme une collaboration dans laquelle l'être humain est replacé au cœur du problème.

Cette collaboration, nous l'avons modélisé en l'articulant autour de la notion de rôle. Nous avons en particulier défini un ensemble de quatre rôles génériques : utilisateur, médiateur, source et traitement d'information. Ces rôles permettent à leur tour de construire des classes de services. Nous avons de plus montré que notre modélisation pouvait se généraliser et prendre en compte les autres applications que nous avons rencontrées, comme le filtrage dans les projets Elin et Ozone. Nous avons également pu valider notre approche en modélisant une application en télémédecine avec le projet Diatélic.

D'un point de vue applicatif, nous nous sommes plongés au cœur de la plate-forme Unimédia afin d'en comprendre les rouages et notamment comment elle unifiait l'accès aux ressources multimédia. Nous avons pu en dégager une architecture abstraite, dans laquelle les agents ont pleinement leur place pour interagir dans le cadre des services.

Dans cette thèse, nous avons exposé nos réflexions pour concevoir les agents contrôlés. Pour cela, nous avons cherché à montrer la palette des possibilités qui s'était offerte à nous, tant en ce qui concerne la planification de comportements, que pour l'adaptation aux autres agents et aux aléas dans les services. Nous n'avons pas proposé de solution unique pour construire un agent contrôlé, mais nous avons envisagé plusieurs façons de réaliser des services, en cherchant à dépasser l'aspect statique pour lequel tout doit être prévu, afin de faciliter le travail de conception et pour les rendre plus adaptatifs.

Dans le cas particulier du problème de la recherche d'informations, les résultats de nos expérimentations ont montré que même si notre modélisation par des MDP souffre d'une complexité importante, elle était utilisable conjointement avec des méthodes d'apprentissage par renforcement pour obtenir un comportement de médiation adaptative.

3. Perspectives

Qu'il s'agisse de travaux théoriques ou d'applications pratiques, de nombreuses perspectives sont possibles.

Un premier challenge est la réduction de la complexité liée à la représentation de la tâche et la quantité de données nécessaire à l'apprentissage. Nous pensons qu'une modélisation tenant compte des liens de dépendance qui peuvent exister entre les attributs pourrait permettre de réduire cette complexité. Par ailleurs, l'usage de modèles hiérarchiques, comme les HMDP pourrait permettre d'effectuer une décomposition des services par niveau et ainsi permettre de nous attaquer à des tâches plus complexes.

Dans nos travaux, nous avons principalement cherché à prendre en compte l'incertitude au niveau des résultats des actions, mais celle-ci n'est pas la seule à intervenir. En effet, l'état du système modélisé est rarement accessible de façon totale et certaine. C'est pourquoi, une seconde perspective serait d'explorer le potentiel des modèles stochastiques à observation partielle comme les POMDP, voire en combinant avec l'approche hiérarchique, nous rapprocher des travaux sur les HPOMDP *(Pineau et Thrun 2000)*.

De plus, nous pensons qu'il pourrait être intéressant de se pencher sur les méthodes qui permettent de généraliser les connaissances acquises lors de l'apprentissage par renforcement des comportements,

comme le font les systèmes à classeurs *(Gérard et Sigaud 2001)*. Une possibilité serait donc d'abstraire le modèle qui est constitué au cours de l'apprentissage et d'étudier comment repartir de politiques déjà connues.

Les perspectives pour la fourniture de services adaptatifs avec une approche multi-agent sont multiples et nous pensons qu'il serait intéressant de distribuer davantage les fonctionnalités à l'instar des plates-formes agents existantes, comme Jade (The Java Agent DEvelopment framework), ou OAA *(Martin et al. 1999)* qui nous paraissent bien plus souples pour mettre en œuvre une composition des services par interactions multi-agent. A cet effet, nous pensons que le déploiement d'agents contrôlés, tant pour assister les utilisateurs que les experts, et leur communication favorisée par des médiateurs, pourraient permettre à chacun de publier et d'utiliser les services qu'il souhaite, soit de façon atomique, soit par composition.

Références Bibliographiques

(AIPS 1998) AIPS-98 Planning Competition Committee, The planning Domain Definition Language Version 1.2 Manual, October 1998. (Ch. 4)

(Albrecht et al. 1998) Albrecht D. W., Zukerman I. et Nicholson A., Bayesian Models for Keyhole Plan Recognition in an Adventure Game. In Proceedings of User Modeling and User-Adapted Interaction (UMUAI'98), Vol. 8(1-2), pp. 5-47, 1998. (Ch. 5, C)

(Allen et al. 2000) Allen J., Byron D., Dzikovska M., Ferguson G, Galescu L. et Stent A., An Architecture for a Generic Dialogue Shell. In Natural Language Engineering, Cambridge University Press, Vol. 6, 2000. (Ch. 6)

(Aristote -328) Aristote, L'éthique à Nicomaque, Livre III.3, 1112b, 328 avant JC. (Ch. 4)

(Assadourian et Maillet 2000) Assadourian F. et Maillet Y, UNIMEDIA : Application générique, Langage de scénarisation, Guide de l'utilisateur, v1.0, MIC2, 2000. (Ch. 2)

(Auberger 1998) Auberger M., Student modelling in educational titles using agents, PhD Thesis of the Wirtschaftsuniversität Wien, 1998. (Ch. 5)

(Barbuceanu et Fox 1995) Barbuceanu M. et Fox M. S., COOL: A language for describing coordination in multiagent systems. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95), 1995. (Ch. 1)

(Basseville et Cordier 1996) Basseville M. et Cordier M. O., Surveillance et diagnostic de systèmes dynamiques : approches complémentaires du traitement de signal et de l'intelligence artificielle. Publication interne de l'IRISA n°1004, 40 p, Mars 1996. (Ch. 5)

(Bauer 1998) Bauer M., Towards the automatic acquisition of plan librairies, In Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98), pp. 484-488, 1998. (Ch. 5)

(Bellman 1957) Bellman R., Dynamic Programming, Princeton University Press, p 83, 1957. (Ch. 4)

(Bellot 2002) Bellot D., Fusion de données avec des réseaux bayésiens pour la modélisation des systèmes dynamiques et son application en télémédecine. Thèse de Doctorat de l'Université Henri Poincaré, Nancy, 2002. (Ch. 3, 5)

(Bennacef et al. 1996) Bennacef S., Devillers L., Rosset S., et Lamel L.,Dialog in the RAILTEL telephone-based system. In Proceedings of International Conference on Spoken Language Processing, (ICSLP'96), Vol. 1, pp. 550-553, Philadelphie, USA, 1996. (Ch. 6)

(Blum et Furst 1997) Blum A. et Frust M. L., Fast Planning Through Planning Graph Analysis. In Artificial Intelligence, Vol. 90(1-2), pp. 281-300, February 1997. (Ch. 4)

(Blum et Langford 1999) Blum A. L. et Langford J. C., Probabilisitc Planning in the Graphplan Framework. In Proceedings of the 5th European Conference on Planning (ECP'99), pp. 319-332, Durham, United Kingdom, 1999. (Ch. 4)

(Boissier 1999) Boissier O., Cours sur les Systèmes Multi-Agent de DEA-CCSA - Comunication et Coopération dans les Systèmes à Agents de l'Ecole Nationale Supérieure des Mines de Saint-Etienne, 1999. (Ch. 1)

(Bond et Gaesser 1998) Bond A.H. et Gasser L., Readings in Distributed Artificial Intelligence. Morgan Kaufmann Publishers, San Mateo, CA, 1988. (Ch. 1)

(Bonet et Geffner 1998) Bonet B. et Geffner H., HSP: Heuristic search planner. Entry at the 4th International Conference on Artificial Intelligence Planning Systems (AIPS'98) Planning Competition, Pittsburgh, 1998. (Ch. 4)

(Bridge 2000) Bridge D., Artificial Intelligence Teaching Materials, University College Cork, Computer Science Department, 2000. (http://www.cs.ucc.ie/~dgb/courses/ai/lectures.html) (Ch. 4)

(Buffet 2000) Buffet O., Mémoire de DEA Informatique de l'Université Henri Poincaré, Nancy, 2000. (Ch. 4)

(Carley et Gasser 1999) Carley K. M. et Gasser L., Computational Organization Theory. Dans Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Weiss G., MIT Press, Cambridge, MA, pp. 299-330, 1999. (Ch. 1)

(Chadès et al. 2002) Chadès I., Scherrer B. et Charpillet F., An heuristic Approach for solving Decentralized-POMDP: Assessment on the Pursuit Problem. In the 17th ACM Symposium on Applied Computing (SAC'2002), Madrid, 2002. (Ch. 5)

(Chalupsky et al. 1992) Chalupsky H., Finin T., Fritzson R., McKay D., Shapiro S. et Wiederhold G., An overview of KQML: A knowledge query and manipulation language. Technical Report, KQML Advisory Group, April 1992. (Ch. 1, C)

(Charpillet et al. 1998) Charpillet F., Chadès I. et Gallone J.M., Stochastic and Distributed Anytime Task Scheduling. In Proceedings of the 10th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'98), 1998. (Ch. 4)

(Charpillet et Boyer 1997) Charpillet F. et Boyer A., Progress : an Approach for Defining and Monitoring Non Deterministic Design-to-Time Methods. In Proceedings of the 9th International Conference on Tools with Artificial Intelligence (ICTAI'97), pp. 502-507, Newport Beach, CA, 1997. (Ch. 4)

(Chevrier 2002) Chevrier V., Contributions au domaine des systèmes multi-agents. Mémoire d'habilitation à diriger des recherches. Université Henri Poincaré, Nancy, 2002. (Ch. 1)

(Cohen et al. 1989) Cohen P. R., Greenberg M. L., Hart D. M. et Howe A. E., Trial by fire: Understanding the design requirements for agents in complex environments. In AI Magazine, Vol. 10(3), pp. 34-48, 1989. (Ch. 4)

(Côté et al. 2001) Côté M., Chaib-draa B. et Troudi N., NetSA : une architecture multiagent réutilisable pour les environnements riches en informations. Dans Information, Interaction, Intelligence, Cépaduès, Toulouse, pp. 39-78, 2001. (Ch. 3)

(Crosbie et Spafford 1996) Crosbie M et Spafford E. H., Evolving Event-Driven Programs. In Proceedings of the 1st Annual Conference on Genetic Programming, pp 273-278, 1996. (Ch. 5)

(Currie et Tate 1991) Currie K. W. et Tate A., O-Plan: the Open Planning Architecture. In Artificial Intelligence, Vol. 52(1), pp. 49-86, 1991. (Ch. 4)

(Dasgupta et González 2002) Dasgupta D. et González F., An Immunity-Based Technique to Characterize Intrusions in Computer Networks. In IEEE Transactions on Evolutionary Computations, Vol. 6(3), pp. 1081-1088, June 2002. (Ch. 5)

(Demazeau 1995) Demazeau Y., From interactions to collective behaviour in agent-based systems. In Proceedings of the 1st European Conference on Cognitive Science, St. Malo, 1995. (Ch. 1)

(Demazeau et Briot 2001) Demazeau Y. et Briot J-P., Principes et Architecture des Systèmes Multi-Agents. Traité IC2, Hermès, Paris, France, Novembre, 2001. (Ch. 1)

(Deneubourg 1991) Deneubourg J. L., Gross S., Beckers R. et Sandini G., Collectively self-solving problems, in Self organization, emergent properties and learning. A. Babloyantz editors, Plenum, 1991. (Ch. 1)

(Dey et Abowd 2000) Dey A. K. et Abowd G. D., Towards a Better Understanding of Context and Context-Awareness. In the CHI 2000 Workshop on The What, Who, Where, When, and How of Context-Awareness, La Haye, Pays Bas, 2000. (Ch. 5)

(Dialoca 2002) Dialoca, Des applications stratégiques, 2002. (http://www.dialoca.com) (Ch. 2)

(Dietterich 2000) Dietterich T. G., Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. In Journal of Artificial Intelligence Research, Vol. 13, pp 227-303, 2000. (Ch. 4)

(Dorigo et al. 1999) Dorigo M., Di Caro G. et Gambardella L. M., Ant Algorithms for Discrete Optimization. In Artificial Life, MIT Press, Vol. 5(2), pp. 137-172, 1999. (Ch. 1)

(Draper et al. 1994) Draper D., Hanks S et Weld D., Probabilistic planning with information gathering and contingent execution. In Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems (AIPS'94), pp. 31-37, 1994. (Ch. 4)

(Drogoul et Dubreuil 1993) Drogoul A. et Dubreuil C., A Distributed Approach to N-Puzzle Solving. In Proceedings of the 12th International Workshop on Distributed Artificial Intelligence, pp. 95-108, Seattle, 1993. (Ch. 1)

(Drummond 1989) Drummond M., Situated Control Rules. In Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasonning, Toronto, pp. 103-113, 1989. (Ch. 4)

(Durfee 1999) Durfee E. H., Distributed Problem Solving and Planning. Dans Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Weiss G., MIT Press, Cambridge, MA, pp. 121-164, 1999. (Ch. 4)

(Durfee et al. 1989) Durfee E. H., Lesser V. R. et Corkill D. D., Trends in Cooperative Distributed Problem Solving, In IEEE Transactions on Knowledge and Data Engineering, Vol. 1(1), pp. 63-83, 1989. (Ch. 1)

(Durfee et al. 1998) Durfee E. H., Kiskis D. L. et Birmingham W. P., The agent architecture of the University of Michigan Digital Library. Dans Readings in Agents. Huhns M. N. et Singh M. P. (Editeurs), Morgan Kaufmann, pp 98-108, 1998. (Ch. 3)

(Durfee et Lesser 1991) Durfee E. H. et Lesser V. R., Partial global planning: A coordination framework for distributed hypothesis formation. In IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21(5), pp. 1167-1183, 1991. (Ch. 4)

(Dury et al. 1999) Dury A., Le Ber F. et Chevrier V., A reactive approach for solving constraint satisfaction : Assigning land use to farming territories. In Proceedings of the 5th International Workshop on Agent Theories, Architectures, and Languages (ATAL'98), 1999. (Ch. 1)

(Dutech 1999) Dutech A., Apprentisage d'environnement : Approches cognitives et comportementales. Thèse de Doctorat de l'Ecole Nationale de l'Aéronautique et de l'Espace, 1999. (Ch. 4)

(Edelkamp et Helmert 2000) Edelkamp S. et Helmert M., On the Implementation of MIPS. In Workshop on Model-Theoretic Approaches to Planning at the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS'2000), Breckenridge, Colorado, 2000. (Ch. 4)

(El Fallah-Seghrouchini et al. 1999) El Fallah-Seghrouchini A., Haddad S. et Mazouzi H., A Formal study of interaction in Multi-Agent Systems, In Proceedings of the 14th International Conference on Computers and their Applications, (CATA'99), Cancun, Mexique, 1999. (Ch. 1)

(Errico 1997) Errico B., Intelligent Agents and User Modelling. PhD thesis, Dipartimento Informatica e Sistemistica, Università di Roma "La Sapienza", July 1997. (Ch. 5)

(Ferber 1995) Ferber J., Les Systèmes Multi-Agents. Vers une intelligence collective. Interéditions, 1995 (Ch. 1)

(Fikes et Nilsson 1971) Fikes R. E. et Nilsson N. J., STRIPS: A new approach to the application of theorem proving to problem solving. In Artificial Intelligence, Vol. 2, pp. 189-208, 1971. (Ch. 4)

(FIPA 2002) FIPA, FIPA ACL Message Structure Specification ID 00061, Fipa 2000, 2002. (Ch. 1, C)

(Foisel 1998) Foisel R., Modèle de réorganisation de systèmes multi-agents: Une approche descriptive et opérationnelle, Thèse de Doctorat de l'Université Henri Poincaré, November, 1998. (Ch. 1)

(Fourman 2000) Fourman M. P., Propositional Planning. In Workshop on Model Theoretic Approaches to Planning, AIPS 2000, Breckenridge Colorado, 2000. (Ch. 4)

(Fox et Long 2001) Fox M. et Long L., PDDL 2.1: An extension to PDDL for expressing temporal planning domains. Technical Report, Department of Computer Science, University of Durham, UK, 2001. (Ch. 4)

(Franklin et Graesser 1996) Franklin S. et Graesser A., Is It an Agent, or just a Program ?: A Taxonomy for Autonomous Agents. In Proceedings of Agent Theories, Architectures and Languages (ATAL'96) Workshop, Springer Verlag, Vol. 1193, 1996. (Ch. 1,3)

(Friedman-Hill 2003) Friedman-Hill E. J., Jess, The Expert System Shell for the Java Platform Version 6.1b3, Distributed Computing Systems Sandia National Laboratories, 2003. (http://herzberg.ca.sandia.gov/jess) (Ch. 8)

(Gelernter et Carriero 1992) Gelernter D. et Carriero N., Coordination Languages and Their Significance. Communications of the ACM, n°35(2), pp. 96-107, 1992. (Ch. 1)

(Georgeff et Lansky 1987) Georgeff, M. P. et Lansky, A. L., Reactive reasoning and planning. In Proceedings of the 6th National Conference of the American Association for Artificial Intelligence (AAAI'87), pp. 677-682, Seattle, WA. Mogran Kaufmann, 1987 (Ch. 4)

(Gérard et Sigaud 2001) Gérard P. et Sigaud O., Généralisation et Apprentissage Latent dans les Systèmes de Classeurs Extraction des connaissances et apprentissage : Apprentissage automatique et évolution artificielle, Vol. 3(1), pp. 87-114. Hermès, 2001. (Ch. C)

(Gini 1997) Gini M., Intelligent agents: Accomplishments and new challenges, Course Introduction at the University of Minnesota, January 1997. (Ch. 1)

(Goddeau et al. 1996) Goddeau D., Meng H., Polifroni J., Seneff S. et Busayapongchaiy S., A Form-Based Dialogue Manager For Spoken Language Applications. In Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP'96), Vol. 2, pp. 701-704, Philadelphia, PA, USA, 1996. (Ch. 6)

(Gonzales et Wuillemin 1998) Gonzales C. et Wuillemin P. H., Réseaux bayésiens en modélisation d'utilisateurs. Sciences et Techniques éducatives, Vol. 5(2), p. 173, 1998. (Ch. 5)

(Gorin et al. 1997) Gorin A. L., Riccardi G. et Wright J.H., How may I help you ?, In Speech Communication, Vol. 23, pp. 113-127, 1997. (Ch. 6)

(Grislin-Le Sturgeon et Péninou 1998) Grislin-Le Sturgeon E. et Péninou A., Les interactions Homme-SMA : réflexions et problématiques de conception. Systèmes Multi-Agents de l'interaction à la Socialité. Dans Journées Francophones de l'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA'98), Hermès, pp. 133-145, 1998. (Ch. 1)

(Gutknecht 2001) Gutknecht O., Proposition d'un modèle organisationnel générique de systèmes multi-agents Examen de ses conséquences formelles, implémentatoires et méthodologiques. Thèse de Doctorat de l'Université Montpellier II, 2001. (Ch. 1,3)

(Gutknecht et Ferber 1998) Gutknecht O. et Ferber J., Un méta-modèle organisationnel pour l'analyse, la conception et l'exécution de systèmes multi-agents. Dans Journées Francophones de l'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA'98), pp. 267, 1998. (Ch. 1,3)

(Heeman 2002) Heeman P., Spoken Language Systems - CSE550 - Cours du Department of Computer Science and Engineering de l'OGI School of Science & Engineering à l'Oregon Health and Science University, 2002. (http://www.cse.ogi.edu/~heeman/cse550/) (Ch. 6)

(Hoffmann et Nebel 2001) Hoffmann J. et Nebel B., The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence, Vol. 14, pp. 253-302, 2001. (Ch. 4)

(Horvitz et al. 1998) Horvitz E., Breese J., Heckerman D., Hovel D. et Rommelse K., The Lumière Project: Bayesian user modelling for inferring the goals and needs of software users. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98), pp. 256-265, Madison, WI, July 1998. (Ch. 5)

(Howard 1960) Howard R. A., Dynamic Programming and Markov Processes. MIT Press, Cambridge, Massachussets, 1960. (Ch. 4)

(Howard et Matheson 1981) Howard R. et Matheson J., Influence diagrams. In Howard, R. and Matheson, J., editors, Readings on the Principles and Applications of Decision Analysis, Vol. 2, pp. 721-762. Strategic Decisions Group, Menlo Park, CA, 1981. (Ch. 5)

(Huber et al. 1994) Huber M. J., Durfee E. H. et Wellman M. P., The Automated Mapping of Plans for Plan Recognition. In Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI'94), Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, pp. 344-351, 1994. (Ch. 5)

(IMC 1996) Internet Mail Consortium, vCard, The Electronic Business CardVersion 2.1. A versit Consortium Specification, September 18, 1996. (http://www.imc.org/pdi/vcardwhite.html) (Ch. 5)

(ISO 1994) Technologies de l'information, Interconnexion de systèmes ouverts (OSI), Modèle de référence de base: Le modèle de base - ISO/IEC 7498-1:1994. (Ch. 2)

(Jaakkola et al. 1994) Jaakkola T., Jordan M. I. et Singh S.P., On the convergence of stochastic iterative dynamic programming algorithms. In Neural Computation, Vol. 6(6), 1994. (Ch. 4)

(Jameson 1999) Jameson A., User Adaptive Systems An integrated Overview. Tutorial presented at the 7th International Conference on User Modeling, June 20-24, 1999. (Ch. 5)

(Jeanpierre 2002) Jeanpierre L., Apprentissage et adaptation pour la modélisation stochastique de systèmes dynamiques réels. Thèse de Doctorat de l'Université Henri Poincaré, Nancy, 2002. (Ch. 3, 5)

(Jensen 1994) Jensen K., An Introduction to the Theoretical Aspects of Coloured Petri Nets. In: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.): A Decade of Concurrency, Lecture Notes in Computer Science, Springer-Verlag, Vol. 803, pp. 230-272, 1994. (Ch. 5)

(JORF 1978) Journal Officiel de la République Française, Loi n° 78-17 relative à l'informatique, aux fichiers et aux libertés, 6 Janvier 1978. (Ch. 5)

(JSAPI 1998) Java Speech API Programmer's Guide, Version 1.0, Sun Microsystems, 1998. (http://java.sun.com/products/java-media/speech) (Ch. 8)

(JSGF 1998) Java Speech Grammar Format Specification, Version 1.0, Sun Microsystems, 1998. (http://java.sun.com/products/java-media/speech) (Ch. 8)

(Kaelbling et al. 1996) Kaelbling L. P., Littman M. L. et Moore A. P., Reinforcement Learning : A survey. In Journal of Artificial Intelligence Research, Vol. 4, pp. 237-285, 1996. (Ch. 4)

(Kanal et al. 1994) Kanal L., Kumar V., Kitano H., et Suttner C., Parallel Processing for Artificial Intelligence. Elsevier Science Publishers, Amsterdam, 1994. (Ch. 1)

(Kautz 1987) Kautz H., A Formal Theory of Plan Recognition, PhD Thesis, Department of Computer Science, University of Rochester, 1987. (Ch. 5)

(Kautz et Selman 1998) Kautz H. et Selman B., BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving. In Working notes of the Workshop on Planning as Combinatorial Search (AIPS'98), Pittsburgh, PA, 1998. (Ch. 4)

(Kobsa 1990) Kobsa A., User Modeling in Dialogue Systems: Potentials and Hazards. In AI & Society, Vol. 4(3), pp. 214-240, 1990. (Ch. 5)

(Kohler et al. 1997) Koehler J, Nebel B., Hoffmann J. et Dimopoulos Y., Extending Planning Graphs to an ADL Subset. In Proceedings of the 4th European Conference on Planning (ECP'97), 1997. (Ch. 4)

(Kuokka et Harada 1998) Kuokka D. et Harada L., Matchmaking for Information Agents. Dans Readings in Agents. Huhns M. N. et Singh M. P. (Editeurs), Morgan Kaufmann, pp 91-97, 1998. (Ch. 3)

(Laasri et al. 1988) Laasri H., Maitre B., Mondot T., Charpillet F. et Haton J. P., ATOME: A Blackboard Architecture with Temporal and Hypothetical Reasoning. In Proceedings of the 8th European Conference on Artificial Inelligence (ECAI'88), pp. 5-10, Munich, Germany, 1988. (Ch. 4)

(Labidi et Lejouad 1993) Labidi S. et Lejouad W., De l'intelligence Artificielle Distribuée aux Systèmes Multi-Agents, Rapport de Recherche, Programme n°2, Calculs Symbolique programmation et Génie Logiciel, INRIA N°2004, 39 p, 1993. (Ch. 1)

(Lagoudakis 1996) Lagoudakis M. G., Planning and Intelligent Systems: An Introductory Overview, Technical Report of the Center for Advanced Computer Studies, University of Louisiana, Lafayette, CACS TR-96-2-1, December 1996. (Ch. 4)

(Larousse 1998) Le petit Larousse 1998, Larousse-Bordas, 1998. (Ch. 1,2)

(Lederman 2000) Lederman M., La suite logicielle Unimédia, Présentation générale de l'offre MIC2, v1.3, MIC2, 2000. (Ch. 2)

(Lesh et al. 1999) Lesh N., Rich C. et Sidner C., Using Plan Recognition in Human-Computer Interaction. In Proceedings of the 7th International Conference on User Modeling (UM'99), pp. 23-32, 1999. (Ch. 5)

(Lesser et Corkill 1984) Lesser R. V. et Corkill D. D., The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. In AI Magazine Vol. 4(3), pp. 15-33, 1984. (Ch. 4)

(Levin et al. 1998) Levin E., Pieraccini R. et Eckert W., Using Markov Decision Process for Learning Dialogue Strategies. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'98), Seattle, USA, 1998. (Ch. 6, C)

(Levin et al. 1999) Levin E., Pieraccini R., Eckert W., Di Fabbirizio P. et Narayanan S., Spoken Language Dialogue: From theory to practice. In International Workshop on Automatic Speech Recognition and Understanding (ASRU'99), Keystone, Colorado, USA, 1999. (Ch. 6)

(Levin et al. 2000) Levin E., Pieraccini R. et Eckert W., A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. In IEEE Transactions on Speech and Audio Processing, Vol. 8(1), pp. 11-23, Janvier 2000. (Ch. 6)

(Levin et Pieraccini 1997) Levin E. et Pieraccini R., A stochastic model of computer-human interaction for learning dialogue strategies. In proceedings of the 5th European Conference on Speech Communication And Technology (Eurospeech'97), Vol. 4, pp. 1883-1886, Rhodes, Grèce, 1997. (Ch. 6)

(Long et Fox 1998) Long D. et Fox M., Efficient Implementation of the Plan Graph in STAN. In Journal of Artificial Intelligence Research (JAIR), Vol. 10, pp. 87-115, 1998. (Ch. 4)

(M. Do et S. Kambhampati 2001) Do M. B. et Kambhampati S., Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. In Artificial Intelligence, Vol. 132(2), pp. 151-182, 2001. (Ch. 4)

(Maes 1994) Maes P., Agents that Reduce Work and Information Overload. In Communication of the ACM, Vol. 37(7), pp. 31-40, 1994. (Ch. 1)

(Maia 2002) Rapport d'activité du projet MAIA, Institut National de Recherche en Informatique et en Automatique, 2002. (Ch. 1)

(Maillet 2000) Maillet Y., Noti-Mail, Notification d'E@Mail MIC2, v1.1, 2000. (Ch. 7)

(Malone et Crowston 1994) Malone T. et Crowston K., The Interdisciplinary study of coordination, ACM Computers Surveys, Num. 26(1), pp. . 87-119, 1994. (Ch. 1)

(Marcenac 1997) Marcenac P., Modélisation de systèmes complexes par des agents. In Techniques et Sciences Informatiques, Hermès, 1997. (Ch. 1)

(Martin et al. 1999) Martin D. L., Cheyer A. J. et Moran D. B., The Open Agent Architecture: A framework for building distributed software systems. In Applied Artificial Intelligence, Num. 13(1-2), pp. 91-128, January-March 1999. (Ch. 3)

(Moore 1994) Moore A. W. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In Cowan J. D., Tesauro G. and Alspector J. editors; Advances in Neural Information Processing Systems 6, pp 711-718, Morgan Kaufmann, 1994. (Ch. 4)

(Nareyek 1998) Nareyek A., A Planning Model for Agents in Dynamic and Uncertain Real-Time Environments. In Proceedings of the AIPS Workshop on Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments, Technical Report, WS-98-02, 7-14. AAAI Press, Menlo Park, California, 1998 (Ch. 4)

(Newell et Simon 1961) Newell A. et Simon H. A., GPS, a program that simulates human thought. In Lernende Automaten Billing H. (éditeur), pp. 109-124. R. Oldenbougg, Munich, Allemagne, 1961. (Ch. 4)

(Nicolle 2000) Nicolle A., L'organisation dans les systèmes multi-agents. Journées Colline, Nancy, 2000. (Ch. 1)

(Nwana 1996) Nwana H. S., Software Agents: An Overview. In Knowledge Engineering Review, Vol. 11(3), pp. . 205-244, 1996. (Ch. 1)

(Pearl 1988) Pearl J., Probabilistic reasoning in intelligent systems : Networks of plausible inference. Morgan Kaufman Publishers, Inc., San Mateo, CA, 1988. (Ch. 5)

(Pearl 2000) Pearl J., Models, Reasoning, and Inference. Cambridge, Cambridge University Press, 2000. (Ch. 1)

(Peot et Smith 1992) Peot M. A. et Smith D. E., Conditional Nonlinear Planning. In proceedings of the First International Conference on Artificial Intelligence Planning Systems (AIPS'92), pp. 189-197, College Park, Maryland, 1992. (Ch. 4)

(Pieraccini et al. 1997) Pieraccini R., Levin E. et Eckert W. AMICA: the AT&T Mixed Initiative Conversational Architecture - In proceedings of the 5th European Conference on Speech Communication And Technology (Eurospeech'97), Vol. 4, pp. 1875 - 1878, 1997 (Ch. 6)

(Pineau et al. 2001) Pineau J., Roy N. et Thrun S., A Hierarchical Approach to POMDP Planning and Execution. In Workshop on Hierarchy and Memory in Reinforcement Learning (ICML 2001), Williams College, MA, June 2001. (Ch. 4)

(Pineau et Thrun 2000) Pineau J. et Thrun S., Hierarchical POMDP Decomposition for A Conversational Robot., Janvier 2000. (http://www-2.cs.cmu.edu/~thrun/papers/pineau.hier-pomdp.html) (Ch. C)

(Pohl 1997) Pohl W., Logic-Based Representation and Inference for User Modeling Shell Systems, PhD Dissertation, University of Essen, 1997. (Ch. 5)

(Puterman 1994) Puterman M. L., Markov Decision processes. Discrete stochastic dynamic programming. Wieley-Interscience, New York, 1994. (Ch. 4)

(Quinlan 1986) Quinlan J. R., Induction of decision Trees. In Machine Learning, Kluwer Academic Publishers, Boston, Vol. 1, pp. 81-106, 1986. (Ch. 5, 6)

(Rao et Georgeff 1995) Rao A. S. et Georgeff M. P., BDI agents: from theory to practice, In Proceedings of the First International Conference on Multiagent Systems (ICMAS'95), San Francisco, 1995. (Ch. 1)

(Reddy et al. 1976) Reddy R., Erman L., Fennell R. et Neely R., The HEARSAY speech understanding system: An example of the recognition process. In IEEE Transactions on Computers C-25, pp. 427-431, 1976. (Ch. 1)

(Ricart et Brizzi 1998) Ricart M. et Brizzi T., PMMU, Protocole Multi-Média Unifié - UMM-API : Description, v2.0, 1998. (Ch. 2)

(Rich 1979) Rich E., User Modeling via Stereotypes. In Cognitive Science, Vol. 3, pp. 329-354, 1979. (Ch. 5)

(Rosenblatt 1959) Rosenblatt F., The perceptron: a probabilistic model for information storage and retrieval in the brain. In Psychological Review, Vol. 65, pp. 386-408, 1959. (Ch. 5)

(Roy et al. 2000) Roy N., Pineau J. et Thrun S., Spoken Dialogue Management Using probabilistic Reasoning. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'2000) , Hong Kong, 2000 (Ch. 6, 8, C)

(Rusmevichentong et Van Roy 2001) Rusmevichientong P. et Van Roy B., A Tractable POMDP for a Class of Sequencing Problems. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI'2001), 2001. (Ch. 5)

(Russell et Norvig 1995) Russell S. et Norvig P., Artificial Intelligence: A Modern Approach, The Intelligent Agent Book. Prentice Hall Series in Artificial Intelligence, 1995. (Ch. 1,2)

(Sacerdoti 1974) Sacerdoti E. D., Planning in a Hierarchy of Abstraction Spaces, In Artificial Intelligence, Vol. 5(2), pp. 115-135, 1974. (Ch. 4)

(Sacerdoti 1975) Sacerdoti E. D., The nonlinear nature of plans. In Proceedings of the 4th International Joint conference on Artificial Intelligence (IJCAI'75), pp. 206-214, Tbilisi, Georgie, URSS, 1975. (Ch. 4)

(Sachet et Maillet 2000) Sachet F. et Maillet Y., UNIMEDIA : Application générique, Macros Guide de l'utilisateur, v1.4, MIC2, 2000. (Ch. 2)

(Sansonnet 2001) Sansonnet J.-P., Dialogical Agents : How can ordinary people and active components interact together in a more natural and resilient way, Rapport Technique du LIMSI, novembre 2001. (Ch. 3)

(Schoppers 1987) Schoppers M. J., Universal Plans for Reactive Robots in Unpredictable Environments. In Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI'87), pp. 1039-1046, Milan, Italy, 1987. (Ch. 4)

(Searle 1969) Searle J. L., Speech Acts: An Essay in the Philosophy of Language, Cambridge, Eng.: Cambridge University Press. 1969. (Ch. 1)

(Seneff et al. 1998) Seneff S., Hurley E., Lau R., Pao C., Schmid P. et Zue V., Galaxy-II: A Reference Architecture for Conversational System Development. In Proceedings of the 5th International Conference on Spoken Language Processing, Vol. 3, pp. 931-935, Sydney, Australie, 1998 (Ch. 6)

(Shardanand et Maes 1995) Shardanand U. et Maes P., Social Information Filtering: Algorithms for Automating "Word of Mouth", In Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems, Vol. 1, pp. 210-217, 1995. (Ch. 5)

(Singh et al. 2002) Singh S., Litman D., Kearns M. et Walker M., Optimizing dialogue management with Reinforcement Learning: Experiments with the NJFun System. In Journal of Artificial Intelligence Research (JAIR), Vol. 16, pp. 105-133, 2002. (Ch. 6)

(Smith 1980) Smith R. G., The contract net protocol : High-level communication and control in a distributed problem solver. In IEEE Transactions on Computers, Vol. C-29(12), pp. 1104-1113, Décembre 1980. (Ch. 4)

(Stefik 1981) Stefik M., Planning and Meta-Planning (MOLGEN: Part 2). In Artificial Intelligence, Num. 16, pp. 141-169, 1981. (Ch. 4)

(Stone et Veloso 2000) Stone P. et Veloso M., Multiagent systems: a survey from a machine learning perspective. In Autonomous Robotics, Vol. 8(3), 2000. (Ch. 5)

(Sussman 1975) Sussman G. J., A computer Model of Skill Acquisition. 133 p, Elsevier Science / North-Holland, Amsterdam, London, New-York, Janvier 1975. (Ch. 4)

(Sutton et Barto 1998) Sutton R. S. et Barto A. G., Reinforcement Learning: An introduction. MIT Press, Cambridge, MA, 1998. (Ch. 4)

(Tambre et al. 2000) Tambe M., Pynadath D. V., Chauvat N., Das A. et Kaminka G. A., Adaptive Agent Integration Architectures for Heterogeneous Team Members. In Proceedings of the International Conference on Multi-Agent Systems (ICMAS'2000), pp. 301-308, 2000. (Ch. 3)

(Ten Hagen 2001) Ten Hagen S. H., Continuous State Space Q-Learning for Control of Nonlinear Systems. PhD Thesis of the Computer Science Institute, University of Amsterdam, Pays Bas, 2001. (Ch. 4)

(Tsitsiklis 1994) Tsitsiklis N. J., Asynchronous stochastic approximation and Q-learning. In Machine Learning, Vol. 16(3), 1994. (Ch. 4)

(Vercouter 2000) Vercouter L., Conception et mise en oeuvre de systèmes multi agents ouverts et distribués. Thèse de Doctorat de l'université Jean Monnet et de l'Ecole des Mines de Saint-Etienne, 2000. (Ch. 3)

(Wærn 1996) Wærn A., Recognizing Human Plans: Issues for Plan Recognition in Human - Computer Interaction. PhD Thesis of the Royal Institute of Technology and Stockholm University, 1996. (Ch. 5)

(Wahlster et Kobsa 1986) Wahlster W. et Kobsa A., Dialogue-based user models. In Proceedings of IEEE, Vol. 74(7), pp. 948-960, 1986. (Ch. 5)

(Warren 1976) Warren D. H. D., Generating Conditional Plans and Programs. In proceedings of the Summer Conference on Artificial Intelligence and Simulation of Behavior (AISB'76) Summer Conference, Edinburgh, pp. 344-354 , 1976. (Ch. 4)

(Watkins 1989) Watkins C., Learning from Delayed Rewards. PhD Thesis of the King's College, University of Cambridge, England, 1989. (Ch. 4,6)

(Weld 1994) Weld D. S., An Introduction to Least Commitment Planning. In AI Magazine, Vol. 15(4), pp. 27-61, Eté-automne 1994. (Ch. 4)

(Wilkins 1984) Wilkins D. E., Domain-independent Planning: Representation and Plan Generation. In Artificial Intelligence, Vol. 22, pp. 269-301, 1984 (Ch. 4)

(Wilkins et Myers 1998) Wilkins D. E. et Myers K. L., A Multiagent Planning Architecture. In Proceedings of Artificial Intelligence Planning Systems (AIPS'98), pp. 154-162, 1998. (Ch. 4)

(Winograd et Florès 1986) Winograd T. et Florès F., Understanding computers and cognition : A new foundation for design. Ablex Publishing Corp, Norwood, New Jersey, 1986. (Ch. 1,3)

(Wooldridge 2002) Wooldridge M., An Introduction to Multiagent Systems, 340 p, John Wiley & Sons Publishers, Chichester, England, 2002. (Ch. 1)

(Wooldridge et Jennings 1995) Wooldridge M. et Jennings N. R., Intelligent Agents: Theory and Practice. In The Knowledge Engineering Review, Vol. 10(2), pp. 115-152, 1995. (Ch. 1)

(Young 1999) Young S., Probabilistic Methods in Spoken Dialog Systems. In Proceedings of the Royal Society, London, September 1999. (Ch. 6, C)

(Young 2000) Young S., Talking to Machines (Statistically Speaking) In the 7th International Conference on Spoken Language Processing (ICSLP'02), September, 2002. (Ch. 6)

(Zilberstein 1995) Zilberstein S., Models of Bounded Rationality, In AAAI Fall Symposium on Rational Agency, Cambridge, Massachusetts, Novembre 1995. (Ch. 1)

Publications

Conférences internationales avec comité de lecture et actes

Charton R., Boyer A. et Charpillet F, Learning of mediation strategies for heterogeneous agents cooperation. In Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03). Sacramento, Etats-Unis, 3-5 Novembre, 2003.

Charton R., Boyer A. et Charpillet F, Providing users with adapted services: Dynamic building of dialogues to make heterogeneous agents cooperate. In Proceedings of Special sessions on Multi-agent Systems and Applications of the 2nd IEEE International Symposium on Signal Processing and Information Technology (ISSPIT'2002), Marrakech, Maroc, 18-21 Décembre, 2002.

Charton R., Boyer A. and Charpillet F, Towards bringing heterogeneous agents to cooperation: an architecture for multimedia services. Poster in Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS'02), Palazzo Re Enzo, Bologne, Italie, 15-19 Juillet, 2002.

Boyer A., Charpillet F. and Charton R., Reinforcing interaction between teachers and students in distance learning systems. Poster of the 20th ICDE World Conference on Open Learning and Distance Education (WCOLDE'2001). Düsseldorf, Allemagne, 01-05 Avril, 2001.

Communication à des colloques

Charton R., Boyer A. et Charpillet F, Faire coopérer des agents hétérogènes par apprentissage de médiation. Dans les Actes des Secondes Journées Francophones des Modèles Formels de l'Interaction (MFI'03) , A. Herzig, B. Chaib-draa, P. Mathieu (éditeurs), Cépaduès-Editions, pp 61-70, Lille, France, 20-22 mai 2003.

Boyer A., Charpillet F. et Charton R., Utilisation des agents intelligents dans le commerce électronique. Colloque Form-Ami Information et Education, Le Ciment des Nations, d'INFO 2000 à e-Europe", Commission Européenne, Société de l'information (Form-Ami 2000). Palais du Pharo, Marseille, 13 octobre 2000.

Autres publications, rapports techniques et mémoires

Charton R., Agent approach for Dialoca Services: Architecture and user-mediator interaction. White-Paper Dialoca, Avril 2002.

Charton R., UNIMEDIA White-Paper Dialoca, 2000.

Charton R. Prise en compte de la validation, de la vérification et de la preuve dans le processus de construction de spécifications. Mémoire de DEA de l'Université Henri Poincaré, Nancy, 1998.

Annexe I - Un exemple de modélisation par planification pour le service Nancy-Tour

1. Description du service

Dans le cadre du programme de Recherche et Développement "*La Route Automatisée*", auquel participe l'INRIA, plusieurs équipes travaillent sur un véhicule électrique expérimental, baptisé CyCab et construit par la société RoboSoft. Ce véhicule est destiné à être utilisé comme transport public en libre-service pour la ville de demain. Afin d'illustrer ce thème de recherche et de présenter une vision de ce que pourrait être le tourisme assisté par ordinateur (TAO), l'équipe Maia a réalisé, en collaboration avec les sociétés Dialoca (encore MIC2 à cette date), Neoxy et Lorasi.fr, la démonstration *Nancy-Tour*, le 17 octobre 2000, à Nancy. Au-delà de cette démonstration, le challenge est de réussir à offrir aux usagers du véhicule un véritable assistant touristique intelligent : "*Passer en voiture devant un théâtre et consulter le programme de la semaine; réserver sa table en ligne au moment où l'on passe devant un restaurant ; tout savoir sur le monument historique devant lequel on se gare …*". Lors de la démonstration, illustrée Figure 87 l'usager circulait sur la place Stanislas à bord du CyCab et des informations touristiques apparaissaient sur l'écran embarqué en fonction de la position du véhicule calculée par GPS. L'usager pouvait à tout moment appuyer sur un bouton pour appeler un télé-opérateur humain afin de lui demander une information touristique. Le dialogue une fois établi oralement par téléphone, il était également possible au télé-opérateur de "pousser" des pages hypertexte à destination de l'écran embarqué. Lors de la démonstration, Dialoca a utilisé la plate-forme Unimédia et a défini un service spécifique afin de gérer les communications audio et hypertextes entre le véhicule mobile et le poste du télé-opérateur.

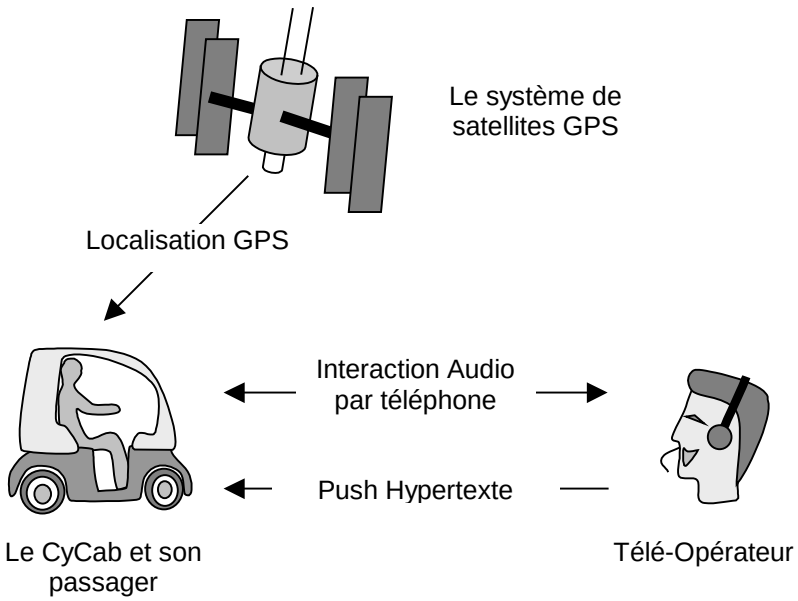


Figure 87 - Interactions utilisées pour la démonstration

2. Notre modélisation

Afin de valider notre architecture générique de communication pour les hSMA, nous avons modélisé une version automatisée du service Nancy Tour dans lequel l'usager dialogue, non plus avec un interlocuteur humain, mais avec un assistant touristique virtuel. La Figure 88 reprend les quatre niveaux de l'architecture présentée au chapitre 7. A gauche, le CyCab et son passager sont considérés comme formant un agent physique non-contrôlé. A droite, nous avons placé un agent logiciel non-contrôlé partenaire, capable de servir de source d'informations. Au centre, la médiation est assurée par un agent logiciel contrôlé.

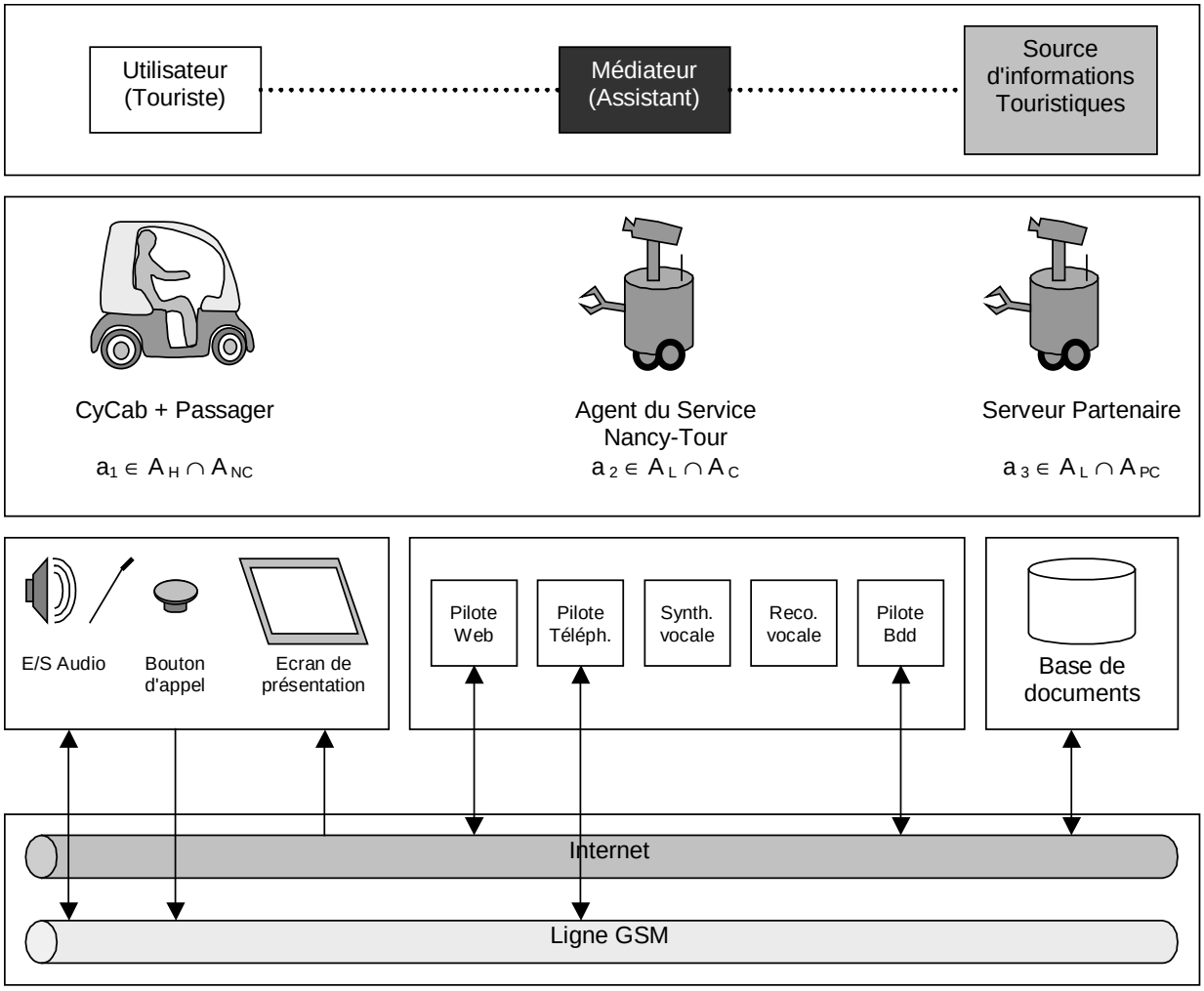


Figure 88 - Architecture agent pour le service Nancy-Tour totalement automatisé

Dans le fonctionnement de ce service, le passager peut appeler l'assistant virtuel au besoin en appuyant sur le bouton. L'assistant peut dialoguer avec le passager et transmettre vers le véhicule des informations hypertexte ("push").

On peut formuler deux remarques sur cette modélisation :

- On a supposé ici que le véhicule était guidé par l'humain. Si ce n'était pas le cas, nous aurions pu ajouter un agent physique pour modéliser le véhicule.
- Dans une autre version du service que nous avons étudié, le médiateur pouvait appeler un guide touristique expert.

3. Vers la production du comportement de médiation

Afin d'étudier le potentiel des systèmes de planification, nous avons écrit un domaine et un problème de planification reprenant un bon nombre de points du service, l'objectif étant de montrer qu'il était possible de produire un premier comportement d'agent.

3.1. Définition PDDL du domaine

En premier lieu, nous donnons ci-dessous la définition du domaine. Tout d'abord, nous définissons les différents prédicats qui permettent de construire les états, puis vient la liste des opérateurs.

Nous avons regroupé les opérateurs qui permettent de gérer les (dé)-connexions sur les flux média, puis les transmissions vocales (audio), hypertextes (visuelles) et enfin nous avons ajouté des opérateurs s'apparentant aux comportements des agents :

- l'assistant qui est appelé par l'appui du bouton et qui va essayer de trouver ce qui intéresse l'usager,
- l'usager (simulé), qui ne connaît pas la ville, va dire à l'assistant ce qu'il veut faire,
- l'assistant, en fonction de ce que l'usager veut faire, va identifier la classe de l'usager et effectuer une proposition, et enfin
- l'usager qui va voir l'information et obtenir satisfaction.

```
;---------------------------------------------------------------------------
; DIALOCA (C) - MAIA - LORIA
;
; Romaric CHARTON
;
; Fichier de définition de domaine (opérateurs) PDDL  pour une simulation
; de tourisme assisté par ordinateur (Nancy-Tour).
;---------------------------------------------------------------------------

(define (domain nancy_tour)
(:requirements :strips)
(:predicates

 (document ?doc) (utilisateur ?util) (systeme ?syst)
 (interet ?modele_utilisateur ?document)
 (profil ?utilisateur ?modele_utilisateur) (profil_infere ?entite ?modele_utilisateur)
 (expression_vocale ?e) (expression_typique ?modele_utilisateur ?expression)
 (expression_invite ?ex)
 (peut_proposer ?entite ?objet)
 (interface ?systeme ?dispositif) (connecte ?dispositif ?flux)
 (dispositif_visuel ?dispo) (dispositif_push ?dispo) (dispositif_vocal ?dispo)
 (dispositif_appel ?dispo) (flux_vocal ?flux) (flux_donnees ?flux)
 (deconnecte ?dispositif)
 (affichage ?dispositif ?document) (perception ?entite ?percept)
 (dans ?entite ?entite)
 (memoire_video ?dispo ?doc)
 (satisfaction ?util)
 (selection ?entite ?doc)
 (appuye ?dispositif)
 (appel ?dispo1 ?dispo2 ?flux)
 (doitTransmettre ?syst ?exp)
 (prononce ?expression ?utilisateur)
 (transmis ?expression ?flux)
 (reception_vocal ?dispo ?recinv)
 (diffusion ?dispo ?exp)
)

; CONNEXIONS
; -------------------------------------------------------------
; L'utilisateur, qui est dans un système qui possède un bouton d'appel
; dans ses interfaces va déclencher un appel sur la ligne vocale s'il appuye sur ce bouton

(:action appel_teleoperatrice
:parameters (?util ?syst ?syst2 ?flux ?disp ?disp1 ?disp2)
:precondition (and
  (utilisateur ?util)
```

```
  (dans ?syst ?util)
  (systeme ?syst)
  (interface ?syst  ?disp)  (dispositif_appel ?disp)  (appuye ?disp)
  (interface ?syst  ?disp1) (dispositif_vocal ?disp1) (deconnecte ?disp1)
  (interface ?syst2 ?disp2) (dispositif_vocal ?disp2) (deconnecte ?disp2)
  (systeme ?syst2)
  (flux_vocal ?flux))
:effect(and (not (appuye ?disp)) (appel ?disp1 ?disp2 ?flux)))

; ---------------------------------------------------------------
; Lorsqu'un dispositif vocal en appelle un autre sur un flux, et qu'ils sont initialement
déconnectés
; alors les dispositifs se connectent sur le flux

(:action reponse_appel_vocal
:parameters (?flux ?disp1 ?disp2 ?syst ?ex)
:precondition (and
  (systeme ?syst) (expression_vocale ?ex) (expression_invite ?ex)
  (dispositif_vocal ?disp1) (deconnecte ?disp1)
  (dispositif_vocal ?disp2) (deconnecte ?disp2)
  (flux_vocal ?flux)  (appel ?disp1 ?disp2 ?flux))

:effect (and
  (connecte ?disp1 ?flux) (not (deconnecte ?disp1))
  (connecte ?disp2 ?flux) (not (deconnecte ?disp2))
  (not (appel ?disp1 ?disp2 ?flux)) (doitTransmettre ?syst ?ex)))

; ---------------------------------------------------------------
; Connexion de deux dispositifs de données sur un flux

(:action connexion_donnees
:parameters (?flux ?disp1 ?disp2)
:precondition (and
  (dispositif_push ?disp1) (deconnecte ?disp1)
  (dispositif_visuel ?disp2) (deconnecte ?disp2)
  (flux_donnees ?flux))
:effect (and
  (connecte ?disp1 ?flux) (not (deconnecte ?disp1))
  (connecte ?disp2 ?flux) (not (deconnecte ?disp2))))

; DECONNEXIONS
; ---------------------------------------------------------------
; Déconnexion de deux dispositifs de données sur un flux

(:action deconnexion_donnees
:parameters (?flux ?disp1 ?disp2)
:precondition (and
  (dispositif_push ?disp1)  (connecte ?disp1 ?flux)
  (dispositif_visuel ?disp2)  (connecte ?disp2 ?flux)
  (flux_donnees ?flux))
:effect (and
  (deconnecte ?disp1) (not (connecte ?disp1 ?flux))
  (deconnecte ?disp2) (not (connecte ?disp2 ?flux))))

; ---------------------------------------------------------------
; Déconnexion de deux dispositifs vocaux sur un flux

(:action deconnexion_vocal
:parameters (?flux ?disp1 ?disp2)
:precondition (and
  (dispositif_vocal ?disp1)  (connecte ?disp1 ?flux)
  (dispositif_vocal ?disp2)  (connecte ?disp2 ?flux)
  (flux_vocal ?flux))
:effect (and
  (deconnecte ?disp1) (not (connecte ?disp1 ?flux))
  (deconnecte ?disp2) (not (connecte ?disp2 ?flux))))

; TRANSMISSIONS VOCALES
; ---------------------------------------------------------------------
; Envoi d'un message vocal par le système

(:action envoi_audio
:parameters (?syst ?exp ?dispo1 ?dispo2 ?flux)
:precondition (and
  (systeme ?syst) (expression_vocale ?exp)
  (doitTransmettre ?syst ?exp) (interface ?syst ?dispo1)
  (dispositif_vocal ?dispo1)   (dispositif_vocal ?dispo2)
  (flux_vocal ?flux)
  (connecte ?dispo1 ?flux) (connecte ?dispo2 ?flux) )
:effect (and (reception_vocal ?dispo2 ?exp) (not (doitTransmettre ?syst ?exp))))
```

```
; ------------------------------------------------------------
; Un dispositif audio qui a reçu une expression vocale
; la diffuse

(:action diffusion_vocale
:parameters (?exp ?dispo)
:precondition (and (expression_vocale ?exp) (dispositif_vocal ?dispo)
  (reception_vocal ?dispo ?exp))
:effect (diffusion ?dispo ?exp))

; ------------------------------------------------
; Envoi d'une phrase prononcée d'un utilisateur

(:action envoi_expression_typique
:parameters (?util ?syst ?expr ?dispo ?flux)
:precondition (and
  (utilisateur ?util) (systeme ?syst) (dans ?syst ?util)
  (dispositif_vocal ?dispo) (interface ?syst ?dispo) (flux_vocal ?flux)
  (connecte ?dispo ?flux) (prononce ?expr ?util)(expression_vocale ?expr))
:effect
  (transmis ?expr ?flux))

; ------------------------------------------------------------------------
(:action reception_vocal_expression
:parameters (?entite ?expr ?disp_rec ?flux)
:precondition (and
  (systeme ?entite) (interface ?entite ?disp_rec) (dispositif_vocal ?disp_rec)
  (connecte ?disp_rec ?flux) (flux_vocal ?flux) (expression_vocale ?expr)
  (transmis ?expr ?flux))
:effect (and (perception ?entite ?expr)(not (transmis ?expr ?flux))))

; TRANSMISSIONS DONNEES / VISUELS
; ------------------------------------------------------------------------
; Une entité qui a sélectionné un document et qui dispose d'une interface
; qui est un dispositif d'émission dispo1 sur un flux de donnés

(:action envoi_visuel
:parameters (?syst ?doc ?dispo1 ?dispo2 ?flux)
:precondition (and
  (document ?doc) (systeme ?syst) (selection ?syst ?doc)
  (interface ?syst ?dispo1) (dispositif_push ?dispo1) (dispositif_visuel ?dispo2)
  (flux_donnees ?flux) (connecte ?dispo1 ?flux) (connecte ?dispo2 ?flux))
:effect (memoire_video ?dispo2 ?doc))

; ------------------------------------------------------------
; Un dispositif d'affichage qui a un document en mémoire vidéo
; affiche ce document

(:action affichage_visuel
:parameters (?doc ?dispo)
:precondition (and
 (document ?doc) (dispositif_visuel ?dispo) (memoire_video ?dispo ?doc))
:effect (affichage ?dispo ?doc))

; PARTIE ASSISTANT
; ------------------------------------------------------------------------
; Si une entite perçoit une expression typique d'un profil, alors
; elle sait qu'elle a affaire à un interlocuteur de ce profil

(:action inference_profil
:parameters (?entite ?profil ?expr)
:precondition (and
  (systeme ?entite) (expression_vocale ?expr) (perception ?entite ?expr)
  (expression_typique ?profil ?expr))
:effect (profil_infere ?entite ?profil))

; ------------------------------------------------------------------------
; Le robot opérateur virtuel va sélectionner le document à envoyer
; en fonction du profil qu'il a inféré et de l'intérêt qu'ont
; les personnes ayant ce profil pour ce document

(:action selection_visuel
:parameters (?entite ?doc ?profil)
:precondition (and
  (systeme ?entite) (profil_infere ?entite ?profil) (interet ?profil ?doc)
  (document ?doc)   (peut_proposer ?entite ?doc))
:effect (selection ?entite ?doc))
```

```
; PARTIE USAGER
;---------------------------------------------------------------
; L'utilisateur qui est dans un système possédant un dispositif
; d'affichage affichant un document perçoit ce document

(:action perception_visuelle_de_document
:parameters (?util ?doc ?dispo ?syst)
:precondition (and
   (utilisateur ?util) (systeme ?syst) (dans ?syst ?util)  (dispositif_visuel ?dispo)
   (interface ?syst ?dispo)  (document ?doc) (affichage ?dispo ?doc))
:effect (perception ?util ?doc))

;---------------------------------------------------------------
; L'utilisateur qui est dans un système possédant un dispositif
; audio diffusant une expression_vocale perçoit ce expression_vocale

(:action perception_de_message_vocal
:parameters (?util ?exp ?dispo ?syst)
:precondition (and
   (utilisateur ?util) (systeme ?syst) (dans ?syst ?util)(dispositif_vocal ?dispo)
   (interface ?syst ?dispo) (expression_vocale ?exp) (diffusion ?dispo ?exp))
:effect (perception ?util ?exp)
)

; ----------------------------------------------------
; Réponse à une demande de but en une phrase par l'utilisateur selon son profil

(:action Enonciation_expression_vocale
:parameters (?util ?profil ?recinv ?expr)
:precondition (and
   (utilisateur ?util) (profil ?util ?profil)(perception ?util ?recinv)
   (expression_invite ?recinv) (expression_vocale ?expr) (expression_typique ?profil ?expr))
:effect
  (prononce ?expr ?util))

; ---------------------------------------------------------------
; Un utilisateur qui a percu un document pour lequel il a un intêret est satisfait

(:action satisfaction_interet_document
:parameters (?util ?doc ?modele)
:precondition (and
   (utilisateur ?util) (profil ?util ?modele) (interet ?modele ?doc)
   (document ?doc) (perception ?util ?doc))
:effect (satisfaction ?util))
)
```

3.2.Définition PDDL du problème

Nous donnons maintenant le problème particulier avec les constantes, l'état initial et l'état final. Afin de pouvoir prendre en compte différents utilisateurs, la fin de la définition de l'état initial permet de changer le passager du véhicule en modifiant les commentaires.

```
;-------------------------------------------------------------------------
; DIALOCA (C) - MAIA - LORIA
;
; Romaric CHARTON
;
; Fichier de définition de problème (faits et buts) PDDL  pour une simulation
; de tourisme assisté par ordinateur (Nancy-Tour).
;-------------------------------------------------------------------------

(define
(problem nancy_tour_service)
(:domain nancy_tour)
(:objects

; Définition des documents pouvant être transmis par le système
musee_lorrain           ; un grand classique de Nancy
musee_beaux_arts        ; va de paire avec le précédent
aquarium                ; des poissons en tout genre
boutique_muzik          ; la meilleure musique de Nancy
magasin_automne         ; un grand centre commercial
hotel_bonne_nuit        ; les lits y sont hyper confortables
```

```
gite_dodo               ; Morphée y donne les clefs
taverne_ecossaise       ; rousse, blonde ou brune ?
discotheque_caveau      ; avec le meilleur écho
resto_le_gourmand       ; pour les Ga
creperie_suzette        ; les meilleurs crepes bretonnes de lorraine

; Définition des profils utilisateurs pris en compte
culturel                ; plutôt musée, visites
pratique                ; resto, hôtel et shopping
loisirs                 ; fête et copains

; Définition des expressions de but (Ce qui dit l'utilisateur)
apprendre
decouvrir
acheter
manger
boire
dormir
sortir
faire_la_fete

; Définition d'une expression de prompt (Ce que demande le système)
Que_souhaitez_vous

; Définition des utilisateurs (les passagers possibles)
mamie_potin
petit_luc
grande_zoe
philou
lola

; Définition des systèmes
cycab
operatrice_virtuelle

; Définition des canaux de communication
ligne_internet
ligne_gsm

; Elements d'interface (dispositifs)
bouton_appel
poste_cycab
navigateur_cycab
push_operatrice
poste_operatrice
)

(:init

; Déclaration des documents
(document musee_lorrain)
(document musee_beaux_arts)
(document aquarium)
(document boutique_muzik)
(document magasin_automne)
(document hotel_bonne_nuit)
(document gite_dodo)
(document taverne_ecossaise)
(document discotheque_caveau)
(document resto_le_gourmand)
(document creperie_suzette)

; Déclaration des propositions possibles sur les objets
(peut_proposer operatrice_virtuelle musee_lorrain)
(peut_proposer operatrice_virtuelle musee_beaux_arts)
(peut_proposer operatrice_virtuelle aquarium)
(peut_proposer operatrice_virtuelle boutique_muzik)
(peut_proposer operatrice_virtuelle magasin_automne)
(peut_proposer operatrice_virtuelle hotel_bonne_nuit)
(peut_proposer operatrice_virtuelle gite_dodo)
(peut_proposer operatrice_virtuelle taverne_ecossaise)
(peut_proposer operatrice_virtuelle discotheque_caveau)
(peut_proposer operatrice_virtuelle resto_le_gourmand)
(peut_proposer operatrice_virtuelle creperie_suzette)

; Lien des profils utilisateurs
(profil mamie_potin culturel)
(profil petit_luc loisirs)
(profil grande_zoe culturel)
(profil philou pratique)
```

```
(profil philou loisirs)
(profil lola culturel)
(profil lola pratique)

; Utilisateurs
(utilisateur mamie_potin)
(utilisateur petit_luc)
(utilisateur grande_zoe)
(utilisateur philou)
(utilisateur lola)

; Description de l'intêret d'un document pour les profil
(interet culturel musee_lorrain)
(interet culturel musee_beaux_arts)
(interet culturel aquarium)
(interet loisirs magasin_automne)
(interet pratique boutique_muzik)
(interet pratique hotel_bonne_nuit)
(interet pratique gite_dodo)
(interet loisirs taverne_ecossaise)
(interet loisirs discotheque_caveau)
(interet pratique resto_le_gourmand)
(interet pratique creperie_suzette)

; Définition des expressions
(expression_vocale apprendre)
(expression_vocale decouvrir)
(expression_vocale acheter)
(expression_vocale manger)
(expression_vocale boire)
(expression_vocale dormir)
(expression_vocale sortir)
(expression_vocale faire_la_fete)
(expression_vocale Que_souhaitez_vous)

; Définition des expressions de requête
(expression_invite Que_souhaitez_vous)

; Définition des expressions typiques selon les profils

(expression_typique culturel apprendre)
(expression_typique culturel decouvrir)
(expression_typique pratique acheter)
(expression_typique pratique manger)
(expression_typique pratique boire)
(expression_typique pratique dormir)
(expression_typique loisirs sortir)
(expression_typique loisirs faire_la_fete)

; Définition des systèmes
(systeme cycab)
(systeme operatrice_virtuelle)

; Définition des dispositifs
(dispositif_vocal poste_cycab)
(dispositif_vocal poste_operatrice)
(dispositif_visuel navigateur_cycab)
(dispositif_push push_operatrice)
(dispositif_appel bouton_appel)

; Constitution des systèmes
(interface cycab bouton_appel)
(interface cycab poste_cycab)
(interface cycab navigateur_cycab)
(interface operatrice_virtuelle poste_operatrice)
(interface operatrice_virtuelle push_operatrice)

; Etat initial des dispositifs
(deconnecte poste_cycab)
(deconnecte poste_operatrice)
(deconnecte navigateur_cycab)
(deconnecte push_operatrice)

(flux_vocal ligne_gsm)
(flux_donnees ligne_internet)

;----------------------------
; Le passager du Cycab

;(dans cycab mamie_potin)
```

```
;(dans cycab petit_luc)
;(dans cycab grande_zoe)
;(dans cycab philou)
(dans cycab lola)
(appuye bouton_appel)
)

(:goal (and
;  (satisfaction mamie_potin)
;  (satisfaction petit_luc)
;  (satisfaction grande_zoe)
;  (satisfaction philou)
  (satisfaction lola)
  (deconnecte poste_cycab)
  (deconnecte poste_operatrice)
  (deconnecte navigateur_cycab)
  (deconnecte push_operatrice)
 )
)
)
```

4. Résultats

En utilisant IPP 4.0, nous obtenons les plans partiels suivants.

Pour l'utilisateur "Philou"

```
0:    Connexion_Donnees Ligne_Internet Push_Operatrice Navigateur_Cycab
      Appel_Teleoperatrice Philou Cycab Operatrice_Virtuelle Ligne_Gsm Bouton_Appel
      Poste_Cycab Poste_Operatrice
1:    Reponse_Appel_Vocal Ligne_Gsm Poste_Cycab Poste_Operatrice Operatrice_Virtuelle
      Que_Souhaitez_Vous
2:    Envoi_Audio Operatrice_Virtuelle Que_Souhaitez_Vous Poste_Operatrice Poste_Cycab
      Ligne_Gsm
3:    Diffusion_Vocale Que_Souhaitez_Vous Poste_Cycab
4:    Perception_De_Message_Vocal Philou Que_Souhaitez_Vous Poste_Cycab Cycab
5:    Enonciation_Expression_Vocale Philou Pratique Que_Souhaitez_Vous Acheter
6:    Envoi_Expression_Typique Philou Cycab Acheter Poste_Cycab Ligne_Gsm
7:    Reception_Vocal_Expression Operatrice_Virtuelle Acheter Poste_Operatrice Ligne_Gsm
8:    Inference_Profil Operatrice_Virtuelle Pratique Acheter
      Deconnexion_Vocal Ligne_Gsm Poste_Cycab Poste_Operatrice
9:    Selection_Visuel Operatrice_Virtuelle Boutique_Muzik Pratique
10:   Envoi_Visuel Operatrice_Virtuelle Boutique_Muzik Push_Operatrice Navigateur_Cycab
      Ligne_Internet
11:   Deconnexion_Donnees Ligne_Internet Push_Operatrice Navigateur_Cycab
      Affichage_Visuel Boutique_Muzik Navigateur_Cycab
12:   Perception_Visuelle_De_Document Philou Boutique_Muzik Navigateur_Cycab Cycab
13:   Satisfaction_Interet_Document Philou Boutique_Muzik Pratique
```

Pour l'utilisateur "Lola"

```
0:    Connexion_Donnees Ligne_Internet Push_Operatrice Navigateur_Cycab
      Appel_Teleoperatrice Lola Cycab Operatrice_Virtuelle Ligne_Gsm Bouton_Appel Poste_Cycab
      Poste_Operatrice
1:    Reponse_Appel_Vocal Ligne_Gsm Poste_Cycab Poste_Operatrice Operatrice_Virtuelle
      Que_Souhaitez_Vous
2:    Envoi_Audio Operatrice_Virtuelle Que_Souhaitez_Vous Poste_Operatrice Poste_Cycab
      Ligne_Gsm
3:    Diffusion_Vocale Que_Souhaitez_Vous Poste_Cycab
4:    Perception_De_Message_Vocal Lola Que_Souhaitez_Vous Poste_Cycab Cycab
5:    Enonciation_Expression_Vocale Lola Culturel Que_Souhaitez_Vous Apprendre
6:    Envoi_Expression_Typique Lola Cycab Apprendre Poste_Cycab Ligne_Gsm
7:    Reception_Vocal_Expression Operatrice_Virtuelle Apprendre Poste_Operatrice Ligne_Gsm
8:    Inference_Profil Operatrice_Virtuelle Culturel Apprendre
      Deconnexion_Vocal Ligne_Gsm Poste_Cycab Poste_Operatrice
9:    Selection_Visuel Operatrice_Virtuelle Musee_Lorrain Culturel
10:   Envoi_Visuel Operatrice_Virtuelle Musee_Lorrain Push_Operatrice Navigateur_Cycab
      Ligne_Internet
11:   Deconnexion_Donnees Ligne_Internet Push_Operatrice Navigateur_Cycab
      Affichage_Visuel Musee_Lorrain Navigateur_Cycab
```

```
12:   Perception_Visuelle_De_Document Lola Musee_Lorrain Navigateur_Cycab Cycab
13:   Satisfaction_Interet_Document Lola Musee_Lorrain Culturel
```

5. Critique des résultats et conclusion

Les plans générés ont été repris et représentés sur la Figure 89. Nous avons tout d'abord laissé de côté la partie de gestion des dispositifs et des flux pour nous focaliser sur les étapes de la communication. Nous avons ensuite réparti les tâches entre l'usager et l'assistant virtuel.

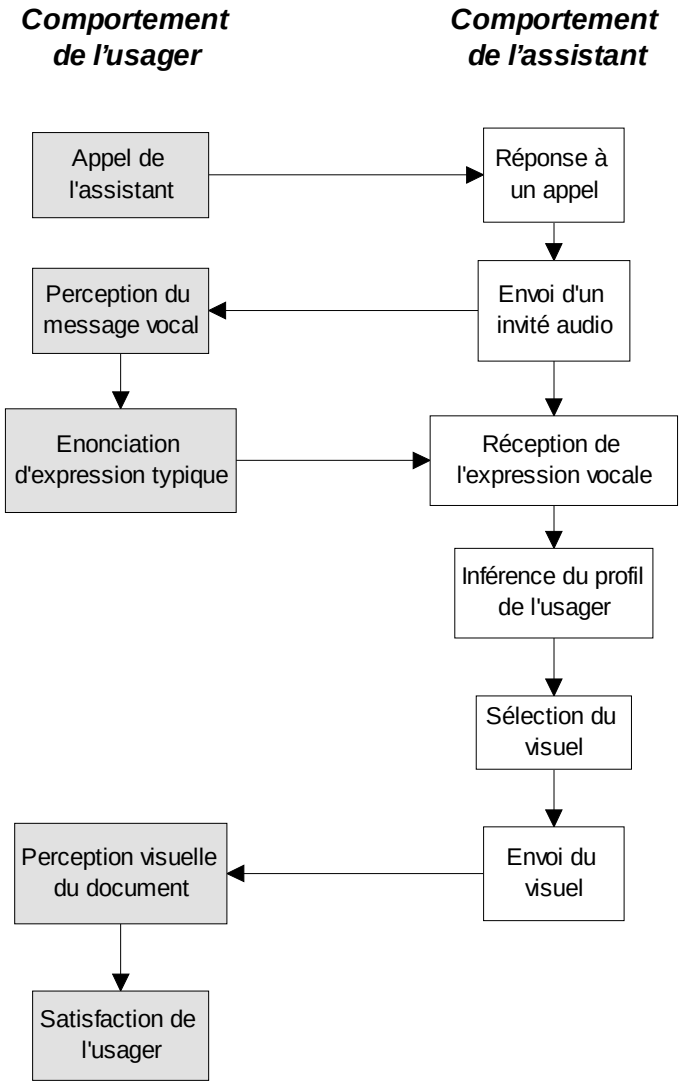


Figure 89 - Vue d'un plan généré comme une interaction

Ainsi, par planification, nous pouvons donc obtenir automatiquement un schéma d'interaction de type Utilisateur-Médiateur. Plusieurs critiques peuvent cependant être faites. Tout d'abord, nous obtenons un plan dans lequel le comportement de l'usager est simulé à partir d'un comportement défini typiquement, or dans des cas réels, cette condition est très rarement vérifiable. Ensuite, il n'y a pas de gestion de comportements conditionnés ou explicitement répétitifs possibles, au mieux, le plan généré offre la possibilité de paralléliser deux comportements indépendants (comme le calcul de profil et la déconnexion dans l'étape 8). Enfin, aucune gestion de l'incertitude ne peut être prise en compte par cette première approche.

Annexe II - Pseudo-code du Q-Learning

Dans cette annexe, nous donnons le pseudo-code que nous avons utilisé dans notre classe ***Q-Learner*** pour implanter l'algorithme du Q-Learning de *(Watkins 89)*.

1. Attributs de la classe

Nom de l'attribut	***Type***	***Description***
Paramètres de la modélisation		
nbEtats	entier	Nombre d'états du modèle
nbActions	entier	Nombre d'actions du modèle
tMin	double	Température de Boltzmann minimale
tMax	double	Température de Boltzmann maximale
tDelta	double	Décroissance arithmétique de la Température de Boltzmann
tFactor	double	Décroissance géométrique de la Température de Boltzmann
gamma	double	Coefficient gamma (escompte du gain)
tauxDepartAlpha	double	Taux d'apprentissage avant seuil de décroissance inverse par visite
seuilAlpha	entier	Seuil de décroissance inverse par visite du Taux d'apprentissage
Variables courantes du MDP		
etat	entier	Etat s_t
nvEtat	entier	Nouvel état s_{t+}
action	entier	Dernière action effectuée
recompense	double	Récompense obtenue r_{t+1}
temperature	double	Température de Boltzmann
Données d'apprentissage		
qvaleur	Tableau [nbEtats, nbActions] de doubles	Q-Valeurs Q(s,a)
nbVisite	tableau [nbEtats] d'entiers	Compteur de visites de l'état pour le taux d'apprentissage

2. Méthodes de la classe

2.1. Constructeur de classe

A la construction d'une instance, nous récupérons le nombre d'états et d'actions, puis nous montons la température de Boltzmann au maximum. Nous initialisons également les Q-Valeurs à zéro.

```
Constructeur   Q-Learner

Paramètres     entier nbS
               entier nbS
               entier initial

Résultats      ∅

// Initialisation du modèle
nbEtats   ← nbS
nbActions ← nbA
temperature ← tMax
nbVisite ← {0 ... 0}
qvaleur ← { {0 ... 0} , ... , {0 ... 0} }
etat ← initial
```

2.2. Fonction de décision

Cette fonction constitue la principale interface du module qui est utilisé dans la boucle d'apprentissage. Nous pouvons faire décroître la température de Boltzmann à la fois de façon géométrique et arithmétique.

```
Fonction       decider

Paramètres     entier  nouvEtat       // le nouvel état
               double  nouvRec        // la nouvelle récompense

Résultats      entier                 // l'action sélectionnée

Variables      entier  memAction

début
       // On stocke les nouvelles valeurs
       nvEtat ← State;
       recompense ← nouvRec;

       // On appelle la fonction qui met à jour la table des Qvaleurs
       QLearningUpdate

       // Mise à jour de la température
       temperature ← (temperature * tFactor) + tDelta;

       // Replace la température dans ses bornes
       si (temperature<tMin) alors  temperature ← tMin fsi
       si (temperature>tMax) alors  temperature ← tMax fsi

       // Met à jour le nombre de visites du nouvel état
       etat ← nvEtat
       nbVisite[etat] ← nbVisite[etat] + 1

       // Sélection de l'action
       memAction ← QLearningSelect();

       retourner memAction;
fin
```

2.3. Fonction de Boltzmann

Cette fonction permet d'obtenir les probabilités d'effectuer chaque action selon la température du système.

```
Fonction       boltzmann
paramètres     entier  s              // état demandé
résultat       double[]               //Table des probabilités d'actions

variables      double[nbActions]      tableProba
               double                 tempExp        // Stockage temporaire
               double                 somme          // Somme des exponentielles
               entier                 a

début
       temp ← 0
       somme ← 0
                                                    Q(s,a)
       // On place d'abord toutes les valeurs des  e  T    pour tout a
       // dans tableProba et on fait leur somme

       pour a de 0 à (nbActions-1) faire
           temp ← exponentielle(qvaleur[s][a]/temperature)
           tableProba[a] ← temp
           somme ← somme + temp
       fpour

       // On redivise toutes les exponentielles par leur somme
       pour a de 0 à (nbActions-1) faire
           tableProba[a] ← tableProba[a] /somme;
       fpour

       retourner tableProba;
fin
```

2.4. Fonction de sélection

Cette fonction utilise un tirage aléatoire sur une distribution de Boltzmann pour sélectionner l'action à accomplir.

```
Fonction       QLearningSelect
Paramètres     ∅
Resultat       entier                 // action à accomplir
variables      double[] probas        // Distribution de probabilité sur les actions à
sélectionner
               double  tirage

début
       // On récupère la table qui donne pour chaque action la probabilité P(a).
       probas ← boltzmann(etat)
       tirage ← 0
       action ← 0

       // Tirage aléatoire sur [0;1]
       tirage ← aléa(0,1)

       // On cherche l'action a correspondant au tirage
       tant que ((tirage>0) et (action<nbActions-1) faire
              tirage ← tirage - probas[action]
              si (tirage ≥ 0) alors action ← action + 1 fsi
       ftq

       retourne action
fin
```

2.5. Fonction de mise à jour des Q-Valeurs

Le taux d'apprentissage α dépend du nombre de visites. On lui donne un taux de départ, puis il décroît comme $\frac{1}{\text{nbVisite}}$ avec un décalage d'origine *seuilAlpha*.

```
Fonction       QLearningUpdate
Paramètres     ∅
Résultats      ∅
Variables      entier  actionTestee
               entier  maxa                  // mémoire de la meilleur action
               double  max                   // mémoire de la meilleure valeur
               double  apprend               // Coefficient d'apprentissage
début
       maxa ←0
       max ←0

       si (nbVisite[nvEtat]>seuilAlpha) alors
              apprend ←  1/ ( nbVisite[nvEtat]-seuilAlpha+(1/tauxDepartAlpha))
       sinon
              apprend ← tauxDepartAlpha
       fsi

       pour actionTestee de 0 à (actionTestee-1) faire
              si (max ≤ qvaleur[nvEtat , actionTestee])  alors
                     max ← qvaleur[nvEtat , actionTestee];
                     maxa ← actionTestee;
              fsi
       fpour

       // Mise à jour de la Q-Valeur
       qvaleur[etat, action]← (1-apprend)*qvaleur[etat, action]  // conservation
                        + apprend   *(recompense +gamma*max);      // modification
fin
```

Annexe III - Script du service E-Nots

Le script présenté dans cette annexe correspond à une version simplifiée du service Dialoca.

Lorsqu'un message arrive au niveau du pilote E-Mail, le pilote demande à l'administration de la plate-forme de déclencher l'application de notification. Il lui transmet ensuite les variables suivantes : l'adresse du destinataire (To), l'adresse de l'expéditeur (From), et le sujet du message (Subject).

ArgMail :

```
       SEQ {  getstrings, NULL, "argmail.mod"
              , SEQ {PARAMS
                     , PARSE(VAR_STRING(MsgVocAtt, ""))
                     , PARSE(VAR_STRING(MsgVocObj, ", dont l'objet est "))
                     , PARSE(VAR_STRING(MsgSMS, "Objet : "))
              }
              , SEQ {BRANCH}
       };
```

Dans la première étape, l'application demande tout d'abord plus d'informations sur le message : le corps du message (MsgFile), les fichiers attachés (Attach) et leur nombre (CntAttach). On voit que l'on a un destinataire NULL, donc l'action est transmise au pilote E-Mail. Trois chaînes sont préparées dans les variables pour les futures interactions vocales. Le branchement fait simplement passer l'application à l'étape suivante.

IdentDest :

```
       SEQ {  getstrings , "bdd:BddUserPool", "identbdd.mod"
              , SEQ {PARAMS, DUMP(To)}
              , SEQ {BRANCH, SEQ { CountBdd , SEQ { "1", IdentOrg}, DestUnkown } }
       };
```

L'application va maintenant tenter d'identifier le destinataire en consultant la base de données des abonnés à partir de l'adresse E-Mail (To). En retour, elle récupère le nom du destinataire, (NomDest) deux numéros de téléphone (NumTel pour l'appel vocal et Portable pour la notification SMS). Elle récupère aussi deux valeurs booléennes (NotifVoc et NotifSMS) indiquant comment l'abonné souhaite être notifié. S'il n'y a pas d'abonné, alors l'application saute à l'étape DestUnknown.

IdentOrg :

```
       SEQ {          getstrings     , "bdd:BddUserPool"   , "identorgbdd.mod"           //
NomOrg
              , SEQ {PARAMS, DUMP(From)}
              , SEQ {BRANCH  , SEQ {CountBdd, SEQ { "1", SapMsg}, AssignNomOrg   }
              }
       };
```

Dans l'étape IdentOrg, l'application recherche dans la base de données si l'expéditeur du message est connu. Si oui, alors elle stocke son nom (NomOrg) et saute à l'étape SapMsg, mais sinon, elle passe à l'étape AssignNomOrg.

AssignNomOrg :

```
SEQ {   NULL, NULL, NULL
        , SEQ {PARAMS, SET(NomOrg, From)}
        , SEQ {NOWAIT}
        , SEQ {BRANCH  , SapMsg}
};
```

Cette étape, sans action, recopie simplement le contenu de la variable From dans NomOrg. Ainsi, elle se servira par la suite de l'adresse e-mail pour nommer l'expéditeur.

SapMsg :

```
SEQ {   send, "exe:ExeUserPoolLinux", "sapmsglh.mod"
        , SEQ {PARAMS  , DUMP(MsgFile)}
        , SEQ {BRANCH
                , SEQ { CntAttach
                        , SEQ { "0", TstObjet}
                        , SEQ { "1", FmtMsgAttUn}
                        , FmtMsgAttPlus
                }
        }
};
```

Dans cette étape, le moteur de synthèse de la parole est appelé et reçoit le contenu du message (MsgFile) à synthétiser. Il produit un fichier audio encodé A-Law. A partir de là, l'application commence donc à préparer les différents messages de notification qui seront utilisés par la suite lorsqu'elle sera par exemple en ligne avec un utilisateur.

FmtMsgAttUn :

```
SEQ {   NULL, NULL, NULL,
        SEQ {PARAMS,
                PARSE(DUMP_WITH_EXT(MsgVocAtt, ", avec ",
                 EVAL(CntAttach)," fichier attaché"))
                }
        , SEQ {NOWAIT}
        , SEQ {BRANCH, TstObjet}
};
```

Dans cette étape interne, l'application procède à un formatage des informations vocales à diffuser pour un fichier attache (MsgVocAtt) puis elle saute à l'étape TstObjet.

FmtMsgAttPlus :

```
SEQ {   NULL, NULL, NULL
        , SEQ {PARAMS,
                PARSE(DUMP_WITH_EXT(MsgVocAtt, ", avec ",
                 EVAL(CntAttach), " fichiers attachés"))
                }
        , SEQ {NOWAIT}
        , SEQ {BRANCH, TstObjet}
};
```

Cette étape est similaire à la précédente mais gère plus d'un fichier attaché.

TstObjet :

```
SEQ {   NULL, NULL, NULL, SEQ {PARAMS}, SEQ {NOWAIT}
        , SEQ {BRANCH  , SEQ {Subject , SEQ {"", FmtObjVide}, FmtObj} }
};
```

Dans cette étape, on regarde simplement si le sujet du message est vide, auquel cas on se branche à l'étape suivante, sinon, on passe à l'étape FmtObjet.

FmtObjVide :

```
SEQ {   NULL    , NULL , NULL
        , SEQ {PARAMS
                , PARSE(DUMP_WITH_EXT(MsgVocObj, "vide"))
                , PARSE(DUMP_WITH_EXT(MsgSMS, "vide"))
        }
        , SEQ {NOWAIT}
        , SEQ {BRANCH  , CallOut}
};
```

Dans cette étape, on prépare les informations vocales et SMS indiquant qu'il n'y a pas d'objet puis on passe à l'étape CallOut.

FmtObj :

```
SEQ {   NULL, NULL, NULL
        , SEQ {PARAMS
                , PARSE(DUMP_WITH_EXT(MsgVocObj,": ", EVAL(Subject)))
                , PARSE(DUMP_WITH_EXT(MsgSMS, EVAL(Subject), EVAL(MsgVocAtt)))
        }
        , SEQ {NOWAIT}
        , SEQ {BRANCH
                , SEQ { NotifSMS
                        , SEQ { "f"
                                , SEQ { NotifVoc
                                        , SEQ { "t", CallOut }
                                        , FinService
                                }
                        }
                        , SMSOut
                }
        }
};
```

L'application prépare les informations vocales et SMS contenant le sujet, puis selon les options de notifications, elle se branche aux étapes SMSOut, Callout ou FinService.

SMSOut :

```
SEQ {   send, "gen:SMSUserPool", "sendSms.mod"
        , SEQ {PARAMS, DUMP(NomOrg, Portable, MsgSMS)}
        , SEQ {BRANCH, SEQ { NotifVoc, SEQ { "f", FinService }, CallOut} }
};
```

Dans cette étape, l'application demande l'envoi d'un mini-message par le pilote SMS à destination de l'abonné. Elle regarde si l'abonné souhaite être notifié vocalement sur son téléphone et dans ce cas, elle passe à l'étape CallOut, sinon elle va à l'étape FinService.

CallOut :

```
SEQ {   send, "voc:OutCall", "calluser.mod"
        , SEQ {PARAMS, DUMP(NumTel)}
        , SEQ {BRANCH
                , SEQ { ReturnCode
                        , SEQ { "Media_UserInactivity", Fin}
                        , Signal
                }
        }
};
```

Cette étape effectue l'appel du destinataire par téléphone à l'aide du pilote Vocal. Si l'appel échoue, alors l'application passe à l'étape Fin.

Signal :

```
SEQ {   getstrings, "voc:OutCall", "siguserlh.mod"
        , SEQ {PARAMS  , DUMP(NomOrg) , DUMP(MsgVocObj), DUMP(MsgVocAtt)          }
        , SEQ {BRANCH
                , SEQ { ReturnCode
                        , SEQ { "Media_UserInactivity",
                         FinService}
                }
                , SEQ { Choix
                        , SEQ { "oui", LectMsg}
                        , SEQ { "non", RecMsgReq}
                }
        }
};
```

Dans cette étape, l'application notifie l'utilisateur sur la ligne téléphonique à l'aide de la synthèse vocale. Elle demande à l'utilisateur s'il souhaite une lecture du contenu du message. L'application se poursuit différemment selon la réponse comme dans les étapes précédentes, mais elle vérifie également qu'il y a toujours une activité sur la ligne téléphonique.

LectMsg :

```
SEQ {   send, "voc:OutCall", "sigmsg.mod"
        , SEQ {PARAMS  , DUMP_WITH_EXT(MsgFile, ".alw")           }
        , SEQ {BRANCH  , RecMsgReq    }
};
```

Dans cette étape, l'application transmet au pilote vocal, le nom du fichier audio qui avait été préparé dans l'étape SapMsg, afin que celui-ci le diffuse par téléphone.

RecMsgReq :

```
SEQ {   getstrings, "voc:OutCall", "recmsgreqlh.mod"
        , SEQ {PARAMS}
        , SEQ {BRANCH, SEQ { ChoixRec , SEQ { "oui", RecMsg}, SEQ { "non", Fin}}}
};
```

Ici, l'utilisateur entend un intitulé statique lui demandant s'il souhaite ou non répondre au courrier électronique qu'il a reçu. Sa réponse est affectée à la variable ChoixRec qui sert d'aiguillage pour la suite du service.

RecMsg:

```
SEQ {   record, "voc:OutCall", "recordmsglh.mod"
        , SEQ {PARAMS
                , SET(FileRec, SessionId)
                , PARSE(DUMP_WITH_EXT(FileRec, "_msg"))
                , DUMP(FileRec)
        }
        , SEQ {BRANCH
                , SEQ { ReturnCode
                        , SEQ {"Media_UserInactivity", ConvToWav}
                }
                , ConvToWav
        },
        };
```

Cette étape permet l'enregistrement du message de réponse que l'utilisateur souhaite renvoyer à l'expéditeur du courrier électronique. L'application demande au pilote vocal de déclencher

l'enregistreur de son depuis la ligne téléphonique. Le message est enregistré dans un fichier audio dont le nom est composé à partir du numéro de session utilisateur.

ConvToWav :

```
SEQ {   send, "exe:ExeUserPoolLinux", "convtowav.mod"
        , SEQ {PARAMS  , DUMP(FileRec)}
        , SEQ {BRANCH  , MailRespSend }
};
```

Ici, l'application appelle un utilitaire permettant la conversion du fichier audio A-Law vers le format "wave" standard afin que l'expéditeur puisse en faire la lecture.

MailRespSend :

```
SEQ {   send, NULL, "respsend.mod"
        , SEQ {PARAMS
                , PARSE(
                        VAR_STRING(TextMail,
"Veuillez consulter la réponse dans le fichier vocal attaché."))
                , FILE_DUMP(BodyMail, "body", "-n -a", TextMail)
                , DUMP(To), DUMP(From), DUMP(Subject)
                , DUMP_WITH_EXT(FileRec, ".wav")
        }
        , SEQ {BRANCH, Fin}
};
```

Dans cette étape, l'application demande au pilote e-mail d'envoyer un message de réponse à l'expéditeur. Le texte du message invite simplement à écouter le fichier en pièce jointe qui est le message audio qu'elle vient d'enregistrer au téléphone.

Disconnection :

```
SEQ {   NULL, NULL, NULL, SEQ {PARAMS}, SEQ {NOWAIT}
        , SEQ {BRANCH
                , SEQ {"PoolName"
                        // Raccroche, on arrete
                        , SEQ {"OutCall", FinService}
                }
        }
};
```

Cette étape est appelée lorsque l'application reçoit un évènement de déconnexion. Elle passe simplement à l'étape FinService.

DestUnkown :

```
SEQ {   disconnect, NULL, NULL
        , SEQ {PARAMS, LOG_STAT("Le destinataire ", To, " est inconnu")    }
        , SEQ {BRANCH
                , FinService
        }
};
```

Cette étape permet d'effectuer une déconnexion dans le cas où le destinataire serait inconnu.

Fin :

```
SEQ {   disconnect, "voc:OutCall", NULL, SEQ {PARAMS}, SEQ {BRANCH , FinService}  };
```

Dans cette étape, l'application demande au pilote vocal de raccrocher la ligne téléphonique.

FinService :

```
SEQ {   NULL    , NULL  , NULL
        , SEQ {PARAMS
                , FILE_REMOVE("-a", MsgFile)
                , PARSE(DUMP_WITH_EXT(MsgFile, ".alw"))
                , FILE_REMOVE("", MsgFile)
        }
        , SEQ {NOWAIT}
        , SEQ {BRANCH
                , SEQ { CntAttach
                        , SEQ { "0"
                                , SEQ { ChoixRec
                                        , SEQ { "oui", FinService3}
                                        , FinService4
                                }
                        }
                        , SEQ { SUPEQ(1)
                                , SEQ { ChoixRec
                                        , SEQ { "oui", FinService2}
                                        , FinService1
                                }
                        }
                }
        }
} ;
```

Dans cette étape, l'application supprime le fichier du message lu au téléphone et passe aux étapes de nettoyage suivantes. Selon le nombre de fichiers joints et selon que le destinataire a répondu oralement ou non, elle passe à FinService1, FinService2 ou à FinService3 qui supprimment les fichiers audio.

FinService1 :

```
SEQ {   NULL, NULL, NULL
        , SEQ {PARAMS  , FILE_REMOVE("", Attach)}
        , SEQ {NOWAIT}
        , SEQ {BRANCH, FinService4}
} ;
```

FinService2 :

```
SEQ {   NULL    , NULL  , NULL
        , SEQ {PARAMS  , FILE_REMOVE("", Attach), FILE_REMOVE("", FileRec) }
        , SEQ {NOWAIT}
        , SEQ {BRANCH, FinService4}
} ;
```

FinService3 :

```
SEQ {   NULL    , NULL  , NULL
        , SEQ {PARAMS  , FILE_REMOVE("", FileRec)}
        , SEQ {NOWAIT}
        , SEQ {BRANCH, FinService4}
} ;
```

FinService4 :

```
SEQ {   NULL    , NULL  , NULL  , SEQ {PARAMS} , SEQ {NOWAIT} , SEQ {BRANCH} } ;
```

Cette étape sans aucune activité constitue la fin normale du service.

Index