Proposition de l'architecture de l'agent gestionnaire du modèle de l'apprenant dans un système tuteur multi-agents en apprentissage de la lecture : contribution au projet AMICAL

Didier Fragne

THESE

présentée en vue de
l'obtention du titre de

DOCTEUR

de

L'UNIVERSITE BLAISE PASCAL
CLERMONT-FERRAND II

Discipline : Linguistique, Logique et Informatique
Spécialité : Informatique

par

Didier FRAGNE

Proposition de l'architecture de l'agent gestionnaire du modèle de l'apprenant dans un système tuteur multi-agents en apprentissage de la lecture : contribution au projet AMICAL

Présentée et soutenue le 18 décembre 2009, devant le jury composé de :

Directeur de thèse	**Michel CHAMBREUIL** Professeur émérite à l'Université Blaise Pascal, Clermont-Ferrand II
Rapporteurs	**Adina Magda FLOREA** Professeur à l'Université polytechnique de Bucharest, Roumanie
	Brahim CHAÏB-DRAA Professeur à l'Université Laval, Québec, Canada
Examinateur	**Olivier GUINALDO** Maitre de conférences à l'Université d'Auvergne, Clermont-Ferrand I

Remerciements

Au terme de ce travail, je souhaite remercier toutes les personnes qui, de près ou de loin, ont contribué à son bon déroulement.

Mes premiers remerciements s'adressent aux membres du jury qui m'ont fait l'honneur d'examiner ce travail. Je tiens tout d'abord à remercier Brahim Chaïb-Draa et Adina Magda Florea d'avoir accepté d'être rapporteurs de cette thèse. Je les remercie également pour les nombreux échanges fructueux que nous avons eus ainsi que pour leur disponibilité et leurs encouragements. Je remercie également Olivier Guinaldo d'avoir accepté d'examiner ce travail.

Je tiens également à remercier Michel Chambreuil qui a encadré cette thèse. Je le remercie tout spécialement pour m'avoir donné l'opportunité de réaliser ma thèse au sein du projet et de l'équipe AMICAL, projet pluridisciplinaire qui m'a donné l'occasion à la fois de rencontrer des gens de spécialités différentes à la mienne et de découvrir des domaines, pour moi, jusqu'alors inconnus.

Je remercie nos expertes en apprentissage de la lecture Annie Chambreuil et Véronique Quanquin pour leur aide et leurs conseils. Je remercie également Paul Lotin, ingénieur de recherche, pour m'avoir fait découvrir le monde Apple.

Je remercie également l'ensemble des doctorants avec qui j'ai eu de nombreux échanges positifs et avec lesquels j'ai eu d'excellents moments et tout particulièrement : Rabiha, Nabila, Victor, Hicham et Sofiane.

Enfin, je tiens à adresser mes derniers remerciements aux personnes qui m'ont toujours soutenu dans les moments difficiles et sans qui ce travail n'aurait certainement jamais pu être terminé : mes parents, André et Michèle et mon épouse Rabiha.

A tous, un grand merci.

A Michèle et André, mes parents,
A Rabiha, mon épouse,
A Clarisse, ma grand mère,
A Jean-Claude, mon oncle.

TABLE DES MATIÈRES

INTRODUCTION

1. Objectifs et problématiques

L'apprentissage de la lecture fait – aujourd'hui encore – de plus en plus débat au sein de la société française. En France, environ 400000 enfants de l'école primaire sont en difficulté de lecture. Selon une étude épidémiologique réalisée en 2007, 12,7% des enfants scolarisés en CE1 ont un retard de lecture de douze mois ou plus. Pour 3,5% d'entre eux le retard atteint même dix-huit mois ou plus (Billard et al., 2007). On peut également citer les indicateurs publiés récemment par le Haut Conseil de l'Éducation qui fait apparaître que 15% des élèves de CM2 connaissent des difficultés sévères ou très sévères de compréhension et de repères méthodologiques et culturels. Ces enfants ne sont donc pas capables de comprendre l'ensemble du sens du texte qu'ils ont à lire. Le rapport Pirls 2006 (Progress International Reading Literacy Study) réalisé dans 40 pays classe la France au 27ème rang sur 40 (Mullis et al., 2007).

Ces problèmes trouvent leurs racines dès le début du cours préparatoire, c'est-à-dire là où les enfants commencent à apprendre à lire. Une solution consiste, et c'est là le rôle du projet AMICAL, à s'adapter le plus possible à l'enfant pour lui permettre d'apprendre à son rythme et de corriger, dès le début, ses problèmes d'apprentissage. L'environnement AMICAL (Architecture Multi-agents Compagnon pour l'Apprentissage de la Lecture) se veut un système tuteur intelligent multi-agents en apprentissage de la lecture visant, comme tout système tuteur intelligent, à s'adapter le plus possible à l'enfant avec lequel il travaille, donc à individualiser son enseignement.

Pour individualiser leur enseignement, les systèmes tuteurs intelligents ou plus généralement les EIAH (Environnements Informatiques pour l'Apprentissage Humain) disposent d'un modèle de l'apprenant. Celui-ci contient les croyances du

système sur l'enfant, c'est-à-dire ce que ce dernier sait, ce qu'il ne sait pas, les erreurs qu'il commet, etc.

Dans le cadre de l'environnement multi-agents AMICAL, cette thèse se propose d'étudier les problématiques liées à la conception de l'agent en charge de la gestion de ce modèle de l'apprenant. Nous devrons donc réfléchir à un agent s'intégrant à l'architecture multi-agents déjà existante. Cela signifie que nous ne remettrons pas en cause l'architecture de l'environnement ni sa pertinence.

Nos problématiques seront à la fois de réfléchir à une formalisation du modèle de l'apprenant dans un domaine particulier qu'est l'apprentissage de la lecture ainsi que de proposer des mécanismes de construction de ce modèle, intégrés au sein de l'agent gestionnaire de ce modèle. Par gestion du modèle de l'apprenant, nous entendons à la fois la construction (ou modélisation) du modèle de l'apprenant et la réponse aux interrogations des autres agents de l'environnement concernant les informations contenues dans ce modèle. Néanmoins, nous nous focaliserons dans cette thèse quasiment exclusivement sur la construction de ce modèle par l'agent.

La contribution apportée par nos travaux est de deux ordres. Premièrement, nous proposons dans ce document, une représentation du modèle de l'apprenant sous forme de système multi-agents cognitifs où chaque agent sera en charge d'une partie du modèle. Cette tendance n'a en effet pas vraiment été explorée, mais nous pensons que cette voie est porteuse de nombreux avantages. Deuxièmement, nous proposons de voir l'architecture de l'agent gestionnaire du modèle de l'apprenant lui-même comme un système multi-agents. De plus, comme le modèle de l'apprenant représente une partie des croyances de l'agent gestionnaire du modèle de l'apprenant, et comme nous avons décidé de représenter ce modèle au travers d'un système multi-agents, nous avons donc représenté une partie des croyances de notre agent par un système multi-agents.

2. Contexte de travail et précisions

Le projet AMICAL a pour objectif de mettre en œuvre un environnement individualisé d'aide à l'enseignement et à l'apprentissage de la lecture pour des enfants en scolarité normale, en début CP. Il se veut à la fois un projet d'études théoriques et pratiques. Les recherches théoriques concernent l'apprentissage de la

lecture dans des domaines aussi variés que la pédagogie, la didactique ou encore la psychologie cognitive. Les recherches pratiques, quant à elles, consistent en la conception d'un environnement multi-agents d'apprentissage individualisé en utilisant les technologies d'intelligence artificielle comme les agents cognitifs. Ce projet fait donc appel à de très nombreuses expertises théoriques et pratiques composant l'équipe de recherches : chercheurs en linguistique, en apprentissage de la lecture, en intelligence artificielle ainsi que des enseignants de terrain ou auprès d'IUFM.

La question de l'utilisation de systèmes informatiques pour aborder le problème de l'apprentissage de la lecture n'est pas nouvelle (Blanchard al., 1987). On peut citer quatre orientations parmi les travaux existants :

1. des systèmes qui cherchent à simuler des aspects de l'activité de lecture (Just et Carpenter, 1987), (Seidenberg et McClelland, 1989),
2. des systèmes qui sont développés comme outils d'aide à la recherche sur la lecture ou sur l'écriture. Ce sont des instruments de laboratoire destinés à une évaluation des performances des élèves dans des conditions préalablement définies (Sprenger-Charolles, 1993),
3. des systèmes qui mettent à la disposition d'un élève des ressources de natures diverses. On peut situer dans cette catégorie des dictionnaires ou des "livres électroniques" proposant différentes fonctionnalités de recherche d'informations à partir d'un texte ou des aides à la lecture (par exemple, un mot est prononcé par cliquage sur ce mot) (Olson et Wise, 1992).
4. enfin, des systèmes qui se présentent comme des systèmes d'Enseignement Assisté par Ordinateur (EAO) traditionnels, par la nature des médias utilisés, de l'analyse des réponses de l'élève, ou du mode de conduite des activités proposées. Ces systèmes concernent dans leur grande majorité des aptitudes mises en œuvre dans l'activité de lecture, et constituent le plus souvent, un entraînement à la lecture ou à l'écriture. Ils peuvent correspondre à un moment très particulier de l'apprentissage de la lecture ou couvrir un plus large éventail. Nous citerons, pour le français, des systèmes tels que ELMO, LUCIL, et pour l'anglais, le projet LISTEN (Mostow et al., 1993).

Certains des systèmes d'EAO ou des systèmes ressources intègrent, des éléments permettant l'analyse de performances. Par exemple, un logiciel d'aide à

l'apprentissage de l'écriture présenté en 1992 par l'AFL (Association Française pour la Lecture) capte tous les essais de l'élève dans sa démarche d'écriture et donne ensuite directement accès à l'enseignant à l'ensemble des informations stockées. Mais ces données sont livrées à l'état brut, c'est-à-dire sans aucune tentative d'interprétation de la part du système pour conduire à une représentation d'états de savoir de l'élève.

L'environnement du projet AMICAL se voulant être un environnement d'aide à l'apprentissage et à l'enseignement de la lecture dans une perspective d'individualisation de l'enseignement par le système informatique, celui-ci doit donc pouvoir mettre en œuvre les expertises permettant cette individualisation. Cela signifie que ces expertises doivent être explicitées et représentées, mais aussi qu'elles doivent être intégrées dans une architecture permettant leur gestion coopérative. Cette caractéristique de l'environnement AMICAL le situe par rapport à l'EAO de la lecture et montre qu'il ne peut être classé selon aucune des trois premières orientations. De plus, contrairement aux autres systèmes existants, l'environnement AMICAL n'est pas uniquement un système d'entrainement pour l'apprenant mais a pour vocation première de lui apprendre à lire.

Pour cela, l'environnement AMICAL est constitué de trois modules ayant chacun un rôle différent :

1. Un module ressources permettant à l'enfant d'accéder à de nombreuses ressources concernant la lecture, comme un dictionnaire.
2. Un module exploration qui permet à l'enfant, comme son nom l'indique, d'explorer d'une manière autonome les différents concepts de la lecture. Par exemple, l'enfant peut, à partir de mots donnés, construire une phrase.
3. Un module tutoriel, qui lui, est le cœur même de l'environnement AMICAL. Ce module est en quelque sorte l'enseignant qui apprend à lire à l'enfant. C'est lui qui propose à celui-ci les différentes séquences d'activités, déterminées en fonction de son niveau, pour lui apprendre à lire.

A partir de maintenant, quand nous parlerons de l'environnement AMICAL, nous nous placerons toujours dans le cadre du module tutoriel. C'est dans ce module que nous situons également nos travaux de recherches.

Bien que nous évoquerons plus précisément le fonctionnement du module tutoriel

au chapitre 2, nous allons présenter succinctement le fonctionnement de celui-ci pour situer notre travail. Comme nous l'avons dit précédemment, l'objectif de ce module est d'apprendre à lire à un enfant de CP. Pour ce faire, le système va, à chaque fois où il va travailler avec l'enfant, présenter une séquence d'activités à la fois pour faire acquérir des connaissances à l'enfant et pour que l'environnement acquiert lui aussi des informations sur l'enfant, sur ses connaissances, ses lacunes etc.. Le fonctionnement du module suit trois étapes :

1. La première étape correspond à ce que nous appelons la planification didactique qui consiste à planifier la séquence d'activités. Elle se décompose elle-même en deux sous-étapes :
    1. Construction de l'objectif de la séquence (le quoi). A partir du modèle de l'apprenant et d'autres connaissances, nous déterminons l'objectif à atteindre pour la prochaine séquence d'activités.
    2. Détermination de la séquence d'activités à présenter (le comment). A partir de l'objectif précédemment déterminé, nous déterminons la séquence d'activités individualisées que nous allons présenter à l'enfant.
2. La deuxième étape consiste en la présentation de cette séquence à l'enfant en mémorisant les interactions entre celui-ci et le système dans un fichier que l'on nommera compte-rendu par la suite.
3. La troisième et dernière étape, celle qui correspond à cette thèse, consiste à analyser ces comptes-rendus et à mettre à jour le modèle de l'apprenant en conséquence.

Pour situer les ambitions de l'environnement AMICAL par rapport aux autres systèmes existants et donc pour signaler la complexité de notre tâche, nous présentons maintenant les spécificités de celui-ci. L'environnement AMICAL se distingue des autres systèmes par :

1. la volonté d'enseigner un domaine dans toute sa globalité et dans toute sa complexité, alors que les autres se limitent à une partie d'un domaine : l'addition au lieu de l'arithmétique par exemple,
2. le domaine d'enseignement choisi, plus complexe et pas réellement « tranché » : l'apprentissage de la lecture. En effet, il n'existe pas de didactique de la lecture et des débats existent encore sur les méthodes d'apprentissage de celle-ci : méthode globale versus méthode syllabique,
3. le degré de finesse des connaissances prises en compte dans le modèle de l'apprenant. Par exemple, au lieu de nous intéresser uniquement à la

connaissance globale de la lettre, nous nous intéressons à toutes les sous-connaissances de cette lettre : nom, son, graphies,

4. le nombre de connaissances prises en compte dans le modèle de l'apprenant. Nous ne nous intéressons pas uniquement aux connaissances du domaine (lettres, mots, …) mais également à d'autres aspects tels que les aspects méta-cognitifs, comportementaux et cognitifs,
5. le choix d'utiliser un système multi-agents cognitifs comme architecture de l'environnement.

Avant de présenter le plan de cette thèse, il nous a paru important d'apporter quelques remarques entourant la réalisation de ce travail.

Pour simplifier notre travail, nous nous situerons dans un système autonome où l'enseignant n'intervient pas et où l'on ne prendra en compte que la gestion d'un seul apprenant sans se poser la question de savoir comment traiter plusieurs apprenants en même temps.

Cette thèse se situe au carrefour de nombreuses disciplines informatiques (ingénierie des connaissances, représentation des connaissances, agents). La compréhension du domaine de l'apprentissage de la lecture fut plus difficile et plus longue car ne faisant pas partie des connaissances « de base » de tout informaticien. L'analyse et la modélisation du domaine n'ont pu être réalisées qu'en partie, en raison de la taille et de la complexité du domaine abordé. En effet, la complexité de notre problème est en grande partie liée à la complexité du domaine étudié : l'apprentissage de la lecture ainsi qu'au fait d'avoir choisi l'enseignement de ce domaine dans tout son ensemble.

Comme l'objectif de cette thèse consiste en la conception d'un agent sur un domaine où il n'existe ni théorie ni didactique, cette thèse, pour arriver à cette fin, a nécessité de proposer et de prendre position sur de nombreux points.

De plus, bien qu'un premier prototype de l'agent ait été développé, celui-ci est très limité en raison de nombre de travaux toujours en cours de réalisation au sein de l'équipe AMICAL ; travaux aussi bien en apprentissage de la lecture qu'en informatique. De fait, une évaluation de celui-ci n'a pu être réalisée en raison du manque de données sur lesquelles s'appuyer. Ce prototype a essentiellement permis de retenir les outils à utiliser pour les développements futurs et à montrer

leur mise en oeuvre concrète.

3. Plan de travail

Bien qu'il existe différentes méthodes de conception agent, telles que GIAI (Wooldridge et al., 2000) ou MAS-CommonKADS (Iglesias et al., 1998), nous avons choisi de n'utiliser aucune de ces méthodes car ne correspondant pas vraiment à notre problématique. Pour concevoir notre agent, nous avons choisi de considérer notre agent comme n'importe quel autre logiciel et de suivre la démarche consistant simplement à comprendre ce que fait l'agent avant de le concevoir. Le plan général de cette thèse suivra donc globalement cette démarche. Tout d'abord, nous nous intéresserons au domaine d'application de l'agent, c'est-à-dire à la modélisation de l'apprenant, d'un point de vue général d'abord (chapitre 1) puis dans le cas particulier d'AMICAL (chapitre 2). Ensuite, nous proposerons une architecture multi-agents pour notre agent et nous en expliquerons le fonctionnement général (chapitre 3) avant de rentrer dans le détail de chacun des sous-agents (chapitre 4). Enfin, nous terminerons en présentant l'implémentation d'un premier prototype de cet agent (chapitre 5). Nous conclurons évidemment en évoquant les différentes perspectives possibles pour ce travail.

CHAPITRE 1 : SYNTHÈSE EN MODÉLISATION DE L'APPRENANT

Ce premier chapitre a pour principal objectif de présenter ce qu'est la modélisation de l'apprenant en montrant les différentes informations modélisées par différents environnements d'apprentissage et en présentant les techniques les plus répandues, à la fois pour représenter les connaissances de l'apprenant et pour élaborer ce modèle.

La modélisation de l'apprenant a pour objectif de construire un modèle informatique rendant compte de l'état cognitif et/ou psychologique de l'apprenant à partir des comportements observables de celui-ci à l'interface d'un système informatique. Le comportement de l'apprenant représente la suite des interactions entre le système informatique et l'apprenant. Par état cognitif ou psychologique, il convient de comprendre : ses connaissances, ses lacunes mais aussi ses savoir-faire, ses intentions, voire même ses émotions. Par comportement observable, nous entendons la séquence d'actions observables que l'apprenant exécute à l'interface du système informatique et qui sont utilisées comme des entrées du processus de modélisation de l'apprenant. Le système utilise ces observables pour créer des interactions qui vont susciter des comportements chez l'apprenant à l'interface informatique, et ainsi générer de nouveaux observables. Cette boucle permet au système de valider ou de réfuter ses hypothèses sur la connaissance de l'apprenant.

Ce modèle informatique peut remplir quatre fonctions :

- *Simuler et prédire le comportement d'un apprenant*. Dans certaines applications, il est nécessaire de tester l'efficacité de différentes conceptions de systèmes afin de prévoir leurs performances dans des situations d'usage différentes avant de le mettre à la disposition de vrais utilisateurs.
- *Evaluer les connaissances et les compétences de l'apprenant*. Le modèle

est une source d'informations pour aider les enseignants dans leurs pratiques d'enseignement. Il peut également servir à l'apprenant lui-même.
- *Adapter le système aux besoins de l'apprenant.* Le modèle permet de lui venir en aide pendant la réalisation d'une tâche.
- *Organiser l'interaction*. Le modèle constitue une source de critères de décisions pour le système. Il permet ici de déterminer les différentes activités d'apprentissage dans lesquelles placer l'apprenant pour lui faire acquérir une connaissance particulière.

Bien qu'une ouverture à l'enseignant soit envisagée, c'est essentiellement pour les deux dernières utilisations que le modèle de l'apprenant au sein de l'environnement AMICAL est construit : déterminer la prochaine session de travail avec l'apprenant et éventuellement adapter cette session en cours d'exécution.

Un système informatique d'apprentissage, plus généralement appelé Environnement Informatique pour l'Apprentissage Humain (EIAH) s'intéresse particulièrement aux modèles de l'apprenant pour personnaliser l'enseignement en fonction de l'apprenant. Ces systèmes considèrent que chaque apprenant doit être vu comme un individu unique dans une réalité qui lui est unique, ayant des besoins particuliers ainsi que des connaissances et expériences propres, ses préférences, ses styles et ses stratégies personnels. Le modèle permet donc au système de raisonner sur chaque apprenant, afin de répondre à ses besoins pour la réalisation de l'apprentissage visé.

Le processus d'élaboration d'un modèle de l'apprenant se déroule en trois étapes (Py et Hibou, 2006) :

1. *Conception* : consiste à définir la nature et la signification des informations qui vont figurer dans le modèle ainsi que leurs représentations informatiques.
2. *Initialisation* : consiste à établir un modèle initial (de départ) pour que le système puisse commencer à travailler avec l'apprenant. Ce modèle peut être renseigné soit avec les caractéristiques d'un apprenant pris comme référence, soit avec des valeurs par défaut ou bien encore être vide. Cette dernière solution est réputée plus fiable puisque le modèle est créé uniquement à partir du comportement réellement observé.
3. *Diagnostic* : il s'agit là d'acquérir concrètement les valeurs du modèle et

de les mettre à jour, pour un apprenant donné, au cours du temps, à partir des observables recueillis à l'interface.

La première phase, celle de conception d'un modèle de l'apprenant, est la phase la plus importante. C'est dans cette phase que l'on s'interroge sur ce que l'on cherche à modéliser et sur le comment on va le faire. Elle pose généralement trois questions (Kass, 1987), (Nicaud et Vivet, 1988), (Holt et al., 1993) :

1. déterminer les informations précises sur l'apprenant devant être modélisées.
2. déterminer leur mode de représentation informatique,
3. déterminer les mécanismes de mise à jour de ce modèle à partir des évènements produits par l'apprenant sur l'interface. Ce processus d'inférences à partir de ces évènements est généralement désigné par le termes de « diagnostic ».

Ce chapitre est organisé essentiellement en suivant les trois questions précédentes. L'idée étant de faire un tour, non exhaustif bien entendu, de l'existant concernant les réponses possibles à ces trois questions. L'environnement AMICAL[1] étant un système tuteur intelligent, nous commencerons par rappeler ce qu'est ce type de système ainsi que son architecture générale avant de détailler les différents rôles et types de modèle de l'apprenant. Nous répondrons ensuite à la première question en présentant sur quelques exemples représentatifs d'environnements existants, ce que peut contenir un modèle de l'apprenant. Puis, nous répondrons à la deuxième question en présentant les techniques les plus utilisées de représentation des connaissances d'un apprenant au sein d'un modèle. Ensuite, nous répondrons à la troisième question en présentant différentes méthodes de diagnostic, celles-ci dépendant essentiellement du type de connaissance que l'on cherche à modéliser. Nous conclurons enfin ce chapitre en présentant les spécificités de nos recherches dans le cadre de l'environnement AMICAL.

1. Les systèmes tuteurs intelligents

Comme nous l'avons dit, l'environnement AMICAL est un système tuteur intelligent. Il convient donc de rappeler ce qu'est ce type de système avant de

1 Comme nous l'avons dit en introduction, l'environnement AMICAL est à considérer dans le cadre de cette thèse comme étant ramener uniquement à son module tutoriel.

parler de manière plus détaillée de modèle et modélisation de l'apprenant.

L'histoire de l'intelligence artificielle (IA) montre que les efforts des chercheurs dans le domaine des systèmes tuteurs intelligents ont abouti à la production d'un ensemble de modèles et de techniques ayant comme objectif d'améliorer les conditions d'enseignement et d'apprentissage des apprenants, enfants comme adultes. Ainsi, dans un premier temps, on voit apparaître un bon nombre de systèmes pédagogiques qui se sont contentés de traduire sous forme informatique les supports traditionnels de l'enseignement comme les cours, les recueils d'exercices, etc. Cependant, l'émergence de ces systèmes a donné l'espoir d'aller plus loin et d'apporter des solutions aux problèmes de l'enseignement et de l'apprentissage.

L'arrivée de l'ordinateur et surtout l'augmentation continuelle de ses performances a permis la réalisation de programmes d'enseignement toujours plus évolués. Ces programmes peuvent assurer l'enseignement d'un cours découpé en leçons optimisées pour un apprenant particulier mais aussi poser des questions et analyser les réponses de l'apprenant. Néanmoins, de nombreuses questions demeurent (Bruillard, 1997). Si les tout premiers systèmes se sont révélés extrêmement basiques, les systèmes tuteurs intelligents, dès les années 70, se sont attaqués aux problèmes de fond de l'apprentissage (O'Shea et Self, 1983).

L'apparition des techniques d'intelligence artificielle a donné l'espoir d'apporter une réponse aux critiques des tout premiers systèmes, notamment par la capacité à résoudre des problèmes et par la capacité d'expliquer les solutions. Le domaine s'est donc recentré autour du thème de l'Enseignement Intelligemment Assisté par Ordinateur (EIAO), et en particulier celui des Systèmes Tuteurs Intelligents (STI par la suite).

1.1. Architecture d'un STI

Un STI est censé afficher un comportement proche de celui d'un tuteur humain. Cette propriété se traduit notamment par une capacité à gérer l'interaction en adaptant la situation à l'individu qui utilise ce système.

Un STI doit posséder des connaissances de natures diverses : sur le domaine

enseigné, sur la façon d'enseigner, sur l'apprenant et sur les moyens de communiquer. C'est la raison pour laquelle un STI est généralement composé de quatre modules : le module de l'expert du domaine, le module pédagogique, le modèle de l'élève ou de l'apprenant et l'interface (Sleeman et Brown, 1982), (Wenger, 1987), (Nicaud et Vivet, 1988). Les trois premiers modules correspondent respectivement au quoi, au comment et à qui (Hartley et Sleeman, 1973) cité dans (Bruillard, 1997). Le quatrième module est le lieu de communication entre le STI et l'apprenant. Il s'agit généralement d'interfaces graphiques présentées sur l'écran de l'ordinateur et sur lesquelles peut agir l'apprenant, par exemple, pour donner sa solution.

L'architecture classique peut être représentée de la façon suivante (Bruillard, 1997) :

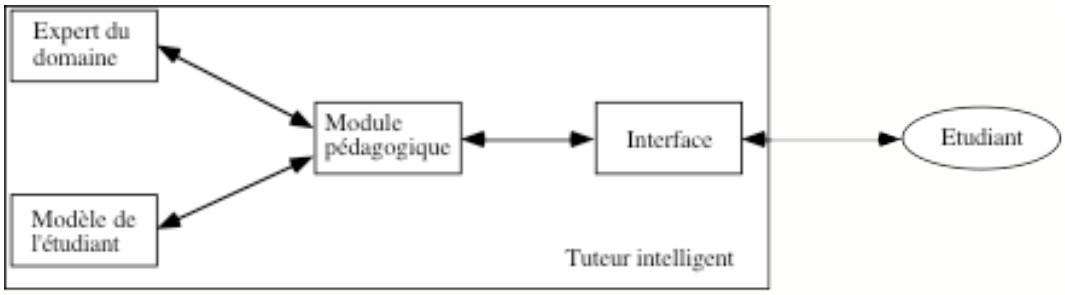

Figure 1.1 : Architecture d'un STI

1.2. Le module de l'expert du domaine

Une des principales caractéristiques d'un STI, comme pour un enseignant, est d'être compétent dans la matière qu'il enseigne. Cette compétence permet au tuteur d'exposer son raisonnement et de justifier ses décisions. Pour cela, le STI doit contenir la connaissance spécifique au domaine ainsi que les processus de raisonnement intervenant lors de la résolution de problèmes (Swartz, 1992). Ce module remplit donc deux fonctions : être la source des connaissances à présenter à l'apprenant et évaluer la performance de celui-ci. En tant que source de connaissances, il détermine les connaissances à faire acquérir à l'apprenant tout au long de son apprentissage. En tant qu'évaluation de l'apprenant, il compare les solutions fournies par l'apprenant à celles d'un expert placé dans la même

situation. Ceci suppose donc qu'il doit être lui-même capable de résoudre les problèmes proposés à l'apprenant.

1.3. Le module pédagogique

Il contient l'ensemble des spécifications sur la manière dont le système doit construire ses interventions (Self, 1987). Il interagit avec l'apprenant plus ou moins directement en sélectionnant les problèmes qu'il doit résoudre, en le guidant vers la solution, en critiquant ses performances, en lui fournissant une aide appropriée lorsque l'apprenant le lui demande, en montrant des exemples. Il s'appuie également sur des connaissances relatives aux stratégies pédagogiques qui sont utilisées par les tuteurs humains, telles que l'apprentissage par l'exemple, l'apprentissage par analogie, etc. Selon Wenger (Wenger, 1987), il existe deux niveaux de prises de décisions : à un niveau global tout d'abord, le STI décide à partir de connaissances du domaine quelles activités sont proposées à l'apprenant, puis, lors de l'exécution de ces activités, il décide à un niveau local quelle est sa forme d'intervention avec l'apprenant. Cette intervention prend en compte le modèle de l'apprenant et des stratégies pédagogiques. Se posent alors plusieurs types de problèmes : le moment d'intervention, le type d'intervention, le type de dialogue. Le STI peut intervenir au cours de la résolution du problème proposé avant que l'apprenant n'ait achevé sa solution ou au moment de l'évaluation sous forme de conseils, d'explications ou de commentaires d'évaluation. Différentes formes de dialogue sont envisageables : le dialogue socratique consiste à poser des questions à l'apprenant, alors que le dialogue à initiative mixte permet à l'apprenant de poser des questions au tuteur, et vice-versa.

1.4. L'interface

Les interactions entre l'apprenant et le STI sont devenues dynamiques grâce à l'interface (Baker et Lund, 1997). Son rôle principal est de faciliter la communication des informations entre le système et l'apprenant. Bien qu'elle puisse paraître comme secondaire, elle joue un rôle important dans un STI. Elle doit offrir à l'apprenant des activités au travers desquelles il pourra acquérir ou consolider des concepts ou des démarches. Pour le STI, l'interface permet de présenter à l'apprenant des activités mettant en jeu les connaissances que l'on

souhaite lui faire acquérir et reflétant les actions et stratégies didactiques du module pédagogique. Elle constitue aussi le lieu à partir duquel l'apprenant fourni des informations au système à travers ses actions. Pour l'apprenant, l'interface est le support du problème à résoudre : c'est par elle, au travers de l'activité, qu'il construit ses représentations des concepts et procédures du domaine. La conception des interfaces requiert une analyse fine des activités proposées à l'apprenant, des interactions possibles et de leur évaluation.

1.5. Le modèle de l'apprenant

Pour permettre une adaptation dynamique et individualisée, un STI doit disposer d'un ensemble d'informations sur l'apprenant en situation d'apprentissage. Le modèle de l'apprenant constitue donc un ensemble d'informations relatives à l'état des connaissances exactes ou erronées, de l'apprenant. Sa construction s'établit principalement par comparaisons avec les connaissances de l'expert. Il sert ainsi à construire un diagnostic qui pourra servir au STI pour prendre une décision de nature didactique : questionnement, explications, remédiation, etc. (Self, 1992).

Dans les STI, l'état des connaissances de l'apprenant est généralement représenté comme un sous-ensemble des connaissances du module expert. Le modèle de l'apprenant est alors construit en comparant la performance de l'apprenant avec celle que l'expert aurait produite dans les mêmes circonstances (Goldstein, 1982).

Les informations du modèle de l'apprenant doivent être pertinentes et fiables pour le module pédagogique dans la conduite de l'apprentissage. Elles sont obtenues à partir de l'expertise du module expert et de l'interprétation des actions de l'apprenant. Ces informations sont de nature diverses : caractéristiques sur l'apprenant, connaissances maîtrisées, processus de résolution de problèmes, connaissances erronées, comportements face au problème posé ou à l'outil informatique. Ces informations contenues dans le modèle de l'apprenant sont par nature hypothétiques et donc sujettes à variation, d'abord parce que le modèle lui-même est une approximation des connaissances de l'apprenant, ensuite parce que l'apprenant en situation d'apprentissage a des comportements variables voire incohérents, enfin parce que le fait d'apprendre remet en cause les hypothèses précédemment établies. Dans un STI, le modèle de l'apprenant est donc central et

en relation avec chacun des trois autres modules.

2. Précisions sur le modèle de l'apprenant

Comme nous l'avons dit plus haut dans ce chapitre, les EIAH ou STI sont caractérisés par le fait de prendre en compte la connaissance de l'apprenant pour générer de nouvelles situations d'apprentissage, c'est-à-dire de posséder un modèle de l'apprenant. La modélisation de l'apprenant est donc le processus qui construit ce modèle à partir des interactions observées entre le système et l'apprenant.

Depuis longtemps, la modélisation de l'apprenant a posé de nombreuses questions tant sur sa faisabilité que sur son efficacité. Les années 80 ont été marquées par des divergences idéologiques sur la possibilité de la construction automatique d'un modèle informatique représentatif de l'état des connaissances de l'apprenant (Self, 1988), (Webb et al., 2001). De plus, Piaget a influencé la conception des systèmes qui sont devenus plus interactifs, et ainsi ont donné un rôle plus actif et autonome à l'apprenant. A cette époque, les chercheurs ont reconnu la très grande difficulté de cette modélisation. Pourtant, certains chercheurs dont John Self ont été les grands défenseurs du modèle de l'apprenant. Ce dernier a montré que la modélisation de l'apprenant ne s'opposait pas à l'approche constructiviste de l'apprentissage et qu'un modèle de l'apprenant relativement imprécis pouvait être utile au système (Self, 1988), (Akhras et Self, 1996). De nos jours, la modélisation de l'apprenant est largement étudiée.

2.1. Rôles du modèle de l'apprenant

Il existe plusieurs fonctions ou rôles que devrait remplir un modèle de l'apprenant. Plusieurs auteurs proposent une liste de fonctions pouvant être remplies par un modèle de l'apprenant. Nous présentons trois auteurs dans la suite de ce paragraphe.

Self (Self, 1987), (Self, 1994) a identifié vingt utilisations différentes d'un modèle de l'apprenant dans des STI existants. Il a retenu six fonctions principales qui dépendent des caractéristiques du modèle de l'apprenant et de l'existence d'un modèle du processus d'apprentissage :

1. *Fonction corrective* : le modèle de l'apprenant doit pouvoir aider à éliminer les connaissances erronées de l'élève.
2. *Fonction élaborative* : le modèle de l'apprenant doit permettre de compléter les connaissances de l'apprenant en choisissant le prochain sujet à aborder. Ce choix peut se faire par comparaison entre l'expert et l'apprenant, par analyse interne de la connaissance de l'apprenant ou laissé à l'apprenant parmi une liste d'exercices choisis en fonction de son modèle de l'apprenant.
3. *Fonction stratégique* : le modèle de l'apprenant est utilisé pour contrôler l'interaction et la stratégie d'enseignement suivie par le STI. Par exemple, le STI va l'utiliser pour choisir le plan de déroulement de la session ou bien encore le style d'interaction préférable avec cet apprenant.
4. *Fonction diagnostique* : le modèle de l'apprenant doit servir à la construction d'un diagnostic plus fiable et plus précis surtout dans le cas où plusieurs interprétations seraient possibles.
5. *Fonction prédictive* : le modèle de l'apprenant peut être utilisé par le STI pour prédire le comportement de l'apprenant face à un problème afin de limiter l'espace de recherche lors de la détermination du plan de la session. La prédiction peut porter sur la performance de l'apprenant ou sur les effets des actions didactiques pour choisir la meilleure et élaborer le nouveau modèle.
6. *Fonction évaluative* : le modèle de l'apprenant peut être utilisé pour mesurer l'efficacité potentielle d'un système en comparant différentes stratégies didactiques avec un apprenant simulé. Le contenu du modèle pout également servir à évaluer les performances de l'apprenant.

VanLehn (VanLehn, 1988), quant à lui, a proposé une autre classification qui contient quatre fonctions. Cette classification n'est pas loin de celle de Self. Pour lui, le modèle de l'apprenant doit permettre :

1. d'augmenter la connaissance de l'apprenant en passant au thème suivant après la maîtrise du thème en cours,
2. d'intervenir au moment des erreurs et offrir des conseils non sollicités,
3. de générer les problèmes de façon dynamique plutôt que seulement gérer l'enchaînement de problèmes prédéfinis,
4. d'individualiser les explications suivant le niveau de la connaissance de l'apprenant.

D'après Ragnemalm (Ragnemalm, 1996), il y a quatre utilisations du modèle de l'apprenant :

1. Importance du modèle pour la planification de l'enseignement : quels contenus doivent-ils être enseignés ?
2. Présentation du contenu d'enseignement : quelles expériences conviennent-elles en vue de l'apprentissage du contenu ?
3. La rétroaction du système doit prendre en compte les connaissances mobilisées précédemment par l'apprenant, aussi bien que le contexte d'apprentissage courant.
4. Le traitement des conceptions erronées : en les signalant à l'apprenant, en fournissant un contre-exemple ou en soulevant une discussion.

Les rôles attribués au modèle de l'apprenant sont assez ambitieux, et en fait aucune approche existante ne les prend totalement en compte. Pour remplir ces différents rôles, le contenu du modèle est décisif. Avant de passer au contenu de certains environnements, nous allons présenter les différents types de modèles de l'apprenant existants.

2.2. Types de modèles de l'apprenant

Il existe plusieurs types de modèles de l'apprenant (Holt et al., 1993), (Kass, 1987). Un modèle de l'apprenant est dit :

- *implicite*, lorsque les informations décrivant le comportement de l'apprenant et influençant le déroulement de l'interaction avec le système sont incorporées à ce dernier,
- *explicite*, lorsque les informations sur l'apprenant sont intégrées et codées dans le système de manière explicite dans le but de gérer l'interaction avec l'apprenant,
- *statique*, lorsque les connaissances de l'apprenant sont déterminées avant toute utilisation et ne peuvent être l'objet d'une modification en cours de session,
- *dynamique*, lorsque l'on peut ajouter ou modifier les données en cours de session,
- *spécifique*, lorsqu'il peut être adapté à une catégorie d'apprenants,
- *de surface*, lorsqu'il contient des informations limitées qui ne peuvent expliquer l'état cognitif de l'apprenant,

- *de profond*, lorsqu'il contient des informations plus représentatives de l'état cognitif de l'apprenant.

3. Contenu du modèle de l'apprenant

Nous explorons dans cette section différentes informations modélisées dans différents environnements d'apprentissage.

Comme nous l'avons déjà dit, ce modèle de l'apprenant représente ce que le système « sait » de l'apprenant. Ces informations peuvent être de nature cognitive, comportementale ou psychologique. Les tout premiers modèles portaient sur des aspects cognitifs et mettaient l'accent sur les connaissances déclaratives, procédurales et heuristiques. Plus récemment sont apparus des modèles visant à représenter des aspects psychologiques : émotions et motivations.

La modélisation de l'apprenant peut concerner un ou plusieurs aspects de l'apprenant : les concepts, règles ou procédures de résolution maitrisées, les conceptions erronées, la vitesse de résolution de problèmes, la motivation d'apprendre, la capacité de réflexion sur la connaissance apprise, les aspects métacognitifs, etc. Le choix du contenu va dépendre essentiellement du domaine d'enseignement, des objectifs didactiques et pédagogiques du système, des types possibles d'interactions avec l'apprenant, etc.

Nous présenterons ici succinctement quelques environnements et les types d'informations que ceux-ci recueillent. L'objectif est de présenter les différents types d'informations que l'on peut rencontrer dans des environnements d'apprentissage et ainsi de voir ce qu'un modèle de l'apprenant peut contenir.

3.1. L'activité de l'apprenant

L'environnement Essaim (Despres et Leroux, 2003) est un outil de suivi pédagogique synchrone d'activités à distance fournissant au tuteur humain des informations sur l'activité de l'apprenant lui permettant de prendre les décisions d'intervention opportunes.

Il permet de visualiser le parcours de l'apprenant, c'est-à-dire la nature et la durée des activités de celui-ci grâce à une représentation structurée de l'activité d'apprentissage. Le temps et le type de tâche constituent les deux dimensions de l'activité. Chaque séquence d'activités est représentée par un rectangle dont la largeur exprime la durée de la séquence et dont la coloration est d'autant plus sombre que le degré d'interaction est élevé.

3.2. Le comportement et les concepts associés

L'environnement Compounds (Danna, 1997) est un tuteur pour l'apprentissage des mots composés anglais pour les étudiants francophones. Il propose à l'apprenant un enchaînement d'exercices simples comportant une dizaine de questions. Les exercices consistent à produire un composé à partir d'une définition donnée ou bien à retrouver la définition à partir du composé.

Un modèle d'expertise de la connaissance experte a été élaboré à partir de travaux linguistiques sur les composés anglais. Le modèle de l'apprenant est dérivé de ce modèle d'expertise. Ce dernier est conçu à deux niveaux :

- *Le modèle comportemental* est constitué d'un ensemble d'entités comportementales. Chaque entité est formée d'un type de comportement, qui dénote l'association entre un type de composé et un type de définition, et d'un coefficient de certitude, qui indique la probabilité de voir l'apprenant manifester ce comportement.
- *Le modèle conceptuel* décrit le degré d'acquisition, par l'apprenant, des concepts nécessaires à la maîtrise du processus de composition. La hiérarchie de ces concepts est issue du modèle d'expertise. Le modèle de l'apprenant est un décalque de cette hiérarchie où chaque concept est étiqueté par des valeurs numériques indiquant la probabilité que l'apprenant maîtrise ou non le concept.

3.3. Les plans et les intentions

L'environnement Mentoniezh (Py, 2001) est un environnement pour l'apprentissage de la démonstration en géométrie au collège. L'apprenant doit construire une démonstration en articulant des pas de preuve. Ces pas de preuve

étant des triplets de la forme : hypothèses-théorème-conclusion. Un problème possède en général plusieurs solutions, c'est-à-dire que des preuves différentes peuvent conduire à la même conclusion finale.

La démonstration est vue comme l'exécution d'un plan, dont les actions élémentaires sont les pas de preuve. La mise en œuvre d'un ou plusieurs plans au cours de la démonstration traduit les intentions de l'apprenant.

Mentoniezh utilise une méthode de reconnaissance de plan pour détecter, parmi les démonstrations potentielles, celle qui est visée par l'apprenant, à partir des actions que celui-ci réalise. La base de plans de référence est constituée des démonstrations solutions pour l'exercice considéré, les plans incorrects n'étant pas représentés explicitement dans ce modèle.

3.4. Les heuristiques

L'environnement West (Burton et Brown, 1982) est l'un des premiers environnements modélisant ce type d'informations. Il combine l'entraînement au calcul arithmétique et l'apprentissage de stratégies pour un jeu à deux joueurs, de type « jeu de l'oie », dans lequel l'apprenant affronte l'ordinateur. Le jeu consiste à effectuer un parcours de soixante-dix cases numérotées. Chaque joueur à son tour tire au hasard trois chiffres, avec lesquels il compose une expression arithmétique, dont la valeur indique le nombre de cases dont il peut avancer. Des cases spéciales permettent de prendre des raccourcis ou de rejouer immédiatement.

Le modèle de l'apprenant dans West comprend les connaissances élémentaires nécessaires au jeu (maîtrise des opérateurs arithmétiques, des priorités) mais aussi les stratégies possibles, recueillies par observation de parties réelles. Quatre stratégies ont été repérées :

1. mettre le plus grand écart possible entre soi et l'adversaire,
2. produire le plus grand nombre possible à partir de trois chiffres,
3. utiliser le plus souvent possible les cases spéciales,
4. parcourir la plus grande distance possible.

West évalue la maîtrise de ces stratégies en comparant le comportement du joueur

à celui de l'expert artificiel. Chaque stratégie est affectée de quatre coefficients indiquant respectivement le nombre de situations où l'apprenant a employé ou non la stratégie à bon ou à mauvais escient.

3.5. Les motivations

L'environnement Moods (Vicente et Pain, 2002) est un environnement pour l'apprentissage des nombres en japonais par des élèves anglophones.

Le modèle de l'apprenant décrit les motivations de l'apprenant à deux niveaux :

1. *les traits*, qui expriment les caractéristiques permanentes,
2. *les états*, qui expriment des caractéristiques plus éphémères.

Quatre traits sont définis :

1. *le trait contrôle*, indique si l'apprenant aime avoir le contrôle sur la situation,
2. *le trait défi*, indique si l'apprenant apprécie d'être confronté à des situations de défi,
3. *le trait indépendance*, indique si l'apprenant aime travailler seul,
4. *le trait fantaisie*, indique si l'apprenant apprécie que l'environnement fasse appel à l'imagination.

Cinq états sont définis :

1. le degré de confiance en soi,
2. l'intérêt manifesté pour l'interface (sons, graphiques),
3. l'intérêt manifesté pour les aspects cognitifs de la tâche,
4. l'effort fourni,
5. la satisfaction ressentie dans l'accomplissement de la tâche.

Comme nous pouvons le remarquer sur les quelques exemples représentatifs cités précédemment, le modèle de l'apprenant peut contenir des informations très différentes en fonction de ce que l'on cherche à modéliser.

Le choix du projet AMICAL a été de prendre en compte tout ce qui devait être appris par l'apprenant pour savoir lire ainsi que tout ce qui pouvait avoir une influence sur l'apprentissage de celui-ci. Nous verrons plus en détails dans le

chapitre 2 ce que contient notre modèle mais nous avons distingué cinq catégories d'informations devant figurer dans notre modèle de l'apprenant : les connaissances du domaine, les aspects cognitifs, les aspects métacognitifs, le comportement et les informations générales sur l'enfant. L'environnement AMICAL a donc des objectifs ambitieux quant au contenu de ce modèle de l'apprenant.

Voyons à présent les différentes techniques permettant de représenter les connaissances figurant dans le modèle de l'apprenant.

4. Techniques de représentation de la connaissance

Parmi les aspects fondamentaux d'un STI se trouve la représentation des connaissances figurant dans ce système. En effet, les informations présentes dans le modèle de l'apprenant doivent être représentées formellement pour pouvoir être exploitées par le système informatique. Depuis le début des STI, on a vu apparaître de nombreuses techniques de représentation des connaissances. Il en existe de nombreuses pouvant être utilisées et nous en présentons les plus répandues dans cette section ainsi que les avantages et inconvénients de chacune.

Une technique de représentation des connaissances est une façon de représenter la connaissance dans un STI pour qu'elle soit utilisée pour inférer des nouvelles connaissances.

4.1. Les systèmes à base de règles

Un système à base de règles comporte trois parties :

1. une base de règles,
2. une base de faits,
3. un moteur d'inférences.

La base de règles contient l'ensemble des règles de production du système. Cette base représente la connaissance opératoire que l'expert a du domaine du problème. Le moteur d'inférences, quant à lui, contrôle l'activité du système et est souvent couplé à un module permettant d'expliquer le raisonnement du système. L'élément de base des systèmes à base de règles est la règle de production. Une

règle de production est de la forme : SI <condition> ALORS <action>.

Cette technique de représentation de la connaissance consiste à définir un ensemble de règles de production qui modélisent la connaissance considérée (Kass, 1987). Elle est utilisée dans un grand nombre de STI. Les systèmes à base de règles sont les plus aptes à modéliser des connaissances de type procédural (Haton et al., 1991). Le fameux système Mycin pour le diagnostic médical, les tuteurs basés sur la théorie ACT* (Anderson, 1983), le tuteur Buggy (Brown et Burton, 1978) sont des exemples de systèmes dont la connaissance est représentée sous forme de règles de production.

4.1.1. Avantages de cette technique

Les systèmes à base de règles permettent notamment d'exprimer facilement des processus de résolution de problèmes en termes de buts et de sous-buts qui sont atteints en réalisant des successions d'actions. Ils permettent ainsi en général de bien résoudre les problèmes de causalité ou de diagnostic. Ils offrent également un cadre déclaratif pour exprimer des connaissances procédurales ou des savoir-faire. La connaissance est exprimée de façon uniforme par des règles et des faits (Barr et Feigenbaum, 1981).

4.1.2. Inconvénients de cette technique

Dans ces systèmes à base de règles, la connaissance n'est pas structurée et l'information concernant un objet quelconque est éparpillée au sein des différentes règles qui parlent de cet objet. De plus, les règles expriment souvent une connaissance apparente, superficielle, susceptible d'occulter le raisonnement profond de l'expert humain (Haton et al., 1991).

4.2. Les réseaux sémantiques

Un réseau sémantique est un graphe composé d'un ensemble de nœuds qui représentent des concepts d'entité, attribut, objet, événement, état, etc. et d'un ensemble d'arcs orientés, liant deux nœuds, qui représentent des relations binaires

entre ces concepts et étiquetés où chaque étiquette spécifie le type de relation modélisée. Les réseaux sémantiques ont été mis en œuvre par Quillian (cité dans (Haton et al., 1991)) comme un modèle psychologique explicite de la mémoire associative humaine : la mémoire est vue comme un réseau d'unités d'informations. Ces unités étant activées par un mécanisme qui propage des signaux à travers le réseau, la « procédure d'activation ».

Cette représentation a été utilisée dans le cadre des tuteurs à leur tout début (Kass, 1987). Le système Scholar utilise un réseau sémantique pour représenter la connaissance factuelle de la géographie de l'Amérique du Sud (Nicaud et Vivet, 1988), (Kass, 1987).

4.2.1. Avantages de cette technique

Les réseaux sémantiques sont bien adaptés aux domaines où les concepts sont simples et fortement liés entre eux. La représentation à base de réseaux sémantiques rend visible les diverses relations existantes entre les objets. Sur cette représentation sous forme de graphe, le mécanisme de filtrage permet de récupérer des informations explicites ou implicites de la base de réseaux à la manière des associations mentales de l'être humain.

4.2.2. Inconvénients de cette technique

L'inconvénient majeur des réseaux sémantiques est que le concepteur du réseau a du mal à déterminer le niveau de détail et les structures générales nécessaires à la bonne expression d'une proposition. De plus, la seule information attachée à un nœud est son nom, complété par une étiquette. Cette simplification pose plusieurs problèmes. Le premier problème rencontré est qu'il est difficile d'exprimer des propositions ayant des quantificateurs universels, existentiels et numériques. Le deuxième problème est que les réseaux sémantiques offrent une représentation statique du monde, ce qui rend difficile la modélisation de l'évolution de l'information.

4.3. Les systèmes basés sur la logique

La logique mathématique figure parmi les premiers outils utilisés par l'intelligence artificielle pour formaliser la connaissance. La logique la plus utilisée est la logique des prédicats du premier ordre. La connaissance de l'apprenant est alors représentée sous la forme d'un ensemble de formules auxquelles sont associées des valeurs de vérité, ainsi qu'un mécanisme qui permet d'inférer de nouvelles formules à partir des connaissances possédées jusque-là.

On peut distinguer parmi les logiques utilisées dans les STI actuels :

- *La logique classique*. Ce formalisme a été utilisé dans certains systèmes pour représenter la connaissance de l'apprenant. Le conseiller en programmation Lisp, Scent (Brecht et al., 1989), est un exemple d'un tel système.
- *Les logiques multivaluées*. Ce sont des formalismes logiques qui utilisent plus de deux valeurs de vérité. Par exemple, on peut exprimer la notion d'inconnu lorsque l'apprenant ne croit rien à propos d'une proposition. SMIS (Student Modeling Inference System) (Ikeda et al., 1989) est un exemple de système qui utilise ce formalisme.
- *La logique probabiliste*. Ce formalisme utilise des valeurs de vérité qui peuvent s'échelonner sur l'ensemble des réels compris entre 0 et 1. La valeur de vérité associée à une formule représente la probabilité que cette formule soit vraie. Le tuteur West (Burton et Brown, 1982) est un exemple de ce formalisme.

4.3.1. Avantages de cette technique

Le cadre formel de la logique mathématique permet de manipuler directement certaines notions cognitives élémentaires en leur donnant un sens précis. La logique fournit un formalisme clair et non ambigu. Cette clarté vient, d'une part, du fait que la signification d'une formule ne dépend que de sa structure et de la signification donnée à ses composants atomiques et, d'autre part, du fait que le langage d'expression logique est proche du langage naturel.

4.3.2. Inconvénients de cette technique

La logique présente de gros inconvénients qui ont ralenti son utilisation dans les

STI. Le premier inconvénient, et non des moindres, est que la représentation d'objets complexes est difficile. En effet, un objet est décrit par son nom qui ne porte pas d'information sur son sens et sa structure. Les composants, propriétés et relations des objets sont, eux aussi, décrits par des noms. Ces éléments sont liés entre eux et avec l'objet par des formules logiques. Comme l'ensemble des formules de la base n'a pas de structure, la connaissance d'un objet est disséminée dans des formules différentes, non organisées. Il y a donc une grande distance entre le modèle et les éléments du monde, qui complique l'acquisition des connaissances, la compréhension et la modification de la représentation, et la vérification de sa cohérence. Le deuxième problème vient du fait que les mécanismes de raisonnement utilisent des algorithmes généraux qui ne sont pas toujours assez efficaces pour la résolution de problèmes nécessitant un grand volume de connaissances. Le dernier problème tient au fait que la logique des prédicats ne permette pas de représenter des connaissances incomplètes.

4.4. Les systèmes basés sur les réseaux de Bayes

La technique la plus en vogue à l'heure actuelle en modélisation de l'apprenant est sans conteste celle des réseaux de bayes, également appelés réseaux bayésiens ou réseaux de croyances. C'est pour cette raison que nous la présenterons un peu plus longuement que les autres. Ce sont des outils probabilistes qui se sont développés dans le cadre des recherches sur la prise en compte de l'incertain en intelligence artificielle.

Un réseau bayésien est un graphe orienté acyclique dont les nœuds sont des variables aléatoires (Pearl, 1988) et où les dépendances entre ces variables s'expriment au travers de la topologie du graphe.

Plus formellement, un réseau bayésien est un couple (G, V) où G est un graphe acyclique orienté, V est un ensemble $\{X_1,\ldots, X_n\}$ de variables aléatoires. La loi jointe de $X_1,\ldots, X_n$ se factorise en $P(X_1,\ldots, X_n) = \prod (P(X_i \mid pa(X_i))$ où $pa(X_i)$ est l'ensemble des parents de X_i dans le graphe. La détermination de la loi jointe ne nécessite par conséquent que la connaissance des probabilités à priori pour les nœuds n'ayant pas de parents et des probabilités conditionnelles des nœuds sachant leurs parents.

L'exemple classique généralement utilisé pour présenter les réseaux de bayes est celui de la pelouse de Sherlock Holmes (Becker et Naïm, 1999). Holmes quitte la maison et, passant entre son jardin et celui de son voisin Watson, constate que la pelouse est humide et se demande s'il a bien débranché son arroseur automatique.

Le réseau est constitué de quatre nœuds booléens :

- P (pluie) représente le fait qu'il ait plu la nuit précédente,
- A (arroseur) représente l'oubli de l'arroseur automatique,
- J (jardin) représente l'état du jardin d'Holmes,
- V (voisin) représente l'état du jardin du voisin, Watson.

On peut représenter la situation par le graphe suivant :

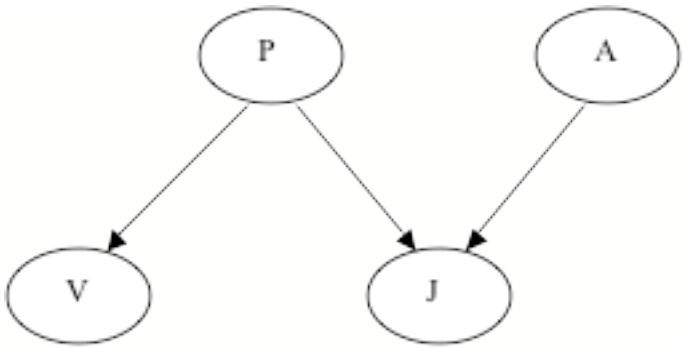

Figure 1.2 : Réseau bayes de l'exemple

Les arcs entre les variables s'interprètent en termes de causalité : la pluie est l'unique cause possible à l'humidité du jardin de Watson, alors que la pelouse d'Holmes peut avoir été arrosée soit par la pluie, soit par l'arroseur automatique.

On a donc P(A,P,J,V) = P(J|P,A) x P(V|P) x P(A) x P(P). On obtient les lois de probabilités : P(P = vrai) = 0,4 ; P(V = vrai) = 0,4 ; P(J = vrai) = 0,64 ; P(A = vrai) = 0,4. Nous ne rentrerons pas dans le détail des calculs, l'objectif étant simplement de faire une rapide présentation du concept.

Une observation dans un réseau bayésien est un événement du type $X_i = x_i$. Faire une inférence dans un réseau bayésien consiste à mettre à jour les probabilités dans le réseau après avoir fait une conjonction d'observations : sachant que la variable aléatoire (ou le vecteur de variables) E prend pour valeur e, on cherche à

déterminer la loi d'une ou plusieurs autres variables aléatoires, c'est-à-dire que l'on calcule P(X|E = e). Autrement dit, les croyances sont actualisées par les observations.

Revenons sur l'exemple d'Holmes. L'humidité du jardin de Watson laisse à penser qu'il a plu et donc qu'Holmes n'a pas de raison d'aller vérifier s'il a oublié d'éteindre son arroseur. Regardons sur le graphe. On peut mettre à jour les probabilités sachant que les pelouses des deux jardins sont humides : P(P = vrai) = 1 ; P(V = vrai) = 1 ; P(J = vrai) = 1 ; P(A = vrai) = 0,4. On constate que P(A = vrai | J = vrai, V = vrai) = 0,4 = P(A = vrai). On peut donc conclure qu'Holmes n'a pas de raison d'aller vérifier s'il a oublié d'éteindre son arroseur, puisque la probabilité qu'il ait fonctionné n'est pas plus importante que dans le cas général.

Cet exemple ne comporte que quatre nœuds et l'on peut donc envisager de faire les calculs à la main. Mais plus le graphe est grand, plus le problème de l'inférence devient grand. Différents algorithmes de propagation – que nous ne présentons pas ici car n'étant pas l'objet de cette thèse – ont été élaborés à cet effet. Ce problème d'inférences est NP-difficile.

Construire un réseau de Bayes suppose :

1. déterminer la structure du réseau,
2. définir les tables de probabilités.

Cette construction peut se faire par recueil d'expertise, par apprentissage automatique à partir de données mais également en combinant les deux approches. Dans le cas de la modélisation de l'apprenant, on détermine généralement la structure avec l'aide d'experts de terrain et on définit les tables de probabilités à partir de données recueillies en situation réelle.

Dans le cadre de la modélisation de l'apprenant, nous présentons deux environnements utilisants les résaux de bayes : Hydrive et Andes.

Hydrive (Mislevy et Gitomer, 1996) est un système pour l'apprentissage de la maintenance du système hydraulique du F15. Trois types de connaissances sont prises en compte dans le système :

1. les connaissances techniques en avionique,
2. les connaissances stratégiques,

3. les connaissances procédurales.

L'ensemble de ces connaissances est structuré sous forme d'un arbre à la racine duquel se trouve le nœud « compétence générale » et dont les trois types de connaissances mentionnées constituent les premières ramifications. Selon les réponses apportées par l'apprenant face à un problème concret, le système infère la probabilité que telle ou telle compétence ou connaissance soit acquise. La structure et les tables de probabilités ont été déterminées par recueil d'expertise, cependant les probabilités ont été affinées de manière à ce que les prédictions du réseau concordent avec le jugement humain. Le diagnostic qui est effectué est épistémique, le but étant d'inférer l'état des connaissances de l'apprenant à partir de ses actions face à un scénario de panne.

Andes (Conati et al., 2002) est un environnement concernant l'apprentissage de la résolution de problèmes de physique dans lequel il s'agit non seulement d'évaluer les compétences d'un apprenant, mais également de reconnaître le plan qu'il suit et de prédire ses actions à venir lors de résolution de problèmes. À chaque problème est associé un graphe solution qui contient la description de toutes les solutions acceptables. Celui-ci est obtenu à l'aide d'un résolveur à base de règles, qui est ensuite converti en réseau de bayes. Deux types de nœuds sont présents : les nœuds définis dans le domaine de la physique qui modélisent les connaissances en physique et les nœuds spécifiques à la tâche qui ne relèvent que du problème étudié. L'examen et la mise à jour des probabilités des nœuds relatifs au domaine permettent une évaluation épistémique de l'apprenant et celui des nœuds spécifiques à la tâche donne des indications quant au plan suivi et aux actions envisagées.

4.4.1. Avantages de cette technique

Le principal avantage des réseaux de bayes est de permettre de représenter facilement les liens de causalités au sein du modèle de l'apprenant. En effet, généralement, une information au sein du modèle n'est pas complètement isolée. Celle-ci est souvent liée à une autre, dans le sens où la connaissance d'une information influe sur la connaissance d'une autre information. Par exemple, la motivation de l'apprenant à apprendre influe directement sur sa capacité d'attention à effectuer une tâche. De plus, le modèle de l'apprenant étant par nature

hypothétique, l'utilisation des probabilités permet de traiter facilement et naturellement cet incertain.

4.4.2. Inconvénients de cette technique

Les principaux inconvénients de ces réseaux sont essentiellement liés à sa construction. En effet, comme nous l'avons dit, la structure du réseau est généralement élaborée avec des experts alors que les probabilités sont initialisées avec des données statistiques obtenues en situation réelle. Ceci nécessite donc, et ce n'est pas toujours le cas suivant les domaines, de disposer d'experts pouvant aider à déterminer la structure et de disposer de tests permettant de recueillir ces probabilités initiales.

La conclusion que l'on peut tirer de cette présentation des techniques de représentation de la connaissance d'un apprenant est que le type de représentation doit être choisi essentiellement en fonction de la nature des informations que l'on souhaite représenter. Par exemple, pour l'enseignement d'un domaine comme la soustraction, qui est typiquement considérée comme de la connaissance procédurale, la technique employée est très souvent celle des règles de production.

Néanmoins, aucune de ces techniques n'est réellement adaptée à nos problématiques. En effet, comme nous l'avons dit notre modèle de l'apprenant doit contenir des connaissances très variées. Celles-ci sont de fait de nature différentes et ne peuvent donc être représentées dans un formalisme unique. C'est pour cette raison, comme nous l'expliquerons au chapitre 3, que nous avons choisi de représenter le modèle de l'apprenant sous forme de système multi-agents où chaque agent du modèle sera en charge d'une partie de celui-ci. Ceci permet d'avoir une plus grande souplesse en terme de représentation des connaissances de l'apprenant.

Maintenant que nous avons exploré les différentes techniques de représentation des connaissances de l'apprenant, intéressons nous aux différentes techniques utilisées pour construire un modèle de l'apprenant.

5. Construction du modèle de l'apprenant

La construction du modèle de l'apprenant est la phase la plus importante dans le STI. En effet, le modèle de l'apprenant représentant la source de toutes les interactions du système avec l'apprenant, cette construction est donc, par nature, elle aussi, importante. Celle-ci passe par la phase de diagnostic. Il s'agit d'acquérir les valeurs du modèle et de les mettre à jour, pour un apprenant donné, au cours du temps, à partir des observables recueillis à l'interface.

5.1. Précisions concernant le diagnostic

Le processus de diagnostic est appelé ainsi par analogie avec les systèmes d'acquisition d'informations sur les systèmes mécaniques ou médicaux. Selon Wenger (Wenger, 1987), le diagnostic concerne non seulement l'acquisition d'informations sur l'apprenant mais aussi la synthèse de ces informations. Par diagnostic, nous entendrons donc à la fois l'acquisition de nouvelles hypothèses sur l'apprenant et la mise à jour du modèle en fonction de ces nouvelles hypothèses.

Les méthodes de diagnostic utilisées dans les STI actuels sont nombreuses et diffèrent selon le domaine d'application. Nous pouvons les classifier selon les critères suivants (Kass, 1987), (Wenger, 1987), (Paiva et John, 1994) :

- Les sources d'informations peuvent être :
  - *Implicites* : le diagnostic se base sur des observations, comme par exemple le cas du système ACM (Wenger, 1987).
  - *Explicites* : l'apprenant est considéré comme une source d'informations possible, comme par exemple les systèmes ACM (Sleeman et Brown, 1982), (Wenger, 1987), (Kass, 1987) et WHY (Wenger, 1987), (Self, 1987).
- Les données fournies au diagnostiqueur peuvent être :
  - *Incrémentales* : le système affine son modèle de l'apprenant au fur et à mesure des interactions, en analysant le comportement de celui-ci.
  - *Globales* : le système attend d'avoir les réponses à tous les exercices avant de commencer son diagnostic.
- Le diagnostic peut être :
  - *Actif* : des exercices ou des tests peuvent être élaborés spécifiquement pour constituer le modèle de l'apprenant ou pour lever des ambiguïtés : le diagnostic reste invisible pour l'apprenant mais influe sur le

comportement du système.
    - *Passif* : le diagnostic n'interfère pas avec l'interaction : il est totalement invisible pour l'apprenant.
    - *Interactif* : le système pose des questions directes à l'apprenant pour renseigner le modèle : le diagnostic est alors perceptible par l'utilisateur
- Le moment de l'acquisition peut être :
    - *En ligne* : le diagnostic est réalisé au moment où l'élève utilise le système.
    - *Hors ligne* : le diagnostic est réalisé une fois que l'apprenant a fourni ses réponses au système.

5.2. Problèmes de diagnostic

Trois problèmes se posent lors de la détermination d'un système de diagnostic des réponses de l'apprenant (Haton et al., 1991) :

1. L'identification des connaissances qui expliquent le comportement de l'apprenant. Ces informations sur l'état cognitif de l'apprenant concernent le fait qu'il possède des connaissances et que ces connaissances soient celles d'un expert ou pas.
2. L'identification des erreurs qui peuvent être commises par l'apprenant. Le problème est alors d'arriver à élaborer un système de diagnostic qui permette d'acquérir des informations fiables à partir de réponses incorrectes, et ce pour le plus grand nombre d'erreurs possibles. Ces informations doivent provenir d'un diagnostic profond en plus de celui de surface, elles devraient renseigner sur les causes des erreurs.
3. L'explosion combinatoire qui peut résulter de la procédure de diagnostic. Le problème survient notamment lorsque, dans le cas de l'utilisation d'un catalogue d'erreurs, le diagnostiqueur cherche une combinaison d'un certain nombre d'erreurs simultanées qui puisse expliquer le comportement de l'apprenant.

5.3. Cadre théorique du diagnostic

Une formalisation logique du diagnostic a été proposée dans (Cialdea, 1992).

Diagnostiquer, c'est effectuer un métaraisonnement nécessitant à la fois de savoir résoudre des problèmes sur le domaine considéré, de savoir comparer la solution correcte à celle de l'apprenant et de savoir engendrer des hypothèses sur les conceptions incorrectes qui expliquent les réponses de l'apprenant.

Le modèle formel de diagnostic repose sur trois structures :

1. le modèle de l'apprenant S,
2. le modèle de l'expert E,
3. le catalogue de bugs B.

Ce sont des théories du premier ordre, associées à une collection d'axiomes et à un système logique permettant d'effectuer des déductions. La formule centrale du modèle indique que la méta-théorie MT permet de dériver une formule α (la réponse de l'apprenant) d'un ensemble de formules Σ issues des ensembles de connaissances correctes et incorrectes. Cet ensemble Σ représente les connaissances mobilisées par l'élève pour produire sa réponse :

$$MT \rightarrow derivable_C(\Sigma,\alpha)$$

où C vaut soit E si la réponse est correcte, soit E ∪ B si la réponse est incorrecte, α est une formule (la réponse de l'apprenant), et Σ est un ensemble fini de formules.

Expliquer la réponse de l'apprenant, c'est donc construire une ou plusieurs dérivations, les comparer et retenir la meilleure selon un critère donné.

Ce cadre logique peut être vu comme une formalisation abstraite des diagnostiqueurs implantés dans la plupart des systèmes.

5.4. Méthodes de diagnostic

Les premiers systèmes ont utilisé des méthodes de diagnostic relativement simples, le plus souvent basées sur la comparaison entre l'apprenant et le module de l'expert. Les limites inhérentes à ce type de méthodes ont conduit les chercheurs à s'intéresser aux techniques d'inférences issues de l'intelligence artificielle et à les transposer pour les appliquer au problème du diagnostic (Py, 1998). Ces travaux ont permis de s'affranchir de la nécessité de disposer d'un

modèle de l'expert, notamment grâce à l'apprentissage automatique et de traiter les informations contradictoires ou non monotones par des approches logiques et numériques.

VanLehn (VanLehn, 1988) a proposé une classification des différentes techniques de diagnostic. Ces méthodes diffèrent selon le type de la connaissance que l'on cherche à modéliser. Nous distinguerons ici trois types de connaissances : les connaissances procédurales, les connaissances conceptuelles et les processus de résolution de problèmes de l'apprenant. Nous présentons dans la suite quelques méthodes les plus couramment utilisées dans chacun des cas.

5.4.1. Connaissances procédurales

Les connaissances procédurales spécifient les structures de contrôle directement utilisables dans la réalisation de l'action. Elles renvoient aux capacités perceptivo-cognitives et perceptivo-motrices. Elles ne sont pas ou très difficilement communicables. Elles reposent sur des systèmes d'association plus ou moins complexes entre des stimuli, des comportements et des états mentaux. Elles sont fortement automatisées.

On peut distinguer deux techniques de diagnostic pour ce type de connaissances de l'apprenant.

5.4.1.1. Diagnostic par traçage de modèle

La méthode de diagnostic par traçage de modèle ou model tracing a été mise au point par Anderson et son équipe lors de l'élaboration des systèmes basés sur la méthode ACT (Anderson, 1983). L'idée de base consiste à utiliser le modèle de performance pour suivre l'état de la solution d'un apprenant à l'intérieur d'une tâche et à utiliser le modèle d'apprentissage pour déterminer l'état des connaissances de celui-ci au fur et à mesure qu'il accomplit les différentes tâches.

Dans le traçage de modèle, on distingue les connaissances que l'apprenant manifeste pendant qu'il résout un problème, le modèle de performance, de celles qu'il possède véritablement, le modèle d'apprentissage, en se référant aux

différentes règles de production représentant les connaissances du domaine. Dans le modèle de performance, on observe le comportement de l'apprenant en situation de résolution de problèmes et on tente d'identifier un ensemble de règles ou mal-règles du système qui pourrait expliquer un tel comportement. Cette identification permet alors de suivre les états cognitifs de l'apprenant en temps réel.

Le principal avantage de cette technique de diagnostic, d'un point de vue pratique, réside dans la simplicité de sa mise en oeuvre (Wenger, 1987). De plus, d'un point de vue pédagogique, le fait que la correction des erreurs soit effectuée dès que celles-ci surviennent est souhaitable selon Anderson.

La principale limite est qu'elle nécessite un catalogue de mal-règles afin de reconnaître les étapes incorrectes de l'apprenant. Ceci oblige, pour que le système soit efficace, à disposer d'un catalogue exhaustif de ces mal-règles.

5.4.1.2. Diagnostic par induction

Dans cette technique, la connaissance sous-jacente au comportement correct ou non de l'apprenant est représentée en termes de conditions sous lesquelles l'apprenant utilise les différentes entités de connaissances primitives. Le diagnostic consiste donc à induire, pour chaque primitive, l'espace des problèmes pour lesquels cet opérateur est appliqué par l'apprenant. Un espace vide signifie que l'opérateur n'est pas utilisé dans les solutions de l'apprenant.

Le processus de diagnostic est composé ici de trois étapes :

1. donner un ensemble de problèmes à résoudre à l'apprenant et récolter les solutions proposées par celui-ci,
2. identifier les séquences d'opérateurs,
3. inférer les conditions d'application des opérateurs apparaissant dans les séquences telles que l'utilisation de tous les opérateurs applicables à un problème permette de retrouver la solution qui a été construite par l'apprenant.

On considère un ensemble de réponses données par l'apprenant (les exemples) et on cherche à en inférer les règles abstraites modélisant son comportement. Les règles qui le composent ne sont pas prédéfinies : elles sont construites à partir de

primitives (des briques de base) neutres vis-à-vis de la correction, dont la combinaison produit des règles correctes ou incorrectes.

Un exemple représentatif de cette approche est le système ACM (Automated Cognitive Modeling) (Ohlsson et Langley, 1987) qui porte sur le domaine de la soustraction. Un problème de soustraction est vu comme un ensemble de chiffres organisés en colonnes et en rangées (haut, milieu, résultat). Langley et Ohlsson font l'hypothèse que les procédures de l'apprenant peuvent être représentées par des règles de production. Ils définissent un ensemble d'opérations primitives sur la soustraction, ainsi qu'un ensemble de conditions primitives.

Les opérateurs élémentaires sont par exemple :

- ajouter10(N, C, R) : ajoute 10 au nombre N situé au rang R, colonne C.
- soustraire(N1, N2, C) : calcule la différence entre les nombres N1 et N2 en colonne C et écrit ce chiffre comme résultat pour la colonne C.
- reportSommet(C) : reporte le chiffre en haut de la colonne C comme résultat pour cette colonne.

ACM cherche à inférer les conditions des règles appliquées par l'apprenant qui effectue une soustraction. La résolution d'un problème de soustraction est considérée comme un parcours dans un espace de recherche, dont les arcs sont des actions primitives et les nœuds des points de croix entre différentes actions possibles.

Pour analyser la réponse d'un apprenant, ACM reconstruit le chemin solution, c'est-à-dire la séquence d'actions exécutées par l'apprenant. Ce chemin est obtenu par une exploration de l'espace de recherche guidée par des heuristiques donnant la priorité au chemin le plus court et le plus proche du chemin solution de l'expert.

Ensuite, il s'agit, à partir du chemin solution, de déterminer les conditions d'application des opérateurs qui ont été utilisés par l'apprenant. Pour cela, ACM parcourt à nouveau le chemin solution, en déterminant à chaque point de croix tous les opérateurs applicables. L'opérateur effectivement appliqué par l'apprenant constitue un exemple positif, les autres constituent des exemples négatifs. Un mécanisme de discrimination permet alors d'inférer les conditions d'application des opérateurs : ce sont les conditions qui écartent les exemples négatifs et retiennent l'exemple positif. Le résultat final est un ensemble de règles qui

exprime le comportement de l'apprenant.

5.4.2. Connaissances conceptuelles

Les connaissances conceptuelles représentent les connaissances de l'apprenant sur des concepts. Là encore, il existe également différentes techniques. Nous en présentons ici rapidement deux.

5.4.2.1. Diagnostic orienté contraintes

Ohlsson (Ohlsson, 1992) propose une technique de diagnostic, dite orientée contraintes, qui permet d'obtenir des informations de type crédit/débit. L'idée de cette technique est que les concepteurs du STI doivent énoncer les évènements à mémoriser sous la forme de contraintes. Ces dernières forment un sous-ensemble plus ou moins grand des concepts du domaine enseigné. Le diagnostic est alors l'ensemble de ces exigences qui sont respectées par la réponse de l'apprenant en ce qui concerne le crédit, le débit étant l'ensemble des contraintes qui ne sont pas respectées.

Une application de cette technique de diagnostic est présentée de façon théorique dans (Ohlsson, 1992) pour le domaine de la soustraction multi-colonnes, par exemple, certaines contraintes concernent la gestion de la retenue.

5.4.2.2. diagnostic dit générer-tester

Cette technique consiste à générer un ensemble de candidats pour ensuite les tester un par un afin de déterminer celui ou ceux qui conviennent.

Dans le cadre de la modélisation de l'apprenant, l'algorithme de diagnostic basé sur cette méthode est le suivant :

1. générer un ensemble de diagnostics candidats à partir d'heuristiques de diagnostic généralement ad-hoc,
2. calculer les modèles de l'apprenant correspondant à ces diagnostics,
3. déterminer les réponses que chaque modèle prédit,
4. comparer les prédictions avec la réponse réelle de l'apprenant.

Lorsque l'on souhaite utiliser cette technique, il est nécessaire de déterminer les heuristiques dépendantes du domaine afin d'augmenter la rapidité du diagnostic, sous peine d'être peu efficace à cause de l'explosion combinatoire inhérente à la phase de génération (Wenger, 1987).

5.4.3. Processus de résolution de problèmes

On cherche ici à savoir quel cheminement l'apprenant a suivi pour résoudre le problème qui lui a été proposé. Plusieurs méthodes d'identification du processus de résolution de problèmes ont été élaborées. Nous distinguerons entre autres trois méthodes : l'arbre de décision, la recherche par chemin et la reconnaissance de plan.

5.4.3.1. Arbre de décision

Cette méthode consiste à construire l'espace des processus de résolution de problèmes possibles sous forme d'un arbre dont les feuilles sont toutes les réponses imaginables. Cette technique est donc applicable dans le cas où seule l'étape finale est visible par le système. Elle permet un diagnostic relativement rapide, surtout dans le cas de combinaisons d'erreurs puisque ces combinaisons sont calculées hors ligne. Cependant, l'explosion combinatoire de cette technique reste un inconvénient dans le cas où le diagnostiqueur détermine plusieurs chemins possibles.

5.4.3.2. Recherche du chemin

Cette méthode de diagnostic consiste à élaborer un algorithme qui permette de retrouver en ligne les étapes intermédiaires effectuées par l'apprenant, en d'autres termes le chemin que l'apprenant a emprunté pour construire sa réponse. L'algorithme poursuit itérativement jusqu'à ce qu'une étape corresponde à la solution de l'apprenant ou jusqu'à ce que le problème soit retrouvé. Contrairement à la technique de diagnostic par arbre de décision, cet algorithme ne construit pas tous les chemins possibles qui aboutissent à toutes les réponses diagnostiquables

par le système.

5.4.3.3. Reconnaissance de plan

C'est une méthode très utilisée en modélisation de l'apprenant. Cette méthode consiste à déterminer le plan qui sous-tend les actions effectuées par l'apprenant. Une fois déterminé, le plan permet d'inférer les étapes intermédiaires qui n'ont pas été observées. En d'autres termes, il s'agit de réaliser l'opération réciproque de la planification. Cette technique de diagnostic est plus efficace que celle dite de recherche de chemin lorsqu'il manque beaucoup d'étapes mentales entre deux étapes observées, puisqu'elle permet de retrouver les étapes possibles en se basant sur les buts intermédiaires que l'apprenant a dû se fixer.

En intelligence artificielle, un plan est considéré comme une séquence ordonnée d'actions qui vise à satisfaire un but donné. Chaque action est définie pas ses préconditions et ses effets. La reconnaissance de plans est le processus qui consiste à découvrir, à partir de l'observation de quelques actions, le plan mis en œuvre par l'apprenant.

En modélisation de l'apprenant, la technique de reconnaissance de plans peut être utilisée quand la résolution du problème ne se ramène pas à l'exécution d'un algorithme, comme en arithmétique, mais requiert une démarche heuristique, comme en algèbre et géométrie. Pour ce type de problèmes, il existe en général plusieurs solutions d'efficacité variable. Ces solutions sont considérées comme des plans, et le diagnostiqueur cherche à reconnaître le plan exécuté par l'apprenant à partir des actions observées. Ceci permet de prévoir les prochaines actions, ou bien d'inférer des étapes intermédiaires non observées. La reconnaissance de plans suppose de connaître les plans possibles : c'est une contrainte forte, dans la mesure où leur nombre est souvent élevé. Pour contourner cet obstacle, on peut utiliser une bibliothèque de plans abstraits, qui seront spécialisés, ou bien générer les plans à la demande. La reconnaissance de plans permet d'élaborer un modèle local à un exercice, mais elle ne permet pas de produire un modèle à long terme.

PHI est un système de reconnaissance de plan, basé sur un formalisme logique, destiné aux environnements d'aide intelligents (Bauer et al., 1993). Il a été

appliqué dans le cadre d'un logiciel conseillé pour les utilisateurs du système Unix. PHI utilise une logique modale et temporelle d'intervalles qui permet d'exprimer les plans de l'utilisateur comme des formules logiques. Les spécifications des plans sont des formules du type : [préconditions et Plan] → buts.

La métavariable Plan est remplacée par une formule (le plan) qui vérifie la spécification, en utilisant une base de connaissances propre au domaine. Par exemple, la formule suivante spécifie le plan « afficher le contenu de la boîte aux lettres M et lire le message x » :

(ouverte(M) = vrai et Plan) → ◊ [afficher = en-tête(M)] et ◊ [lire(x,M) = vrai]

La reconnaissance de plan est réalisée par un processus abductif associé à une évaluation probabiliste des hypothèses. Soit T les connaissances du domaine et a l'action observée, le système détermine les plans P tels que : T ∪ {P} → a. Pour déterminer le « meilleur » parmi ces plans, le système exploite des connaissances sur les préférences de l'utilisateur, exprimées sous forme de probabilités associées aux actions et aux buts. À partir d'une distribution de probabilités propre au domaine, le système établit une hiérarchie des hypothèses tel que le plan le plus plausible figure au sommet.

La problématique de l'environnement AMICAL étant de chercher à modéliser des informations de différentes natures, aucune de ces méthodes ne correspond exactement à notre objectif. En effet, pour modéliser les différentes informations de notre modèle, nous avons des activités d'apprentissage différentes et donc la méthode de diagnostic devra donc être réfléchie au sein de chaque activité et non de manière globale. Là encore, pour résoudre ce problème, nous avons choisi de charger un agent particulier d'élaborer le diagnostic pour une activité donnée. Ce qui fait que nous aurons autant d'agents qu'il y aura d'activités différentes au sein de notre environnement. Nous y reviendrons dans le chapitre 3 où nous parlerons de l'architecture de notre agent gestionnaire du modèle de l'apprenant. Dans la suite de cette thèse, nous emploierons le terme d'analyse plutôt que celui de diagnostic.

6. Conclusion

Comme nous l'avons vu au cours de ce chapitre, tout STI individualise en élaborant pour chaque apprenant un modèle de cet apprenant. Cela lui permet par la suite d'adapter son enseignement à cet apprenant. Chaque STI ou EIAH se distingue essentiellement par le nombre et le type de connaissances prises en compte : certains ne prenant en compte que des connaissances du domaine alors que d'autres prennent en compte des aspects motivationnels. De plus, ils se concentrent tous sur une petite partie d'un domaine mais ne s'intéressent jamais à celui-ci dans son ensemble. Contrairement à ces systèmes, le projet AMICAL se propose de prendre en compte le domaine de l'apprentissage de la lecture dans son ensemble et de considérer au sein du modèle de l'apprenant l'ensemble des informations possibles ayant une influence pour l'apprentissage. Nous y avons distinguer cinq catégories à faire figurer dans notre modèle : connaissances du domaine, aspects cognitifs, aspects métacognitifs, comportement et informations générales sur l'apprenant.

Nous avons également parlé des différentes méthodes de représentation des connaissances de ce modèle. Nous avons choisi de « déplacer » le problème de représentation par le fait de représenter notre modèle de l'apprenant lui-même comme un système multi-agents. Nous parlerons ici d'agents de connaissance. Le problème revient alors à nous interroger sur la représentation des connaissances de chaque agent. Nous y reviendrons aux chapitre 3 et 4.

Nous avons également exploré plusieurs types de méthodes de diagnostic et nous avons vu que celles-ci dépendaient du type de connaissance que l'on cherchait à modéliser. Malheureusement, du fait que nous souhaitons prendre en compte l'ensemble des aspects possibles sur l'apprenant et que ceux-ci sont de natures différentes, aucune des méthodes présentées ne répond à notre attente. De ce fait, nous avons choisi de concevoir un agent par type d'activité d'apprentissage et de laisser cet agent s'occuper du diagnostic de l'activité. Nous parlerons ici d'agents d'analyse. Nous le verrons plus loin mais nous décomposons le diagnostic en deux étapes : l'analyse de chaque activité d'apprentissage et la mise à jour du modèle de l'apprenant. C'est pour cette raison que nous parlons d'agents d'analyse car ceux-ci ont en charge l'analyse des activités. Chaque agent d'analyse aura pour rôle d'émettre toutes les hypothèses possibles sur l'apprenant concernant une activité. Nous y reviendrons aux chapitre 3 et 4.

Dans le chapitre suivant, nous explorerons plus en détail la modélisation de

l'apprenant dans AMICAL mais sous un angle théorique uniquement, c'est-à-dire sans parler d'aspects formels ou d'agents. Nous y détaillerons le contenu de notre modèle de l'apprenant et ses cinq catégories ainsi que la méthode générale de construction de notre modèle au sein de notre environnement.

CHAPITRE 2 : MODÉLISATION DE L'APPRENANT DANS AMICAL : POINT DE VUE THÉORIQUE

Comme nous l'avons vu dans le chapitre 1, il est nécessaire au système de disposer d'un modèle de l'apprenant pour conduire un enseignement individualisé et donc pour s'adapter à chaque apprenant. Ce modèle de l'apprenant contient ce que le système « sait » de cet apprenant. L'objectif de ce chapitre est de présenter la modélisation de l'apprenant dans l'environnement AMICAL mais uniquement d'un point de vue théorique, c'est-à-dire en expliquant le fonctionnement général du système. Nous parlerons du contenu du modèle de l'apprenant ainsi que de son élaboration au sein du système : comment et à partir de quoi le système l'élabore.

Pour déterminer le contenu du modèle de l'apprenant ainsi que les éléments permettant l'élaboration de celui-ci, nous avons dû faire de nombreuses réunions avec les experts en apprentissage de la lecture en faisant de l'ingénierie des connaissances. Néanmoins, nous n'avons employé aucune des méthodes existantes telles que CommonKADS (Schreiber et al., 1999), MKSN (Ermine, 2000) ou encore Macao (Aussenac, 1989) car aucune ne correspondait réellement à nos attentes essentiellement pour des raisons de souplesse. Nous nous sommes donc limités à de nombreuses discussions avec les experts.

Le rôle de l'environnement AMICAL, d'un point de vue de l'apprenant, est d'apprendre à lire. Néanmoins, d'un point de vue du système, le rôle de celui-ci est de modéliser l'apprenant en construisant le modèle celui-ci au fil des interactions (Self, 1987). Les interactions entre l'environnement et l'apprenant passent par des interfaces informatiques que nous appelons situations didactiques. Nous traiterons largement dans ce chapitre de ces situations didactiques, lieu primordial de la modélisation de l'apprenant au sein de notre système.

Nous rappelons également que le public visé par notre environnement AMICAL concerne les enfants au début du cours préparatoire (CP), c'est-à-dire le moment de l'école où les enfants commencent à apprendre à lire. On suppose également que ces enfants n'ont aucun déficit cognitif particulier.

Les trois questions auxquelles nous répondrons au cours de ce chapitre sont d'ordre théorique, c'est-à-dire sans référence à la représentation des connaissances ni à la formalisation des processus de construction. Celles-ci sont les suivantes : « Que contient le modèle de l'apprenant ? », « Comment détermine-t-on les valeurs initiales de celui-ci ? » et enfin « Comment construit-on ce modèle et à partir de quoi ? ». Ces trois questions reprennent dans les grandes lignes les trois étapes de l'élaboration d'un modèle de l'apprenant que nous avons définies dans le premier chapitre. Pour répondre à la première question, nous présenterons ce qu'est l'apprentissage de la lecture puis nous en déduirons le contenu de notre modèle de l'apprenant. Ensuite, pour répondre à la deuxième question, nous présenterons la batterie de tests réalisée pour recueillir les informations initiales du modèle et ainsi permettre au système de commencer son enseignement individualisé. Enfin, nous répondrons à la troisième question en expliquant comment, d'un point de vue théorique uniquement, l'environnement AMICAL acquiert des nouvelles informations sur l'apprenant, à partir de quoi et comment il les intègre aux anciennes informations contenues dans le modèle de l'apprenant.

1. Etude du domaine d'application de l'environnement AMICAL

Nous commençons par définir dans cette section le domaine dans lequel se situe l'environnement AMICAL : la lecture, son apprentissage et son enseignement. Cette partie est une rapide synthèse à la fois des ouvrages de références en apprentissage de la lecture parmi lesquels (ONL, 2005), (Fayol et al., 1992) (Gombert et al., 2000), (Ecaille et Magnan, 2002), (Goigoux et Cèbe, 2006) et des nombreuses discussions avec les experts en apprentissage de la lecture. Nous ne rentrons bien évidemment pas dans le détail de l'apprentissage de la lecture car ceci serait en dehors du champ de cette thèse.

1.1. La lecture

La lecture peut se définir comme un processus interactif dans lequel le lecteur utilise les ressources dont il dispose pour construire le sens de l'écrit à partir d'un stimulus visuel ; stimulus étant sous forme de graphies, c'est-à-dire de traits, de courbes ainsi que leurs orientations (Rumelhart, 1985).

On distingue six types de ressources :

1. *Les ressources perceptives* : elles concernent la vision et permettent à un apprenant de discriminer ce qu'il voit.
2. *Les ressources linguistiques* : elles sont relatives à la morphologie, la phonétique, la syntaxe et la sémantique.
3. *Les ressources métalinguistiques* : elles relèvent de la catégorisation du langage et incluent par exemple les catégories syntaxiques (nom, verbe, etc.).
4. *Les ressources cognitives* : elles concernent les capacités de résolution de problèmes. Ce sont soit des capacités générales de comparaison, de discrimination ou de mémorisation, soit des capacités relatives à la lecture comme la compréhension du fonctionnement du système écrit et de ses relations à l'oral.
5. *Les ressources métacognitives* : elles renvoient à la conduite de la construction du sens par le lecteur, à ses capacités à utiliser des stratégies et à les évaluer selon son objectif de lecture.
6. *Les ressources affectives* : elles comprennent les attitudes générales du comportement du lecteur, ses motivations ou ses attentes.

On lit donc pour comprendre. La compréhension n'est toutefois pas spécifique à la lecture. Elle préexiste à l'apprentissage de la lecture et s'exerce à la fois au cours et en dehors de cet apprentissage.

Le processus de lecture se déroule en deux étapes :

1. d'abord, l'identification des mots écrits, c'est-à-dire accéder au sens de chaque mot,
2. puis, la mise en œuvre de l'activité de compréhension à partir de cette identification, c'est-à-dire construire le sens du texte à partir du sens des différents mots.

Le problème auquel se trouve confronté l'enfant qui entre à l'école élémentaire sera donc d'apprendre ou plutôt de commencer à apprendre ces deux étapes. La

lecture est donc un processus complexe mettant en œuvre des savoirs et des savoir-faire de natures diverses et variées. C'est cet ensemble de savoirs et de savoir-faire que l'apprenant doit acquérir.

1.2. L'apprentissage de la lecture

L'apprentissage de la lecture se greffe sur des habiletés cognitives, sociales et linguistiques qui se sont développées depuis le plus jeune âge. La plus importante de ces habiletés est le langage, qui fournit la base de la lecture. Bien avant le début de l'enseignement de la lecture, l'enfant a acquis, à des degrés divers, les différentes dimensions du langage qui lui permettent de comprendre et de produire des énoncés oraux. En apprenant à lire, il transfèrera tout ou partie de ces compétences à l'écrit.

Les connaissances dont dispose l'enfant à son arrivée à l'école sont en général suffisantes pour lui permettre l'apprentissage de la lecture. Il existe cependant déjà d'importantes différences interindividuelles qui tiennent, pour certaines, à la diversité des rythmes de développement et, pour d'autres, à des acquis culturels différents. Sur ce point, les différences sont particulièrement fortes dans le domaine du vocabulaire.

Comme le processus de lecture comporte deux étapes, l'apprentissage de la lecture se doit donc de tenir compte de ceux-ci. Pour cela, l'enfant devra apprendre à :

- identifier les mots écrits,
- utiliser tous les éléments permettant de développer sa compréhension de l'écrit.

Nous présentons rapidement chacun de ces deux aspects.

1.2.1. L'identification de mots

Apprendre à lire, c'est donc d'abord apprendre à identifier les mots écrits.

Les systèmes alphabétiques mettent en correspondance des unités graphiques, les graphèmes (26 lettres – a, b, c,...- ou blocs de lettres – ou, eau,...) avec les unités

abstraites de la langue orale, les phonèmes. Les phonèmes sont les éléments constitutifs de la parole qui permettent des distinctions sémantiques (par exemple, les mots parlés " gâteau " et " château " diffèrent entre eux par le phonème initial). Ces correspondances représentent ce que l'on appelle les correspondances ou associations graphèmes-phonèmes.

Lorsqu'un lecteur rencontre un mot écrit, deux cas sont possibles.

Soit le mot est déjà connu car rencontré au cours de lectures antérieures ; il peut alors être reconnu. Dans ce cas, le lecteur dispose d'un "dictionnaire" des formes écrites auxquelles il peut apparier les mots rencontrés : c'est le lexique orthographique. La reconnaissance est « automatique » et mobilise peu d'attention. Les processus qui interviennent dans la reconnaissance sont cependant très complexes.

Soit le mot est nouveau quant à sa forme ; sa lecture nécessite alors une habileté de décodage intentionnel. Dans ce cas, l'identification du mot mobilise des ressources attentionnelles, opère de gauche à droite et est donc plus lent. Le décodage comprend une décomposition en segments de taille variable selon l'expertise du lecteur. Ces segments sont associés à des représentations phonologiques qui sont ensuite fusionnées pour aboutir à la reconnaissance de la forme orale du mot.

L'identification de mots repose donc soit sur le décodage si le mot est inconnu, soit sur le lexique si le mot est connu. L'apprentissage de cette capacité doit donc porter sur les deux aspects.

Sans rentrer dans le détail, le décodage repose sur la connaissance des correspondances graphèmes-phonèmes. Or, pour comprendre comment fonctionnent ces associations graphèmes-phonèmes, les élèves doivent préalablement avoir pris conscience que la parole peut être segmentée en unités (mots, syllabes, phonèmes) et que les plus petites de ces unités (phonèmes) ont pour contrepartie des lettres ou des groupes de lettres (les graphèmes). Il doivent également comprendre le fonctionnement de l'écrit, c'est-à-dire savoir qu'on lit de gauche à droite, que l'écrit code l'oral, connaître les différents concepts relevant de l'écrit (texte, mot, phrase ou lettre), connaître différents objets tels que les lettres, etc.

Le décodage est "attentionnellement" coûteux lorsqu'il n'est pas automatisé, il implique un effort d'attention trop important pour que le lecteur puisse réserver une part suffisante de son activité mentale pour la compréhension. La conséquence en est la nécessité d'automatiser le plus vite possible l'identification des mots par la constitution d'un lexique orthographique mental (mise en mémoire des formes écrites des mots). Lorsqu'une activité est automatisée, sa mobilisation et sa mise en œuvre sont rapides et peu coûteuses. Il est possible de mener en parallèle deux activités automatisées, une activité automatisée et une autre qui ne l'est pas, mais pas, ou très difficilement, deux activités coûteuses. De là l'importance d'une maîtrise et d'un traitement efficace du lexique orthographique et la nécessité de l'étendre tout au long de la scolarité. La constitution de ce lexique orthographique se fait progressivement. Le décodage de mots nouveaux développe la connaissance de la diversité des correspondances entre graphèmes et phonèmes dans leurs environnements respectifs. La rencontre fréquente des mots induit leur mémorisation et la constitution du lexique mental orthographique. L'automatisation de la reconnaissance des mots se fait d'autant mieux et d'autant plus vite qu'ils ont été rencontrés souvent.

1.2.2. La compréhension

Apprendre à décoder constitue un savoir faire nécessaire, mais il ne suffit pas ; le but étant d'accéder au sens.

L'activité de compréhension consiste en la construction mentale d'une représentation de ce qui est écrit. Elle nécessite de l'attention et souvent un effort important pour coordonner les différents types d'informations et les intégrer en une représentation cohérente. Elle fait appel à des capacités de traitement du lexique, de la syntaxe de phrase et de la syntaxe des textes. Elle mobilise l'ensemble des connaissances du lecteur, à partir desquelles il peut reconstituer l'implicite du texte par des inférences. C'est cet ensemble de capacités que l'enfant doit acquérir ou développer au cours de son apprentissage de la lecture.

1.2.3. Apprentissage long et en parallèle

Une remarque importante quant à l'apprentissage de la lecture est qu'il s'agit d'un apprentissage qui s'effectue sur un temps très long durant toute la formation scolaire et même au-delà de celle-ci. Le travail sur la compréhension commence en CP sur les différents types de textes (récit, dialogue, description) et se poursuit durant toute la formation jusqu'à l'étude de textes littéraires.

Les deux composantes de l'apprentissage que sont l'identification de mots et la compréhension doivent être travaillées en parallèle. De plus, à l'intérieur de chaque composante également, on doit travailler les différents aspects en parallèle. Par exemple, on doit travailler le fonctionnement de l'écrit en même temps que l'enfant travaille sur les correspondances graphèmes-phonèmes.

Après avoir présenté ce que lire et apprendre à lire signifiaient, nous allons voir ce que contient notre modèle de l'apprenant.

2. Contenu de notre modèle de l'apprenant

Pour répondre à la première question concernant les informations qui devaient être modélisées par le système, nous avons donc étudié ce qu'était la lecture et son apprentissage. En collaboration avec les experts en apprentissage de la lecture, nous avons convenu de faire figurer dans notre modèle à la fois tout ce qui devaient être acquis par l'apprenant, c'est-à-dire les connaissances du domaine et tout ce qui pouvait avoir un impact sur l'apprentissage de cet apprenant.

Ainsi, à partir de l'analyse du domaine de l'apprentissage de la lecture et de l'apprentissage en général, nous avons déterminé les cinq sections suivantes devant figurer dans notre modèle :

1. Connaissances du domaine
2. Aspects cognitifs
3. Aspect méta-cognitifs
4. Comportement
5. Informations générales sur l'apprenant

Nous commençons par présenter les connaissances du domaine. Celles-ci sont les connaissances fondamentales de l'environnement AMICAL car ce sont elles qui déterminent le « niveau de connaissance » de l'apprenant par rapport aux

connaissances qu'il est censé acquérir pour savoir lire.

2.1. Les connaissances du domaine

Ces connaissances permettent de dire là où en est l'apprenant concernant son apprentissage de la lecture et c'est pour cela que nous trouvons dans cette catégories toutes les connaissances relatives à l'apprentissage de la lecture. Elles permettent au système de savoir quel est le degré de connaissance de l'apprenant d'une connaissance qu'il est censé acquérir.

Comme nous l'avons dit plus haut, la lecture peut être décomposée en deux parties : l'identification de mots et la compréhension. Nous retrouvons donc ces deux aspects au sein de cette partie.

2.1.1. Généralités sur ces connaissances

Une connaissance du domaine représente le degré de connaissance de l'apprenant sur la connaissance considérée. Par exemple, la connaissance du domaine Objet-Lettre(a) représente le degré de connaissance de l'apprenant sur l'Objet-Lettre(a).

Au lieu de considérer uniquement son degré de connaissance globale d'une connaissance devant être acquise, nous cherchons à modéliser avec le plus de finesse possible les différentes informations que l'on peut avoir sur celle-ci. Nous avons donc, pour une connaissance devant être acquise, plusieurs niveaux d'information permettant au système d'adapter son enseignement avec la plus grande finesse possible. Par exemple, au lieu d'avoir uniquement son degré de connaissance de la lettre *a*, nous disposons de son degré de connaissance sur chaque détail de celle-ci. Nous cherchons donc à déterminer son degré de connaissance du nom de cette lettre, du son lui correspondant et des différentes graphies associées à celle-ci.

Pour cette raison, les connaissances du domaine sont donc des objets complexes que l'on peut représenter par le diagramme UML suivant :

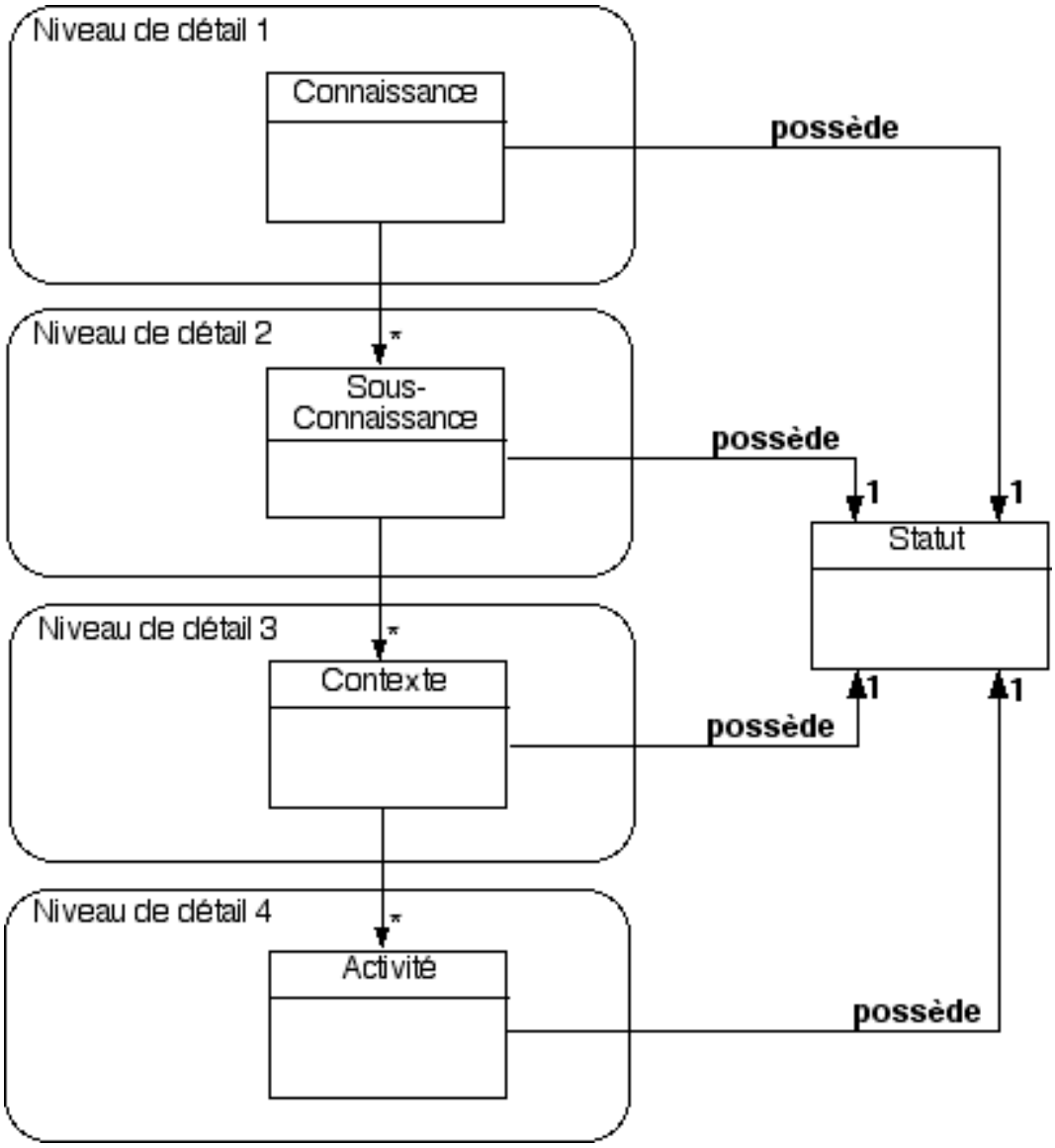

Figure 2.1 : Niveaux de détail d'une connaissance du domaine

Chaque connaissance du domaine est composée de sous-connaissances. Celles-ci représentent le détail de chaque connaissance devant être acquis par l'apprenant. Par exemple, pour un objet lettre, nous avons cinq sous-connaissances que l'apprenant doit acquérir : le nom, le son, la graphie majuscule, la graphie minuscule et la graphie script. Ces sous-connaissances permettent donc au système de connaître finement le degré de connaissance sur chaque détail de la connaissance considérée. Chaque sous-connaissance doit être acquise dans une série de contextes pour être considérée comme acquise. Pour l'objet lettre, les contextes associés à chaque sous-connaissance sont : le mot, l'alphabet, la liste

ordonnée, la liste non-ordonnée, le nuage de lettres. Par exemple, le système peut savoir que le nom de la lettre *a* est acquise dans un mot mais pas dans l'alphabet et ainsi lui présenter une activité où l'apprenant devra par exemple trouver une lettre dans l'alphabet. Enfin, le dernier niveau de détail correspond à celui des activités face auxquelles l'apprenant est placé. Chaque contexte est lui-même associé à différentes activités d'apprentissage. Par exemple, pour la lettre, le contexte de mot peut être associé à plusieurs activités : une activité dans laquelle on lui demande de reconnaître une lettre dans un mot, une autre dans laquelle on lui demande de cliquer sur une lettre qu'il connait d'un mot, etc.

Comme nous le voyons sur la Figure 2.1, nous avons donc plusieurs niveaux d'information sur chaque connaissance du domaine. Chaque niveau de détail possède ce nous appelons un statut. Celui-ci indique son degré de connaissance qui est traduit par une valeur symbolique. Par exemple, un statut « acquis » associé à la sous-connaissance *nom* de la lettre *a*, indiquera que cette sous-connaissance est acquise, alors qu'un statut « en cours d'élaboration » associé à la connaissance du domaine de concept de mot indiquera que celui-ci est en cours d'acquisition. Chaque niveau de détail possède son propre ensemble de statuts. Ces statuts évoluent bien entendu au cours des interactions avec l'apprenant. Par exemple, avant qu'une connaissance ne soit acquise, elle passera par différents statuts différents. Une évolution précise de chaque statut de chaque niveau à été élaboré avec nos experts.

Chaque niveau de détail inférieur influe sur le niveau de détail supérieur. La modification d'un ou de plusieurs statuts de niveau de détail inférieur peut conduire à la modification du statut de niveau de détail supérieur. Par exemple, l'acquisition de toutes les sous-connaissances nous indiquera que la connaissance est acquise et nous devrons donc modifier le statut de la connaissance en conséquence, c'est-à-dire le passer à « acquis », nous signalant que cette connaissance est acquise. Ces modifications sont réalisées par ce que nous appelons des règles de synthèse développées elles aussi en collaboration avec les experts. Nous en présenterons quelques unes au chapitre 4 quand nous parlerons de la formalisation des différentes composantes de notre agent.

Signalons enfin que chaque connaissance du domaine possède son propre ensemble de sous-connaissances, de contextes, d'activités et de statuts. Néanmoins, pour ne pas alourdir l'exposé, nous ne rentrerons pas plus dans le

détail.

Nous terminerons par présenter l'exemple de la connaissance lettre et de ses différents niveaux de détail (Figure 2.2).

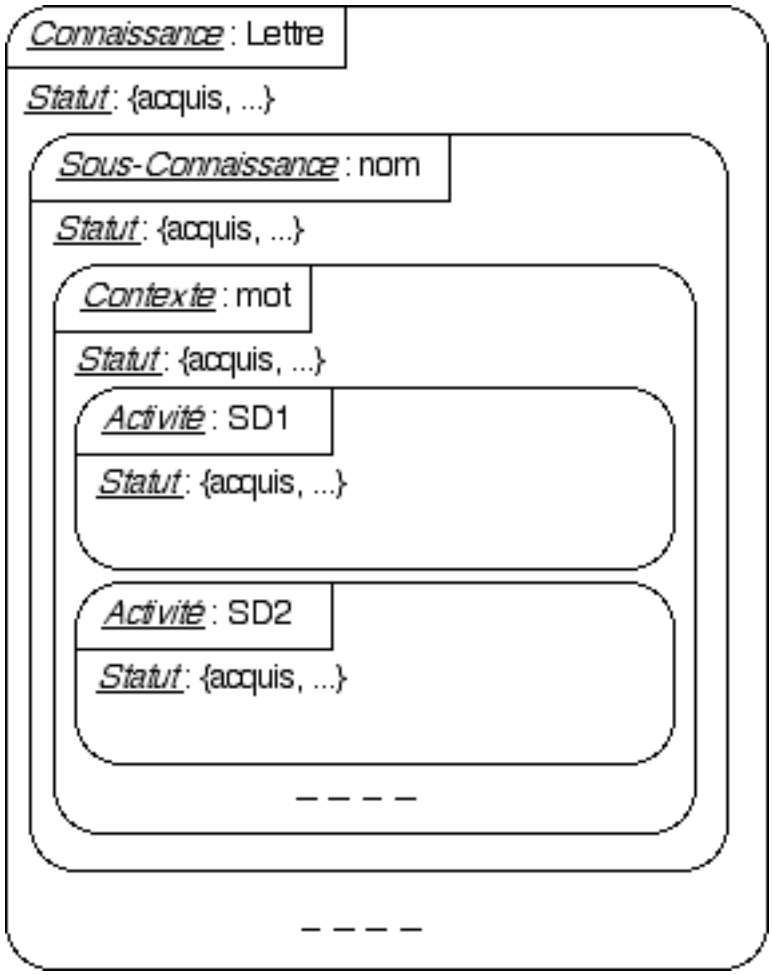

Figure 2.2 : Niveaux de détail de la connaissance objet lettre

Il convient juste ici de remarquer la complexité des différentes connaissances modélisées par notre environnement.

2.1.2. Types de connaissances

Nous distinguons trois types de connaissances du domaine :

1. *les concepts* : il s'agit d'une représentation générale et abstraite d'une réalité ou d'un objet. Ils traduisent en quelque sorte la connaissance de la définition de quelque chose. Les concepts permettent donc de répondre à la question : « Est-ce que l'apprenant sait ce qu'est X ? ». Par exemple, si l'on prend le concept de voiture, le concept nous informerait sur la connaissance de l'apprenant de la définition d'une voiture : le fait qu'il sache qu'elle ait quatre roues, un volant, qu'elle serve à se déplacer d'un point A à un point B, etc.
2. *les objets* : ils représentent les différentes entités concrètes qui existent. Toujours en reprenant l'exemple de la voiture ci-dessus, l'objet correspondrait à des voitures particulières. L'objet nous informerait donc de la connaissance de l'apprenant d'une voiture particulière, par exemple en connaissant ses caractéristiques.
3. *les stratégies* : elles représentent la manière qu'à eu l'apprenant de résoudre un problème particulier.

Comme nous le verrons dans le chapitre 3, ces types de connaissances deviendront par la suite des agents de connaissance.

Après avoir détaillé ce qu'étaient les connaissances du domaine et après avoir donné un exemple détaillé de l'une d'entre elles, nous listons dans la suite les différentes connaissances du domaine présentes dans le modèle et nous donnons quelques explications sur celles-ci.

2.1.3. Concepts

Le premier type de connaissances du domaine correspond aux concepts que l'on rencontre en apprentissage de la lecture. Nous en distinguons essentiellement sept.

Le fonctionnement de l'écrit représente essentiellement la connaissance de l'apprenant de la linéarité de l'écrit (gauche-droite, haut-bas), du fait que l'oral code l'écrit, etc.

Le concept de lettre représente la connaissance de l'apprenant de la définition d'une lettre. Ce concept permet de répondre à la question : « Est-ce qu'il sait ce qu'est une lettre ? ». Il recoupe entre autre sa capacité à discriminer une lettre avec

un chiffre, à discriminer une lettre avec un signe, à discriminer une lettre avec une autre lettre proche, à savoir qu'une lettre à plusieurs graphies, etc.

Le concept de mot représente la connaissance de l'apprenant de la définition d'un mot. Ce concept permet de répondre à la question : « Est-ce qu'il sait ce qu'est un mot ? ». Il recoupe entre autre sa capacité à discriminer un mot avec nombre, à discriminer un mot avec une chaine de caractères, à discriminer un mot avec un pseudo-mot (un mot qui ressemble à un mot mais qui n'existe pas dans la langue), à discriminer un mot avec une image, à savoir qu'un mot est composé de lettres, à savoir qu'un mot est composé de graphèmes, à savoir qu'un mot est composé de syllabes, etc.

Le concept de texte représente la connaissance de l'apprenant de la définition d'un texte. Ce concept permet de répondre à la question : « Est-ce qu'il sait ce qu'est un texte ? ». Il recoupe entre autre sa capacité à discriminer un texte avec une phrase, à discriminer un texte d'un mot, à savoir qu'un texte est un ensemble de phrases, etc.

Le concept de phrase représente la connaissance de l'apprenant de la définition d'une phrase. Ce concept permet de répondre à la question : « Est-ce qu'il sait ce qu'est une phrase ? ». Il recoupe entre autre sa capacité à discriminer une phrase avec un texte, à discriminer une phrase avec un nombre, à discriminer une phrase avec une chaine de caractères, à discriminer une phrase avec un mot, à savoir qu'une phrase est composée de mots, à savoir qu'elle commence par une majuscule et se termine par un point, etc.

Le concept de syllabe représente la connaissance de l'apprenant de la définition d'une syllabe. Ce concept répond à la question : « Est-ce qu'il sait ce qu'est une syllabe ? ». Il recoupe entre autre sa capacité à discriminer auditivement une syllabe d'une autre syllabe, etc.

Le concept d'alphabet représente la connaissance de l'apprenant de la définition de l'alphabet. Ce concept permet de répondre à la question : « Est-ce qu'il sait ce qu'est l'alphabet ? ». Il recoupe entre autre sa capacité à savoir que l'alphabet est une suite de toutes les lettres de la langue, à savoir que l'alphabet a un ordre précis, etc.

Parmi les statuts que l'on peut trouver pour ces concepts, on peut citer : « en cours d'élaboration », « acquis », ...

2.1.4. Objets

Le deuxième type de connaissances du domaine correspond aux objets que l'on rencontre en apprentissage de la lecture. Nous en distinguons essentiellement quatre.

Les lettres sont bien évidemment présentes au sein de notre modèle ; l'objectif étant bien entendu pour l'apprenant de les acquérir toutes. Pour chaque lettre, nous recueillons la connaissance de l'apprenant du nom, du son et des différentes graphies.

Les mots rencontrés par l'apprenant ainsi que leur degré de connaissance sont présents au sein du modèle. Ils permettent de représenter le lexique que l'apprenant est en train de développer.

Les textes rencontrés par l'apprenant au cours des sessions sont également présents. Ceux-ci permettent de savoir quels types de textes ont été travaillés ainsi que leur degré de difficulté et la compréhension de ceux-ci par l'apprenant.

Les associations graphèmes-phonèmes sont elles-aussi présentes au sein du modèle. Elles permettent de savoir où l'apprenant en est par rapport au décodage.

Pour ces objets, on peut trouver les statuts : « non acquis », « acquis », « vu », « travaillé », « reconnu », ...

2.1.5. Stratégies

Les stratégies que nous rencontrons sont de deux types : les stratégies concernant la compréhension et les stratégies concernant l'identification de mot. Pour chaque stratégie, on notera son utilisation ou non ainsi que le contexte dans lequel elle a été mise en oeuvre. Nous nous contentons de donner le nom des différentes stratégies sans en donner la signification pour ne pas obscurcir l'exposé.

Les cinq stratégies de compréhension sont : la stratégie sémantique, la stratégie pragmatique, la stratégie syntaxique, la stratégie élaboration de propositions, la stratégie combinaison-intégration de propositions.

Les huit stratégies d'identification de mots sont : la stratégie comparaison-discrimination, la stratégie contextuelle, la stratégie logographique, la stratégie repérage graphique, la stratégie repérage phonologique, la stratégie alphabétique, la stratégie orthographique, la stratégie analogie.

On peut trouver les statuts : « utilisé », « maitrisé », ...

C'est sur ces connaissances du domaine qu'a porté exclusivement nos recherches en terme de modélisation de l'apprenant. Les autres aspects sont donc présentés à titre d'information.

Contrairement aux connaissances du domaine qui représentent en quelque sorte « son niveau de connaissance de l'apprentissage de la lecture », les quatre dernières catégories de notre modèle de l'apprenant représentent des informations « utilitaires ». Ces informations permettent également au système d'adapter son enseignement mais cette fois davantage en terme de pédagogie, c'est-à-dire de façon d'enseigner.

2.2. Aspects cognitifs

Dans cette section, nous intégrons les capacités cognitives générales de l'apprenant. On distingue :

1. *sa capacité de mémorisation* : il s'agit ici de considérer le niveau de mémorisation de l'apprenant d'une manière globale, c'est-à-dire de répondre à la question : l'apprenant mémorise-t-il en général bien ce qu'il lit ?
2. *Sa capacité d'inférences* : il s'agit ici de considérer si l'apprenant est capable de faire des inférences à partir des textes qui lui sont présentés. Par exemple, si l'apprenant est capable, alors que la réponse n'était pas présente directement dans le texte, de répondre à une question du type : « Que n'y avait-il pas dans le cartable ? »

3. *sa capacité de compréhension* : il s'agit ici de considérer le niveau de compréhension générale de l'apprenant, c'est-à-dire de répondre à la question : l'apprenant comprend-t-il en général bien les textes qu'on lui présente ?

Chaque capacité peut avoir les statuts suivants : très basse, basse, moyenne, haute et très haute.

2.3. Aspects méta-cognitifs

Dans cette section, nous intégrons les aspects méta-cognitifs, c'est-à-dire tout ce qui permet à l'apprenant d'auto-conduire son apprentissage, c'est-à-dire de le conduire en partie tout seul. On distingue :

1. *sa capacité d'auto-contrôle* : il s'agit ici de considérer si l'apprenant est capable ou non de contrôler son apprentissage, par exemple en faisant plusieurs tentatives avant de valider sa réponse.
2. *Sa capacité d'auto-évaluation* : il s'agit ici de considérer si l'apprenant est capable ou non d'évaluer ses réponses, ses performances ou ses stratégies, par exemple en choisissant et en utilisant l'aide proposée par le système au bon moment.
3. *sa capacité d'auto-correction* : il s'agit ici de considérer si l'apprenant est capable ou non de voir ou de corriger ses propres erreurs après les avoir commises.
4. *sa capacité d'auto-régulation* : il s'agit ici de considérer si l'apprenant est capable ou non de réguler son apprentissage, par exemple en demandant la relecture de la consigne de l'activité proposée par le système ou bien en demandant des explications sur les fonctions présentes à l'écran.

Chaque capacité peut avoir les statuts suivants : très basse, basse, moyenne, haute et très haute.

2.4. Aspects comportementaux

Dans cette section, nous intégrons tout ce qui concerne le comportement de l'apprenant. Le terme de comportement est ici à prendre sous un angle

psychologique. Nous distinguons ici :

1. *sa motivation à apprendre* : il s'agit ici de considérer le niveau de motivation qu'a l'apprenant à apprendre. Plus l'apprenant sera motivé, plus celui-ci sera capable d'apprendre facilement et dans de bonnes conditions.
2. *son degré d'attention* : il s'agit ici de considérer si l'apprenant est attentif ou pas. Cela permettrait par exemple de diminuer la durée de la session de travail.
3. *son état émotionnel* : il s'agit de considérer l'état psychologique dans lequel se trouve l'enfant. On sait également qu'en fonction de cet état, l'apprenant sera plus où moins bien disposé à apprendre.

Pour les points 1 et 2, on trouve les statuts suivants : très basse, basse, moyenne, haute et très haute.

Chacun de ces comportements évolue bien évidemment en fonction du temps, si bien que pour prendre en compte ces aspects, il faudrait sans cesse mettre à jour ces valeurs pour que le système soit efficace.

2.5. Informations générales sur l'apprenant

Dans cette partie, nous intégrons les données relatives à l'apprenant et qui peuvent avoir une influence sur sa capacité d'apprendre. On distingue :

1. *le nom* : celui-ci est d'abord présent pour mémoriser le modèle de l'apprenant ainsi que les différents comptes-rendus relatifs à cet apprenant. Il permet également au système de commencer éventuellement par faire acquérir les lettres du prénom de celui-ci pour commencer son apprentissage des lettres.
2. *le sexe* : le système peut éventuellement différencier le choix du thème des textes présentés à l'apprenant en fonction du sexe. Par exemple, pour une fille, on pourra présenter une histoire de poupées alors que pour un garçon on lui présentera un texte sur les robots. Il existe également des différences en terme d'attention entre les deux sexes que le système pourrait exploiter.
3. *s'il est redoublant ou non* : si l'apprenant redouble, peut-être qu'on ne travaillera pas certaines connaissances
4. *les préférences de celui-ci* : pour choisir le texte on peut également le faire en fonction des préférences du thème, on peut également personnaliser

l'interface en fonction des couleurs préférées de l'apprenant, éventuellement pour accroitre sa motivation.

5. *son style d'apprentissage* : il existe plusieurs styles d'apprentissage que nous ne traiterons pas. L'idée serait bien entendu d'adapter la façon d'enseigner du système en fonction de ce style d'apprentissage.
6. *la langue parlée à la maison* : si les parents de cet apprenant ne parlent pas français à la maison, alors l'apprenant aura plus de difficultés que les autres. Cela nécessitera un traitement spécifique de la part du système pour cet apprenant là précisément.
7. *le milieu social de la famille* : suivant le milieu social, les parents possèdent un vocabulaire plus ou moins riche, et donc l'enfant également. Là encore le système devra tenter de développer davantage son vocabulaire au début de l'apprentissage.

Ces informations sont remplies par un tuteur humain et ne sont pas acquises par le système sauf en ce qui concerne le style d'apprentissage. Néanmoins, il n'existe pour l'instant aucune activité permettant de déterminer celui-ci en raison toujours de travaux toujours en cours de la part des experts de l'apprentissage de la lecture.

Dans la suite, quand nous parlerons d'état de savoir de l'apprenant, nous ferons référence aux connaissances du domaine. Parfois, nous parlerons d'état global de l'apprenant pour signifier que nous nous intéressons à toutes les informations de notre modèle. De plus, comme nous l'avons déjà dit, nous nous limiterons uniquement aux connaissances du domaine car ce sont celles sur lesquelles les travaux sont les plus aboutis et sur lesquelles les activités existantes portent.

3. Initialisation de notre modèle de l'apprenant

Pour répondre à la deuxième question que nous posions en introduction de ce chapitre en ce qui concerne l'initialisation de notre modèle de l'apprenant, nous présentons maintenant la batterie de tests réalisée à cet effet. L'initialisation du modèle est une phase importante pour que le système puisse commencer à travailler avec l'apprenant de manière individualisée. Pour initialiser le modèle de l'apprenant de l'environnement AMICAL, nous avons choisi de réaliser une série de tests permettant au système de déterminer certains aspects importants pour pouvoir commencer les acquisitions de manière individualisées.

3.1. Objectifs de la batterie de tests

Pour une plus grande fiabilité, nous avons choisi de partir d'un modèle vide. En effet, au lieu d'avoir des valeurs préétablies pas toujours très justes, il nous a paru plus sûr de concevoir une batterie de tests permettant d'initialiser notre modèle de l'apprenant avec des informations particulières à un apprenant et non avec des informations standards, identiques pour tous les enfants.

3.2. Informations recueillies au cours des tests

Les informations recueillies par la batterie de tests portent essentiellement sur les connaissances du domaine. En effet, au tout début des interactions avec l'apprenant, il est très difficile voire même impossible de déterminer d'autres informations que celles-ci. La batterie de tests permet de déterminer les informations suivante sur l'apprenant :

- sa connaissance du fonctionnement de l'écrit et des différents concepts de la lecture
  - s'il sait discriminer une lettre d'un chiffre, d'un signe
  - s'il sait ce qu'est une majuscule et une minuscule
  - s'il sait discriminer un mot d'un nombre, d'un signe et d'une image.
  - s'il sait discriminer une phrase d'un nombre, d'une chaine de caractères et d'un mot.
  - s'il sait qu'un mot est composé de lettres et trouver la position d'une lettre dans ce mot
  - s'il sait compter le nombre de mot dans une phrase
  - s'il sait discriminer un texte d'un mot et d'une phrase
  - s'il arrive à compter le nombre de syllabes dans un mot
- sa compréhension de l'oral à travers une histoire lue à laquelle on lui demande d'associer une image
- les lettres qu'il connait
- quelques mots qu'il connait

Ces informations vont donc permettre au système de pouvoir commencer à travailler de manière individualisée avec l'apprenant. Sans rentrer davantage dans le détail, nous présentons par la suite quelques écrans de celle-ci.

3.3. Exemples d'écrans

Nous présentons quelques écrans représentatifs de cette batterie de tests.

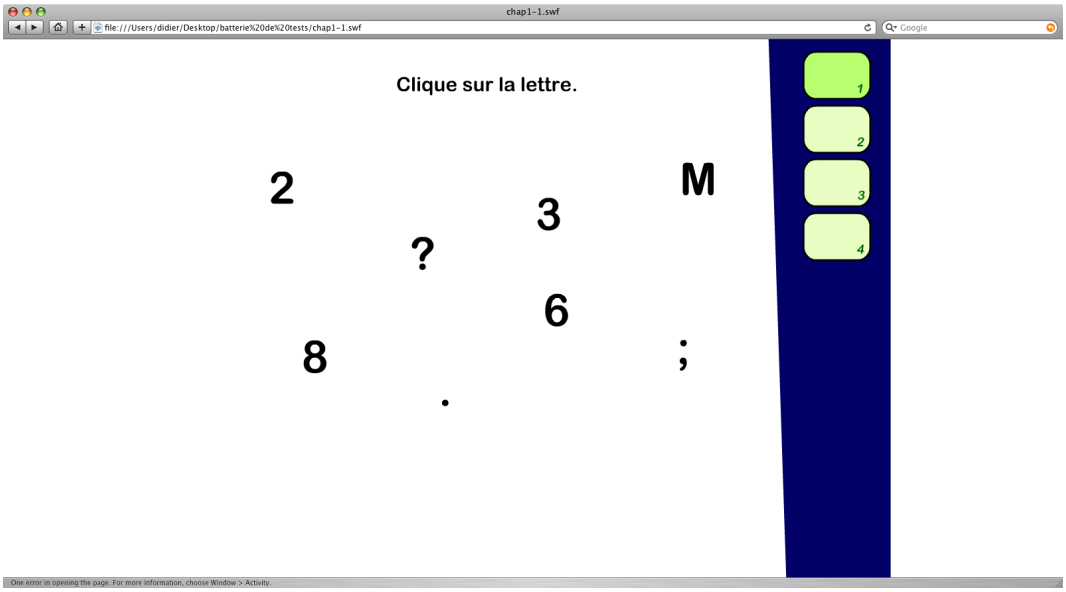

Figure 2.3 : Discrimination d'une lettre avec d'autres formes

Cet écran permet au système de déterminer si l'apprenant sait discriminer une lettre d'un signe de ponctuation et d'un chiffre. On renseigne ici le concept de lettre.

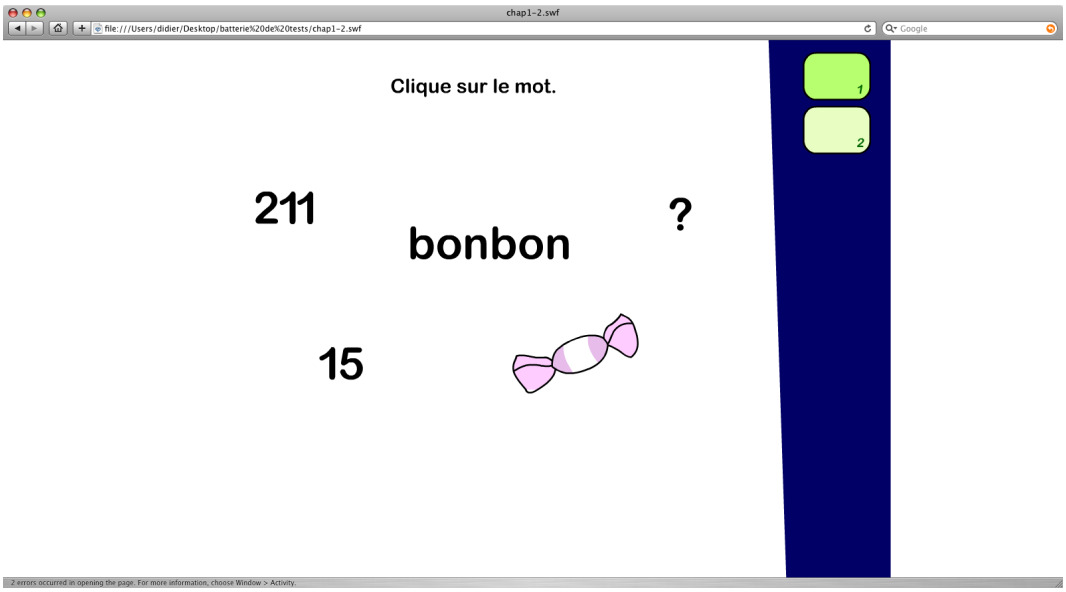

Figure 2.4 : Discrimination d'un mot avec d'autres formes

Cet écran permet au système de déterminer si l'apprenant sait discriminer un mot d'un dessin, d'un signe de ponctuation et d'un nombre. On renseigne ici le concept de mot.

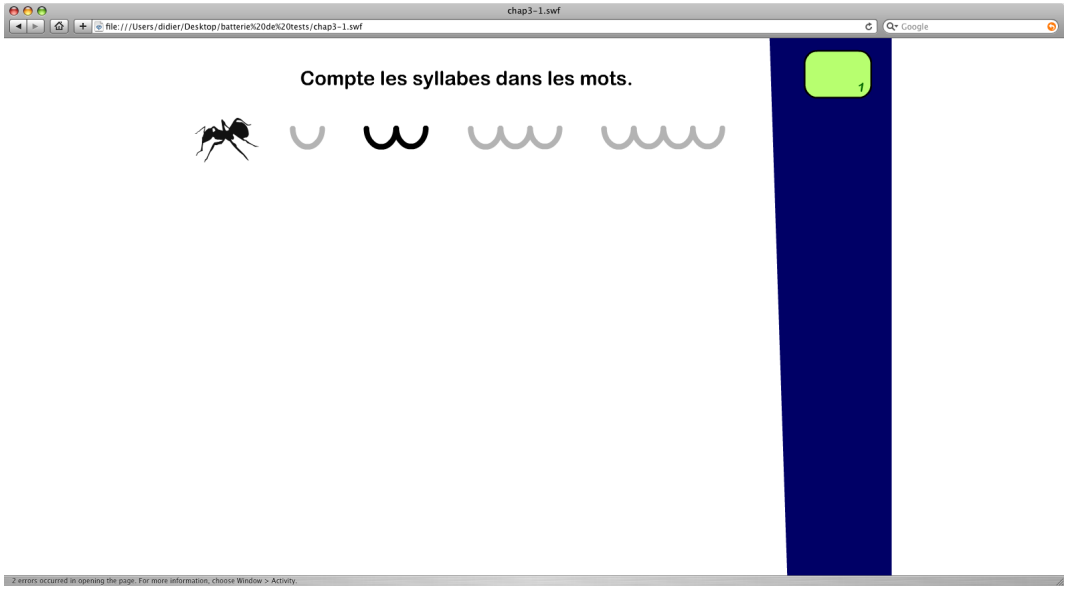

Figure 2.5 : Compter le nombre de syllabes dans un mot oral

Cet écran permet au système de déterminer si l'apprenant sait compter le nombre de syllabes dans un mot. On renseigne ici sa connaissance du fonctionnement de l'écrit.

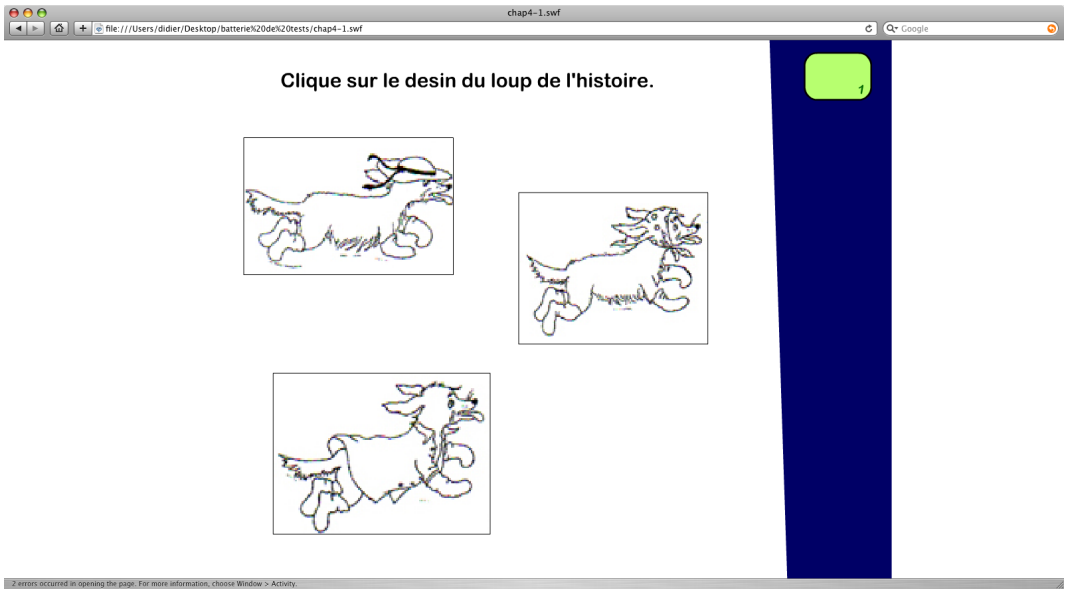

Figure 2.6 : Compréhension orale

Cet écran permet au système de déterminer si l'enfant comprend une histoire lue simple. Pour cela, on lui demande de sélectionner une image parmi celles présentées pour voir s'il a compris l'histoire lue. On renseigne ici son degré de compréhension orale.

4. Processus général d'élaboration de notre modèle de l'apprenant

Une fois le modèle initialisé avec les valeurs acquises pendant l'exécution de la batterie de tests, l'environnement AMICAL peut commencer à travailler avec l'apprenant d'une manière plus efficace. Nous allons maintenant rentrer dans le fonctionnement plus détaillé de l'environnement.

Le rôle du module tutoriel est de proposer à l'apprenant des situations et des conditions lui permettant de construire ses savoirs de manière autonome et personnelle. Du point de vue du système, le rôle de ce module tutoriel peut également être vu comme étant de modéliser l'apprenant (Self, 1987).

4.1. Cycle fonctionnel de l'environnement AMICAL

L'acte d'enseignement peut se décomposer en trois étapes :

1. La première consiste à déterminer les connaissances à faire acquérir à l'apprenant et répond à la question « Qu'est ce que je dois faire acquérir à mon apprenant ? »
2. La deuxième a pour but de proposer des situations d'apprentissage susceptibles de conduire à cette acquisition tout en observant l'apprenant et répond à la question : « Comment je vais lui faire acquérir ces connaissances ».
3. La troisième interprète les observations recueillies pour déterminer les états de savoir de l'apprenant et répond grossièrement à la question : « A partir de ce qu'il a fait, a-t-il acquit ces connaissances ? ».

Ce sont ces trois étapes que nous retrouvons au sein du fonctionnement du module tutoriel de l'environnement AMICAL. Le fonctionnement de ce module est

représenté par un cycle (Cleder, 2002), que nous qualifions de cycle fonctionnel. Le schéma suivant permet de visualiser ce cycle :

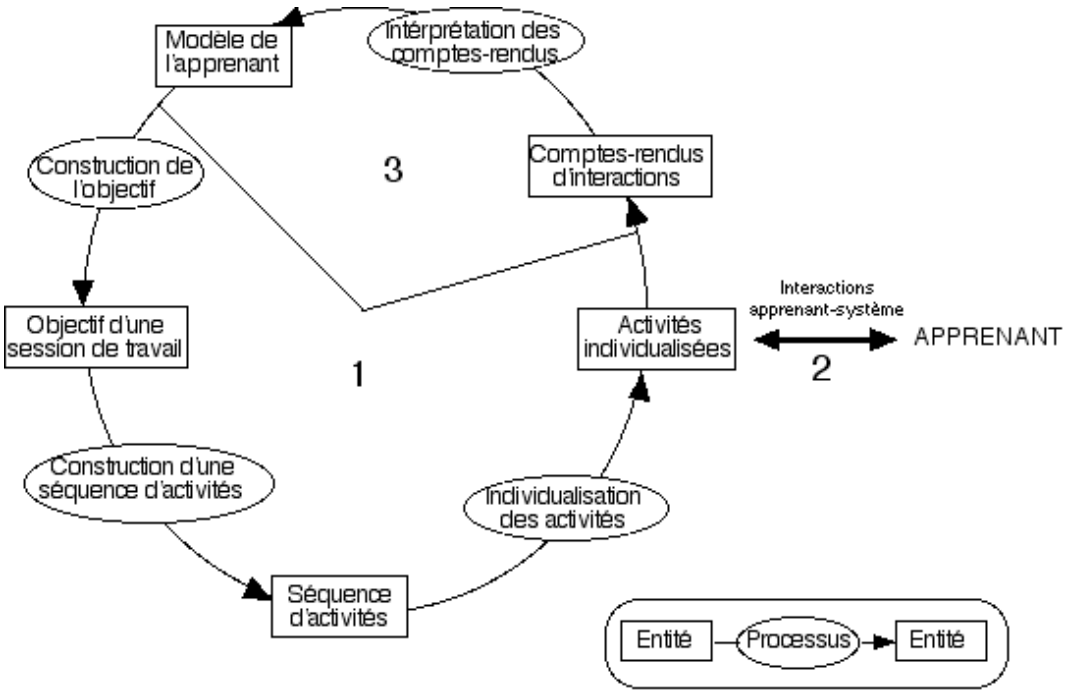

Figure 2.7 : Cycle fonctionnel de l'environnement AMICAL

Comme nous le voyons sur ce schéma, ce cycle se décompose en trois phases ; chacune des phases correspondant à une étape de l'enseignement :

1. la planification didactique,
2. l'exécution de la séquence d'activités par l'apprenant,
3. l'analyse des interactions et la mise à jour du modèle de l'apprenant.

La première phase, la planification didactique, est une succession de deux étapes : la construction de l'objectif de la session de travail et la construction de la séquence de situations didactiques individualisées permettant d'atteindre l'objectif fixé, répondant respectivement à la question « Que doit-on faire acquérir à cet apprenant ? » et « Comment le lui faire acquérir ? ».

La première étape consiste à construire l'objectif pédagogique de la session de travail. Cet objectif est composé d'unités d'objectifs (Cleder, 2002). Celles-ci sont

représentées par des couples ou des triplets : <action ; connaissance> ou <action ; statut ; connaissance>. La différence entre couple et triplet intervient selon le type d'action. Sans rentrer dans le détail, ces unités d'objectifs représentent les intentions d'enseignement que le système cherche à atteindre. Par exemple, un couple <faire-utiliser ; association graphème-phonème a/[a]> signifiera que le système aura pour intention de faire utiliser à l'apprenant l'association graphème-phonème a/[a] au cours de la séquence de travail qu'il aura avec lui. la connaissance peut elle-même être une structure complexe. Par exemple, pour exprimer la volonté de mettre l'apprenant en présence d'un texte lu, on aura l'unité d'objectif <mettre-en-présence ; <texte : lecture-texte>>.

La seconde étape de la planification consiste d'abord à construire la séquence de situations didactiques types puis ensuite à construire la séquence de situations didactiques instanciées pour atteindre l'objectif précédemment construit. La séquence de situations didactiques instanciées étant une individualisation ou un paramétrage de la séquence de situations didactiques types. Les situations didactiques sont sélectionnées dans une bibliothèque en fonction de celles permettant au mieux d'atteindre l'objectif fixé, c'est-à-dire en fonction des unités d'objectif. Le nombre de situations didactiques disponibles au sein de l'environnement est pour le moment relativement restreint pour des raisons de travaux en cours de la part de nos experts en apprentissage de la lecture. Le nombre final devrait probablement tourner autour de plusieurs dizaines voire peut-être même une centaine.

Nous distinguons donc deux types de situation didactique :

1. la *situation didactique type* est un objet théorique, caractérisé par les types de connaissances qu'il met en jeu pour l'apprenant. Celle-ci se présente comme un squelette ou un « moule » d'activité. C'est en quelque sorte la description d'une activité,
2. la *situation didactique instanciée* (ou activité) correspond à l'activité exécutable. Elle correspond à une situation didactique type que l'on a instancié et individualisé et qui dispose d'une interface, lieu d'interaction entre l'apprenant et le système. C'est elle qui contient par exemple le texte que l'on a choisi de présenter à l'apprenant, les mots que l'on a choisis de lui faire découvrir ou encore les lettres que l'on a choisies de lui faire acquérir.

Par la suite, nous emploierons indistinctement les termes de « situation didactique instanciée » et d'« activité ». Nous emploierons également indistinctement les termes de « séquence de situations didactiques instanciées » et de « session de travail ». Nous réserverons le terme de « situation didactique » pour désigner une « situation didactique type ».

Les objectifs et les séquences de situations didactiques ainsi que leur individualisation ne sont pas prédéfinis. Ils sont construits à chaque cycle à partir du modèle de l'apprenant et des connaissances didactiques et pédagogiques sur le domaine. Aussi, chaque situation didactique de la séquence portera sur des connaissances différentes.

La deuxième phase, l'exécution de la séquence d'activités, met l'apprenant face à la séquence à l'interface du système. Durant cette phase, le système enregistre l'ensemble des interactions de l'apprenant sous forme d'un compte-rendu unique comprenant l'ensemble de la session de travail.

La troisième et dernière phase, celle qui nous concerne dans cette thèse, consiste à analyser le compte-rendu mémorisé au cours de l'interaction. Le système cherche à interpréter ces interactions, à inférer de nouvelles informations sur l'apprenant puis à les intégrer aux anciennes informations du modèle de cet apprenant. De manière générale la troisième phase peut être décomposée elle-mêm en deux phases :

1. Etant donnée une séquence de situations didactiques, la première phase que nous appelons *processus d'analyse des activités* consiste à formuler des hypothèses sur les états de savoir de l'apprenant pour chaque situation didactique.
2. La deuxième phase appelée *processus de mise à jour du modèle de l'apprenant* consiste à intégrer les hypothèses issues de la première phase au modèle préexistant.

Comme nous l'avons dit précédemment, toutes les interactions entre le système et l'apprenant se réalisent au sein de ces situations didactiques. Intéressons nous maintenant de plus près à ces situations didactiques, lieu essentiel de la modélisation de l'apprenant.

4.2. Les situations didactiques

Les situations didactiques sont des objets complexes multifacettes (Chambreuil et al., 2000) que l'on peut considérer de différents points de vue :

- par rapport à l'objectif d'enseignement et au processus gestionnaire de la planification didactique, une situation didactique est une unité d'action dont dispose le processus pour atteindre l'objectif d'enseignement,
- par rapport à l'apprenant et au domaine d'apprentissage une situation didactique est un problème à résoudre par l'apprenant,
- par rapport à l'apprenant elle peut aussi être considérée en tant qu'espace d'interactions,
- par rapport aux processus gestionnaire des situations didactiques et gestionnaire du modèle de l'apprenant, une situation didactique est un espace d'observation de l'apprenant.

Les deux premiers aspects renvoient aux connaissances que la situation didactique est susceptible de faire acquérir à l'apprenant et à celles qu'il est susceptible de mettre en œuvre pour résoudre le problème qui lui est posé. On retrouve donc ici les types de connaissances que l'on souhaite évaluer.

Les deux derniers aspects concernent les informations que l'on peut recueillir sur l'interaction de l'apprenant avec le module tutoriel et qui permettront justement d'évaluer les connaissances.

Dans la suite, nous allons rentrer dans le détail de chacun de ces points de vue.

4.2.1. Situation didactique par rapport à l'objectif d'enseignement

Une situation didactique peut être considérée comme étant une action susceptible de permettre d'atteindre l'objectif d'enseignement. Par rapport à cet aspect, aux connaissances sur le domaine d'apprentissage et à celles liées à son enseignement, une situation didactique peut être caractérisée comme une configuration de couples <action : connaissance> ou de triplets <action : statut : connaissance> appelés unités d'objectif ou UO (Cleder, 2002).

Ces couples ou triplets peuvent être de natures différentes selon les actions et les connaissances mises en jeu. Prenons l'exemple d'une situation didactique de type Reconnaissance de mots en Contexte dans laquelle l'apprenant doit identifier un mot dans un texte. De manière générale, elle est susceptible de répondre à une configuration de couples suivante :

```
<faire acquérir : mot>
<faire acquérir  : correspondance mot écrit/mot oral>
<faire mémoriser  : connaissance de mots>
<inciter à utiliser : stratégie identification de mots>
<inciter  à utiliser : capacité à utiliser des aides>
<évaluer  : capacité à utiliser stratégies de résolution de
problèmes>
```

Le premier couple vise à faire acquérir à l'apprenant le concept de mot en tant qu'objet écrit. Le deuxième se rapporte à la correspondance entre un mot oral et sa forme écrite. Le troisième vise à faire acquérir le vocabulaire. Les trois couples suivants concernent plus particulièrement les stratégies de résolution de problèmes. Ainsi, le module tutoriel incite l'apprenant à utiliser des stratégies d'identification de mots pour résoudre le problème qui lui est posé. Il évalue également ses capacités métacognitives c'est-à-dire le contrôle qu'il porte sur son apprentissage notamment sur sa capacité à utiliser et exploiter les aides qui peuvent lui être fournies. Enfin le dernier couple évalue des stratégies de résolution plus générales indépendantes du domaine d'apprentissage de la lecture. Ces capacités concernent l'utilisation ou la réutilisation de stratégies de résolution de problèmes dans des problèmes similaires ou identiques, ses aptitudes à identifier le type de problème posé, à sélectionner les stratégies applicables et à les utiliser.

Par rapport à l'évaluation de l'état de savoir de l'apprenant nous soulignons deux aspects :

- L'individualisation d'une situation didactique type peut conduire à préciser sa configuration d'UO. Ainsi, dans notre exemple, le troisième couple peut comporter l'ensemble de mots à faire mémoriser. On peut également préciser dans le quatrième couple les stratégies d'identification particulières que l'on souhaite voir utiliser par l'apprenant.
- Par rapport à l'objectif d'enseignement, une situation didactique est sélectionnée parce que certains de ses aspects peuvent permettre d'atteindre tout ou partie de l'objectif. Cependant, elle est susceptible de

réaliser l'ensemble des couples de sa configuration et donc de permettre d'évaluer des connaissances autres que celles strictement retenues pour l'objectif.

La caractérisation en termes d'UO de chaque situation didactique de la séquence proposée à l'apprenant est nécessaire à l'agent gestionnaire du modèle de l'apprenant pour savoir quels types d'évaluations sur quels types de connaissances il doit effectuer et par la suite permettre d'évaluer si l'objectif d'enseignement peut être considéré comme atteint ou non.

4.2.2. Situation didactique comme problème à résoudre par l'apprenant

Pour l'apprenant, une situation didactique constitue le lieu dans lequel il construit son savoir. L'apprenant est placé dans une situation dans laquelle il doit accomplir une tâche spécifique. Ceci est vrai même d'une situation didactique présentant des informations à l'apprenant. Une situation didactique de type Présentation de Texte consistant à présenter un texte à l'apprenant vise à placer l'apprentissage dans un contexte signifiant. Elle vise aussi à développer son niveau de compréhension de texte qui sera évalué en posant des questions sur le texte présenté.

Par rapport au problème à résoudre par l'apprenant, une situation didactique peut être caractérisée de deux manières : d'une part, par les composants de la situation d'apprentissage dans laquelle se trouve l'apprenant, d'autre part, par les types de connaissances qu'il doit mettre en œuvre pour résoudre le problème qui lui est posé.

4.2.2.1. Composants de la situation d'apprentissage

Parmi les composants, nous pouvons citer la consigne par rapport à la tâche à accomplir, les données constitutives du problème, les aides dont l'apprenant peut disposer, les traces de sa résolution de problèmes et de ses solutions ou encore les commentaires qui lui sont fournis durant sa démarche. Chacun de ces composants est lui-même un objet complexe pouvant être analysé selon deux perspectives (Chambreuil et al., 2000). La première concerne les informations susceptibles

d'être portées par les composants. La deuxième concerne la prise en compte de ces informations dans l'interprétation des solutions de l'apprenant. Par rapport aux aides par exemple, la première perspective analyse les aides selon leurs types et les critères d'adéquation à l'apprenant. La deuxième perspective analyse les solutions fournies en fonction de l'utilisation ou non des aides données à l'apprenant.

4.2.2.2. Types de connaissances à mettre en œuvre par l'apprenant

Une situation didactique peut par ailleurs être analysée par rapport à l'apprenant du point de vue des connaissances qu'il doit mettre en œuvre dans sa démarche de résolution. Ces connaissances varient en fonction de la situation didactique proposée à l'apprenant.

Les connaissances mises en jeu concernent l'ensemble des connaissances du domaine que nous avons évoqué au paragraphe 2, à savoir ses connaissances sur les concepts de la lecture, sur les objets de la lecture ainsi que les stratégies mises en oeuvre par l'apprenant durant sa résolution de problème. Elles concernent également les autres types d'informations susceptibles de figurer dans le modèle de l'apprenant, comme les aspects cognitifs, comportementaux on méta-cognitifs.

4.2.3. Situation didactique comme espace d'interaction

Une situation didactique est caractérisée d'une part par ses constituants, d'autre part par son scénario pédagogique. L'ensemble des constituant est par définition tout ce qui constitue l'écran d'interactions avec l'apprenant ; chaque constituant ayant un rôle particulier. Certains de ces constituants sont fixes, c'est-à-dire qu'ils ne changent pas et ne dépendent pas de l'apprenant, alors que d'autres, au contraire dépendent de l'apprenant. Par exemple, la position d'un bouton de validation ne changera jamais alors que le texte présenté dans la situation didactique sera déterminé en fonction de l'apprenant, par exemple en fonction du degré de difficulté de celui-ci.

L'ensemble de ces constituants et leur mode d'intervention constituent le scénario pédagogique et définit le déroulement de la situation didactique (Mahmoud,

1997). Un scénario pédagogique peut être décomposé en trois phases :

1. la formulation de la consigne du problème à résoudre,
2. la construction de la solution par l'apprenant ou phase de travail,
3. l'évaluation de la solution.

Nous détaillons à partir d'un exemple les différents types de constituants d'une situation didactique et leur organisation, ainsi que les différents modes d'actions dont dispose l'apprenant pour interagir avec le module tutoriel. L'exemple que nous présentons est une situation didactique de type « reconnaissance autonome de mots dans un texte ». Cette situation demande à l'apprenant de sélectionner tous les mots qu'il pense connaître.

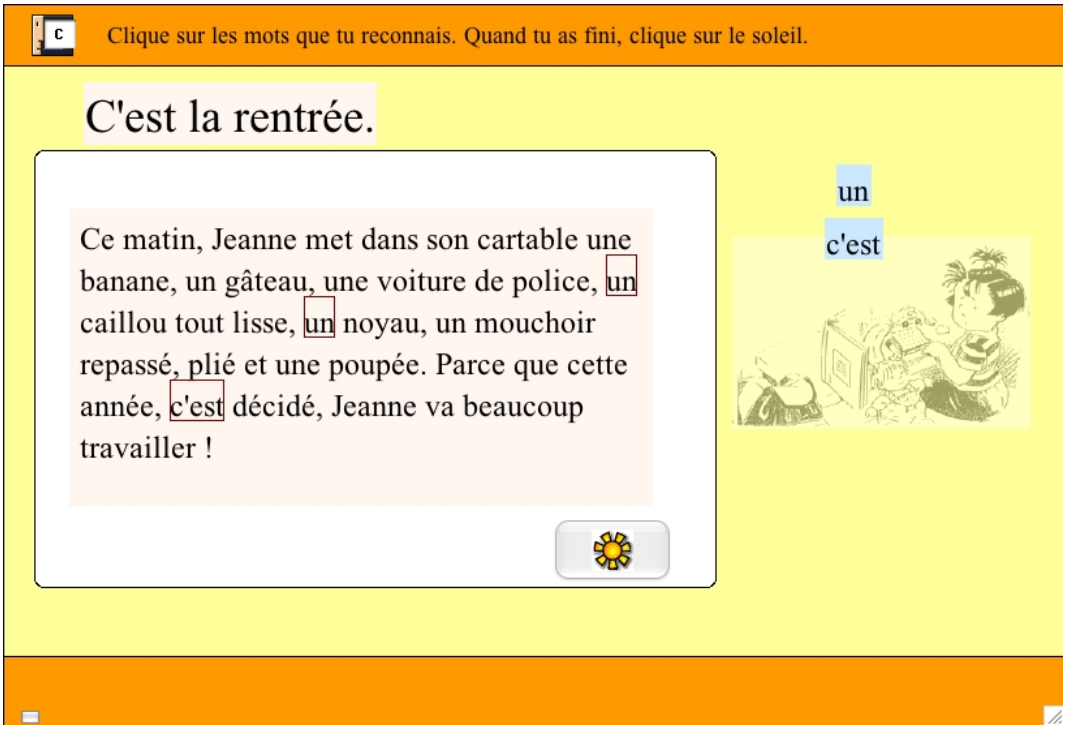

Figure 2.8 : Exemple d'écran d'interactions dans une activité de type reconnaissance autonome de mots dans un texte

Nous définissons trois espaces sur lesquels l'apprenant peut agir pendant les phases du scénario pédagogique : l'espace consigne, l'espace de travail et l'espace d'évaluation.

4.2.3.1. L'espace consigne

L'espace consigne est essentiellement composé du cadre contenant la consigne. C'est la barre qui se situe tout en haut de l'écran en orange. La consigne est ici lue et apparaît sous sa forme écrite. Un bouton permet éventuellement à l'apprenant de réécouter la consigne dans le cas où il ne l'aurait pas comprise ou bien dans le cas où il n'aurait pas eu le temps de l'écouter. Différents choix d'individualisation sont possibles : la consigne peut être lue mais non écrite ou écrite et non lue.

A la consigne est associée une aide permettant de la faire relire. L'utilisation de cette aide apporte des informations sur le comportement de l'apprenant par rapport au problème posé : si elle est systématique, elle peut traduire une économie d'effort ou une stratégie de simplification de la tâche. L'aide de lecture de la consigne est active pendant la phase de travail.

4.2.3.2. L'espace de travail

L'espace de travail est ici constitué du texte dans lequel l'apprenant doit sélectionner les mots qu'il pense connaitre et du bouton de validation, représenté ici sous forme de soleil. C'est la partie de l'écran où l'on voit le texte.

Des aides peuvent être associées à l'espace de travail. Par exemple pour faire relire le texte. Ces aides permettent à l'apprenant de réduire son champ de recherche de la solution. Le bouton de validation oblige l'apprenant à décider de confirmer ses choix. Tant qu'il n'a pas validé sa sélection, il peut changer d'avis et choisir de sélectionner un autre mot ou bien d'annuler sa sélection en recliquant sur le même mot. Cependant les sélections successives sont mémorisées dans le compte-rendu d'interactions. Ils permettent notamment de mettre en évidence les stratégies de résolution de problèmes utilisées par l'apprenant en particulier celles liées à l'identification de mots.

4.2.3.3. L'espace d'évaluation

L'espace d'évaluation permet au système de laisser à la disposition de l'apprenant les traces de ses solutions. Il est ici représenté par la liste des mots sélectionnés en bleu. Les mots sélectionnés dans le texte sont également encadrés de rouge. Cela permet ici à l'apprenant de voir les mots qu'il a déjà sélectionnés.

4.2.3.4. Modes d'interaction de l'apprenant sur une situation didactique

Nous avons donc vu que l'apprenant peut agir sur différents types d'objets à différents moments lors de son interaction avec le module tutoriel. Dans AMICAL, l'apprenant n'a à sa disposition comme périphériques d'entrée que le clavier et la souris. Il peut donc réaliser trois types d'actions :

- saisir du texte,
- déplacer des objets,
- cliquer sur des objets .

Certaines situations didactiques privilégient plutôt certains types d'actions lors de la construction de la solution. Dans l'exemple ci-dessus, l'apprenant doit choisir sa solution en cliquant sur un mot dans le texte. Dans une situation didactique de type Closure, qui consiste à placer des étiquettes dans un texte à trous, il doit sélectionner une étiquette et la placer dans le texte. Dans une situation didactique de type Copie de mots, il doit copier un mot du texte dans un cadre.

Par rapport à l'élaboration du modèle de l'apprenant et à ces types d'actions, nous soulignons deux aspects :

- un même type d'action a des significations différentes selon l'objet qu'il concerne. Ainsi, un clic sur un bouton d'aide indique une décision de demande d'aide de la part de l'apprenant alors qu'un clic sur un mot du texte peut indiquer un choix de solution. Un clic sur le bouton de validation est une confirmation de la solution choisie. Un clic sur une zone inactive, s'il est répétitif peut traduire un problème de maîtrise de l'outil informatique.
- un même type d'action sur un même type d'objet peut prendre un sens différent selon le contexte dans lequel il intervient. Ainsi, un clic sur un mot du texte peut constituer un choix de solution. Par contre, si l'apprenant a demandé une aide de lecture de mots en cliquant sur le bouton d'aide correspondant, le clic sur le mot désigne l'élément à faire lire. Il ne suffit donc pas de considérer l'action de base (le clic) de manière isolée mais il est nécessaire de la situer dans l'action fonctionnelle de plus haut niveau (la demande d'aide) dans laquelle elle s'inscrit.

Ces deux aspects illustrent la complexité d'interprétation des actions de l'apprenant : elle doit prendre en compte les fonctionnalités des actions et le

contexte dans lequel elles interviennent. Pour pouvoir réaliser les interprétations, il est d'abord nécessaire de recueillir les interactions de l'apprenant avec le module tutoriel.

4.2.4. Situation didactique comme espace d'observation de l'apprenant

Une situation didactique peut être considérée comme un espace d'observations de l'apprenant dans la mesure où elle permet de recueillir les interactions de l'apprenant avec le système. Or, pour élaborer notre modèle de l'apprenant il est nécessaire d'effectuer une observation précise et fine de l'apprenant. Se posent alors deux types de problèmes : le premier concerne le type d'observation et leur utilisation, le deuxième concerne la forme et la granularité des informations recueillies.

4.2.4.1. Types et utilisation des observations

L'observation de l'apprenant est réalisée de manière implicite c'est-à-dire que le système mémorise les interactions sans solliciter l'apprenant. Dans AMICAL, nous ne prenons en compte que les actions de l'apprenant sur les interfaces qui lui sont proposées.

Les observations recueillies lors de l'interaction peuvent être utilisées de plusieurs manières :

- elles permettent par exemple de reconstituer le déroulement effectif de l'interaction pour un observateur extérieur,
- elles peuvent également fournir des informations permettant d'améliorer les interfaces proposées ou de fournir de nouvelles fonctionnalités à l'apprenant,
- elles sont enfin et surtout susceptibles de fournir des informations sur la démarche de résolution de problèmes suivie par l'apprenant et des éléments sur ses états de savoir.

C'est bien entendu dans le cadre de cette dernière utilisation que la modélisation de l'apprenant est réalisée. Néanmoins, l'idée de permettre à un enseignant

extérieur de visualiser les interactions de l'apprenant avec le module tutoriel a été réalisée dans le cadre d'un outil de visualisation de comptes-rendus (Lotin et al., 1999). Il est également envisageable que l'enseignant puisse intégrer des hypothèses au modèle à l'aide d'interfaces spécifiques, notamment pour remplir les éléments constitutifs de la partie informations générales au sein du modèle mais pas exclusivement.

L'interprétation des observations en termes d'états de savoir de l'apprenant est liée à la forme et la granularité des informations mémorisées.

4.2.4.2. Forme et granularité des observations recueillies

Le système mémorise pour chaque action provenant du système ou de l'apprenant un ensemble de paramètres identifiant l'action et l'objet manipulé. Considérons par exemple, un fragment de compte rendu d'interaction d'une situation didactique de type « reconnaissance autonome de mots dans un texte »

```
[5] 15:36:04 (S:185s, A:0s) : Debut SDE :
[5] 15:36:10 (S:192s, A:7s) : Selection : mot "banane"
[5] 15:36:19 (S:201s, A:16s) : Selection : mot "noyau"
[5] 15:36:38 (S:219s, A:34s) : Validation :
```

Cette visualisation du compte-rendu n'est qu'une vision pour l'expert. Pour le système, nous avons retenu le formalisme XML que nous introduirons et expliquerons au chapitre 4.

Cette partie de compte-rendu contient tous les événements mémorisés pour la situation didactique « reconnaissance autonome de mots dans un texte ». L'apprenant n'a ici sélectionné que deux mots : banane et noyau.

Chaque ligne est composée de la façon suivante :

- le numéro entre crochet désigne le numéro de l'activité correspondante dans le compte-rendu. Ici le numéro cinq renvoie à l'activité de type « reconnaissance autonome de mots dans un texte »,
- on trouve ensuite l'heure à laquelle l'évènement a été mémorisé,
- entre parenthèses ensuite, on trouve le temps écoulé en secondes depuis le début de la séquence (S:) et le temps écoulé depuis le début de l'activité

(A:),
- le type d'action réalisé par l'apprenant. Ici, on trouve soit la sélection d'un mot, soit la validation des réponses,
- suivant le type d'action, on trouve l'objet de cette action, c'est-à-dire ce sur quoi a porté l'action. Ici, la sélection porte sur un mot.

Cette portion de compte-rendu nous indique que l'apprenant a d'abord sélectionné le mot banane puis le mot noyau et qu'ensuite il a validé sa réponse.

Les différentes formes de caractérisation des situations didactiques que nous venons de présenter permettent une individualisation fine de l'apprentissage et fournissent des comptes-rendus d'interactions nécessaires au processus de construction du modèle de l'apprenant.

Nous présentons maintenant la construction du modèle de l'apprenant à partir de ces comptes-rendus d'interactions présentés précédemment.

4.3. Construction du modèle de l'apprenant

Après avoir évoqué les situations didactiques sous tous leurs aspects car étant la source principale de la modélisation de l'apprenant au sein de l'environnement, nous présentons maintenant le processus de construction du modèle de l'apprenant qui, à partir de l'analyse des interactions avec ces situations didactiques, mettra à jour le modèle de l'apprenant. La présentation se fera toujours de manière théorique. Nous rentrerons dans le détail concret du fonctionnement au chapitre 4, quand nous parlerons des différentes composantes de l'agent gestionnaire du modèle de l'apprenant.

Comme nous l'avons dit, l'ensemble des interactions entre le système et l'apprenant est mémorisé dans un compte-rendu unique pour la session de travail. Ce compte-rendu contient l'ensemble des comptes-rendus de chaque situation didactique mais également l'objectif que la séquence a cherché à atteindre. Il contient également diverses informations telles que la date ou le nom de l'apprenant. Néanmoins, les seules informations pertinentes pour évaluer l'état de savoir de l'apprenant se trouvent dans les sous-comptes-rendus. Une fois la séquence de travail terminée, ce compte-rendu peut-être exploité pour mettre à

jour le modèle de l 'apprenant.

La première chose réalisée pour construire le modèle consiste à découper ce compte-rendu en autant de sous-compte-rendu qu'il y a de situations didactiques : $CR\text{-}SDT_1$ à $CR\text{-}SDT_n$; n représentant le nombre d'activités dans la session de travail. Nous parlerons à partir de maintenant de compte-rendu en lieu et place de sous-compte-rendu. On trouve trois types d'informations dans le compte-rendu de chaque activité :

1. *l'objectif que l'activité cherche à atteindre* : il est constitué par la configuration d'unités d'objectif de la situation didactique. Elle définit les types de connaissances à évaluer.
2. *les éléments d'individualisation utilisés* au cours de l'exécution de l'activité : il s'agit des différents paramètres de l'activité. On trouve ici, à la fois le contenu variable de l'activité (texte présenté, mots demandés, …) ainsi que différents autres paramètres (nombre d'essais, aides proposées, …). Ces paramètres permettent l'évaluation d'éléments particuliers des connaissances de l'apprenant.

– *les interactions elles-même, entre le système et l'apprenant* : ce sont les différentes actions qu'a mis en oeuvre l'apprenant (sélection, validation, ...)

Ensuite, de manière générale la construction du modèle de l'apprenant peut être décomposée en deux phases :

1. Pour chaque situation didactique de la séquence, la première phase que nous appelons *processus d'analyse* consiste à formuler des hypothèses sur l'état de savoir de l'apprenant pour cette activité particulière à partir du compte-rendu correspondant.
2. La deuxième phase appelée *mise à jour du modèle de l'apprenant* consiste à intégrer toutes les hypothèses issues de la première phase au modèle de l'apprenant existant et de le mettre à jour en conséquence.

On peut schématiser la construction du modèle de la manière suivante :

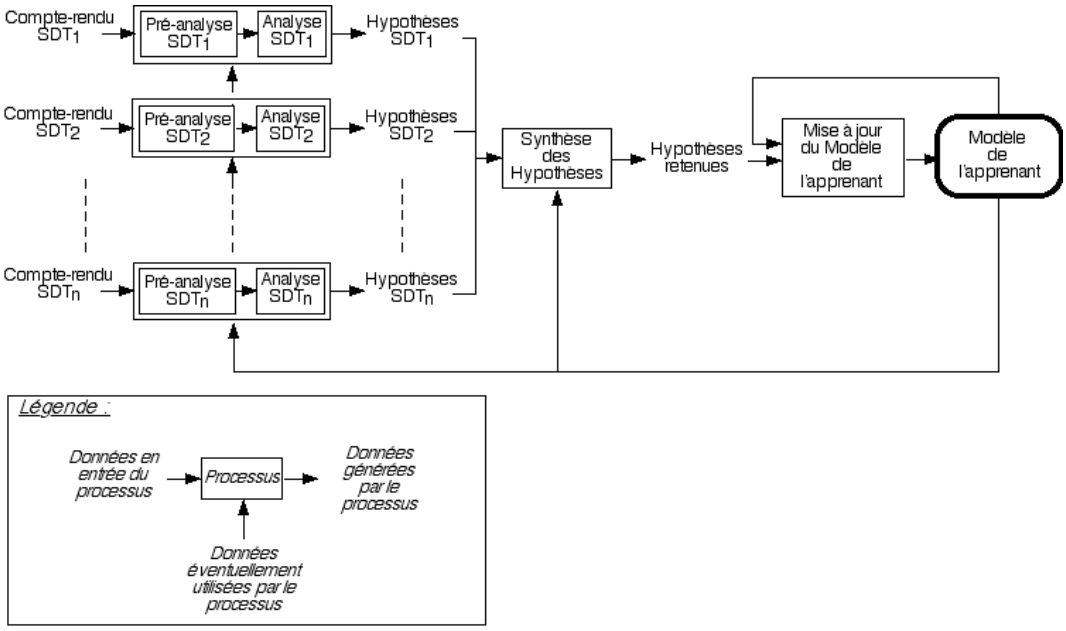

Figure 2.9 : Processus d'analyse et de mise à jour du modèle de l'apprenant

4.3.1. Processus d'analyse

Le processus d'analyse correspond à l'ensemble des sous-processus d'analyse des différentes activités. Comme on peut le remarquer sur ce schéma, chaque processus d'analyse prend en entrée le compte-rendus d'une activité de la séquence de travail que l'on vient de proposer à l'apprenant. Les processus d'analyse fonctionnent en parallèle ; ceci permettant bien évidemment de gagner du temps par le processus. Eventuellement, on pourrait envisager que chaque processus puisse communiquer avec les autres si le besoin se faisait sentir d'échanger des informations. De plus, on peut noter que chaque processus d'analyse pourra être spécialisé dans l'analyse particulière d'un type d'activité. Ceux-ci deviendront par la suite autant d'agents d'analyse qu'il y aura d'activités au sein de la bibliothèque du système.

Avant d'analyser une activité, nous commençons par faire ce que l'on appelle une pré-analyse. Cette pré-analyse a pour principal objectif de déterminer certaines données à partir du compte-rendu. La pré-analyse a par exemple pour rôle de :

- calculer le temps qu'a duré l'activité. On peut ainsi savoir si l'apprenant à été très rapide ou non pour réaliser l'activité. Ceci peut jouer un rôle dans le cas par exemple où la réponse a été la bonne et que le temps a été très bref. On pourrait en conclure que sa maitrise est plus élevée que s'il avait passé plus de temps.
- calculer éventuellement le nombre de clics « vides », c'est-à-dire hors zone d'interaction. Ceci peut traduire un manque de maitrise de l'interface ou bien de la souris.
- compter le nombre de réponses correctes. Par exemple, combien de mots juste a-t-il trouvés ?

A partir des différentes informations contenues dans le compte-rendu et de la pré-analyse, le processus d'analyse va formuler des hypothèses sur l'état des connaissances de l'apprenant. Ces hypothèses vont porter à la fois sur les connaissances du domaine (concepts, objets ou stratégies) ainsi que sur les autres aspects cognitifs, comportementaux ou méta-cognitifs. Ces hypothèses dépendent de l'activité considérée.

Par exemple, dans une situation didactique de type « identification de mot en contexte », qui a pour but de faire identifier à l'apprenant un mot dans un texte, les hypothèses porteront sur :

- *la connaissance de mots particuliers* : l'apprenant doit acquérir un nombre de mots constituant son vocabulaire en particulier au tout début de l'apprentissage. Cette situation didactique permet de faire identifier ou de faire reconnaître des mots. L'identification porte sur des mots qui n'ont jamais été explicitement travaillés par l'apprenant au contraire de la reconnaissance qui concerne des mots déjà étudiés.
- *des connaissances sur le concept de mot* : on observe si l'apprenant connaît le mot en tant qu'unité signifiante repérable. En fait, c'est l'absence de cette connaissance qui est prise en compte selon le type d'élément validé par l'apprenant : si l'apprenant clique sur des signes de ponctuation et des espaces et valide son choix, alors cette connaissance est considérée comme non avérée.
- *des connaissances sur les stratégies spécifiques de résolution de problèmes* : on observe les stratégies susceptibles d'être mises en œuvre par l'apprenant. La situation peut en particulier avoir été choisie pour inciter l'apprenant à utiliser certaines stratégies : elles sont alors

explicitement présentes dans l'objectif pédagogique et sont impliquées par le choix des paramètres d'individualisation

Chaque processus d'analyse peut en cas de besoin consulter le contenu du modèle de l'apprenant dans l'hypothèse où l'interprétation de l'état de savoir nécessiterait une information disponible déjà présente dans le modèle de l'apprenant. Les hypothèses de tous les processus d'analyse sont ensuite combinées pour former l'ensemble des hypothèses retenues grâce à une synthèse de ces hypothèses. Eventuellement, certaines hypothèses peuvent être contradictoires et doivent donc faire l'objet d'un traitement particulier.

4.3.2. Intégration des hypothèses au modèle de l'apprenant

L'ensemble des hypothèses retenues est ensuite intégré au modèle de l'apprenant. Ce processus a donc pour objectif de mettre à jour les valeurs des informations du modèle ou bien de modifier plus en profondeur le modèle lui-même.

Comme nous l'avons vu dans ce chapitre les connaissances du domaine sont des objets complexes multi-niveaux. L'intégration de ces hypothèses peut donc conduire, soit à ajouter simplement l'hypothèse à la connaissance considérée si l'on a aucune information antérieure, soit à modifier le statut d'un niveau qui lui-même peut entrainer la modification du statut des autres niveaux. Par exemple, on passera le statut d'une connaissance à acquis si, par exemple, toutes les sous-connaissances ont été déterminées comme acquises. Ces modifications sont réalisées par des règles de synthèse qui portent sur tous les niveaux de chaque connaissance.

Nous ne rentrerons pas davantage dans le détail du fonctionnement théorique. L'objectif était de présenter la modélisation de l'apprenant dans le cadre de l'environnement avant de passer au fonctionnement plus concret de celui-ci et de l'agent développé dans le cadre de cette thèse. Nous reviendrons donc de manière concrète sur la construction de ce modèle au travers du fonctionnement de l'agent gestionnaire du modèle de l'apprenant, l'agent en charge de ce travail au sein de l'environnement mutli-agents.

5. Conclusion

Comme nous l'avons vu tout au long de ce chapitre, l'environnement AMICAL a choisi de modéliser de très nombreux aspects différents concernant l'apprenant. Nous en avons déterminé cinq : les connaissances spécifiques au domaine, les aspects cognitifs, les aspects méta-cognitifs, le comportement et les informations générales sur l'apprenant. Nous nous limiterons dans le cadre de cette thèse uniquement aux connaissances du domaine. Chaque connaissance du domaine est en fait un objet complexe multi-niveaux où chaque connaissance est composée de sous-connaissances elles-même associées à des contextes qui eux-même sont associés à des activités. Chaque niveau possède un statut représentant le degré de connaissance de l'apprenant à ce niveau de l'objet. Par exemple, le statut acquis signifiant que la connaissance est acquise par l'apprenant.

Nous avons également vu que le module tutoriel, du point de vue du système, avait pour rôle de construire le modèle de l'apprenant à partir de l'analyse des interactions avec les différentes situations didactiques ou activités ; chaque activité demandant à l'apprenant de mettre en oeuvre des connaissances ou compétences différentes. L'analyse de ces interactions conduit à émettre des hypothèses sur l'état de savoir de l'apprenant ; hypothèses ensuite intégrées au modèle existant en modifiant les informations déjà contenues dans celui-ci.

Dans le chapitre suivant, nous évoquerons l'architecture et le fonctionnement de l'environnement multi-agents AMICAL dans lequel est intégré l'agent que nous cherchons à réaliser dans cette thèse : l'Agent Gestionnaire du Modèle de l'Apprenant. Nous présenterons ses différents rôles ainsi que l'architecture que nous avons retenue pour celui-ci. Le fonctionnement détaillé et la formalisation sera présentée au chapitre 4.

CHAPITRE 3 : ARCHITECTURE DE L'AGENT GESTIONNAIRE DU MODÈLE DE L'APPRENANT

Au cours des deux chapitres précédents, nous nous sommes intéressés à ce qu'était la modélisation de l'apprenant uniquement d'un point de vue théorique. D'abord en présentant, dans le chapitre 1, ce que recoupait la modélisation de l'apprenant en général puis, dans le chapitre 2, en spécifiant comment, théoriquement, au sein de l'environnement AMICAL, elle se déroulait. Nous avons donc présenté les informations qu'on cherchait à modéliser, les situations didactiques à partir desquelles on tentait de les modéliser, l'analyse des interactions avec ces situations didactiques et enfin l'intégration des hypothèses issues de l'analyse au modèle existant.

L'environnement AMICAL, comme nous l'avons dit, étant toujours considéré uniquement du point de vue de son module tutoriel, est un système tuteur intelligent ayant pour objectif d'apprendre à lire à un enfant. Celui-ci a été développé sous forme de système multi-agents où chaque module du système tuteur intelligent est représenté par un agent. C'est un agent particulier de ce système qui est l'objet de cette thèse : l'Agent Gestionnaire du Modèle de l'Apprenant (AGMA par la suite). Nous utilisons ici le terme gestionnaire pour regrouper deux rôles essentiels de l'agent : la modélisation de l'apprenant et la réponse aux requêtes sur le contenu du modèle de l'apprenant de la part des autres agents du système. Bien que nous évoquerons le rôle de réponse aux différentes requêtes, c'est la modélisation de l'apprenant qui fera l'objet de toutes nos attentions dans ce document.

Au chapitre 2, nous avons parlé des situations didactiques et le rôle qu'elles avaient dans la modélisation de l'apprenant. Dans la suite de ce document et dans le cadre de cet agent, seuls les comptes-rendus de celles-ci seront évoqués. Nous

ne parlerons pas de la conception de ces situations didactiques ni de leur réalisation informatique car ceci ne relève pas de nos recherches. Nos recherches se sont exclusivement portées sur l'exploitation des comptes-rendus d'interactions de ces situations didactiques par AGMA pour construire le modèle de l'apprenant.

Comme nous l'avons déjà évoqué au chapitre précédent, la modélisation de l'apprenant comporte deux étapes :

1. d'abord, l'analyse des différentes situations didactiques présentées à l'apprenant au cours de la session de travail permet d'émettre les différentes hypothèses quant à l'état de savoir de l'apprenant ; état de savoir dépendant bien évidemment de la situation didactique considérée.
2. ensuite, l'intégration de ces hypothèses au modèle de l'apprenant existant.

L'architecture retenue pour notre agent tiendra donc bien évidemment compte de ces deux étapes. Nous verrons que AGMA sera lui-même considéré comme un système multi-agents où les agents représenterons à la fois les différents processus d'analyse et le modèle de l'apprenant lui-même. Les processus d'analyse seront représentés par des agents d'analyse où chaque agent aura pour charge d'analyser un type de situation didactique particulière alors que le modèle de l'apprenant sera lui représenté par des agents de connaissance où chaque agent sera responsable de la modélisation d'une connaissance particulière du modèle de l'apprenant.

Ce chapitre a ainsi pour principal objectif de présenter l'architecture retenue pour AGMA. Le plan de ce chapitre sera le suivant : nous commencerons tout naturellement par présenter les notions d'agents et de systèmes multi-agents pour situer notre travail. Ensuite, nous présenterons l'architecture multi-agents de l'environnement AMICAL au sein duquel opère notre agent. Puis, nous expliciterons les rôles qu'occupe AGMA au sein de l'environnement AMICAL ainsi que ses différentes interactions avec les autres agents du système. Enfin, nous présenterons l'architecture retenue pour AGMA. Les détails concernant les différents éléments constitutifs d'AGMA ne seront exposés dans le détail qu'au chapitre suivant.

1. Agents et Systèmes Multi-Agents

L'intelligence artificielle (IA) classique a pour principal objectif de construire des

programmes exécutant des tâches complexes à partir de connaissances expertes centralisées dans un système unique. Un des défauts majeurs de cette approche est de regrouper dans une même base de connaissances les expertises et les compétences d'entités très différentes collaborant à la résolution de problèmes complexes. L'intelligence artificielle distribuée (IAD) pose le problème de la manière suivante : au lieu de concentrer toute l'intelligence au sein d'une même entité, distribuons et répartissons cette intelligence au sein d'un groupe d'entités appelées agents. Chacun des agents de ce groupe sera autonome, capable de planifier et d'exécuter des actions dans un environnement et de collaborer avec les autres agents au prix de conflits éventuels (Erceau et Ferber, 1991).

1.1. Définition de la notion d'agent

Bien que le terme d'agent soit utilisé dans une multitude de contextes et de disciplines – en économie par exemple, on parlera d'agents économiques – il n'en existe cependant pas de définition unique. Le terme d'agent reste aujourd'hui un concept relativement flou et dont le sens n'est pas communément admis. En effet, il existe de très nombreuses définitions qui dépendent surtout des auteurs et de l'objectif des agents qu'ils développent. Certains auteurs vont même jusqu'à considérer qu'un interrupteur peut être vu comme un agent.

La notion d'agent ne renvoie pas à une architecture précise, mais se pose plutôt comme une abstraction ou un concept. Le concept d'agent permet alors de décrire une entité, dans le cas qui nous intéresse une entité logicielle, capable d'agir avec un certain degré d'autonomie pour accomplir un certain objectif. Cette abstraction peut être associée à certaines fonctionnalités utiles suivant les agents que nous recherchons à réaliser. On définira donc plutôt le terme d'agent par les fonctionnalités qu'on lui associe comme la pro-activité, la persistance, l'autonomie, les capacités sociales ou bien encore la réactivité.

La difficulté de définir précisément ce qu'est un agent vient du mot lui-même. En effet, sans rentrer dans l'étymologie du mot, « agent » signifie très grossièrement « qui agit ». Le problème, partant de ce sens très flou, c'est que l'on pourrait considérer que tout agit à un niveau ou un autre. La question revient ensuite à savoir comment on agit, avec qui, etc. Toute la problématique de la classification des agents revient à définir, comme dans chaque classification (la systématique en

phylogénétique par exemple), l'objectif de cette classification et les bons critères pour la réaliser.

La notion d'agent étant floue, nous partirons de trois définitions pour tenter de poser et de définir ce concept. Une première tentative de définition peut être celle de Russell et Norvig (Russell et Norvig, 2003) pour qui un agent est tout ce que peut percevoir son environnement par le biais de capteurs et d'agir sur ce même environnement par le biais d'effecteurs. Formellement, on peut représenter cela par une fonction qui, à toute séquence de perception fait correspondre une action que l'agent peut exécuter : f : $P^* \rightarrow A$.

Cette définition d'agent est assez générale et peut inclure des agents humains dont les capteurs peuvent être les yeux et les effecteurs peuvent être les mains, des agents robotiques dont les capteurs peuvent être les caméras et les effecteurs peuvent être les roues, ou des agents logiciels dont les capteurs et les effecteurs peuvent être une interface graphique.

Une deuxième définition plus précise est celle de Ferber (Ferber, 1995), qui donne une définition minimale selon laquelle un agent est une entité physique ou virtuelle :

- capable d'agir dans un environnement,
- pouvant communiquer directement avec d'autres agents,
- mû par un ensemble de tendances sous forme d'objectifs qu'elle cherche à optimiser,
- possédant des ressources propres,
- capable de percevoir de manière limitée son environnement,
- ne disposant que d'une représentation partielle de cet environnement,
- possédant des compétences et offrant des services,
- pouvant éventuellement se reproduire,
- dont le comportement tend à satisfaire ses objectifs en fonction des ressources, des compétences, de ses perceptions et des communications qu'elle reçoit.

La troisième définition retenue qui a été donnée plus récemment est la définition de Jennings et Wooldridge, reprise par Chaïb-draa (Wooldridge et Jennings, 1995), (Chaïb-draa, 2000), (Chaïb-draa et al., 2001) selon laquelle : « un agent est un système informatique, situé dans un environnement, et qui agit d'une façon

autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu ». On trouve plus de détails sur cette définition dans (Gaguet, 2001) :

- *L'agent est situé* : il est capable d'agir sur son environnement à partir de la perception qu'il en a. L'action que peut mener l'agent lui permet de modifier son environnement et conditionne ses prises de décisions ultérieures.
- *L'agent est autonome* : il est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne. Ce qui pousse l'agent à agir est une caractéristique qui lui est propre. L'autonomie n'est pas seulement relative à son comportement, mais porte également sur sa gestion de l'utilisation des différentes ressources dont il dispose, par exemple les connaissances dont il a besoin pour décider et agir. Un exemple très parlant de l'autonomie est probablement celui de la sonde *deep space one* de la NASA qui avait pour mission de photographier une comète. Au lieu de piloter la sonde depuis la Terre par des humains, celle-ci devait se piloter toute seule pour atteindre la comète et la photographier. Celle-ci avait été conçue comme un agent qui devait donc prendre ses propres décisions en fonction de la situation, sans intervention humaine (*remote agent*).

Comme le soulignent Erceau et Ferber, l'ensemble des fonctionnalités susceptibles de caractériser un agent s'organisent en quatre secteurs (Erceau et Ferber, 1991) :

- Deux d'entre eux concernent l'agent en tant qu'être social, ses relations avec son environnement et les autres agents : on distingue ainsi la dynamique des relations et leur actualisation des mécanismes nécessaires à leur planification.
- Les deux autres secteurs concernent les capacités propres de l'agent : de ses capacités personnelles de perception, décision et action aux modèles que l'agent a de lui-même, des autres et de l'univers.

Pour synthétiser ces quelques définitions que nous avons posées, nous définirons un agent comme un concept abstrait auquel, au minimum, on associera quatre caractéristiques :

- *un agent est autonome* : il est capable de se « débrouiller tout seul », c'est-à-dire sans l'intervention d'un autre agent réel ou virtuel, et de prendre ses propres décisions pour atteindre ses objectifs,
- *un agent est réactif* : il réagit (vite) aux changements de l'environnement

dans lequel il se trouve,
- *un agent est pro-actif* : il doit prendre l'initiative d'agir pour atteindre ses objectifs sans qu'on lui ordonne de le faire,
- *un agent est sociable* : il doit être capable d'interagir avec d'autres agents dans le cas d'un système multi-agents.

1.2. Types d'agents

Bien qu'il existe d'autres façons de catégoriser les agents (Müller, 1999), (Nwana, 1996), nous avons choisi de rester sur la décomposition classiquement retenue : agent réactif versus agent cognitif. En fait cette décomposition revient à décomposer les agents, en quelque sorte, suivant leur « niveau d'intelligence ». Par niveau d'intelligence, nous entendons capacités décisionnelles, c'est-à-dire selon qu'elles soient orientées vers l'atteinte de buts explicites ou bien uniquement conditionnées par des perceptions.

1.2.1. Agents réactifs

La tendance réactive (Demazeau et Müller, 1991), (Ferber, 1995) prétend qu'il n'est pas nécessaire que chaque agent soit individuellement intelligent pour que l'ensemble du système soit lui-même intelligent. Les agents réactifs réagissent aux événements sans planification des buts poursuivis et n'ont pas de capacité d'anticipation des événements futurs. Néanmoins, certains comportements intelligents peuvent émerger d'un système multi-agents réactifs. Un agent réactif est donc un agent simple, de faible granularité, ne possédant pas de représentation explicite de son environnement, dont le comportement ne consiste qu'à répondre à des stimulus de ce même environnement. De plus, il ne dispose que d'un langage de communication réduit. Il n'a pas de connaissance sur les autres, ni de capacité de raisonner sur les messages qu'il reçoit ou de développer des stratégies de contrôle (Demazeau et Müller, 1991).

Les agents réactifs sont ceux de plus bas niveau. Ils ne disposent que d'un protocole et d'un langage de communication réduits, leurs capacités répondant uniquement à la loi stimulus/action.

1.2.2. Agents cognitifs

La tendance cognitive est la plus représentée. Un système d'agents cognitifs se caractérise par un petit nombre d'agents, chacun disposant d'une base de connaissances explicites lui permettant de résoudre sa tâche et de gérer les interactions avec les autres agents.

Là encore le terme choisi, à savoir celui de "cognitif" est également très général. La cognition concerne tout ce qui a trait à la connaissance. Classiquement, un agent cognitif dispose de connaissances explicites du monde et au minimum est capable de raisonner dessus ou d'agir en fonction de celles-ci. Ensuite, on peut lui adjoindre toutes les autres caractéristiques que l'on peut trouver sous le terme cognition : apprentissage de nouvelles connaissances, émotions, etc. en ajoutant des modules supplémentaires à l'architecture de base.

Les agents cognitifs sont donc des agents qui disposent de connaissances explicites sur le monde qui les entoure ainsi que sur eux-même. Il existe plusieurs types d'agents cognitifs. L'intelligence artificielle traite essentiellement des agents cognitifs rationnels ou agents rationnels. Ceux-ci choisissent les actions qui leur permettent de maximiser leurs chances d'atteindre leurs objectifs en fonction des connaissances dont ils disposent. On trouve également les agents intentionnels qui possèdent des buts et plans explicites leur permettant d'accomplir leurs tâches. Le modèle le plus connu d'agent intentionnel est le modèle BDI ou Beliefs Desires Intentions (Rao et Georgeff, 1995). Ce modèle définit trois composantes: les croyances, les désirs et les intentions. Un agent BDI représente le type d'agent correspondant le plus au fonctionnement, en terme de prises de décisions, d'un être humain.

Dans toute la suite de cette thèse, nous nous plaçons dans le cadre d'agents cognitifs rationnels sans étudier la rationalité des agents qui dépasse le cadre de cette thèse.

1.3. Définition de la notion de système multi-agents

Après avoir évoqué les différents types d'agents qui existaient, nous donnons dans cette section une brève introduction aux systèmes multi-agents, nous discutons de

leurs différences avec les systèmes mono-agents, et nous évoquerons les deux utilisations possibles d'une modélisation multi-agents.

Comme nous l'avons dit un agent, dans sa définition la plus simple, peut être vu comme tout ce qui peut percevoir son environnement par le biais de capteurs et d'agir sur cet environnement par le biais d'effecteurs. Toutefois, les agents sont rarement des systèmes autonomes. Dans de nombreuses situations, ils doivent coexister et interagir avec d'autres agents de plusieurs manières différentes pour atteindre leur but. Par exemple, on peut citer les agents logiciels sur Internet ou bien les robots qui jouent au football. Ceux-ci pour marquer un but doivent collaborer et travailler en équipe.

Ce système qui consiste en un groupe d'agents qui peuvent interagir les uns avec les autres est appelé un système multi-agents, et le sous-champ de l'IA qui a trait aux principes et conception des systèmes multi-agents est l'intelligence artificielle distribuée (IAD).

Néanmoins, un système multi-agents n'est pas uniquement qu'un ensemble d'agents, mais comporte également d'autres éléments. Pour Ferber (Ferber, 1995), un système multi-agents est constitué des éléments suivants :

- d'un environnement,
- d'un ensemble d'agents,
- d'un ensemble de relations entre ces agents,
- d'un ensemble d'objets pouvant être créés, modifiés ou détruits par ces agents.

De son côté, Chaib-Draa (Chaïb-draa et al., 2001) précise qu'un système multi-agents possède les caractéristiques suivantes :

- chaque agent a des informations ou des capacités de résolution de problèmes limitées, ainsi chaque agent a un point de vue partiel,
- il n'y a aucun contrôle global du système multi-agents,
- les données sont décentralisées,
- le calcul est asynchrone.

1.4. Caractéristiques d'un système multi-agents

Nous présentons ici les aspects spécifiques et fondamentaux qui caractérisent un système multi-agents et qui les distinguent d'un système mono-agent. Nous faisons ici un rapide tour de ces différents aspects.

1.4.1. Conception des agents du système

Quand on conçoit un système multi-agents, une question qui nécessite d'être posée et qui n'existe pas dans le cadre d'un système mono-agent, est celle de l'hétérogénéité des différents agents que l'on cherche à développer. En effet, dans le cadre d'un système multi-agents, tous les agents ne sont pas nécessairement développés par la même personne ni même nécessairement dans le même contexte matériel-logiciel et peuvent donc être conçus de différentes manières. Par exemple, les différences de conception sont liées soit au matériel (par exemple le football des robots sur la base de différentes mécaniques des plates-formes), soit au logiciel (par exemple les agents logiciels fonctionnement sur des systèmes d'exploitation différents). La différence de conception peut également être liée à des choix différents quant aux types des agents développés, à leurs prises de décisions, etc.

Un exemple typique de ce type de système multi-agent sont les softbots ou software robot, ce que l'on pourrait traduire par « robot logiciel ». Un softbot est un agent logiciel qui fait office d'assistant intelligent personnel : l'utilisateur effectue une requête auprès du softbot et celui-ci va tenter de satisfaire cette requête.

On dit souvent que ces agents sont hétérogènes contrairement à l'homogénéité des agents qui sont conçus d'une même manière.

1.4.2. L'environnement

Comme pour les systèmes mono-agent, les agents d'un système multi-agents sont situés dans un environnement. Dans le cas d'un système multi-agents, on distingue deux types d'environnement : l'environnement du système multi-agents et l'environnement de chaque agent. L'environnement du système multi-agents représente en quelque sorte l'espace commun de tous les agents du système alors

que l'environnement de chaque agent est représenté par l'environnement du système et par les autres agents du système.

L'environnement peut être :

- *accessible ou inaccessible* : si les capteurs d'un agent lui donnent accès à l'état complet de l'environnement suffisant pour choisir une action, l'environnement est accessible à l'agent.
- *déterministe ou non déterministe* : il est déterministe pour un agent si le prochain état de l'environnement est déterminé par l'état courant et par l'action de l'agent.
- *épisodique ou non épisodique* : il est épisodique si les prochaines évolutions ne dépendent pas des actions déjà réalisées.
- *statique ou dynamique* : il est statique s'il ne change pas pendant que l'agent réfléchit.
- *discret ou continu* : il est discret si le nombre de percepts distincts et d'actions est limité.

L'environnement est un aspect très important des systèmes multi-agents.

1.4.3. Les interactions entre agents

Au sein d'un système multi-agents, les agents doivent pouvoir interagir entre eux pour atteindre leurs objectifs. Ces interactions sont l'essence même d'un système multi-agents et portent sur les mécanismes leur permettant d'interagir et sur les différentes formes d'interactions. Celles-ci peuvent se définir en relation avec trois composantes des agents :

- *les buts qu'ils poursuivent* : ils peuvent être identiques ou compatibles, on parle alors d'agents coopératifs, sinon on parle d'agents antagonistes,
- *leurs capacités à résoudre les tâches qui leur incombent* : les agents peuvent réaliser seuls les tâches ou faire appel aux capacités des autres agents,
- *les ressources auxquelles ils accèdent* : celles-ci sont nécessairement limitées et leur utilisation peut générer des conflits ou des perturbations si plusieurs agents cherchent à accéder aux mêmes ressources. Ces situations peuvent être résolues par des mécanismes de gestion de conflits ou de coordination d'actions.

Il existe de nombreux types d'interactions possibles au sein d'un système multi-agents. Ferber (Ferber, 1995) propose huit types d'interactions envisageables :

1. l'indépendance des agents,
2. la collaboration simple,
3. l'encombrement,
4. la collaboration coordonnée,
5. la compétition individuelle pure,
6. la compétition collective pure,
7. les conflits individuels pour des ressources,
8. les conflits collectifs pour des ressources.

Le choix du type des interactions que l'on met en oeuvre entre les agents dépend bien entendu du système que l'on cherche à concevoir ou à modéliser. Par exemple, dans une équipe de football robotique, le type d'interactions entre les agents seraient bien entendu la collaboration et non la compétition.

1.4.3.1. Le contrôle du système

Le contrôle correspond à la coordination des actions des agents. Celui-ci peut être réalisé de plusieurs manières. On distingue le contrôle centralisé et le contrôle distribué. Bien que nous présentons les deux modes, les recherches actuelles s'orientent davantage vers un contrôle distribué (Chaïb-draa et al., 2001).

Le contrôle centralisé suppose l'existence d'un agent centralisateur possédant toutes les connaissances pour gérer l'interaction des agents du système et le comportement global de celui-ci. Il peut modifier ce comportement de manière dynamique en planifiant les actions des agents. Le contrôle planifié centralisé distingue encore deux cas :

- la planification centralisée pour agents multiples suppose qu'il existe un agent planificateur central qui traite la coordination des actions et l'allocation des tâches. Les autres agents ne sont alors que des exécutants.
- la coordination centralisée par plans partiels ne centralise que la coordination. Chaque agent construit son plan partiel qu'il envoie à l'agent centralisateur qui tente de synthétiser tous les plans partiels en un plan global en éliminant les conflits potentiels.

Dans le contrôle distribué, un agent sélectionne lui-même les tâches qu'il peut accomplir et négocie des alliances avec les autres agents pour résoudre les tâches qu'il ne peut traiter tout seul. Le contrôle distribué peut se faire par planification distribuée : il n'existe pas d'agent centralisateur ni pour planifier de plan global ni pour coordonner un ensemble de plans partiels. Chaque agent planifie ses actions en fonction de ses propres buts. La difficulté porte sur la résolution de conflits potentiels mais aussi sur la gestion des situations dans lesquelles les actions des uns peuvent être utiles à la réalisation des buts des autres.

1.4.3.2. Les communications

Pour pouvoir interagir entre eux, les agents doivent être capable de communiquer. Chaque agent doit donc avoir la possibilité à la fois de recevoir et d'envoyer des messages. La question de la communication entre les agents soulève essentiellement deux problèmes : le langage de communication à utiliser et le protocole à utiliser pour mener à bien cette communication.

Plusieurs langages de communication spécialisés ont vu le jour. Le Knowledge Query and Manipulation Language (KQML) (DARPA, 1993), et plus récemment, le standard FIPA-ACL (ACL pour Agent Communication Language) (FIPA, 2002) créée par la Foundation for Intelligent Physical Agents (FIPA). Ce dernier standard repose en particulier sur la théorie des actes de langage de John Searle (Searl, 1969) où les messages sont des actions ou des actes communicatifs, car ils sont prévus pour effectuer une certaine action en vertu de l'envoi. Un exemple simple de message ACL peut être :

```
(inform
        :sender i
        :receiver j
        :content "in(clermont-ferrand, france)"
        :language Prolog)
```

L'agent *i* informe l'agent *j* (performative *inform*) que Clermont-Ferrand est en France (in(clermont-ferrand, france) en langage Prolog).

L'environnement AMICAL utilise le langage ACL.

Le protocole de communication permet de définir d'une manière abstraite la séquence de messages échangés dans le but d'effectuer une tâche donnée entre plusieurs agents. Un exemple de protocole est le FIPA-ContractNet-Protocol qui permet de définir comment effectuer un appel d'offres (FIPA, 2002b). Il existe différentes variantes au protocole : soit l'émetteur connait le destinataire, c'est ce que l'on appelle la communication point-à-point, soit l'émetteur envoie le message à tous les agents sans connaître le destinataire intéressé par l'information, c'est la communication par diffusion (broadcast). La communication point-à-point suppose que les liaisons de communication soient fixe contrairement à la diffusion. Cette dernière permet aux agents de changer de place ou bien de disparaître puis de réapparaitre en fonction des besoins.

1.5. Principales utilisations des systèmes multi-agents

Nous terminerons cette rapide présentation des systèmes mutli-agents en distinguant deux grandes utilisations de ceux-ci : d'une part, le développement de logiciels et d'autre part, la simulation de phénomènes sociaux ou biologiques.

En génie logiciel, la technologie des systèmes multi-agents est considérée comme une méthode nouvelle et prometteuse de construction de logiciels. Un système logiciel complexe pourrait être considéré comme une collection d'un grand nombre d'agents autonomes de petite taille, chacun ayant ses propres propriétés et fonctionnalités et interagissant entre eux. Dans le cas du génie logiciel, l'idée est donc d'utiliser les agents pour élaborer de nouvelles applications plus autonomes, plus intelligentes, plus distribuées, etc.

Dans le cas de la biologie ou des sciences humaines ou sociales, les systèmes multi-agents peuvent être utilisés pour la modélisation et la simulation de systèmes existants. Ces systèmes peuvent être biologiques, sociaux, économiques, etc. Par exemple, on peut modéliser les théories économiques sous forme de systèmes multi-agents où chaque individu peut être modélisé par un agent rationnel qui tend à maximiser sa satisfaction ou ses profits. On peut aussi modéliser les colonies de fourmis ou bien encore d'autres espèces animales.

Après ce rapide tour d'horizon des agents et des systèmes multi-agents, nous revenons sur l'environnement multi-agents AMICAL et nous en présentons son architecture avant de nous focaliser sur l'agent de cette thèse.

2. L'environnement multi-agents AMICAL

Le module tutoriel de l'environnement AMICAL est réalisé en tant que communauté d'agents cognitifs rationnels collaborant pour conduire une session didactique individualisée pour un apprenant particulier.

L'architecture de l'environnement AMICAL est préexistante à nos travaux et c'est donc dans le cadre de celle-ci que nous déterminerons une architecture pour l'agent gestionnaire du modèle de l'apprenant. Le choix des différents agents ainsi que les choix concernant les interactions entre ceux-ci ne sont pas de notre fait. Nos recherches n'ont pas porté sur la réflexion de cette architecture générale qui aurait, de toute façon, dépassé le cadre de cette thèse. Nous la prenons donc pour acquise et nous la présentons dans la suite de ce paragraphe.

Cette architecture a été déterminée à partir d'une analyse fonctionnelle du système. Cette analyse a consisté de partir du cycle fonctionnel de l'environnement que nous avons évoqué Figure 2.7 page 79 pour en déterminer les différents agents du système.

Nous rappelons que ce cycle fonctionnel est composé de trois tâches expertes :

1. la planification didactique visant à construire la séquence de travail à présenter à l'apprenant pour lui faire acquérir un certain nombre de connaissances,
2. la gestion de la session de travail visant à la présenter à l'apprenant sur l'écran et à mémoriser les interactions,
3. l'analyse de ces interactions pour construire et mettre à jour le modèle de l'apprenant.

De ce cycle ont été dégagés trois agents : un agent gestionnaire de la planification didactique (tâche 1), un agent gestionnaire des interactions apprenant-système (tâche 2) et un agent gestionnaire du modèle de l'apprenant (tâche 3). De plus, ces différentes tâches nécessitent des connaissances expertes qui ont été regroupées au

sein d'un quatrième agent : l'agent gestionnaire des expertises. Ces quatre agents sont considérés comme des agents de premier niveau. Le terme d'agent de premier niveau est utilisé parce que certains de ces agents seront eux-mêmes composés de sous-agents que l'on qualifiera alors d'agents de second niveau.

2.1. Architecture du module tutoriel AMICAL

Dans le module tutoriel du projet AMICAL, nous distinguons donc quatre agents (Figure 3.1) :

- un agent gestionnaire des expertises : AGE,
- un agent de planification didactique : AGPD,
- un agent responsable interface avec l'apprenant, appelé agent gestionnaire de la séquence de situations didactiques instanciées : AGSSDI
- un agent gestionnaire du modèle de l'apprenant : AGMA.

Il a été également choisi de rajouter un cinquième agent : l'agent médiateur. Celui-ci a pour rôle de faire le relais entre toutes les requêtes des agents.

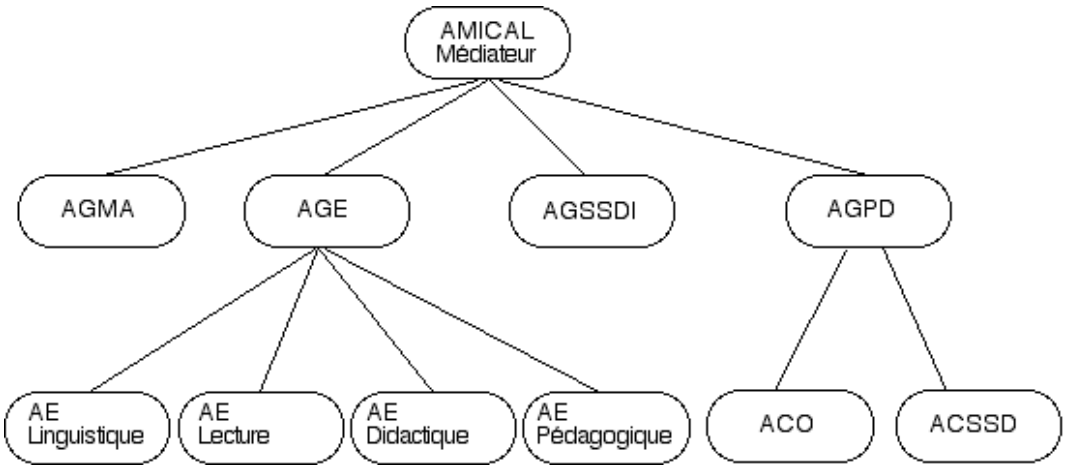

Figure 3.1 : Architecture multi-agents existante de l'environnement AMICAL

Les quatre agents du module tutoriel, sans compter l'agent médiateur, collaborent pour la conduite de sessions d'activités didactiques avec l'apprenant.

Par analogie avec les quatre composantes d'un tuteur intelligent, le module expert du domaine est gérée par AGE, l'interface est gérée par l'agent AGSSDI, le module pédagogique par l'agent AGPD et le modèle de l'apprenant par l'agent AGMA.

Nous décrivons maintenant les fonctionnalités de chacun des agents du module tutoriel.

2.1.1. L'agent gestionnaire des expertises

Comme nous le voyons sur la Figure 3.1, l'agent gestionnaire des expertises, AEG, est lui-même subdivisé en quatre sous-agents : l'agent expert linguistique, l'agent expert lecture, l'agent expert didactique et l'agent expert pédagogique. Nous présentons dans la suite rapidement ces quatre agents.

2.1.1.1. L'agent expert pédagogue

L'agent expert pédagogue, AEP, a pour rôle, comme son nom l'indique, d'être l'expert en pédagogie du système, c'est-à-dire un expert sur la façon d'enseigner. Il intervient essentiellement en phase de planification didactique. Il contribue à la construction d'un objectif d'enseignement correspondant à l'objectif que prévoit un enseignant pour une leçon de lecture avec un apprenant. L'agent expert pédagogue intervient également dans la détermination de la séquence d'activités proposées à l'apprenant.

2.1.1.2. L'agent expert linguistique

L'agent expert linguistique, AELi, a pour rôle de proposer des ressources linguistiques permettant une progression dans l'apprentissage. Ces ressources interviennent lors de la construction de l'objectif d'enseignement par le choix d'éléments à travailler. Une autre de ses tâches intervenant en parallèle de la détermination de la session d'activités est de choisir le texte support de la session (Quanquin, 2000). Nous choisissions en effet de baser l'apprentissage de la lecture sur la présentation d'un contexte signifiant en début de session.

2.1.1.3. L'agent expert lecture

L'agent expert lecture, AELe, est un agent ressource qui participe comme l'agent linguistique à la planification didactique. Il manipule des connaissances spécifiques à l'apprentissage de la lecture notamment sur les représentations du système de l'écrit, les relations de correspondance entre l'écrit et l'oral ou les aspects de compréhension du sens.

2.1.1.4. L'agent expert didactique

L'agent expert didactique, AED, a pour rôle, comme son nom l'indique, d'être l'expert en didactique du système, c'est-à-dire un expert qui gère les progressions sur les différentes connaissances à faire acquérir à l'apprenant. Il intervient essentiellement en phase de planification didactique. Il contribue lui aussi à la construction d'un objectif d'enseignement. Il intervient également dans la détermination de la séquence d'activités proposée à l'apprenant.

2.1.2. L'agent gestionnaire de la planification didactique

L'agent gestionnaire de la planification didactique (Cleder, 2002), (Cherkaoui, 1996), AGPD, a pour but de contribuer à construire un objectif d'enseignement puis de déterminer une séquence d'activités didactiques susceptible de répondre à l'objectif d'enseignement. Cet agent est lui même subdivisé en deux agents :

1. l'agent de construction de l'objectif : ACO
2. l'agent de construction de la séquence de situations didactiques : ACSSD

2.1.3. L'agent gestionnaire des séquences de situations didactiques instanciées

L'agent gestionnaire des séquences de situations didactiques instanciées (Mahmoud, 1997), AGSSDI, est celui qui va gérer l'affichage à l'écran de la séquence de travail. Il propose à l'apprenant des interfaces ou ateliers didactiques mettant en œuvre la séquence de situations didactiques construite par AGPD. Ces ateliers sont construits dynamiquement à partir des données fournies à l'issue de la planification didactique. Pendant l'interaction, il mémorise toutes les interactions

entre l'apprenant et le module tutoriel, qu'elles proviennent du système ou de l'apprenant.

2.1.4. L'agent gestionnaire du modèle de l'apprenant

L'agent gestionnaire du modèle de l'apprenant, AGMA, objet de cette thèse, construit le modèle de l'apprenant à partir de l'analyse des comptes-rendus d'interactions. Ce modèle de l'apprenant est notamment utilisé pour construire l'objectif et la session didactique.

2.1.5. L'agent médiateur

Comme on peut le noter sur la Figure 3.1, l'architecture de l'environnement multi-agents AMICAL se fonde sur l'organisation des communications par agent médiateur (Azmatally, 1997). Toutes les communications entre les agents de premier niveau passent donc par le médiateur qui centralise les communications. Les agents ne se connaissent donc pas directement : c'est le médiateur qui permet la communication.

L'agent médiateur permet aux autres agents de premier niveau de s'abonner aux différents services proposés par les différents agents de premier niveau. Il a donc un rôle de gestion des abonnements à ces services. L'abonnement permet à un agent de demander à être informé des événements susceptibles de l'intéresser. Les échanges entre un abonné et ses fournisseurs se terminent lorsque le demandeur se désabonne ou que l'émetteur ne peut plus émettre. Au début de la phase de résolution de problèmes ou dès qu'il apparaît dans l'architecture, chaque agent transmet au médiateur les tâches qu'il est capable de réaliser et les types d'événements qu'il est susceptible d'émettre. Ces annonces étant faîtes, chaque agent indique à quels événements il souhaite s'abonner. Les abonnements se font par rapport aux annonces déjà constituées. A partir de ces annonces, le médiateur construit des structures de données permettant de mettre en relation, d'une part, les agents qui demandent l'exécution d'une tâche avec ceux sachant la traiter, et d'autre part, les abonnés à un événement avec ceux qui les diffusent.

2.2. Collaboration des agents dans le module tutoriel

d'AMICAL

Nous présentons maintenant un scénario de collaboration de ces agents pour conduire une séquence de situations didactiques. Nous rappelons qu'il se déroule en trois phases : d'abord, la planification didactique visant à déterminer une séquence de situations didactiques pour un apprenant particulier puis, l'exécution de celle-ci en interaction avec l'apprenant et enfin la mise à jour du modèle de l'apprenant à partir de l'analyse du compte-rendu mémorisé durant l'exécution. Le processus est déclenché par la connexion d'un apprenant au module tutoriel. Nous n'évoquerons pas le rôle du médiateur dans ce scénario, mais il convient de se rappeler que toutes les communications entre les agents de premier niveau passent par lui.

2.2.1. Phase 1 : la planification didactique

Cette phase est réalisée par AGPD et ses sous-agents : ACO et ACSSD.

La première étape de cette phase consiste pour ACO à construire un objectif d'enseignement pour un apprenant particulier en collaboration avec AED, AELe, AELi et AGMA. AGPD demande à AGMA de lui fournir des éléments concernant l'apprenant. Il interroge ensuite AED, AELe et AELi pour lui fournir des connaissances à faire acquérir à l'apprenant pour le faire progresser dans son apprentissage. Une fois cet objectif, composé d'UO, construit, ACO l'envoie à l'autre sous-agent d'AGPD : ACSSD.

La seconde étape consiste, une fois l'objectif d'enseignement fixé, pour ACSSD à déterminer une séquence de situations didactiques types ou SDT permettant d'atteindre l'objectif. ACSSD va ensuite individualiser la séquence de SDT et déterminer la séquence de situations didactiques instanciées ou SDI en déterminant les paramètres de chaque SDT. Par exemple, ACSSD va déterminer en fonction de l'objectif de la séquence et du modèle de l'apprenant, les mots à identifier, le texte proposé ou encore les aides proposées à l'apprenant par exemple. L'individualisation conduite par ACSSD se fait en collaboration avec AEP, AGMA et AELe.

La séquence de SDI est alors transmise par AGPD à AGSSDI pour exécution

auprès de l'apprenant.

2.2.2. Phase 2 : exécution de la séquence de SDI

Nous nous plaçons dans l'hypothèse d'un scénario de fonctionnement sans replanification en cours d'exécution d'une séquence de SDI, c'est-à-dire que la séquence ne change pas en cours d'exécution par exemple pour s'adapter aux difficultés de l'apprenant.

AGSSDI exécute chaque SDI de la séquence en interaction avec l'apprenant. Il enregistre tous les événements survenus pendant l'interaction, qu'ils soient initiés par le système ou par l'apprenant. AGSSDI poste le compte-rendu dans l'environnement du système multi-agents, c'est-à-dire qu'il va le stocker dans la base des comptes-rendus destinée à les mémoriser. A la fin de la séquence, AGSSDI informe à la fois AGPD et AGMA que la session de travail est terminée et que le compte-rendu a été posté.

2.2.3. Phase 3 : la mise à jour du modèle de l'apprenant

AGMA reçoit l'information que la session est terminée et donc que le compte-rendu est disponible. AGMA va aller récupérer le compte-rendu déposé par AGSSDI dans la base des comptes-rendus.

Une fois le compte-rendu récupéré, AGMA va pouvoir commencer l'élaboration du modèle de l'apprenant. Il commence donc par analyser chacune des situations didactiques de la session de travail où il va émettre des hypothèses sur l'état de savoir de l'apprenant. Ensuite, il va intégrer ces hypothèses au modèle de l'apprenant déjà existant en le modifiant en conséquence. La mise à jour du modèle de l'apprenant peut nécessiter de faire appel aux agents experts d'AMICAL.

Après avoir présenté l'architecture déjà existante de l'environnement multi-agents AMICAL, nous allons à présent nous intéresser à l'agent qui a fait l'objet de nos recherches : l'agent gestionnaire du modèle de l'apprenant.

3. L'Agent Gestionnaire du Modèle de l'Apprenant : vers une architecture multi-agents

Le but de cette partie est de présenter l'architecture retenue pour notre agent gestionnaire du modèle de l'apprenant. Pour cela, nous commencerons par présenter les différents rôles qu'il devra remplir, puis à partir de ceux-ci, nous en déduirons son architecture. Enfin, nous expliquerons le fonctionnement général de cette architecture pour chacun de ses rôles.

3.1. Rôles d'AGMA au sein de l'environnement multi-agents

Paiva a défini six types de tâches que devrait réaliser un agent de modélisation de l'apprenant au sein d'un système multi-agents (Paiva, 1996) :

1. *des tâches informatives* : l'agent de modélisation doit être capable d'informer d'autres agents du contenu du modèle de l'apprenant,
2. *des tâches d'acquisition* : l'agent doit pouvoir inférer de nouvelles informations sur les apprenants qu'il modélise en fonction d'éléments fournis par les autres agents,
3. *des tâches de diagnostic* : l'agent doit détecter des situations inconsistantes et des événements imprévus apparaissant dans le modèle de l'apprenant,
4. *des tâches de maintenance* : l'agent doit gérer et mettre à jour les données du modèle de l'apprenant,
5. *des tâches d'interrogation* : l'agent doit pouvoir interroger d'autres agents lorsqu'il a besoin de données ou de ressources pour accomplir ses tâches,
6. *des tâches de réponse* : l'agent doit pouvoir satisfaire les demandes des autres agents même s'il doit pour cela effectuer des inférences.

Nous choisissons de séparer ces types de tâches en deux ensembles : un ensemble « modélisation de l'apprenant » et un ensemble « réponse aux requêtes ».

Le premier ensemble « modélisation de l'apprenant » regroupe les tâches d'acquisition, de maintenance, de diagnostic et d'interrogation. Il concerne directement le comportement propre de l'agent, son objectif principal étant de construire et gérer le modèle de l'apprenant. Cette construction passe par une phase d'analyse consistant à interpréter des résultats d'interaction (tâche

d'acquisition) entre le module tutoriel et l'apprenant, et par une phase visant à mettre à jour le modèle de l'apprenant en fonction des résultats de l'analyse (tâches de diagnostic et de maintenance). Si AGMA ne possède pas toutes les connaissances ou compétences pour assurer seul ces deux phases, il pourra éventuellement faire appel à des informations extérieures en interrogeant d'autres agents (tâche d'interrogation).

Le deuxième ensemble « réponse aux requêtes » regroupe les tâches informatives et de réponse. Il vise à renseigner les agents d'AMICAL sur le contenu du modèle de l'apprenant.

Chacun de ces deux ensembles représente un rôle spécifique à remplir par AGMA. Celui-ci doit donc remplir deux rôles :

1. *modéliser l'apprenant* : AGMA doit construire le modèle de l'apprenant à partir des comptes-rendus et mettre à jour le modèle en conséquence,
2. *répondre aux requêtes* : AGMA doit répondre aux requêtes des différents autres agents du système concernant le contenu du modèle de l'apprenant.

C'est dans ces deux rôles que l'architecture d'AGMA sera envisagée.

3.2. Analyse des rôles d'AGMA

Pour déterminer l'architecture interne de notre agent, nous analysons plus précisément chacun de ses rôles et nous en dégagerons les éléments essentiels devant faire partie de l'architecture retenue.

Comme nous l'avons dit, la modélisation de l'apprenant passe essentiellement par deux phases :

1. l'analyse de la séquence de situations didactiques présentée à l'apprenant afin d'émettre des hypothèses quant à l'état de savoir de celui-ci,
2. l'intégration de ces hypothèses au sein du modèle de l'apprenant déjà existant.

De ces deux phases, on peut donc retenir que :

- AGMA doit posséder le modèle de l'apprenant,
- AGMA doit avoir des mécanismes de mise à jour de ce modèle,

- AGMA doit avoir des capacités de communication pour, le cas échéant, pouvoir demander des informations aux autres agents du système,
- AGMA doit avoir des mécanismes d'analyse de la séquence des situations didactiques.

Les deux premiers points peuvent se regrouper en un seul car tous les deux sont liés au modèle de l'apprenant. Nous choisissons donc d'intégrer la mise à jour du modèle de l'apprenant au modèle de l'apprenant lui-même.

La réponse aux requêtes des autres agents du système se déroule d'une manière relativement simple : un agent envoie une requête concernant l'apprenant, AGMA recherche alors dans le modèle de l'apprenant pour voir s'il trouve la réponse, puis il envoie la réponse à l'agent ayant effectué la requête.

De ce scénario, on peut donc retenir que :

- AGMA doit posséder le modèle de l'apprenant pour pouvoir répondre aux questions sur celui-ci,
- AGMA doit avoir des capacités de communication pour recevoir les requêtes et envoyer les réponses.

Les deux points précédents correspondent aux points évoqués dans le rôle de modélisation de l'apprenant : disposer du modèle de l'apprenant et pouvoir communiquer avec l'extérieur.

Si l'on synthétise les éléments que doit posséder AGMA, on peut donc dégager trois fonctions que l'agent doit remplir :

1. il doit être capable de communiquer avec les autres agents du système,
2. il doit être capable d'analyser la séquence de situations didactiques,
3. il doit être capable de disposer du modèle de l'apprenant et de le maintenir à jour.

3.3. Architecture générale retenue

Concevoir un agent monolithique devant réaliser ces différentes fonctions serait complexe et ne serait pas très efficace. C'est pour cela que nous choisissons de réaliser chacune des fonctions précédentes par un module particulier d'AGMA.

Celui-ci sera donc décomposé en trois modules :

1. le *module analyse* : il sera responsable d'analyser les comptes-rendus des séquences de situations didactiques afin d'en émettre toutes les hypothèses possibles quant à l'état de savoir de l'apprenant,
2. le *module modèle* : il sera responsable à la fois de stocker le modèle de l'apprenant et de le mettre à jour en intégrant les hypothèses en provenance du module analyse,
3. le *module communication* : il sera responsable de gérer toutes les communications d'AGMA avec le reste du système. Ce sera en quelque sorte « la gare de triage » d'AGMA, c'est-à-dire que toutes les communications à destination et en provenance de l'extérieur passeront par ce module. Il sera donc également responsable d'aller récupérer les comptes-rendus et de les transmettre au module analyse.

Bien évidemment, ces modules ne vont pas travailler de manière isolée mais doivent travailler de concert pour réaliser les tâches de l'agent. Il doivent donc pouvoir communiquer entre eux. Il existe six liens de communication entre ces modules :

- entre le *module communication* et le *module modèle* : dans le cas d'une requête de la part d'un autre agent du système, le « module communication » devra pouvoir accéder au modèle de l'apprenant et donc communiquer avec le module modèle,
- entre le *module modèle* et le *module communication* : dans le cas de la réponse à la requête précédente, le module modèle doit pouvoir répondre au module communication,
- entre le *module communication* et le *module analyse* : dans le cas de la modélisation de l'apprenant, le compte-rendu doit être transmis au module analyse. En effet, comme nous l'avons dit, le module communication est le seul à pouvoir communiquer avec le reste du système. De ce fait, c'est le module communication qui va aller rechercher le compte-rendu et le transmettre au module analyse,
- entre le *module analyse* et le *module communication* : dans le cas où le module analyse aurait besoin de l'expertise d'un autre agent du système, et comme toutes les communications d'AGMA passent par le module communication, le module analyse devra donc passer lui aussi par le module communication,
- entre le *module analyse* et le *module modèle* : dans le cas de la

modélisation de l'apprenant, les hypothèses émises par le module analyse doivent être intégrées au modèle de l'apprenant existant, le module analyse doit donc les envoyer au module modèle,

- entre le *module modèle* et le *module analyse* : dans le cas où pour effectuer des hypothèses, le module analyse a besoin d'une information déjà contenue dans le modèle de l'apprenant, le module modèle doit donc pouvoir lui répondre.

Ces six liens sont donc en fait trois liens bidirectionnels. De plus, il existe également un autre lien bidirectionnel entre le module communication et le reste du système. Dans un sens, quand un agent fait une requête sur le contenu du modèle de l'apprenant. Dans l'autre sens, pour envoyer la réponse à cette requête ou si, par exemple, pour effectuer l'analyse d'une situation didactique, le module analyse a besoin de la collaboration d'un autre agent du système.

L'architecture modulaire retenue pour AGMA est donc la suivante :

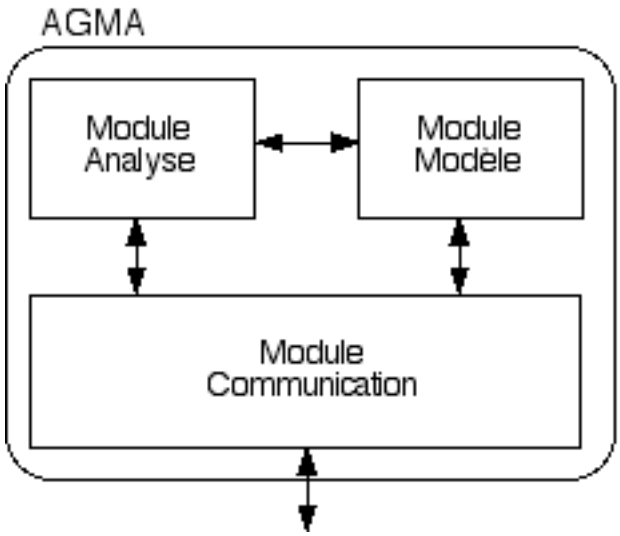

Figure 3.2 : Architecture modulaire d'AGMA

3.4. Contraintes supplémentaires

Dans le cadre de l'apprentissage de la lecture, le temps de réaction de l'environnement est un paramètre non négligeable, essentiellement parce que les

apprenants sont de jeunes enfants et que le temps d'attente doit être le plus court possible pour que ceux-ci ne s'ennuient pas trop vite.

Comme AGMA est au coeur de ce système en raison de l'importance du modèle de l'apprenant, celui-ci doit donc perdre le moins de temps possible à la fois pour modéliser l'apprenant, et donc disposer très rapidement des nouvelles informations sur son état de savoir, que pour le communiquer aux autres agents travaillant avec l'apprenant.

De plus, et même si ce n'est pas l'objet de cette thèse, dans l'hypothèse où AGMA devrait travailler avec plusieurs apprenants en même temps, il devrait là aussi réaliser le plus rapidement possible ses tâches, à savoir la modélisation des apprenants dans ce cas et la réponse aux requêtes du contenu des différents modèles. Dans cette hypothèse, une autre contrainte qui pourrait être fixée à AGMA serait celle de pouvoir réaliser ses deux rôles en même temps : pendant qu'il analyse le compte-rendu d'un apprenant, il devrait être capable de répondre à des requêtes concernant le contenu du modèle d'un autre apprenant. L'évocation de l'hypothèse du travail avec plusieurs enfants n'est citée ici qu'à titre purement informatif et dans l'unique but de souligner la nécessité d'avoir un agent le plus réactif possible.

Ainsi la conception de ces trois modules devra donc tenir compte de cette rapidité d'exécution des différentes tâches.

3.5. Architecture détaillée

Nous proposons donc pour résoudre ce problème de rapidité d'exécution de paralléliser au maximum la réalisation des différentes tâches de l'agent. Nous avons donc opté, dans le cas d'AGMA, de représenter les différents modules comme des systèmes multi-agents où les différents agents pourront travailler en parallèle. Chaque module pourra donc effectuer sa tâche le plus rapidement possible. Voyons-en maintenant les raisons pour chacun des trois modules.

Le module analyse est responsable d'analyser le compte-rendu de la séquence de travail que le système a eu avec l'apprenant, c'est-à-dire d'émettre toutes les hypothèses possibles quant à l'état de savoir de l'apprenant. Comme nous l'avons

dit, une séquence de travail est un ensemble de situations didactiques. Analyser ces situations didactiques les unes après les autres serait long et fastidieux. De plus, nous avons également dit que chaque situation didactique portait sur des connaissances différentes de l'apprenant. De ce fait, chaque situation didactique doit être analysée différemment des autres. Il n'y a donc pas de processus unique permettant d'analyser toutes les situations didactiques. Une façon simple de combiner à la fois la parallélisation nécessaire à la rapidité de l'analyse et la spécialisation de l'analyse de chaque situation didactique, est de représenter le module analyse comme un système multi-agents où chaque agent sera spécialisé dans l'analyse d'un seul type de situation didactique. Ainsi, par exemple, on aura un agent capable d'analyser une situation didactique de type « identification de mots en contexte » et uniquement celle-là. Nous appellerons ces agents : les agents d'analyse ou AA.

Le module modèle est responsable de stocker et de maintenir à jour le modèle de l'apprenant en intégrant les hypothèses du module analyse. Nous rappelons que nous avons limité le modèle de l'apprenant aux connaissances du domaines. Nous avons expliqué que celles-ci étaient des objets complexes multi-niveaux au chapitre 2. La première approche (Fragne et al., 2005), (Fragne et al., 2007), (Fragne, 2007), que nous avons adoptée a été de représenter ces connaissances du domaine par des objets. Un processus était en charge de mettre à jour ces connaissances à partir des hypothèses émises. Malheureusement, le gros inconvénient de cette structure d'objets précédemment imaginée était la complexité de la maintenance d'une telle structure. Là encore, pour augmenter les performances de mise à jour et pour faciliter la mise à jour en décomposant le problème en sous-problèmes, nous avons décidé de décomposer notre modèle de l'apprenant en autant de connaissances du domaine qui existaient et de laisser un agent particulier s'occuper de cette connaissance particulière. Ainsi, chacun des agents disposera d'une et une seule connaissance du domaine qu'il aura pour charge de mettre à jour, en fonction des hypothèses qu'il recevra de la part des agents d'analyse. On aura donc autant d'agent qu'il y aura de connaissances du domaine. Le module modèle est donc lui aussi représenté par un système multi-agents. De ce fait, le modèle de l'apprenant est donc représenté lui aussi sous forme de système multi-agents où chaque agent dispose d'une connaissance particulière qu'il est capable de mettre à jour en fonction des hypothèses qu'il reçoit des différents agents d'analyse. Nous appellerons ces agents : les agents de connaissance ou AK, le K désignant ici knowledge pour éviter la confusion avec

le C de AC, désignant lui communication.

Le rôle du module de communication est, comme nous l'avons dit, de faire l'interface entre les modules composants AGMA et le reste du système. Dans le cas de la modélisation de l'apprenant, nous avons vu que plusieurs agents d'analyse travaillaient en parallèle. Nous avons également dit que ce module pouvait, dans certains cas, avoir besoin de l'aide des agents experts du système. Mais si plusieurs agents d'analyse ont besoin de l'aide des experts en même temps, n'avoir qu'un module monolithique ralentirait grandement l'analyse de ces mêmes agents. De ce fait, l'idée consiste donc à avoir plusieurs agents permettant cette communication, chacun dans le cas précédent, s'occuperait de la demande d'un agent d'analyse. La disponibilité d'une multitude d'agents permettrait également de traiter le cas du travail avec plusieurs apprenants. Dans ce cas, par exemple, un agent irait récupérer le compte-rendu de la séquence, un autre gérerait la communication en provenance d'un autre agent du système, et un troisième s'occuperait de la demande d'un agent d'analyse ayant besoin d'une expertise particulière. Réaliser le module communication sous forme de système multi-agents est donc une excellente option. Nous appellerons ces agents : les agents de communication ou AC.

Finalement, l'architecture détaillée de notre agent est la suivante (nous ne représentons sur le schéma que quelques agents pour chaque module) :

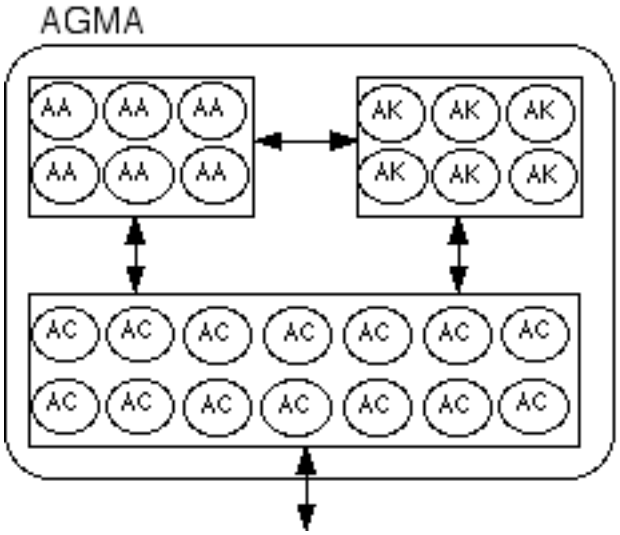

Figure 3.3 : Architecture détaillée d'AGMA

Les agents composant les différents modules d'AGMA étant tous de granularité plus faible qu'AGMA lui-même, ils disposent donc de capacités plus restreintes. Néanmoins, tous ces agents restent des agents cognitifs dans le sens où ils disposent toujours de connaissances explicites leur permettant d'exécuter leurs tâches.

A partir de maintenant, quand nous ferons référence aux modules d'AGMA, nous ferons en fait référence à tous les agents composant ceux-ci.

Dans la suite, nous présentons un peu plus en détail le rôle des agents constituant ces modules. Néanmoins, nous ne rentrerons dans le détail de chaque agent que dans le chapitre suivant.

3.5.1. Module Communication

Le module communication est responsable de gérer toutes les communications entre les modules d'AGMA et les autres agents du système. Il est composé d'agents de communication ou AC.

Dans le cas de la modélisation de l'apprenant, un AC est responsable, en tant que gestionnaire des communications entre AGMA et le reste des agents du système, de :

- recevoir les messages indiquant que la session de travail est terminée et donc que le compte-rendu est disponible,
- récupérer le compte-rendu de la séquence dans la base des comptes-rendus : comme les messages précédents lui indiquent que le compte-rendu est disponible et que celui-ci sait qu'AGMA en a besoin pour modéliser l'apprenant, il va donc aller le chercher,
- décomposer le compte-rendu en autant de sous-comptes-rendus qu'il y a de situations didactiques dans la séquence : nous avons également choisi, pour faciliter le travail des agents d'analyse qui eux ne sont responsables que d'analyser une situation didactique particulière, de lui laisser cette charge.
- d'envoyer ces sous-comptes-rendus aux agents d'analyse concernés,
- de s'occuper des demandes en provenance des agents d'analyse qui ont besoin de l'expertise d'un autre agent du système.

Dans le cas de la réponse aux requêtes des autres agents du système, un AC est responsable de :

- recevoir les requêtes des agents du système,
- comprendre la requête, éventuellement avec l'aide des différents agents experts,
- interroger les différents agents de connaissance pour pouvoir répondre à la requête,
- envoyer la réponse à l'agent ayant effectué la requête.

Par exemple, si AGPD a besoin de la liste des voyelles connues par l'apprenant, AC devra d'abord interroger l'agent expert linguistique pour connaître la liste des voyelles. Ensuite, il devra interroger chaque agent de connaissance correspondant à chaque voyelle pour en connaître le statut. Il construira ensuite la liste des voyelles connues en ne retenant que celles qui ont un statut égal à connu. Enfin, il enverra cette liste à AGPD.

Comme nous l'avons dit, l'architecture du système AMICAL a été réalisée comme devant utiliser un médiateur pour gérer ses communications. Chaque AC doit donc s'enregistrer auprès du médiateur en se « faisant passer » pour AGMA. De ce fait, quand le médiateur voudra envoyer un message à destination d'AGMA, il sélectionnera, via un mécanisme que nous ne détaillerons pas ici, le premier AC disponible et lui enverra le message.

3.5.2. Module Analyse

Le module analyse est responsable d'analyser les différentes situations didactiques de la séquence de travail. Il est composé d'agents d'analyse ou AA.

Chaque agent d'analyse est responsable d'analyser un et un seul type de situation didactique, c'est-à-dire qu'il a pour charge d'émettre toutes les hypothèses possibles quant à l'état de savoir de l'apprenant sur cette situation didactique particulière.

Un AA reçoit un compte-rendu d'un AC lui demandant d'analyser la situation didactique. Une fois ce compte-rendu en sa possession, il va analyser celui-ci et

émettra toutes les hypothèses possibles sur l'état de savoir de l'apprenant. Une fois ces hypothèses émises, il les enverra aux agents de connaissance concernés par la situation didactique. En effet, comme nous l'avons dit, chaque situation didactique porte sur un ensemble de connaissances précis et donc correspond à certain nombre d'agents de connaissance. Par exemple, un AA « identification de mots en contexte » recevra, si la séquence a contenu cette situation didactique, le sous-compte-rendu correspondant de la part d'un AC. Il analysera ensuite celui-ci et émettra toutes les hypothèses possibles sur l'état de savoir uniquement pour cette situation didactique « identification de mots en contexte ». Cet AA enverra ensuite les hypothèses aux AK concernés. En effet, AA sait que cette situation didactique porte sur : le concept de mot, différents objets mots ainsi que différentes stratégies. Il enverra donc les hypothèses concernant le concept de mot à l'AK en charge de ce concept, les hypothèses concernant les objets mots à tous les AK en charge de ces objets mots, et les hypothèses sur les stratégies aux AK en charge de ces stratégies.

Bien qu'il eut été possible de concevoir un fonctionnement différent pour chaque AK, notamment du fait que chaque situation didactique porte sur des connaissances de types différents, nous avons choisi, dans nos recherches, de concevoir tous les AK de la même manière et dotés du même fonctionnement. Nous détaillerons ce fonctionnement ainsi que les raisons de ce choix au chapitre 4.

Nous ajouterons deux autres intérêts à avoir utilisé des agents d'analyse au sein du module analyse :

- d'un point de vue de l'ingénierie des connaissances auprès de nos experts, cette approche est plus facile à réaliser : nous concevons situation didactique par situation didactique et donc nous créons les AA au fur et à mesure,
- cette approche est plus facilement évolutive. En cas d'ajout de situation didactique, il suffit simplement d'ajouter les AA correspondants,

3.5.3. Module modèle

Le module modèle est responsable de stocker et maintenir à jour le modèle de l'apprenant en fonction des hypothèses reçues en provenance des AA. Il est

composé d'agents de connaissances ou AK.

Un agent de connaissance est en charge de stocker et maintenir une connaissance unique du modèle de l'apprenant. Le stockage de cette connaissance n'est pas externe à l'agent mais fait bien entendu partie de ses croyances. La mise à jour de cette connaissance correspond donc aussi, par définition, à mettre à jour ses propres croyances.

Un AK reçoit donc des hypothèses en provenance de plusieurs AA. En effet, comme plusieurs situations didactiques portent sur les mêmes connaissances, des hypothèses sur ces connaissances sont donc émises par plusieurs AA différents. Nous rappelons que la connaissance dont a la charge un AK est un objet complexe multi-niveaux. Ensuite, il intègre ces hypothèses et met à jour en conséquence la connaissance dont il a la charge, et donc il met à jour ses propres croyances, en faisant évoluer le statut d'un niveau de détail de la connaissance et en le répercutant, le cas échéant, sur les niveaux de détail supérieurs.

Le modèle de l'apprenant est donc entièrement réparti au sein des différents AK composant le module modèle. D'un point de vue formel, on peut représenter le modèle de l'apprenant de la façon suivante :

$$MA = \cup_i \text{Bel}(AK_i)$$

où MA représente le modèle de l'apprenant, $\text{Bel}(AK_i)$ représente les croyances (une partie tout du moins) d'un AK. Le modèle de l'apprenant est donc constitué d'une myriade d'AK. Il y aura donc autant d'AK que de connaissances dans le modèle de l'apprenant. Dans le cas des concepts et des stratégies, il existe un AK pour chaque concept et pour chaque stratégie. Par contre, dans le cas des objets, il existe autant d'AK qu'il y a d'instances de types d'objets. Par exemple, concernant les objets lettres, il existera 26 AK, un pour chaque lettre, et concernant les objets mots, il existera autant d'AK que de mots travaillés avec l'apprenant. Il existera donc par exemple un AK sur l'objet mot Arthur et un autre AK sur l'objet mot voiture. Il n'existe pas d'AK correspondant aux types d'objets eux-mêmes. Par exemple, il n'existe pas d'AK pour l'objet lettre ou pour l'objet mot. En effet, ce qui nous intéresse n'est pas la connaissance sur un type d'objet mais bien la connaissance sur l'instance d'un type d'objet.

Bien que nous ayons évoqué au chapitre 2 l'existence de trois types de connaissances : concepts, objets et stratégies, la structure de celles-ci reste identique. Seuls différent leurs sous-connaissances, leurs contextes et leurs statuts. De ce fait, il n'y a donc pas de différences d'architecture et de fonctionnement entre les AK. Chaque AK fonctionne de la même manière. Ceux-ci ne tiennent juste compte que de la spécificité de la connaissance qu'il gère, c'est-à-dire des sous-connaissances, des contextes et des statuts particuliers à cette connaissance. Nous verrons au chapitre 4 comment ceux-ci fonctionnent.

L'intégration des hypothèses au sein du modèle de l'apprenant pourrait éventuellement se dérouler en deux étapes :

1. au sein de chaque AK. Chaque agent met à jour sa propre base de connaissances à partir des hypothèses reçues,
2. éventuellement entre les AK. Chaque agent de connaissance envoie ses conclusions aux agents susceptibles d'être intéressés.

Néanmoins, nos recherches n'ont porté que sur la première étape. La deuxième fera partie des perspectives de ce travail.

Toutes les communications au sein de d'AGMA se déroulent via des messages ACL (FIPA, 2002).

3.6. Fonctionnement général d'AGMA

Nous exposerons ici le fonctionnement général de l'agent dans ses deux rôles : la modélisation de l'apprenant et la réponse aux requêtes. Le chapitre suivant rentrera davantage dans le détail des agents et de leur fonctionnement sur un exemple concret.

3.6.1. Initialisation d'AGMA

Pour des raisons de facilité d'échange avec la multitude d'agents existant au sein d'AGMA, nous avons convenu d'utiliser l'architecture abstraite proposée par FIPA (FIPA, 2002c) au sein de l'architecture multi-agents d'AGMA.

Cette architecture fournit aux différents agents des services pour communiquer entre eux plus facilement et plus sûrement. Cette architecture propose l'utilisation des services de :

1. *message transport* : ce service fournit un moyen sûr aux agents pour transmettre leurs messages,
2. *ACL* : ce service fournit la gestion d'ACL,
3. *agent directory* : ce service permet à un agent de s'enregistrer. De cette manière, les autres agents, en interrogeant ce service, peuvent retrouver facilement cet agent pour communiquer avec lui. On peut comparer ce service aux pages blanches.
4. *service directory* : ce service permet à un agent de s'enregistrer mais cette fois en enregistrant les services qu'il est capable de fournir. Les autres agents, en cas de besoin d'un service particulier, interrogent le *service directory* pour connaître quels agents sont susceptibles de fournir ce service et ainsi pouvoir le contacter. On peut comparer ce service aux pages jaunes.

Nous y reviendrons au chapitre 5, mais c'est pour cela que nous avons utilisé la plateforme Jade car celle-ci fournit nativement ces services.

L'idée est d'intégrer au sein de notre architecture la possibilité pour tous les agents d'AGMA de se rechercher suivant les services qu'ils rendent. C'est pour cela que le *service directory* nous semble très pertinent à utiliser. En effet, au lieu que chaque agent d'AGMA soit obligé de connaître directement les autres agents, il lui suffit de rechercher les agents dont il a besoin auprès du *service directory*. Cela nous fournit l'avantage d'avoir une architecture plus souple et plus dynamique car on pourrait imaginer que les agents concernés puissent changer ou bien ne plus se trouver au même endroit.

Nous détaillerons au chapitre 5, lors de l'implémentation du prototype de l'agent, les conventions de nommage que nous avons retenues pour nos agents. Dans les deux scénarios que nous explorerons dans la fin de ce chapitre, nous avons désigné les agents simplement en les nommant avec le nom de l'élément dont ils ont la charge. Par exemple, un agent responsable de la connaissance K1 sera désigné par AK-K1, alors qu'un agent d'analyse en charge la situation didactique SD1 sera nommé AA-SD1.

Au démarrage d'AGMA, chaque agent s'enregistre auprès du *service directory* :

- chaque AA, pour dire quelle activité il est capable d'analyser,
- chaque AK, pour dire de quelle connaissance il est en charge,
- chaque AC, pour dire qu'il est responsable des communications.

3.6.2. Collaboration des différents modules

Nous présentons ici comment fonctionnent les différents modules pour modéliser l'apprenant et pour répondre aux requêtes des autres agents du système. Comme cette thèse se centre essentiellement sur la modélisation de l'apprenant, nous présenterons uniquement le scénario d'une requête simple en provenance d'AGPD, c'est-à-dire une requête qu'AGMA est capable de traiter seul et donc qui ne nécessite pas l'intervention des agents experts de l'environnement AMICAL.

3.6.2.1. Collaboration des agents dans le rôle « modélisation de l'apprenant »

Nous présentons ici volontairement un exemple fictif permettant de comprendre facilement le fonctionnement d'AGMA. En effet, le scénario de fonctionnement restera le même quelle que soit la séquence de travail et les situations didactiques contenues dans celle-ci, c'est pour cela que nous choisissons une séquence de travail relativement simple. La séquence de travail de l'exemple comporte deux situations didactiques : SD1 et SD2. SD1 portera sur les connaissance K1 et K2, alors que SD2 portera sur la connaissance K2. Les agents qui seront mobilisés dans cet exemple seront les suivants : AC-1, AA-SD1, AA-SD2, AK-K1 et AK-K2, respectivement l'agent de communication numéro 1, l'agent d'analyse responsable de la situation didactique SD1, l'agent d'analyse responsable de la situation didactique SD2, l'agent de connaissance responsable de la connaissance K1 et enfin l'agent de connaissance responsable de la connaissance K2.

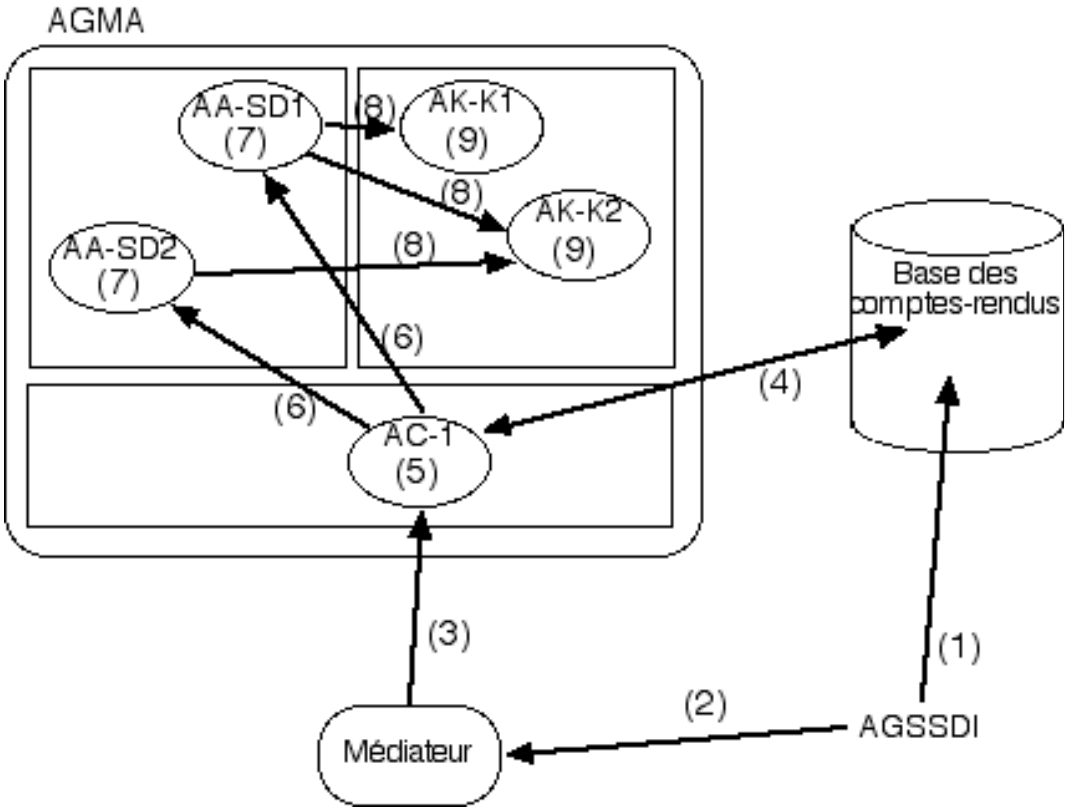

Figure 3.4 : Collaboration des modules d'AGMA pour modéliser l'apprenant

Le scénario se déroule de la manière suivante :

(1) une fois la séquence de travail terminée, l'agent gestionnaire des séquences de situations didactiques instanciées (AGSSDI) poste le compte-rendu dans la base des comptes-rendus,
(2) AGSSDI envoie ensuite un message au médiateur demandant de prévenir que la session est terminée,
(3) le médiateur informe donc AGMA au travers du premier AC disponible, que la session est terminée, AC-1,
(4) comme AC-1 sait qu'AGMA a besoin du compte-rendu pour mettre à jour le modèle de l'apprenant, il va donc aller le chercher,
(5) une fois le compte-rendu de la session récupéré, AC-1 va regarder son contenu et va le découper en autant de sous-comptes-rendus qu'il y a de situations didactiques. Sur l'exemple ci-dessus, la session de travail a

contenu deux situations didactiques, il y aura donc deux sous-comptes-rendus.

(6) chaque sous-compte-rendu est envoyé à l'AA correspondant. Sur l'exemple ci-dessus, la séquence a comporté deux situations didactiques : SD1 et SD2. AC-1 va donc envoyer le sous-compte-rendu de la situation didactique SD1 à l'agent en charge d'analyser SD1, AA-SD1 et le sous-compte-rendu de la situation didactique SD2 à l'agent en charge d'analyser SD2, AA-SD2. Dans le fonctionnement normal d'AGMA, AC-1 irait d'abord interroger le service directory pour savoir quels AA sont responsables des situations didactiques, puis enverra les sous-comptes-rendus à ces AA,

(7) chaque AA ayant reçu un sous-compte-rendu à exploiter va émettre toutes les hypothèses possibles. Sur l'exemple, AA-SD1 et AA-SD2 vont émettre des hypothèses quant à l'état de savoir de l'apprenant,

(8) une fois ces hypothèses émises, elles sont envoyées aux agents qui le nécessitent. Sur l'exemple, comme SD1 porte sur K1 et K2, AA-SD1 va envoyer les hypothèses sur K1 à l'agent de connaissance en charge de K1, AK-K1 et les hypothèses sur K2 à l'agent de connaissance en charge de K2, AK-K2 et comme SD2 porte uniquement sur K2, AA-SD2 va envoyer les hypothèses sur K2 à l'agent de connaissance en charge de K2, AK-K2. Là encore, dans le fonctionnement normal d'AGMA, chaque AA ira interroger le service directory pour savoir quels agents sont responsables des connaissances ayant attrait à la situation didactique qu'il analyse,

(9) Chaque AK ayant reçu de nouvelles hypothèses de la part des AA met à jour sa base de connaissances en conséquence. Sur l'exemple, AK-K2 reçoit des hypothèses à la fois de AA-SD1 et AA-SD2, alors que AK-K1 ne reçoit des hypothèse que de AK-K1.

3.6.2.2. Collaboration des agents dans le rôle « réponse aux requêtes »

Nous prendrons ici un exemple très simple où l'agent est capable de répondre directement à la requête. AGPD demande à AGMA de lui donner le statut de la sous-connaissance SK1 de la connaissance K2 dans le contexte C1. Les agents d'AGMA mobilisés dans cet exemple sont les suivants : AC-1 et AK-K2, respectivement l'agent de communication numéro 1 et l'agent de connaissance responsable de la connaissance K2. Nous supposerons que la réponse à la requête

existe.

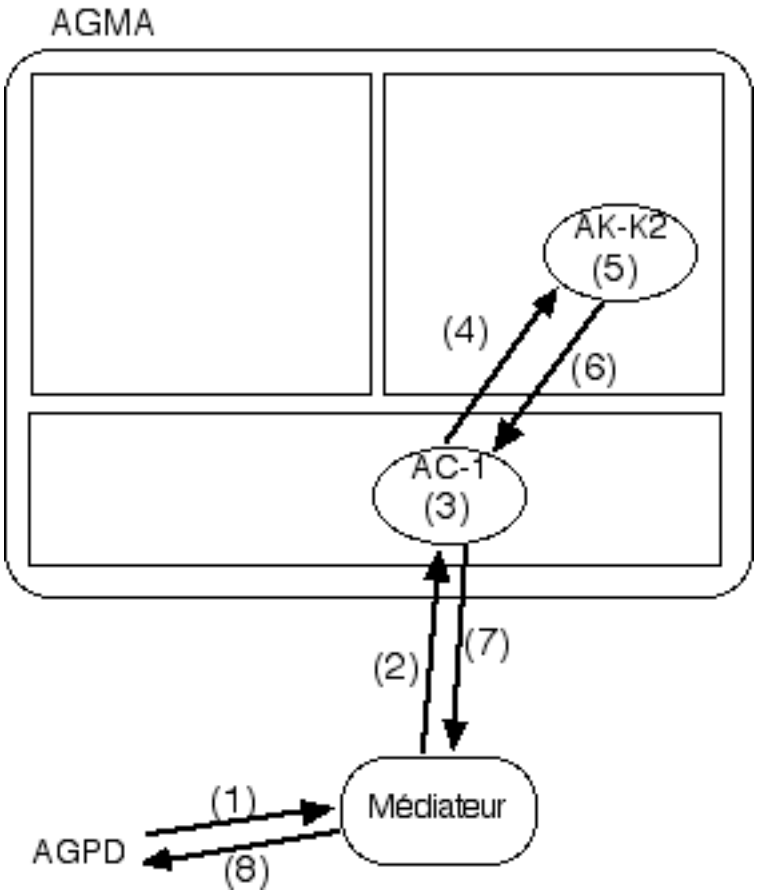

Figure 3.5 : Collaboration des modules d'AGMA pour répondre à une requête

Le scénario se déroule de la façon suivante :

(1) AGPD ayant besoin de disposer d'une information précise pour planifier sa séquence de travail, décide donc de faire appel à AGMA, en charge du modèle de l'apprenant. Pour cela, il envoie sa requête au médiateur.
(2) le médiateur décide donc de faire suivre cette requête à AGMA. Pour cela, il cherche le premier AC disponible, AC-1, et lui envoie la requête,
(3) AC-1 va alors étudier la requête pour savoir sur quelles connaissances elle porte. Sur l'exemple, elle porte sur la connaissance K2. Dans le fonctionnement normal, il aurait consulté le service directory pour avoir l'information,
(4) AC-1 décide donc de transmettre la requête à AK-K2, agent en charge de

cette connaissance K2,

(5) AK-K2 étudie la requête et cherche s'il dispose de la réponse. Ici, il cherchera le statut de la sous-connaissance SK1 dans le contexte C1,

(6) une fois la réponse trouvée, AK-K2 l'envoie à l'agent qui lui en a fait la demande, à savoir AC-1,

(7) AC-1 reçoit la réponse à la requête et la transmet au médiateur pour qu'il la fasse suivre à AGPD,

(8) le médiateur informe AGPD de la réponse à sa requête.

Comme nous l'avons dit, nos recherches se sont essentiellement portées sur la modélisation de l'apprenant et nous n'avons donc présenté qu'un exemple très simple de requête. Bien évidemment, il existe des requêtes beaucoup plus complexes qui nécessitent la combinaison de différentes connaissances ou bien qui nécessitent de faire appel à des agents experts du système. Néanmoins, celles-ci n'ont pas fait l'objet de nos recherches.

4. Conclusion

Nous avons vu au cours de ce chapitre les deux rôles qu'avait AGMA au sein de l'environnement multi-agents AMICAL. Le premier rôle, celui qui a fait l'objet principal de cette thèse, est la modélisation de l'apprenant. Pour cela, il va analyser le compte-rendu de la session de travail qui a été réalisée avec l'apprenant puis intégrer les nouvelles hypothèses issues de cet analyse au sein du modèle de l'apprenant déjà existant. Le deuxième rôle d'AGMA est de répondre aux différentes requêtes en provenance des autres agents de l'environnement.

Nous avons vu qu'AGMA était un agent composé de trois modules : un module communication responsable de la communication entre les modules d'AGMA et le reste du système AMICAL, un module analyse en charge de l'analyse des séquences de travail et un module modèle en charge de stocker et de maintenir à jour le modèle de l'apprenant en intégrant les nouvelles hypothèses faites pendant l'analyse.

Pour des raisons en partie de rapidité d'exécution, nous avons choisi de voir chaque module lui-même comme un système multi-agents. Ainsi, le module communication est composé d'agents de communication, le module analyse

d'agents d'analyse et le module modèle d'agents de connaissance.

Ainsi, en lieu et place d'avoir une analyse linéaire et exécutée en séquence, nous avons vu que la vision d'une multitude d'agents d'analyse, réalisant chacun leur travail en parallèle était beaucoup plus efficace et performant ; chaque agent d'analyse émettant un ensemble d'hypothèses sur l'état de savoir de l'apprenant. Ces hypothèses sont ensuite envoyées au module modèle.

Nous avons également choisi de représenter ce module modèle comme un système multi-agents composé d'agents de connaissance ; chaque agent de connaissance étant en charge de stocker et de maintenir à jour une petite partie du modèle de l'apprenant.

Dans le chapitre suivant, nous allons rentrer dans le détail du fonctionnement des différents agents constituant AGMA. Pour cela, nous suivrons un exemple de modélisation de l'apprenant à partir d'une situation didactique de type «identification de mots en contexte ». Nous verrons dans cet exemple comment l'agent d'analyse et les agents de connaissance correspondant à cette situation didactique travaillent de concert pour réaliser cette tâche.

CHAPITRE 4 : FORMALISATION ET FONCTIONNEMENT DES DIFFÉRENTS MODULES D'AGMA

Dans le chapitre précédent, nous avons introduit l'architecture d'AGMA. Nous avons vu que l'architecture retenue pour celui-ci était constituée de trois modules : le module communication, le module analyse et la module modèle. Nous avons également vu que ces différents modules étaient eux-mêmes des systèmes multi-agents. Il convient donc maintenant de présenter pour chacun des agents, son fonctionnement interne et la représentation de ses connaissances.

Dans ce chapitre, nous détaillerons donc chacun de ces agents, à la fois théoriquement et pratiquement en nous basant sur l'exemple d'une situation didactique que nous suivrons tout au long de son déroulement au sein des différents modules de l'agent. De ce fait, nous expliquerons, à la fois comment fonctionne chacun des agents et dans le même temps comment la modélisation de l'apprenant est réalisée au sein d'AGMA. Répétons le encore une fois, seule la partie modélisation de l'apprenant sera traitée dans ce chapitre.

Nous ne traiterons dans ce chapitre que de l'analyse d'une situation didactique et de l'intégration des hypothèses émises sur celle-ci au sein du modèle de l'apprenant. En effet, la séquence n'étant qu'un ensemble de situations didactiques différentes, le même processus se déroulera sur les autres situations didactiques. Néanmoins, pour parler des agents de coordination et de communication, nous ferons l'hypothèse que le compte-rendu de la situation didactique sera contenu dans un compte-rendu de la session dans laquelle s'est déroulée celle-ci.

Nous utiliserons l'exemple de manière à montrer comment la modélisation est réalisée au travers des différents agents. Nous montrerons comment l'agent de communication récupère le compte-rendu de la séquence puis en extrait les

différents sous-comptes-rendus pour envoyer chacun d'eux à l'agent d'analyse correspondant. Nous nous limiterons par la suite à l'exploitation de la situation didactique exemple. Pour cela, nous montrerons comment l'agent d'analyse correspondant à cette situation didactique émet ses différentes hypothèses puis les envoie aux différents agents de connaissance concernés. Enfin, nous montrerons comment ces différents agents de connaissance intègrent ces hypothèses à leur base de connaissances. Là encore, nous nous limiterons à ne présenter qu'un seul agent de connaissance car tous fonctionnent suivant le même principe.

Nous avons retenu l'utilisation de Prolog comme formalisme pour décrire les connaissances et le raisonnement des différents agents d'AGMA. Une excellente introduction à Prolog peut être trouvée dans (Sterling et Shapiro, 1994) alors que son utilisation en IA est présentée dans (Bratko, 2001). En effet, celui-ci permettra d'exprimer simplement et de manière souple les différents éléments de chaque agent, permettant ainsi de décrire facilement son fonctionnement. Bien entendu, chaque agent dispose de son propre moteur de raisonnement capable d'appliquer les différentes règles que nous formaliserons dans ce chapitre. Néanmoins, nous ne présenterons pas celui-ci car n'ayant que peu d'intérêt pour l'exposé. Nous nous contenterons donc de présenter les croyances et les règles d'inférences utilisées par les différents agents présentés dans ce chapitre.

Nous supposerons également que nous ne travaillons qu'avec un seul apprenant, si bien que nous ne tiendrons pas compte de son prénom dans ce chapitre. De ce fait, il ne sera mentionné nul part dans la description des différents agents.

Ce chapitre suivra la chronologie du déroulement de la modélisation de l'apprenant au sein d'AGMA. Nous commencerons par présenter l'exemple que nous suivrons tout au long de ce chapitre. Ensuite, nous présenterons le premier module de notre agent : le module communication. Nous détaillerons le fonctionnement des agents de communication ainsi que les éléments qu'ils traitent comme les comptes-rendus. Nous continuerons par le deuxième module : le module analyse. Nous expliquerons ici comment l'agent d'analyse correspondant à la situation didactique exemple émet ses hypothèses sur l'état de savoir de l'apprenant. Enfin, nous terminerons en détaillant le troisième module : le module modèle. Nous montrerons ici comment, à partir des hypothèses qu'il reçoit, un agent de connaissance met à jour ses croyances et donc comment il met à jour la connaissance du modèle dont il a la charge. Là encore, nous nous limiterons à un

seul agent de connaissance parmi les agents de connaissance correspondant à la situation didactique exemple. En effet, tous fonctionnant de la même manière, il est donc inutile de tous les présenter.

1. Présentation de l'exemple retenu

Nous allons commencer par présenter rapidement la situation didactique que nous suivrons tout au long de ce chapitre pour suivre la modélisation de l'apprenant au sein des différents agents des différents modules constituant AGMA. Elle s'intitule : « identification de mots en contexte ». Elle consiste à demander à l'apprenant de cliquer sur les mots qu'on lui demande de trouver. A cet effet, l'apprenant dispose de plusieurs essais pour trouver chaque mot. Les mots à trouver, le nombre d'essais possibles ainsi que les aides proposées sont choisis par AGPD lors de la planification didactique. Ces paramètres sont disponibles dans le compte-rendu de la situation didactique. Nous avons choisi cet exemple car il mobilise chez l'apprenant les trois types de connaissances que l'on trouve dans la partie connaissances du domaine de notre modèle de l'apprenant : le concept de mot, différents objets mots ainsi que les stratégies que l'apprenant est susceptible de mettre en oeuvre. Voici l'écran de l'interface qui est montrée à l'apprenant :

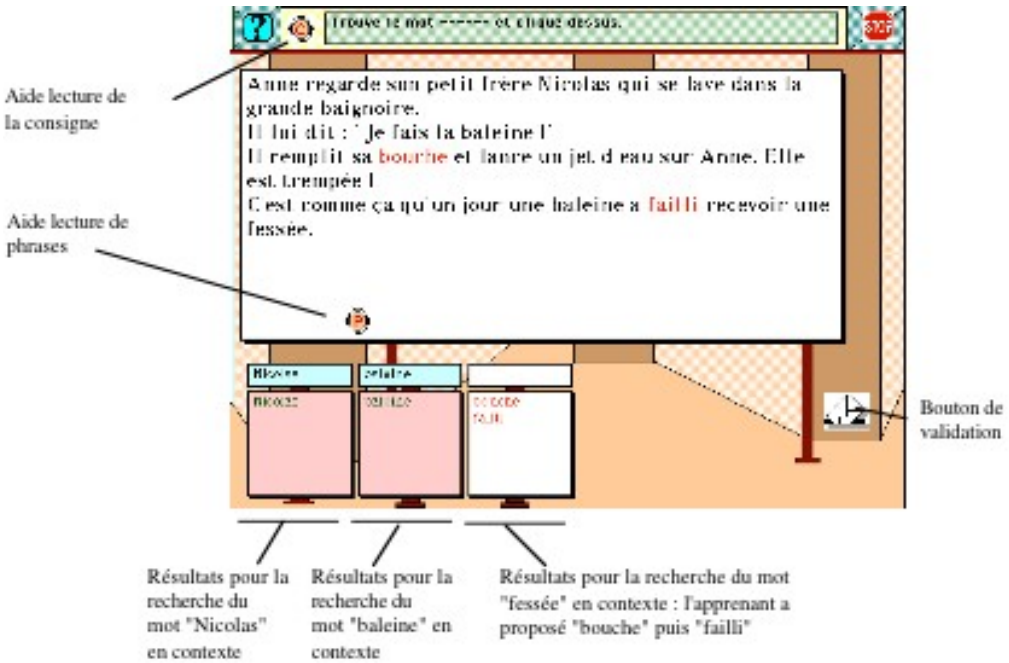

Figure 4.1 : Ecran de la situation didactique «identification de mots en contexte »

L'ensemble des agents d'AGMA mobilisés pour traiter cette situation didactique est représenté par la Figure 4.2 :

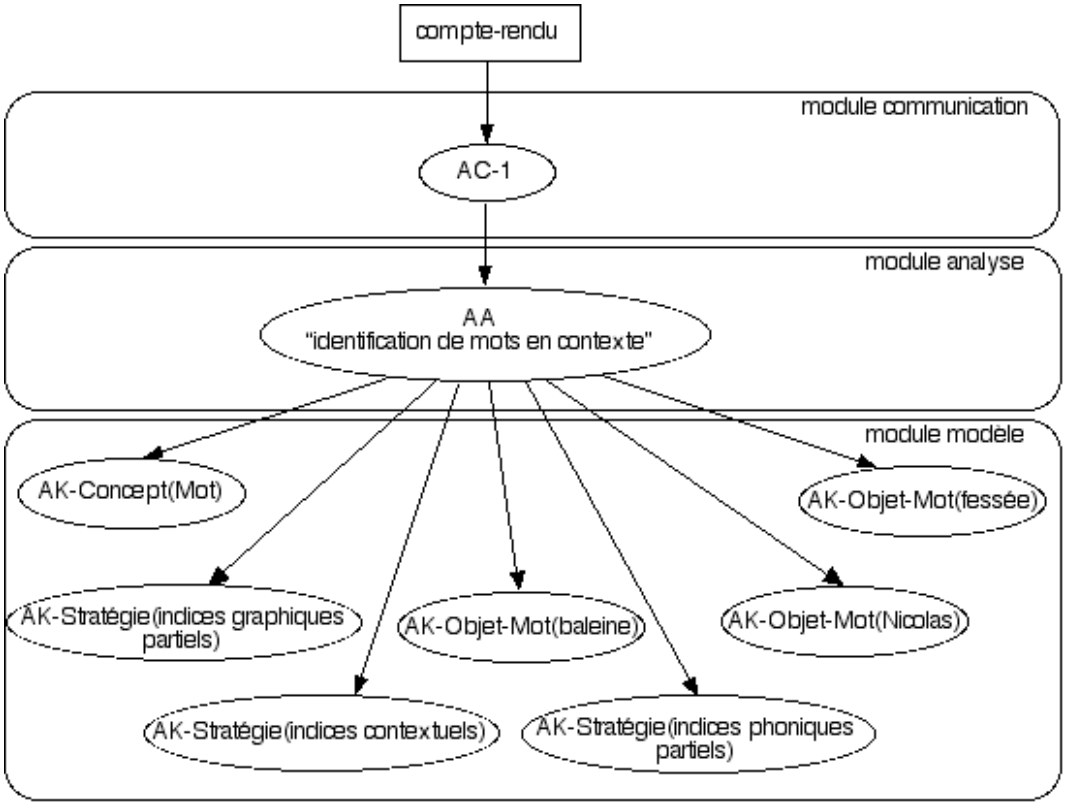

Figure 4.2 : Agents mobilisés pendant la modélisation de l'apprenant pendant le déroulement de l'exemple

Dans l'exemple de cette situation didactique, on retrouve donc naturellement des agents de chacun des modules d'AGMA :

- AC-1[2] qui va aller récupérer le compte-rendu disponible, le découper et envoyer chaque sous-compte-rendu aux différents AA dont celui qui concerne cet exemple : AA « identification de mots en contexte »,
- AA « identification de mots en contexte », qui va analyser le compte-rendu pour la situation didactique correspondante et émettre des hypothèses sur l'état de savoir de l'apprenant en termes de concept, d'objets et de stratégies,
- les différents AK correspondant aux connaissances mobilisées par l'apprenant pendant l'exécution de cette activité. Ceux-ci vont intégrer ces

2 Il s'agit du premier AC disponible choisi par le Médiateur. Nous ne traiterons pas dans ce document de ce mécanisme.

hypothèses à leurs croyances en les mettant à jour si besoin est. AK-Concept(Mot) représente l'agent en charge du concept de mot, les AK-Stratégie représentent les agents en charge des différentes stratégies et les AK-Objet-Mot représentent les agents en charge des différents objets mots (Nicolas, fessée et baleine).

Dans la suite de ce chapitre, nous montrerons la formalisation et le fonctionnement de trois agents : AC-1, AA « identification de mots en contexte », AK-Objet-Mot(Nicolas). En effet, tous les AK fonctionnent de la même manière à ceci près qu'il gère une connaissance ne disposant pas des mêmes sous-connaissances, contextes et statuts, mais le principe reste néanmoins le même. Pour chaque agent, nous expliquerons son fonctionnement général en nous appuyant sur l'exemple concret de la situation didactique.

2. Module communication : les agents de communication (AC)

Le premier module d'AGMA est le module communication. Celui-ci, comme nous l'avons dit, est constitué par les agents de communication ou AC. Ces agents servent d'interface entre les différents modules d'AGMA et le reste de l'environnement AMICAL.

Dans la perspective de la modélisation de l'apprenant qui nous intéresse, nous rappelons qu'un AC est responsable de :

- recevoir les messages indiquant que la session de travail est terminée et donc que le compte-rendu de celle-ci est disponible,
- récupérer le compte-rendu de la séquence,
- décomposer ce compte-rendu en autant de sous-comptes-rendus qu'il y a de situations didactiques dans la séquence,
- envoyer ces sous-comptes-rendus aux agents d'analyse concernés,
- éventuellement, de s'occuper des demandes en provenance des agents d'analyse ayant besoin de l'expertise des autres agent du système.

Dans la suite de ce paragraphe, nous présentons tous points ci-dessus à l'exception du dernier qui n'entre pas en considération dans cet exemple.

Pour envoyer les sous-comptes-rendus aux AA concernés, un AC a besoin de savoir quel AA est capable d'analyser quelle activité. Pour cela, il consultera le service directory. C'est pour cette raison que chaque AA s'enregistre, à l'initialisation d'AGMA, auprès de ce service intégré, pour dire quelle situation didactique il est capable d'analyser.

2.1. Fonctionnement général

Dans l'exemple retenu, le scénario d'intervention d'AC-1 est le suivant :

1. AGSSDI poste un nouveau compte-rendu dans la base des comptes-rendus d'AMICAL, puis en informe le médiateur,
2. le médiateur informe à son tour tous les agents ayant besoin de cette information. AGMA étant l'un de ces agents, le médiateur va lui envoyer le message, au travers du premier AC disponible représentant AGMA auprès de celui-ci, en l'occurrence AC-1,
3. AC-1 reçoit le message de la fin de la session,
4. AC-1 va aller chercher le nouveau compte-rendu dans la base des comptes-rendus,
5. AC-1 va examiner ce compte-rendu pour voir de combien et de quelles situations didactiques il est composé,
6. AC-1 va découper le compte-rendu en autant de morceaux qu'il y a de situations didactiques,
7. AC-1 va construire la liste des AA à contacter en consultant le service directory,
8. AC-1 va envoyer à chaque AA un message lui demandant s'il peut analyser le compte-rendu,
9. si un AA refuse d'analyser son compte-rendu, il en informe AC-1. Néanmoins, nous faisons l'hypothèse que ce cas de figure ne peut pas se produire.
10. Une fois qu'un AA a terminé son analyse, il en informe AC-1.

Le protocole d'interaction d'analyse entre un AC et un AA peut se représenter par le diagramme de séquence UML ci-dessous (Figure 4.3). Nous ignorons volontairement le contenu de ces messages pour ne retenir que les performatives. Nous ne traitons pas non plus les cas où AA refuse d'analyser le compte-rendu et où l'analyse a échoué.

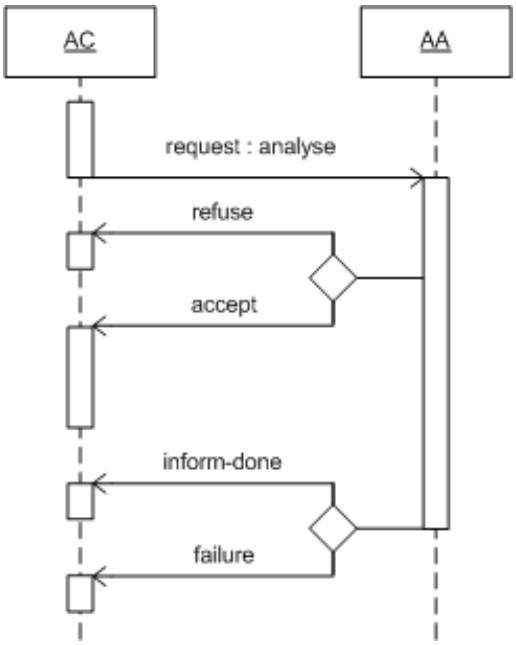

Figure 4.3 : Protocole de demande d'analyse

Ce protocole est très simple : AC demande si AA peut analyser le compte-rendu qu'il lui envoie. AA accepte ou refuse de le faire. AA peut refuser si, par exemple, il est en train de traiter un autre compte-rendu. A la fin de l'analyse, AA informe AC qu'il a fini d'analyser le compte-rendu ou qu'il n'a pas pu effectuer l'analyse. Nous supposons que les cas de refus (refuse) et d'échec (failure) n'existent pas pour le moment au sein de notre agent et nous n'en tiendrons donc pas compte.

Nous ne rentrerons pas dans les détails concernant la réception du message informant AC-1 qu'un nouveau compte-rendu est disponible, ni comment ce même AC-1 va récupérer celui-ci dans la base des comptes-rendus. En effet, nous cherchons uniquement dans ce chapitre à présenter les éléments relatifs à la modélisation de l'apprenant. Nous supposons donc à partir de maintenant, qu'AC-1 dispose du compte-rendu.

2.2. Exploitation du compte-rendu de la session

L'élément le plus important qu'a à traiter AC-1 est le compte-rendu. Celui-ci,

comme nous l'avons dit, représente la suite des interactions mémorisées de l'apprenant interagissant avec le système. Pour l'exploitation au sein du système, celui-ci est représenté par un formalisme XML pour des questions évidentes d'interopérabilité et de possibilité de structuration. De plus, comme nous décidons de garder les comptes-rendus de toutes les sessions, ce formalisme paraît encore plus adapté aujourd'hui en raison notamment de la disponibilité de base de données spécifiquement dédiées au XML. A cet effet, nous utiliserons pour notre implémentation une base de données XML : dbXML. L'utilisation d'un tel lieu de stockage permettrait par exemple d'utiliser des techniques de XML mining, par exemple, pour découvrir de nouvelles règles d'analyse.

Nous présentons ci-dessous une partie d'un compte-rendu XML d'une séquence fourni et construit par AGSSDI, l'agent gérant l'interface avec l'apprenant. Nous parcourons ce fichier XML du début à la fin.

Le fichier XML commence par définir la racine : le compte-rendu. Ce compte-rendu est enregistré pour un apprenant, ici morgane, et à une date et une heure données.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<compteRendu eleve="morgane" date="2008-09-26 14:38:53">
```

La première partie du compte-rendu représente l'objectif de la session de travail. Nous rappelons que cet objectif est composé d'unités d'objectifs, c'est-à-dire les intentions d'enseignement que le système a cherché à atteindre avec la session de travail. Nous ne présentons ici que quelques unités d'objectif : un couple et un triplet.

```
<objectif>
   <unitéObjectif numero="1">
      <action>mettre-en-présence</action>
      <statut></statut>
      <connaissance parametre="lecture-texte">texte</connaissance>
      <priorité>1<priorité>
   </unitéObjectif>
   <unitéObjectif numero="2">
      <action>vérifier</action>
      <statut>construit</statut>
      <connaissance parametre="représentation-conceptuelle-
données-textuelles">texte</connaissance>
      <priorité>2<priorité>
   </unitéObjectif>
```

```
   ...
</objectif>
```

Dans les deux unités d'objectif présentées dans ce compte-rendu, le système a cherché à mettre l'apprenant en face d'un texte lu (UO numéro 1) et à vérifier qu'il est en train de développer sa compréhension en regardant s'il a compris le texte qu'on lui a présenté (UO numéro 2).

La partie suivante du compte-rendu, celle qui nous intéresse pour ce travail dans le cadre de AC-1, est la constitution de la séquence de situations didactiques présentées à l'apprenant.

```
<sequence>
      <sdt numero="1"> … </sdt>
      <sdt numero="2"> … </sdt>

      ...
</sequence>
</compteRendu>
```

Chaque situation didactique, mémorisée entre les balises <sdt> et </sdt>, est représentée dans le système par un identifiant unique. AC-1 dans ce cas va décomposer le compte-rendu en autant de morceaux qu'il y a de situations didactiques dans la séquence. Si cette dernière en comporte cinq, il y aura donc cinq sous-comptes-rendus. Le compte-rendu qui sera envoyé aux différents AA sera donc contenu entre les balises <sdt> et </sdt>.

Voyons maintenant ce que l'on trouve entre ces balises. On présente dans la suite le compte-rendu concerné par la situation didactique « identification de mots en contexte ». Nous nous plaçons à présent entre les balises : <sdt numéro= "2"> … </sdt> où le numéro 2 représente pour le système l'identifiant de la situation didactique « identification de mots en contexte ».

Ce sous-compte-rendu est composé de trois parties : l'objectif de la situation didactique, les paramètres de celle-ci et enfin les interactions entre l'apprenant et le système.

L'objectif, comme pour la séquence, est composé d'unités d'objectif. Nous n'y

reviendrons donc pas.

```
<objectif>
   <unitéObjectif numero="1">
      <action>mettre-en-présence</action>
      <statut></statut>
      <connaissance>texte</connaissance>
   </unitéObjectif>

   ...

</objectif>
```

La deuxième partie concerne les paramètres de la situation didactique. Chaque paramètre porte sur un objet particulier de la situation didactique que l'on peut individualiser.

```
<parametres>
      <parametre objet="texte">
            <titre>La colère de Nicolas</titre>
      </parametre>
      <parametre objet="mot">
            <instance>Nicolas</instance>
            <nombreEssai>3</nombreEssai>
            <declenchementAideEssai>3</declenchementAideEssai>
            <aideProposee>lecture phrase contenant le
mot</aideProposee>
      </parametre>

      ...

<parametres>
```

Dans l'exemple, un paramètre porte sur le texte présenté dans la situation didactique et un autre sur un mot que l'apprenant doit trouver. Chaque objet possède des caractéristiques d'individualisation propres. L'objet texte correspond au texte portant le titre « La colère de Nicolas ». L'objet mot est ici le mot « Nicolas ». Pour trouver ce mot, l'apprenant disposera de trois essais et au troisième essai, l'aide consistant à lire la phrase contenant le mot « Nicolas » sera déclenchée.

La dernière partie est la plus importante pour l'analyse de l'activité. Elle porte sur les interactions qu'a réalisé l'apprenant au cours de l'exécution de la situation didactique. On distingue deux types d'interaction : les actions système et les actions apprenant.

```
<interactions>

      ...

      <interaction numero="20" type="apprenant">
            <time>16:18:30</time>
            <time_activite>100</time_activite>
            <action>selection</action>
            <objet type="mot">Nicolas</objet>
      </interaction>
      <interaction numero="21" type="apprenant">
            <time>16:19:02</time>
            <time_activite>132</time_activite>
            <action>clic</action>
            <objet type="bouton">validation</objet>
      </interaction>

      ...

      <interaction numero="25" type="apprenant">
            <time>16:20:02</time>
            <time_activite>192</time_activite>
            <action>clic</action>
            <objet type="bouton">aide</objet>
      </interaction>

      ...

</interactions>
```

Chaque interaction porte un numéro et correspond à un type : système ou apprenant. Une interaction de l'apprenant correspond à une action (balises <action> </action>) de celui-ci, à un instant donné (balises <time> </time>), sur un objet (balises <objet></objet>) d'un certain type (argument type). Les balises <time_activite> </time_activite> représente le temps écoulé depuis le début de l'activité, c'est-à-dire la situation didactique. Les actions du système représentent ce qu'a fait le système comme lire une consigne ou afficher une information particulière.

2.3. Envois aux agents d'analyse

Une fois cette partie du compte-rendu de la séquence extrait, AC-1 va donc aller consulter le services directory pour trouver quel AA est en charge de cette situation didactique. Une fois l'identifiant de l'agent trouvé, il va lui envoyer le morceau de compte-rendu présenté entre les balises <sdt numero=2> et </sdt>. Il

fera de même pour toutes les autres situations didactiques de la séquence que nous n'évoquerons pas ici.

Dans notre exemple, on supposera que AA « identification de mots en contexte » se nommera AA-31 auprès du service directory. AC-1 va donc envoyer à AA-31 le message ACL ci-dessous, lui demandant d'analyser le morceau du compte-rendu afin d'émettre les hypothèses sur l'état de savoir de l'apprenant.

```
(request
        :sender AC-1
        :receiver AA-31
        :content "analyse?(compte-rendu-sdt2.xml)"
        :language Prolog)
```

Dans la réalisation concrète du prototype de l'agent, le morceau de compte-rendu sera transmis directement dans le message.

Nous verrons dans la section suivante comment AA-31 va exploiter ce morceau de compte-rendu afin d'émettre les hypothèses concernant l'état de savoir de l'apprenant.

3. Module analyse : les agents d'analyse (AA)

Le deuxième module d'AGMA est le module analyse. Celui-ci est constitué d'agents d'analyse (AA).

Il y a autant d'agents d'analyse qu'il y a de situations didactiques au sein du système. Ce type d'agent a pour rôle de faire toutes les hypothèses possibles sur l'état de savoir de l'apprenant à partir du compte-rendu de la situation didactique dont il à la charge.

3.1. Fonctionnement général

On peut représenter très schématiquement le rôle d'un agent par la Figure 4.4 :

Figure 4.4 : Le rôle d'un AA est de produire des hypothèses

AA reçoit donc le compte-rendu de la situation didactique correspondant à ses compétences et à travers son mécanisme interne, va générer un ensemble d'hypothèses contenant toutes les hypothèses possibles quant à l'état de savoir de l'apprenant.

Une fois ces hypothèses émises, l'agent va les envoyer aux différents AK ayant besoin de celles-ci. Pour cela, il consultera lui aussi le service directory pour savoir quel AK gère quelle connaissance du modèle de l'apprenant. AA n'enverra pas le même ensemble d'hypothèses à tous les AK correspondant aux connaissances mises en jeu dans la situation didactique, mais il enverra les bonnes hypothèses aux bons AK. En effet, il ne sert à rien d'envoyer des hypothèses concernant le concept de mot à un agent en charge d'un objet mot particulier, Nicolas par exemple. Les messages envoyés par ce type d'agent porteront la performative *inform* signifiant qu'il informe l'AK d'une nouvelle hypothèse.

3.2. Représentation des connaissances des AA

Comme un compte-rendu est une suite d'actions dans le temps, et que le contenu de celui-ci sera intégré aux croyances de l'agent, il nous a paru alors évident de choisir une représentation capable d'exprimer le temps et de permettre un raisonnement temporel. Par exemple, pour prendre en charge une succession d'évènements de la sorte : sélection d'un mot puis déselection de ce mot, … et émettre des hypothèses dessus, il est nécessaire de disposer d'un paramètre temps. Nous ne nous intéressons pas à représenter des intervalles mais des événements discrets. Le paramètre temps sera donc simplement représenté par un entier. Cet entier correspondra au temps écoulé en secondes depuis le début de l'exécution de la situation didactique. Il existe donc une relation d'ordre total sur toutes les interactions. En effet, comme nous l'avons dit, il n'est pas possible que deux évènements aient eu lieu en même temps.

De plus, pour correspondre au plus près possible à la manière de penser de nos experts en apprentissage de la lecture en terme d'analyse des situations didactiques, il nous a paru plus naturel d'utiliser des règles d'inférences auxquelles nous avons adjoint une représentation du temps. En effet, une règle d'analyse donnée par nos experts correspond à peu de chose près à une règle du type : « Si l'apprenant a fait ça puis ça et puis ça alors on fait l'hypothèse que ça signifie ça ». Les règles d'inférences des AA, que nous verrons plus loin, ressembleront donc à quelque chose de cette sorte, soit en utilisant le temps, soit en ne l'utilisant pas. Ces règles d'inférences seront appelées règles d'analyse par la suite.

L'agent va donc utiliser ses règles d'analyse pour lui permettre, à partir des informations contenues dans le compte-rendu et d'autres connaissances dont il dispose, d'émettre un ensemble d'hypothèses, en prenant en compte le temps.

Pour représenter les croyances de l'agent, nous aurons donc à la fois des croyances ayant un paramètre temps et d'autres n'en disposant pas. Nous aurons également des règles d'inférences utilisant le temps et d'autres ne l'utilisant pas. Ceci permettra à la fois d'exprimer ou non les croyances et règles d'inférences dépendantes du temps.

Les croyances auxquelles on souhaite adjoindre le temps se verront simplement ajouter un argument supplémentaire au début : le temps (*t*), qui sera, comme nous l'avons dit, représenté par un entier. Un exemple fictif de croyance temporalisée pourra avoir cette forme :

```
croyance(t, ...).
```

On définira, pour raisonner sur le temps, un prédicat Prolog, en plus de l'argument temps que l'on rajoute :

```
avant(T1, T2) :- T1 < T2.
```

Ce prédicat sera utilisé pour répondre aux questions de type : l'évènement 1 a-t-il eu lieu avant l'évènement 2. Comme nous l'avons déjà dit, nous enregistrons des évènements discrets et deux événements ne peuvent pas se produire en même temps. De ce fait, ce seul prédicat temporel est donc suffisant pour raisonner sur le temps et répondre à nos besoins.

Nous montrerons dans la suite des exemples concrets de représentation des croyances et des règles d'inférences de ces agents. Nous utiliserons dans la suite un formalisme à la Prolog pour représenter les différentes croyances et règles d'inférences de l'agent.

3.3. Prétraitements du compte-rendu de la situation didactique

AA reçoit donc un compte-rendu au format XML. Nous rappelons que ce compte-rendu ne contient que les informations spécifiques à la situation didactique.

Pour pouvoir mettre en oeuvre son mécanisme d'inférences, AA a besoin de :

1. convertir le compte-rendu XML et intégrer le résultat à ses croyances,
2. lancer un phase de pré-analyse où l'on effectuera certains calculs.

3.3.1. Conversion en croyances pour l'agent

La conversion du fichier XML en prédicats logiques se déroule assez simplement. Elle consiste simplement à prendre chaque entrée du compte-rendu et à la transformer en prédicats temporalisés et non temporalisés. Comme nous avons trois types d'informations dans le compte-rendu XML : les unités d'objectif, les paramètres d'individualisation de la situation didactique et les interactions entre l'apprenant et le système, on retrouvera donc ceux-ci sous forme de prédicats différents. On aura donc trois types de prédicats :

1. *uniteObjectif()* : il correspond à une unité d'objectif. Il s'agit d'un prédicat non temporalisé,
2. *parametre()* : il correspond à un paramètre d'individualisation de la situation didactique. Il s'agit d'un prédicat non temporalisé,
3. *interaction()* : il correspond à une interaction entre l'apprenant et le système. Il s'agit d'un prédicat temporalisé,

Une unité d'objectif ne dépend pas du temps et est donc transformée en une croyance de la sorte :

```
uniteObjectif(<numéro>, <action>, <statut>, <connaissance>).
```

Nous représentons toujours entre chevrons (< >) le rôle de chacun des arguments des formules Prolog.

Ici, <*numéro*> représente le numéro de l'unité d'objectif, <*action*> représente l'action que l'on souhaite réaliser avec cette situation didactique, <*statut*> représente le statut que l'on souhaite observer suivant l'action effectuée et <*connaissance*> représente la connaissance sur laquelle porte l'action.

Pour reprendre l'exemple du compte-rendu XML de la page 146, la conversion aura pour résultat :

```
uniteObjectif(1, mettreEnPresence, _, texte).
```

Cette croyance signifie que l'objectif de cette situation didactique était de mettre en présence l'enfant en face d'un texte.

Un paramètre ne dépend pas lui non plus du temps. Ce qui nous intéresse dans le paramètre, c'est l'objet qui a été individualisé. On transforme donc chaque paramètre en une croyance de la sorte :

```
parametre(<objet>, <valeur_parametre1>,...,<valeur_parametren>).
```

<*objet*> correspond à l'objet qui est individualisé et <*valeur_parametre*$_i$> la valeur correspondant au paramètre i. C'est la position de la valeur qui détermine à quel paramètre elle est associée. Pour chaque objet individulisable, on aura donc une formule de taille différente. Toujours en reprenant l'exemple XML de la page 146, les croyances seront :

```
parametre(texte, "La colère de Nicolas").
parametre(mot, "Nicolas", 3, 3, lecturePhraseTexte).
```

Dans l'exemple, nous avons un texte qui porte le titre « la colère de Nicolas » et le mot « Nicolas » à trouver par l'apprenant sur ce texte.

C'est la position de chaque valeur qui en donne sa signification. Par exemple, le deuxième argument d'un paramètre texte correspond à son titre.

Une interaction quant à elle dépend du temps. On retrouve donc tout naturellement un argument spécifique au temps en début de croyance. On transforme chaque interaction en une croyance de la sorte :

```
interaction(<time_activite>, <type>, <time>, <action>, <objet>,
<type>).
```

<*time_activite*> représente le paramètre temps qui représente le temps écoulé depuis le début de l'exécution de la situation didactique, <*type*> représente le type d'action, <*time*> représente l'heure à laquelle l'action a été réalisée, <*action*> représente l'action qui a été mémorisée, <*objet*> représente l'objet sur lequel a porté l'action et <*type*> représente le type d'objet sur lequel l'action a été effectuée.

Le temps sur lequel l'agent va inférer est représenté par un entier. Cet entier correspond au nombre de secondes depuis le début de la situation didactique. De cette manière, on peut même, contrairement au fait d'utiliser un entier que l'on incrémente à chaque interaction, raisonner sur les écarts de temps eux-mêmes. Par exemple, on pourrait imaginer un cas de figure où le temps qu'a passé l'apprenant à résoudre le problème posé par la situation didactique a été trop long ou au contraire très rapide. L'heure à laquelle à été réalisée l'action est ici juste un paramètre supplémentaire au cas où l'on en aurait besoin éventuellement pour savoir si la situation didactique s'est déroulée l'après-midi ou le matin. En effet, et même si aucune règle n'est présente au sein des agents pour le moment, la chronopédagogie[3] précise que l'heure de la journée pourrait avoir une influence sur la façon d'analyser les interactions. Nous ajoutons donc ce paramètre uniquement pour information.

Dans l'exemple XML de la page 147, on peut représenter les croyances de la manière suivante :

```
interaction(100, apprenant, [16, 18, 30], selection, mot,
"Nicolas").
interaction(132, apprenant, [16, 19, 02], clic, bouton,
validation).
interaction(192, apprenant, [16, 20, 02], clic, bouton, aide).
```

3 Il s'agit d'un type de pédagogie qui prend en compte l'heure de la journée et le rythme biologique des enfants qui lui est associé

3.3.2. Pré-analyse

Avant de lancer le processus d'analyse, c'est-à-dire le processus où l'agent d'analyse va inférer à partir de ses croyances afin d'émettre les différentes hypothèses, il est nécessaire de passer par une phase de pré-analyse.

La pré-analyse dépend elle du type de situation didactique considéré. Elle consiste essentiellement au calcul de différentes durées ou comptages facilitant ainsi les inférences réalisées par l'agent. Dans notre exemple de situation didactique « d'identification de mots en contexte », on distinguera deux catégories de pré-analyse :

1. par mot à trouver
   - mot trouvé ou non trouvé,
   - si le mot est trouvé, à quel essai
2. pour la situation didactique
   - nombre total de mots trouvés et non trouvés
   - durée de l'activité,
   - compter le nombre de fois où l'aide à été utilisée,
   - compter le nombre de clics hors zone.

Ces informations sont déterminées par l'application d'un ensemble de règles de pré-analyse. Par exemple, pour déterminer combien de fois l'apprenant à utiliser l'aide, on appliquera la règle suivante :

```
compterNombreUtilisationAide :-
      findall(X, interaction(X, apprenant, _, clic, bouton, aide),
Liste),
      length(Liste, NombreUtilisationAide),
      assert(nombreDeFoisUtilisationAide(NombreUtilisationAide)).
```

Cette règle va utiliser une partie des croyances précédemment ajoutées. Cette règle, pour compter le nombre d'utilisations de l'aide, va construire une liste des interactions où l'apprenant a cliqué sur le bouton aide (findall). Ensuite, on va mesurer la taille de cette liste (length), donc le nombre de fois où l'aide a été utilisée. Puis, on ajoutera une nouvelle croyance à l'agent (assert).

Un autre exemple de règle est celle permettant de déterminer si un mot a été trouvé ou pas :

```
trouveMot?(Mot) :-
      interaction(T1, apprenant, _, selection, mot, Mot),
      not(interaction(T2, apprenant, _, deselection, mot, Mot)),
      avant(T1, T2),
      interaction(T3, apprenant, _, clic, bouton, validation),
      avant(T1, T3).
```

On considère un mot trouvé si l'apprenant a sélectionné (première ligne de la règle) le mot demandé (Mot) puis s'il ne l'a pas déselectionné un moment après (deuxième et troisième ligne), et si enfin il a validé sa réponse (quatrième et cinquième ligne).

Puis une autre règle basée sur celle ci-dessus pour compter le nombre de mots trouvés ainsi que la liste de ceux-ci :

```
nombreEtListeDeMotsTrouves :-
      findall(Mot, (parametre(mot, Mot, _, _, _), trouveMot?
(Mot)), ListeMotsTrouves),
      length(ListeMotsTrouves, NombreMotsTrouves),
      assert(listeMotsTrouves(ListeMotsTrouves)),
      assert(nombreMotsTrouves(NombreMotsTrouves)).
```

On construit la liste des mots trouvés en appliquant la règle précédente trouveMot? (findall), puis pour connaître le nombre de mots trouvés, il suffit d'en mesure la taille (length) et enfin ajouter deux nouvelles croyances : une concernant la liste des mots trouvés (listeMotsTrouves) et une autre concernant le nombre de mots trouvés (nombreMotsTrouves).

3.4. Mécanisme d'émission d'hypothèses

Une fois la conversion et les calculs effectués, le mécanisme d'inférences rentre alors en action. Celui-ci va générer toutes les hypothèses possibles à partir des actions de l'apprenant et des connaissances dont dispose l'agent sur la situation didactique.

3.4.1. Connaissances sources de l'émission d'hypothèses

Les connaissances dont dispose chaque agent d'analyse sont assez variées, permettant ainsi d'émettre plus d'hypothèses et de manière beaucoup plus fine.

On distingue trois types de connaissances pour chaque AA :

1. les connaissances non variables liées à la situation didactique elle-même et ne dépendant pas du compte-rendu :
   - la position des éléments graphiques permet à l'agent, si l'enfant n'a pas cliqué sur le bon bouton ou n'a pas sélectionné le bon mot, de savoir si le clic était loin ou non de la « cible ».
   - les unités d'objectifs possibles, c'est-à-dire toutes les intentions d'enseignement qu'est susceptible de remplir la situation didactique,
   - les actions possibles sur l'interface utilisateur de la situation didactique,
   - les connaissances sur lesquelles peut porter la situation didactique,
2. les connaissances dépendant du choix de l'agent de planification et donc présentes dans le compte-rendu :
   - les unités d'objectifs exécutées,
   - les éléments variables (contenu, essais, consignes, aides) sont contenus dans le compte-rendu. Il s'agit par exemple du texte choisi à présenter à l'enfant,
   - les connaissances sur lesquelles a porté la situation didactique.
3. les connaissances dépendant des actions de l'apprenant sur la situation didactique et donc présentes dans le compte-rendu
   - la suite des actions effectuées (compte-rendu de l'activité).

Les deux derniers types de connaissance sont ceux dont nous avons discutés dans les paragraphes précédents. Ceux-ci proviennent du compte-rendu et sont intégrés aux croyances de l'agent après conversion du compte-rendu XML. Ces croyances sont donc variables et dépendent du contenu effectif du compte-rendu.

Le premier type, au contraire, correspond aux croyances de l'agent qui n'évoluent jamais. Celle-ci sont donc toujours présentes et ne dépendent que des informations sur la situation didactique déterminées lors de la conception de celle-ci. Par exemple, une croyance sur la position d'un élément graphique sera la suivante :

```
position(<objet>, <type>, [x1, y1], [x2, y2]).
```

On indique la position d'un objet (<*objet*>) d'un certain type (<*type*>) ainsi que les coordonnées du coin supérieur gauche (x1, y1,) puis du coin inférieur droit (x2, y2). Pour définir la position du bouton aide de la situation didactique, on aura la croyance suivante :

```
position(bouton, aide, [50, 50], [100, 100]).
```

Cette croyance en combinaison avec d'autres liées aux interactions de l'apprenant, permettrait par exemple, d'émettre des hypothèses sur les clics vides et savoir si l'apprenant a eu ou non l'intention de cliquer sur un élément particulier de l'interface, mais qui, pour des raisons de problèmes de manipulation de la souris, a cliqué à côté.

3.4.2. Règles d'analyse

Parmi les prémisses de chacune des règles, on trouve les trois types de connaissances de l'agent, c'est-à-dire à la fois les croyances que l'agent à ajoutées en convertissant le compte-rendu et les autres connaissances dont l'agent dispose sur la situation didactique (position des boutons, etc.).

Chaque agent d'analyse possède son ensemble de règles lui permettant de faire des hypothèses. Ces règles emploient, à la fois des croyances temporalisées et d'autres pas.

Ces différentes règles d'analyse ont tout d'abord étaient recueillies en collaboration avec les experts. Pendant la conception d'une situation didactique, les différents intervenants, informaticiens et experts en apprentissage de la lecture, travaillent de concert. Lors de la conception d'une situation didactique, celle-ci est vue sous différents angles :

- en terme de description de ses intentions d'enseignement,
- en terme d'interface graphique,
- en terme d'informations à faire figurer dans le compte-rendu,
- en terme de règles d'analyse des différentes actions de l'apprenant.

Ces règles d'analyse portent essentiellement mais pas exclusivement[4] sur les trois types de connaissances du domaine : les concepts, les objets et les stratégies. Ces règles sont recueillies sous forme de phrases en langue naturelle qu'il convient ensuite de traduire de manière plus formelle.

4 Nous rappelons que, bien qu'il existe d'autres entrées à notre modèle de l'apprenant, nous ne traitons que les connaissances du domaine

Nous allons donner quelques exemples de règles d'analyse en langue naturelle, toujours sur la situation didactique nous servant d'exemple pour ce chapitre, puis nous les transcrirons en règles utilisables par l'AA. Nous rappelons qu'une règle d'analyse en langue naturelle est toujours de la forme : « Si l'apprenant a fait ça puis … puis ça alors on fait l'hypothèse que ça signifie ça ».

Exemple de règle d'analyse numéro 1 : Si au moins un mot a été identifié par l'apprenant sans aide, alors faire l'hypothèse de l'utilisation « supposée » de la stratégie « stratégie par indices contextuels ».

Nous pouvons noter que la règle en langue naturelle recueillie auprès de l'expert en apprentissage de la lecture est de type SI … ALORS …. Deux mots sont important dans la conclusion de la règle : « stratégie par indices contextuels » qui correspond au type de stratégie mise en oeuvre par l'apprenant et le statut de cette stratégie qui doit être supposée.

Nous avons traduit cette règle en langue naturelle de la façon suivante, en règle de l'agent :

```
regleAnalyse1 :-
      nombreMotsTrouves(NombreMotsTrouves),
      NombreMotsTrouves >= 1,
      nombreDeFoisUtilisationAide(NombreUtilisationAide),
      NombreUtilisationAide = 0,
      assert(hypothese(heureSysteme, dateSysteme, strategie-
indice-contextuels, 2, supposee)).
```

La traduction du « au moins » est simplement traduite par un nombre de mots trouvés supérieur ou égal à 1 (première et deuxième ligne) alors que la traduction de l'utilisation de l'aide peut se faire grâce à l'absence de fois où l'apprenant a utilisé l'aide (troisième et quatrième ligne), c'est-à-dire 0 fois. Si les conditions de ces quatre premières lignes sont remplies alors on peut ajouter une hypothèse sur la stratégie « stratégie par indices contextuels » qui est « supposée » (cinquième ligne). Le numéro 2 correspond à l'identifiant de la situation didactique « identification de mots en contexte ».

Exemple de règle d'analyse numéro 2 : Si le prénom a trouvé a été identifié sans aide alors faire l'hypothèse de l'utilisation « supposée » de la stratégie « stratégie par indices contextuels » et faire l'hypothèse de l'utilisation « supposée » de la

stratégie « stratégie indice graphique partiel ». Nous formalisons la règle en :

```
regleAnalyse2 :-
      parametre(mot, Mot, _, _, _),
      prenom(Mot),
      trouveMotSansAide?(Mot),
      assert(hypothese(heureSysteme, dateSysteme, strategie-
indice-contextuels, 2, supposee)),
      assert(hypothese(heureSysteme, dateSysteme, strategie-
indice-graphique-partiel, 2, supposee)).
```

Si le mot à trouver (première ligne) est un prénom (deuxième ligne) et que celui-ci a été trouvé sans aide (troisième ligne) alors on peut ajouter une hypothèse sur la stratégie « stratégie par indices contextuels » qui est « supposée » et une autre hypothèse sur la stratégie « stratégie indice graphique partiel » qui est « supposée » (quatrième et cinquième ligne). Nous ne détaillerons pas la fonction trouveMotSantAide?(Mot) qui renvoie si le mot a été trouvé avec l'aide ou pas. Cette fonction ressemble à trouveMot(Mot) à laquelle on a rajouté la détection de l'utilisation de l'aide.

Enfin, nous prendrons un troisième et dernier exemple de règles d'analyse de la situation didactique « identification de mots en contexte », mais cette fois sur un objet mot.

Exemple de règle d'analyse numéro 3 : Si le mot à trouver est un prénom et qu'il l'a été sans l'utilisation de l'aide alors faire l'hypothèse que le mot est identifié.

```
regleAnalyse3 :-
      parametre(mot, Mot, _, _, _),
      prenom(Mot),
      trouveMotSansAide?(Mot),
      assert(hypothese(heureSysteme, dateSysteme, Mot,
representationOrthographique, texte, 2, identifié)),
```

Si le mot à trouver (première ligne) est un prénom (deuxième ligne) et qu'il a été trouvé sans aide (troisième ligne) alors on peut faire l'hypothèse que le mot est identifié (quatrième ligne). Le contexte de l'hypothèse (cinquième argument) correspond ici au texte, c'est-à-dire l'élément dans lequel on a demandé à l'apprenant d'identifier le mot.

Bien entendu, il existe de nombreuses autres règles et nous n'en avons donné que quelques unes. L'objectif était de montrer la formalisation que nous avons retenue

pour transcrire les règles des experts en apprentissage de la lecture. Ces règles sont bien entendu utilisées par l'agent. Pour réaliser le prototype de ces agents, nous verrons dans le dernier chapitre comment nous avons implémenté réellement ces mécanismes.

L'agent pour appliquer ces règles d'analyse dispose d'un mécanisme similaire à celui des systèmes experts. Nous ne le détaillerons pas ici.

3.4.3. Hypothèses

Après avoir vu les règles d'analyse permettant d'émettre des hypothèses, il convient maintenant de revenir sur la forme générale de ces dernières.

Pour déterminer la forme générale d'une hypothèse, nous devons nous placer du point de vue d'un AK. Comme nous l'avons dit un AK gère une connaissance particulière du modèle de l'apprenant et cette connaissance, comme nous l'avons dit au chapitre 2, est un objet complexe multi-niveaux. Néanmoins, toutes les connaissances ne disposent pas nécessairement des mêmes sous-connaissances, contextes et statuts. Pour gérer tous les niveaux de cette connaissance, un AK doit donc obtenir des hypothèses les plus fines possibles. Pour cela, une hypothèse correspond donc au niveau de détail le plus fin possible de la connaissance. La forme générale peut être formalisée de la façon suivante :

```
hypothese(<heure>, <date>, <connaissance>, <sous-connaissance>,
<contexte>, <activite> <statut_hypothese>).
```

<*heure*> représente l'heure de l'émission de l'hypothèse, <*date*> représente la date de l'émission de l'hypothèse, ces deux informations précédentes correspondent respectivement à l'heure et à la date de l'ordinateur, <*connaissance*> représente la connaissance sur laquelle porte l'hypothèse. Elle sera également utilisée pour déterminer l'agent auquel sera envoyé cette hypothèse, <*sous-connaissance*> représente la sous-connaissance associée à la connaissance de l'hypothèse, <*contexte*> représente le contexte dans lequel a été effectuée l'hypothèse, <*activité*> représente l'identifiant de la situation didactique dans laquelle a été faite l'hypothèse. Cette information correspond tout naturellement à la situation didactique analysée par l'agent. Nous rappelons que chaque situation didactique est représentée par un identifiant unique au sein du

système et <*statut_hypothèse*> représente le statut déterminé par l'analyse, c'est-à-dire en quelque sorte le degré de connaissance qu'a montré l'apprenant au cours de l'exécution de l'activité. Les champs correspondant à sous-connaissance et contexte ne sont pas toujours remplis ; ceux-ci dépendent en effet du type de connaissance auquel ils sont associés.

Sur notre exemple de situation didactique « identification de mots en contexte », nous supposons que l'apprenant à trouvé le mot Nicolas et que l'agent a donc appliqué la règle d'analyse « regleAnalyse3 ». La croyance de l'agent est donc la suivante :

```
hypothese([16,25,35], [19,09,2009], "Nicolas",
representationOrthographique, texte, 2, identifié).
```

Les deux premiers argument représentent respectivement l'heure et la date auxquelles a été émise l'hypothèse. Cette hypothèse porte sur la connaissance Nicolas, c'est-à-dire l'objet mot Nicolas. La sous-connaissance associée est la représentation orthographique du mot. La connaissance du modèle de l'apprenant « objet mot » possède trois sous-connaissance : la représentation orthographique qui correspond à la forme écrite du mot, la représentation phonologique qui correspond à la forme orale du mot et la représentation sémantique qui correspond au sens du mot. Chaque sous-connaissance et associé à un ensemble de contextes : texte, phrase, liste de mots proches, etc. Le contexte est dans cette hypothèse le texte correspondant au contexte de notre situation didactique exemple, qui elle-même est identifiée par son identifiant, le numéro 2. Enfin, le statut de l'hypothèse est ici identifié.

Cette hypothèse traduit donc que la représentation orthographique du mot Nicolas est considéré comme identifié dans un texte et dans la situation didactique « identification de mots en contextes » à 16h25'35" le 19/09/2009.

3.5. Envois des hypothèses

Une fois l'ensemble des hypothèses émises, celles-ci doivent être envoyées aux AK concernés. Pour la situation didactique exemple, AA va envoyer :

- les hypothèses sur le concept de mot à AK « Concept Mot »,
- les hypothèses sur les objets mots, c'est-à-dire les instances de chaque mot,

Nicolas, baleine et fessée, respectivement à l'AK « Objet-Mot(Nicolas) », à l'AK « Objet-Mot(baleine)», à l'AK « Objet-Mot(fessée) ».

- Les hypothèses sur les trois stratégies, stratégie par indices contextuels, stratégie par indices graphiques partiels et stratégie par indices phoniques partiels, respectivement à l'AK « stratégie par indices contextuels », l'AK « stratégie par indice graphique partiel » et l'AK « stratégie par indices phoniques partiels ».

Chaque AK recevra donc une série d'hypothèses différentes et sera informé des nouvelles hypothèses par une série de messages ACL portant la performative *inform*, c'est-à-dire pour informer l'AK.

Pour envoyer les messages, AA va d'abord consulter chaque hypothèse ; et pour chaque hypothèse, voir sur quel type de connaissance elle porte. Ensuite, il consultera le service directory pour avoir le nom de l'agent responsable de gérer cette connaissance et ensuite lui enverra la série de messages contenant les hypothèses. Par exemple, AA sait que Nicolas est un mot et va donc consulter le service directory en recherchant l'AK responsable de la gestion de la connaissance de l'apprenant du mot Nicolas. Une fois les différents AK informés des nouvelles hypothèses, AA rétractera de ses croyances les données correspondant au compte-rendu qui vient d'être analysé.

Nous supposons que nous travaillons toujours avec AA-31 et que l'AK gérant l'objet mot « Nicolas » aura, par exemple, le nom AK-26. Nous supposerons, d'après la règle d'analyse de la page 159, que le mot « Nicolas » a été trouvé sans aide et que l'on a émis l'hypothèse suivante :

```
hypothese([16,25,35], [19,09,2009], "Nicolas",
representationOrthographique, texte, 2, identifié).
```

La première liste représente l'heure de l'ordinateur 16:25:35 alors que la deuxième liste représente la date de l'ordinateur 19-09-2009. Comme l'AK récepteur du message porte sur une connaissance unique, il est donc inutile d'envoyer cette information. C'est pour cela, qu'il n'est pas nécessaire d'envoyer « Nicolas » à l'agent qui gère l'objet mot Nicolas ! Le message informant AK-26 sera alors le suivant :

(inform

:sender AA-31
:receiver AK-26
:content "hypothese([16,25,35], [19,09,2009], representationOrthographique, texte, 2, identifié)"
:language Prolog)

4. Module modèle : les agents de connaissance (AK)

Le troisième et dernier module d'AGMA est le module modèle. Celui-ci est composée d'agents de connaissance.

Comme nous l'avons vu au chapitre précédent, le modèle de l'apprenant est réparti au sein des différents agents de connaissance. Ces agents ont pour unique vocation de tenir à jour une petite partie du modèle de l'apprenant en étant chacun spécialisé uniquement dans cette partie.

Pour continuer l'exemple de ce chapitre, nous nous bornerons dans cette partie à l'agent AK-Objet-Mot(Nicolas) car, comme nous l'avons dit, tous les AK fonctionnent de la même manière. Seuls diffèrent les sous-connaissances, contextes et statuts.

4.1. Fonctionnement général

Une fois que les AA en charge d'analyser les différentes situations didactiques de la séquence ont envoyé leurs hypothèses aux AK concernés, ces derniers vont mettre à jour leur base de connaissances ; base de connaissances représentant une partie du modèle de l'apprenant.

Un AK reçoit donc une série de messages en provenance de différents AA. Les messages portent la performative *inform* comme dans l'exemple donné plus haut. A chaque fois qu'il reçoit une nouvelle hypothèse, un AK va donc l'intégrer à sa base de connaissances et la mettre à jour en conséquence.

L'intégration de ces hypothèses peut dans le plus simple des cas ne consister qu'à l'ajout pur et simple de celle-ci. Par contre, si l'agent possède déjà des hypothèses

antérieures alors il faudra qu'il modifie plus en profondeur sa base de connaissances en synthétisant les connaissances qu'il possède déjà. Ce sont ces différents niveaux de synthèse que nous présentons dans la suite.

La construction de la base de croyances de l'agent se déroulera exclusivement à partir des hypothèses reçues des différents AA et des règles de synthèse que nous détaillerons plus loin.

4.2. Niveaux de synthèse des croyances d'un AK

Les différents niveaux de détail des connaissances du modèle de l'apprenant se déterminent à partir de l'historique des différentes hypothèses reçues par l'agent au cours du temps dans les différentes situations didactiques présentées à l'apprenant. A partir de cet historique, il synthétisera les différents niveaux possibles. Les niveaux de synthèse possibles correspondent aux niveaux de détail d'une connaissance du domaine (Figure 4.5).

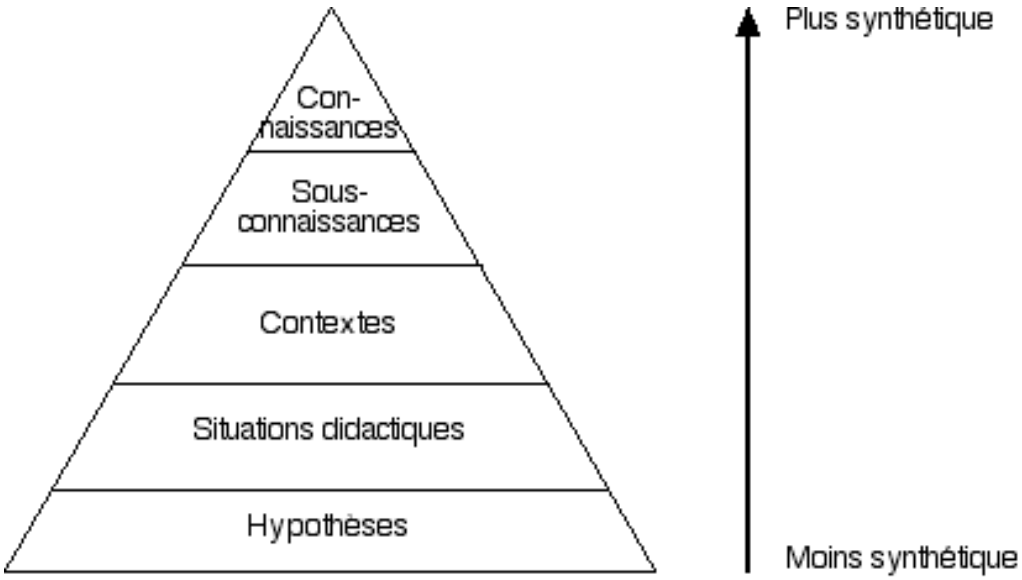

Figure 4.5 : Niveaux de synthèse possibles sur une connaissance du modèle de l'apprenant

Nous détaillons dans la suite de ce paragraphe les différents niveaux de synthèse

possibles.

Pour exploiter l'historique des hypothèses reçues, un AK a donc besoin, là encore, d'utiliser le temps, à la fois pour représenter ses croyances et pour raisonner dessus. Nous utiliserons donc un argument supplémentaire représentant le temps. Cet argument sera lui aussi un entier mais cette fois on utilisera le temps Unix représentant le nombre de seconde depuis le 1er janvier 1970 à 00:00:00UTC. Cela permettra de convertir la date et l'heure de l'hypothèse en un entier unique. Cet entier est bien entendu unique et croissant.

Pour construire sa base de croyances sur une connaissance de l'apprenant donnée, l'agent part des hypothèses qu'il reçoit auxquelles il appliquera des règles lui permettant de synthétiser ces hypothèses en des niveaux toujours plus synthétiques.

Le niveau de croyance le plus bas de l'agent correspond donc au niveau de l'hypothèse. L'agent commencera donc par transformer les hypothèses reçues en une croyance de la forme :

```
bel(<t>, <sous-connaissance>, <contexte>, <activité>,  <statut>).
```

<*t*> représente le temps Unix où l'hypothèse a été émise, <*sous-connaissance*> représente la sous-connaissance de la connaissance gérée par l'agent. Par exemple, une sous-connaissance de la connaissance « lettre A » serait son nom. <*contexte*> représente le contexte dans lequel la sous-connaissance à un statut <*statut*>. Par exemple, un contexte pour la sous-connaissance « nom » de la connaissance « lettre A » serait le « mot ». <*activité*> représente la situation didactique associée au contexte. Nous garderons les mêmes significations dans la suite de ce paragraphe.

Par exemple, l'hypothèse reçue par l'AC-Objet-Mot(Nicolas) aura était convertie en la croyance suivante :

```
bel(1253370335, representationOrthographique, texte, 2,
identifié).
```

Le premier argument représente le temps Unix correspondant à l'heure et à la date de l'hypothèse reçue. Ceci signifie donc qu'à cette date là, le mot « Arthur » a été

identifié dans la situation didactique « identification de mots en contexte », portant l'identifiant 2 ; le contexte de cette situation didactique étant un texte.

4.2.1. Premier niveau de synthèse : la situation didactique

A partir de ces croyances correspondant aux hypothèses fournies par les différents AA, on peut avoir un premier niveau de synthèse. Celui-ci consiste à synthétiser dans le temps les différentes croyances ci-dessus. Si l'on avait travaillé plusieurs fois avec l'apprenant dans la même situation didactique « identification de mots en contexte », l'agent pourrait avoir, en supposant qu'à chaque fois l'apprenant ait bien cliqué sur le mot Nicolas, les croyances suivantes :

```
bel(1253370335, representationOrthographique, texte, 2,
identifié).
bel(1253370435, representationOrthographique, texte, 2,
identifié).
bel(1253370535, representationOrthographique, texte, 2,
identifié).
bel(1253370635, representationOrthographique, texte, 2,
identifié).
bel(1253370735, representationOrthographique, texte, 2,
identifié).
```

La synthèse temporelle consisterait donc à « supprimer » le paramètre temps pour en émettre une synthèse. On aura donc une nouvelle croyance de la forme :

```
bel(<sous-connaissance>, <contexte>, <activité>,  <statut>).
```

On pourrait par exemple avoir une croyance de la forme :

```
bel(representationOrthographique, texte, 2, connu).
```

Dans notre l'exemple, la synthèse temporelle a permis d'ajouter une nouvelle croyance qui signifie que le mot Nicolas est connu dans cette situation didactique. Pour passer d'un ensemble de croyances temporalisées à une croyance synthétique non temporalisée, on applique des règles permettant de le faire. Comme nous l'avons dit, la détermination de ces règles s'est faite auprès des experts en apprentissage de la lecture. Un exemple de règle dans l'exemple précédent pourrait être une règle en langage naturel nous indiquant que « Si l'apprenant a identifié au moins cinq fois dans la même situation didactique alors on suppose

que le mot est connu dans cette situation didactique ». Cette nouvelle croyance ainsi ajoutée pourrait être utilisée pour, par exemple, ne plus demander ce mot à l'apprenant lorsqu'on lui présentera à nouveau la même situation didactique. La règle se formalise de la manière suivante :

```
regleSyntheseTemporelle1 :-
      findall(T, bel(T, SousConnaissance, Ctx, Sd, identifié),
NbFoisIdentifie),
      length(NbFoisIdentifie, Result),
      Result >= 5,
      assert(bel(SousConnaissance, Ctx, Sd, connu)).
```

Cette règle construit la liste de toutes les fois où l'apprenant a identifié le mot Nicolas (première ligne) puis en mesure la taille, c'est-à-dire le nombre de fois (deuxième ligne) et si cette taille est supérieure ou égale à cinq (troisième ligne) alors on ajoute une nouvelle croyance qui synthétise sa connaissance de ce mot dans la situation didactique donnée, en l'occurrence la situation didactique « identification de mots en contexte » portant l'identifiant 2. On suppose ici que cet aspect est connu dans cette situation didactique. Cette règle est générique car elle s'applique quelle que soit la sous-connaissance, quel que soit le contexte et quelle que soit la situation didactique. Les croyances qui ont fait l'objet de cette synthèse temporelle sont bien entendu conservées car on pourrait envisager les utiliser ultérieurement.

4.2.2. Deuxième niveau de synthèse : le contexte

Un autre niveau de synthèse que l'agent peut effectuer est celui consistant à synthétiser les croyances synthétiques des différentes situations didactiques précédentes en une croyance correspondant à une synthèse au niveau contexte. Cette croyance se formalise de la façon suivante :

```
bel(<sous-connaissance>, <contexte>, <statut>).
```

Par exemple, si l'on avait travaillé plusieurs fois sur le mot Nicolas dans différentes situations didactiques se déroulant toutes dans le contexte d'un texte, on pourrait avoir au bout d'un certain temps :

```
bel(representationOrthographique, texte, 2, connu).
bel(representationOrthographique, texte, 25, connu).
```

```
bel(representationOrthographique, texte, 12, connu).
bel(representationOrthographique, texte, 19, connu).
bel(representationOrthographique, texte, 27, connu).
```

Cet ensemble de croyances indiquant que le mot Nicolas est connu dans les situations didactiques 2, 25, 12, 19 et 27.

La croyance synthétisée correspondant à l'exemple serait de la forme :

```
bel(representationOrthographique, texte, connu).
```

Cela indiquerait que la représentation orthographique du mot Nicolas est connue dans un contexte texte.

Là encore, il existe des règles de synthèse permettant de synthétiser les croyances de niveau situation didactique en une synthèse de niveau contexte. De la même manière que précédemment, un exemple de règle de synthèse serait le suivant : « Si le mot est connu dans au moins cinq situations didactiques alors ce mot est connu dans ce contexte ». Cette règle se formalise de la sorte :

```
regleSyntheseContexte1 :-
      findall(SD, bel(SousConnaissance, Ctx, SD, connu),
NbFoisConnu),
      length(NbFoisConnu, Result),
      Result >= 5,
      assert(bel(SousConnaissance, Ctx, connu)).
```

Cette règle construit la liste des situations didactiques dans lesquelles le mot Nicolas est connu (première ligne) puis en mesure la taille, c'est-à-dire le nombre de situations didactiques (deuxième ligne) et si cette taille est supérieure ou égale à cinq (troisième ligne), donc connu dans plus de cinq situations didactiques, alors on ajoutera une croyance indiquant que le mot Nicolas est connu dans le contexte texte (quatrième ligne). Cette règle est générique car elle s'applique quelle que soit la sous-connaissance et quel que soit le contexte.

4.2.3. Troisième niveau de synthèse : la sous-connaissance

Un autre niveau de synthèse que l'agent peut effectuer est celui consistant à synthétiser les croyances synthétiques des contextes en une croyance synthétique

au niveau sous-connaissance. Comme nous l'avons vu au chapitre 2, il existe plusieurs contextes. Pour le mot, par exemple, on peut trouver : le texte (comme dans notre exemple) mais aussi une phrase ou une liste de mots proches ou éloignés. On a donc quatre contextes possibles pour l'objet mot. Cette croyance se formalise de la façon suivante :

```
bel(<sous-connaissance>, <statut>).
```

Par exemple, si l'on avait travaillé plusieurs fois sur le mot Nicolas dans d'autre contexte, on pourrait avoir au bout d'un certain temps :

```
bel(representationOrthographique, texte, connu).
bel(representationOrthographique, liste_mot_proches, connu).
bel(representationOrthographique, liste_mot_eloignes, connu).
```

La croyance synthétisée correspondant à l'exemple serait de la forme :

```
bel(representationOrthographique, connu).
```

Cela indiquerait que la représentation orthographique du mot Nicolas est connue.

Un exemple de règle de synthèse serait alors le suivant : « Si le mot est connu dans tous les contextes alors la sous-connaissance est acquise ». Cette règle se formalise de la sorte :

```
regleSyntheseSousConnaissance1 :-
      findall(Ctx, bel(SousConnaissance, Ctx, connu), NbCtxConnu),
      length(NbCtxConnu, Result),
      Result = 4,
      assert(bel(SousConnaissance, acquis)).
```

Cette règle construit la liste des contextes dans lesquelles le mot Nicolas est connu (première ligne) puis en mesure la taille, c'est à dire le nombre de contextes (deuxième ligne) et si cette taille est égale à quatre (troisième ligne), donc dans tous les contextes, alors on ajoutera une croyance indiquant que la sous-connaissance `SousConnaissance` du mot Nicolas est acquise (quatrième ligne), en l'occurrence ici la représentation orthographique.

4.2.4. Quatrième niveau de synthèse : la connaissance

Un dernier niveau de synthèse que l'agent peut effectuer est celui consistant à synthétiser les croyances synthétiques des sous-connaissances en une croyance synthétique au niveau connaissance. Cette croyance représentera en quelque sorte le « Graal » de la connaissance gérée par l'agent puisque c'est ce niveau de synthèse qui indiquera que la connaissance est acquise. Cette croyance se formalise de la façon suivante :

```
bel(<statut>).
```

Dans notre exemple, on pourrait avoir :

```
bel(acquis).
```

Cela indiquerait que l'objet mot Nicolas est acquis. Le système n'aurait donc plus besoin de travailler cette connaissance avec l'apprenant.

La règle de synthèse consiste donc à dire que : « Si toutes les sous-connaissances d'une connaissance sont acquises alors la connaissance est acquise ». Cette règle se formalise de la façon suivante :

```
regleSyntheseConnaissance1 :-
      findall(SousConnaissance, bel(SousConnaissance, acquis),
NbSousConnaissancesConnues),
      length(NbSousConnaissancesConnues, Result),
      Result = 3,
      assert(bel(acquis)).
```

Cette règle construit la liste des sous-connaissances acquises (première ligne) puis en mesure la taille, c'est-à-dire le nombre de sous-connaissances acquises (deuxième ligne) et si cette taille est égale à trois (troisième ligne), donc correspond au nombre de sous-connaissances total, alors on ajoutera une croyance indiquant que la connaissance est acquise(quatrième ligne), en l'occurrence le mot Nicolas.

Nous n'avons présenté ici qu'une partie des règles de synthèse existantes et des croyances existantes pour ce type d'agent. L'objectif était uniquement de présenter les différents niveaux de synthèse de la connaissance gérée que l'on pouvait avoir et non d'être exhaustif quant aux règles utilisées. Nous n'avons donc pas abordé tous les changements de statut possibles.

On peut donc résumer les niveaux de synthèse pour chaque connaissance suivant deux axes :

- un axe de synthèse temporelle : il consiste à synthétiser les croyances dans le temps grâce à l'application de règles de synthèse temporelle;
- un axe de synthèse hiérarchique : il consiste à synthétiser les connaissances d'un niveau en un niveau de connaissance supérieur.

5. Conclusion

Au cours de ce chapitre, nous avons donc expliqué comment fonctionnait chacun des différents types d'agents d'AGMA. Nous avons expliqué sur des exemples concrets, quelle était la forme des croyances de chaque agent ainsi que la forme des règles qu'il pouvait mettre en oeuvre.

Les AC sont les agents initiateurs de la modélisation de l'apprenant. Un AC commence par aller chercher le nouveau compte-rendu XML présent dans la base des comptes-rendus. Ensuite, il décompose celui-ci en autant de sous-comptes-rendus qu'il y a de situations didactiques dans la séquence. Chaque sous-compte-rendu sera envoyé à l'AA en charge de l'analyse de la situation didactique correspondante.

Les AA sont les agents qui sont responsables d'émettre toutes les hypothèses possibles sur une situation didactique. Il existe un AA par situation didactique et celui-ci n'est capable d'analyser que celle-ci. A partir des données présentes dans le compte-rendu et des connaissances dont il dispose, il va émettre, grâce à des règles d'analyse, ces dites hypothèses. Pour cela, il passera d'abord par une phase de conversion du compte-rendu XML qu'il a reçu. Il intégrera ensuite le résultat de sa conversion à ses croyances. Ensuite, l'agent mettra en oeuvre une phase de pré-analyse dans laquelle des calculs seront effectués. On comptera par exemple le nombre de fois où l'aide a été utilisée ou bien encore le nombre de réponses correctes. Cette phase de pré-analyse a pour but de faciliter la phase d'analyse. Ensuite, cette dernière consiste à appliquer un ensemble de règles permettant d'émettre des hypothèses sur l'état de savoir de l'apprenant. Une fois ces hypothèses réalisées, elles sont envoyées aux AK concernés.

Les AK sont les « derniers maillons » de la modélisation de l'apprenant. Ils ont pour responsabilité de gérer une petite partie du modèle de l'apprenant. Pour cela, ils vont intégrer les hypothèses reçues de la part des AA et mettre à jour en conséquence leurs bases de connaissances représentant cette partie du modèle de l'apprenant. Leurs bases de connaissances possèdent plusieurs niveaux de synthèse permettant ainsi un description fine de la connaissance gérée. Pour réaliser ces synthèses, chaque AK dispose d'un ensemble de règles de synthèse propre.

Dans le chapitre suivant, nous présenterons les outils retenus ainsi que leur mise en oeuvre pour réaliser un premier prototype d'AGMA. Nous nous appuierons sur le contenu de ce chapitre pour voir comment la formalisation présentée sous forme Prolog s'est vue implémentée dans une application réelle.

CHAPITRE 5 : SPÉCIFICATIONS ET IMPLÉMENTATION D'UN PROTOTYPE D'AGMA

Ce chapitre a pour but de présenter la base et les outils pour implémenter un futur prototype complet d'AGMA. En effet, en raison du manque de certains travaux, nous n'avons pu implémenter qu'une partie restreinte de notre agent. De plus certains autres agents de l'environnement AMICAL étant toujours en cours de développement, il a également été impossible de réaliser AGMA dans un environnement AMICAL fonctionnel. Néanmoins, l'objectif était ici plus de présenter les outils et la manière de les mettre en œuvre que de réaliser un agent complet. Nous présenterons donc dans ce chapitre les outils retenus pour concevoir AGMA et la manière de les mettre en œuvre à travers quelques exemples.

Dans le chapitre précédent, nous avons présenté les différents agents d'AGMA ainsi que leur fonctionnement : les agents de communication, les agents d'analyse et les agents de connaissance. Nous avons montré comment l'agent d'analyse allait récupérer le compte-rendu et le découper pour l'envoyer aux agents d'analyse en charge de chacune des situations didactiques de la séquence. Ensuite, nous avons vu comment ces mêmes agents d'analyse exploitaient le sous-compte-rendu correspondant à la situation didactique dont il avait la charge : ils commençaient par le traduire en croyances ; puis à partir de ces croyances, lançaient une phase de pré-analyse ; puis, à partir des règles d'analyse dont ils disposaient, ils émettaient leurs hypothèses quant à l'état de savoir de l'apprenant puis les envoyaient aux agents de connaissance concernés. Enfin, nous avons détaillé la façon dont ces agents de connaissance synthétisaient leurs croyances en de nouvelles croyances synthétiques à partir des hypothèses qu'ils recevaient des agents d'analyse.

Nous avons donc développé un prototype d'AGMA de manière autonome, c'est-à-dire sans l'intégrer au système AMICAL car un prototype complet de celui-ci n'est pas fonctionnel à l'heure actuelle, en raison du fait que nombre d'agents ne sont pas encore réalisés car étant toujours encore des travaux en cours. De plus, l'objet de cette thèse n'étant pas la réalisation informatique d'un prototype de l'environnement AMICAL, nous nous sommes exclusivement centré sur AGMA. Nous ne traiterons ni n'implémenterons donc pas les autres agents du système mais, en cas de besoin, nous simulerons les messages de ceux-ci au travers de « faux messages » envoyés à AGMA. De plus, nous nous contenterons donc de développer AGMA dans le cadre de son rôle de modélisation de l'apprenant et uniquement celui-ci. Développer AGMA d'une manière autonome n'est pas problématique car l'élément principal pour la modélisation de l'apprenant est le compte-rendu de la séquence de travail. Nous n'avons donc pas réellement besoin des autres agents du système pour développer un premier prototype.

Du fait que nous n'ayons pu développer qu'un prototype ne travaillant que sur quelques situations didactiques, tout simplement car nous n'en disposions pas d'autres, il a donc été impossible d'évaluer l'efficacité de l'agent. Néanmoins, comme nous l'avons dit, l'essentiel dans le développement de ce prototype était de retenir les outils les plus pertinents à mettre en oeuvre pour réaliser une application fonctionnelle et de montrer comment les utiliser.

Dans la suite de ce chapitre, nous parlerons de PAGMA pour désigner le prototype d'AGMA que nous avons réalisé.

Ce chapitre montrera comment nous avons implémenté dans une application réelle les agents que nous avons présentés au cours des deux précédents chapitres. Néanmoins, nous ne développerons essentiellement que l'implémentation des éléments les plus pertinents dans le cadre de PAGMA : l'utilisation de la plateforme multi-agents Jade (jade) et le moteur d'inférences Drools (drools). Le plan de celui-ci sera le suivant : nous commencerons par définir le cahier des charges assigné à ce prototype. Ensuite, nous détaillerons les outils que nous avons retenus pour développer notre prototype ainsi que les raisons qui nous ont poussées à les choisir. Ensuite, nous spécifierons quelques généralités concernant l'ensemble des agents développés dans le cadre de ce prototype. Puis, nous verrons comment les éléments détaillés au chapitre précédent sont implémentés au sein de chaque type d'agent. Enfin, nous présenterons les écrans principaux du

prototype.

1. Cahier des charges

1.1. Diagramme de cas d'utilisation

Le petit diagramme de cas d'utilisation ci-dessous permet de définir les fonctionnalités qui seront réalisées par notre prototype d'agent.

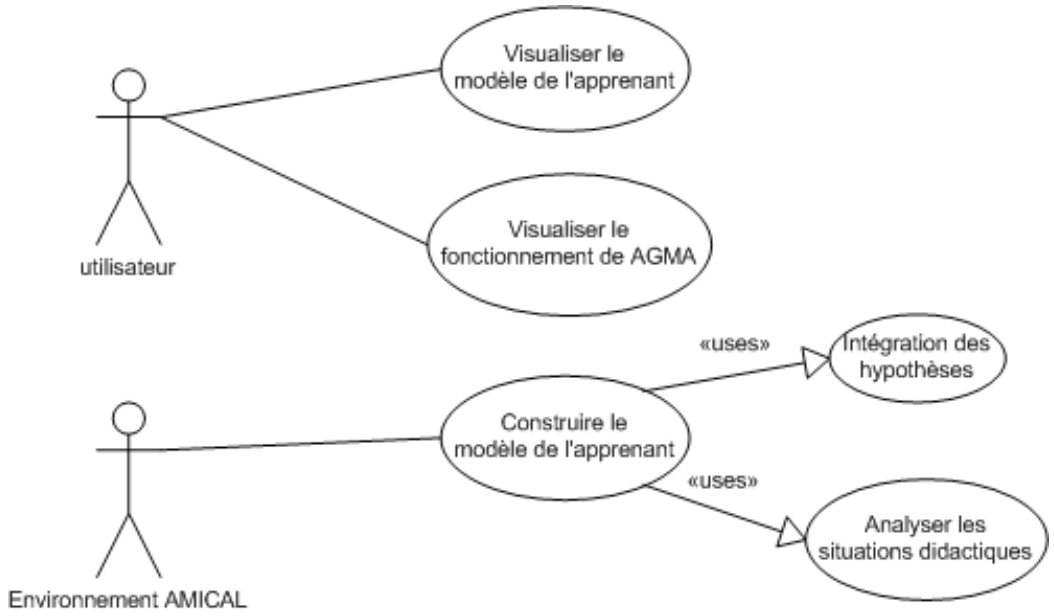

Figure 5.1 : Diagramme de cas d'utilisation d'AGMA

Comme nous le voyons sur la Figure 5.1, l'utilisateur de l'agent pourra donc à la fois visualiser le modèle de l'apprenant et visualiser AGMA en fonctionnement. Pour cela, il disposera donc de deux interfaces graphiques : l'une permettant la visualisation du modèle de l'apprenant et l'autre permettant de voir le fonctionnement d'AGMA, c'est-à-dire essentiellement de voir les différents messages et actions des différents agents composant AGMA.

L'autre fonctionnalité que nous développerons concernera bien entendu le thème principal de cette thèse, à savoir la construction du modèle de l'apprenant. Pour cela, il conviendra bien entendu de développer les processus capables d'analyser les situations didactiques ainsi que d'intégrer les hypothèses au modèle existant, c'est-à-dire tout simplement de réaliser l'architecture multi-agents détaillée dans les chapitres précédents : les agents de communication, les agents d'analyse et les agents de connaissance.

1.2. Spécifications et contraintes

Les écoles disposant de machines variées (PC ou Mac) et donc par là même de systèmes d'exploitation tout aussi variés, il nous a paru plus qu'évident qu'il convenait de choisir une plateforme logicielle pouvant fonctionner sur plusieurs types de systèmes d'exploitation et de plateforme, au lieu de développer une version pour chaque système. En effet, les PC des écoles pouvant tourner sur Windows ou bien Linux et les Mac utilisant Mac OS X, il convient donc de choisir la plateforme logicielle pouvant fonctionner sur tous ces types de systèmes d'exploitation sans avoir à tenir compte des spécificités de chaque environnement matériel et logiciel et donc sans avoir à écrire plusieurs versions.

Même si l'environnement AMICAL n'est pas complètement fonctionnel aujourd'hui, nous devons quand même penser au futur. Quand celui-ci sera développé, ses agents seront probablement répartis sur plusieurs ordinateurs et devront donc pouvoir communiquer entre eux facilement. Il convient donc également de prendre en compte les capacités réseaux de la plateforme logicielle retenues.

PAGMA étant un système multi-agents utilisant des normes FIPA comme les messages ACL ou le *service directory*, nous devons donc utiliser une plateforme multi-agents nous permettant à la fois de concevoir les différents agents de notre prototype de la manière la plus souple possible et de disposer de ces normes sans avoir à les développer nous-mêmes.

Les agents d'AGMA utilisant des règles d'inférences à la fois en ce qui concerne les règles d'analyse et les règles de synthèse que nous avons abordées au chapitre précédent, nous avons également besoin de disposer d'un moteur d'inférences que

nous devrons intégrer aux agents. Bien évidemment, au lieu de le développer nous-mêmes, il serait préférable de disposer d'un moteur déjà existant que nous n'aurions plus qu'à intégrer au sein de nos agents. Ainsi, nous n'aurions juste qu'à concevoir les différentes règles de nos agents.

Bien évidemment, l'élément fondamental de la modélisation de l'apprenant, à savoir le modèle de l'apprenant devra être stocké d'une manière ou d'une autre. Cela reviendra à s'interroger sur le stockage des croyances des différents agents de connaissance.

Nous devrons aussi prévoir le stockage des différents comptes-rendus. Pour cela, il conviendrait tout naturellement d'adopter une base de données native XML.

Notre prototype devant donc exploiter à la fois des technologies agents ainsi que des capacités de moteur d'inférences, il convient donc de réfléchir à l'intégration facile de tous ces éléments au sein de la plateforme logicielle retenue et de choisir ces outils en fonction de celle-ci.

De plus, les écoles françaises ayant des moyens financiers très restreints, il convient donc d'envisager d'utiliser en priorité des éléments logiciels gratuits ou open source.

2. Outils retenus

En partant des spécifications que nous avons décrites dans le paragraphe précédent, nous allons maintenant détailler les outils que nous avons retenus ainsi que les raisons qui nous ont poussé à les choisir.

2.1. Plateforme Java

Concernant le choix de la plateforme logicielle, il nous a paru évident de porter tout naturellement notre choix sur la plateforme Java (java) et son langage associé. En effet, Java ou le célèbre « Write once, run everywhere » permet de n'écrire qu'une fois le programme et de l'exécuter sur les systèmes d'exploitation les plus utilisés : Windows, Unix, Linux et Mac OS. Rapidement, on peut signaler

que Java est à la fois la plateforme elle-même et le langage qui permet de développer sur cette plateforme. Java est un langage de programmation orienté objet qui permet de développer tous les types d'applications informatiques. De plus, il est très facile de développer des composants pour cette plateforme.

Le choix de la plateforme logicielle étant fait, toutes les technologies choisies par la suite devront être utilisable avec Java.

2.2. Plateforme multi-agents Jade

En ce qui concerne la réalisation des agents en Java, deux choix étaient possibles. Le premier choix aurait été de réaliser les agents nous-mêmes (Bigus et Bigus, 2001), en implémentant les différentes fonctionnalités comme la communication entre les agents, la prise en charge d'ACL ou encore le service directory. Le deuxième choix, plus naturel celui-là, a été de choisir une plateforme multi-agents en Java, à la fois la plus souple possible pour développer nos agents, la plus complète possible et si possible gratuite. Nous avons retenu la plateforme Jade (jade), (Bellifemine et al., 2007) en raison de ces critères.

Jade permet de développer nos agents avec une grande liberté quant au fonctionnement de ceux-ci tout en implémentant les différents éléments de la norme FIPA que nous avons évoqués au chapitre 3 comme la prise en charge d'ACL ou les différents services d'annuaires permettant d'enregistrer les agents.

Jade est une plateforme qui facilite donc le développement des systèmes multi-agents en intégrant :

- un environnement d'exécution où les agents vivent une fois ceux-ci lancés,
- une bibliothèque de classes qui permet de développer les agents et qui fournit à ceux-ci tous les éléments leur permettant de communiquer entre eux et de s'enregistrer ou de consulter les différents services d'annuaires,
- des outils graphiques permettant l'administration et la visualisation des différents agents s'exécutant sur la plateforme.

Développer un agent en Jade consiste simplement à concevoir une classe Java qui étend la classe Agent de Jade implémentant deux méthodes :

1. *setup()* qui correspond à l'initialisation de l'agent. C'est dans cette

méthode, par exemple, que l'agent va s'enregistrer auprès des différents types d'annuaires.

2. *takeDown()* qui correspond à la fin de l'agent. C'est dans cette méthode, par exemple, que l'agent va se désenregistrer de ces mêmes annuaires.

Ensuite, il convient d'ajouter à cet agent précédemment défini les comportements dont il a besoin pour effectuer sa tâche. En effet, un agent Jade doit posséder des comportements dans le but de réaliser ce pour quoi il est conçu. Les comportements sont en principe lancés dans la méthode *setup()*. Jade fournit plusieurs types de comportements prédéfinis suivant ce que doit réaliser l'agent, par exemple des comportements ne s'exécutant qu'une seule fois ou bien au contraire en boucle. Néanmoins, on peut concevoir nos propres comportements si les tâches à réaliser sont plus complexes. Développer le comportement de l'agent se fait en créant une classe Java étendant la classe Behaviour de Jade, que l'on intègre ensuite à la classe de l'agent précédemment réalisée.

2.3. Moteur de raisonnement Drools

Pour le moteur de raisonnement des agents, nous avons retenu le moteur d'inférences Drools (drools), là encore, pour des raisons d'intégration facile à Java, pour sa puissance ainsi que pour sa gratuité. De plus, Drools est aujourd'hui sous la coupe de JBoss. Drools nous permet également de mettre en œuvre un raisonnement temporel, ce qui est nécessaire pour au moins deux types d'agents : les agents d'analyse et les agents de connaissance.

Drools est un moteur d'inférences qui permet de raisonner sur des objets et qui utilise le célèbre algorithme Rete en version objet. Ce qui est utile dans le cadre de nos agents puisque nous avons bien entendu représenté les croyances de ceux-ci, elles aussi comme des objets.

Drools utilise essentiellement deux types de concepts :

- *la working memory* : il s'agit de la base de faits qui permet de stocker les faits représentés par des objets,
- *la rule base* : il s'agit de la base de règles d'inférences qui permet, à partir des faits de la working memory, de faire des inférences et ainsi obtenir de nouveaux faits ou bien modifier ou supprimer ceux qui existent.

Développer avec Drools revient donc essentiellement à écrire les différentes règles permettant d'inférer à partir des objets intégrés à la working memory. Un fait en Drools sera donc un objet qui sera intégré à la working memory. Par exemple, un objet Personne disposant des attributs nom, prénom et age. Une règle en Drools sera quant à elle intégrée à la rule base. Une règle permettant d'utiliser l'objet Personne serait par exemple :

```
rule "regle1"
when
      Personne(age >= 18)
then
      System.out.println("La personne est majeure");
end
```

Dans cet exemple, nous définissons une règle (rule) qui porte le nom regle1. Celle-ci indique que si l'age d'une personne est supérieur à 18 alors on en conclut que celle-ci est majeure. Le when correspond au si et porte sur l'objet Personne et le then correspond au alors et affiche ici un message à l'écran.

Nous verrons dans la suite des exemples concrets concernant les règles d'analyse et de synthèse.

2.4. Stockage des comptes-rendus XML : dbXML

Pour le stockage des comptes-rendus XML, nous nous sommes tout naturellement tournés vers une base native au format XML. Notre choix s’est porté sur la base XML dbXML (dbxml), maintenant exploitée par Oracle. Celle-ci étant à la fois gratuite, exploitable en Java et soutenu par un grand constructeur de base de données, le choix a donc paru évident.

En plus du stockage des compte-rendus XML, nous devons également prévoir l'exploitation des documents XML. En effet, un agent de communication doit pouvoir découper ce fichier en sous-comptes-rendus alors qu'un agent d'analyse doit pouvoir lire ce document XML pour le convertir en croyances. Pour réaliser cette tâche, nous choisissons la bibliothèque dom4j (dom4j) qui fournit la

possibilité de lire et de traiter facilement les fichiers XML en Java. Nous la choisissons pour sa facilité d'utilisation et son utilisation très répandue au sein des projets open source.

2.5. Stockage des modèles de l'apprenant : Hibernate et Postgresql

Pour stocker les modèles de l'apprenant, en fait une partie des connaissances des agents de connaissances, nous aurions aimé choisir une base objet car facilement utilisable avec le moteur Drools. En effet, Drools utilisant des objets pour représenter les différents faits, il était donc naturel d'envisager ce choix. Malheureusement, les bases de données objet étant d'un coût considérable, et voulant rester sur des outils libres, nous n'avons pu en retenir une. Notre choix s'est donc porté sur une base de données relationnelle, PostqreSQL (postgresql) et sur un outil de mapping relationel-objet Hibernate (hibernate). Ainsi à travers la combinaison de ces deux outils, nous pouvons simuler une base de données objets. Hibernate permettra aux agents d'utiliser des objets alors que ceux-ci seront stockés dans une base de données relationnelle.

Au final, tous les éléments retenus s'intègrent parfaitement à la plateforme Java, sont gratuits et libres d'utilisation et donc facilement distribuables au sein des écoles françaises.

3. Démarche générale suivie pour l'implémentation

Comme l'objectif est de développer un prototype mais que malheureusement nous ne disposons pas encore de toutes les situations didactiques possibles, nous avons donc dû faire avec celles qui étaient disponibles.

Nous avons bien évidemment réfléchi à la conception de chaque type d'agent avant de commencer à l'implémenter. Par contre, quand nous avons dû réaliser chaque agent particulier du prototype, il s'est avéré nécessaire, pour ne pas trop se perdre, de le faire de manière méthodique.

Partant des activités qui existaient, nous avons retenu la démarche suivante pour

simplifier le développement de notre prototype :

1. prendre une activité existante,
2. développer l'agent d'analyse correspondant,
3. développer les différents agents de connaissance concernés s'ils n'existent pas ou bien les modifier s'ils existent.

La démarche consiste donc par la suite à itérer sur ces trois étapes pour chaque situation didactique disponible. L'ajout de nouvelles activités au sein du prototype revient donc « simplement » à réaliser les étapes 2 et 3.

4. Précisions sur l'implémentation

On peut synthétiser les différentes couches d'outils retenus par le schéma suivant :

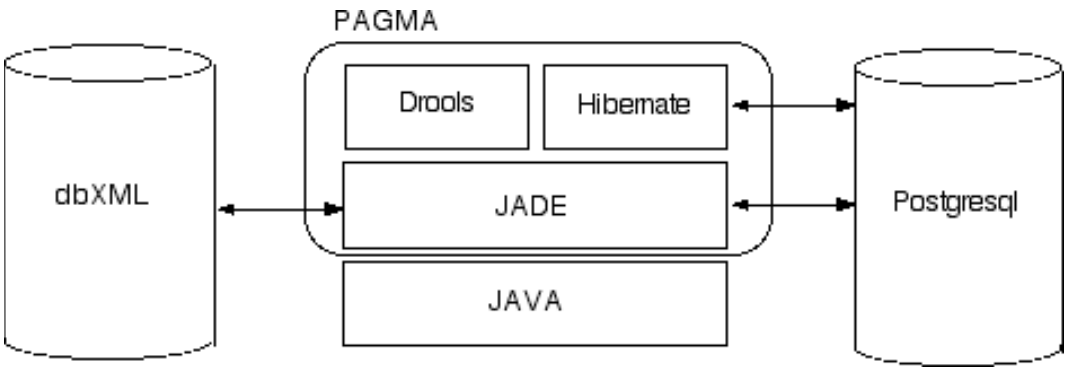

Figure 5.2 : Couches logiciels de PAGMA

L'ensemble du prototype fonctionne bien évidemment sous un environnement Java. Tous les agents développés au cœur de ce prototype le sont sous forme d'agent Jade mais tous n'intègrent pas les mêmes modules. Certains disposent de Drools et Hibernate et d'autres pas.

Les agents de communication doivent accéder à la base des comptes-rendus et utilisent donc un driver pour accéder à la base dbXML. Le fonctionnement d'un AC ne nécessitant pas l'utilisation de règles d'inférences, il n'intègre donc pas Drools et par là même n'intègre pas non plus l'utilisation d'Hibernate pour accéder à postgresql. Par contre, ils doivent pouvoir lire un fichier XML pour en extraire les sous-comptes-rendus et doivent donc pour cela utiliser la bibliothèque dom4j.

Les agents d'analyse et de connaissance, quant à eux, utilisent des règles d'inférences. Pour cela, ils doivent utiliser Drools, ce qui conduit donc à disposer d'un ensemble de croyances intégrées dans la working memory. Les agents d'analyse, en plus, utiliseront la biliothèque dom4j pour convertir le fichier XML en croyances.

4.1. Stockage des croyances et règles d'inférences

La working memory étant un lieu de stockage temporaire et seulement utilisée quand l'agent est exécuté, nous devons stocker les croyances incluses dans celle-ci de manière durable. Au lieu de les stocker dans des fichiers ou bien de les écrire « en dur », nous avons préféré les stocker toutes au même endroit, c'est-à-dire dans une base de données. Dans ce but, ces agents utilisent Hibernate pour stocker les croyances dans la base de données postgresql. Hibernate fournit à l'agent un mécanisme permettant de convertir les objets Java intégrés à la working memory en données dans des tables de la base de données relationnelle. Pour cela, Hibernate utilise un fichier lui permettant de savoir à quel champ de quelle table de la base de données correspond tel attribut de l'objet de la working memory.

De plus, les règles d'inférences des agents sont stockées dans des fichiers qui sont chargés au lancement de l'agent. Au lieu de stocker ces fichiers dans un répertoire de l'application, nous avons également préféré stocker ceux-ci au même endroit. Nous avons pour cela décidé de les stocker eux aussi dans une base de données. En effet, cela permettrait de ne modifier, d'ajouter ou de ne supprimer des éléments que dans la base de données au lieu d'avoir à éditer chaque fichier. Cela permettrait par exemple de faire les modifications à distance. Par exemple, on pourrait envisager que cette base de données ne se trouve pas physiquement au même endroit que l'agent. On pourrait imaginer que les différents agents se trouvent sur les ordinateurs à l'école alors que la base de données se trouverait dans notre laboratoire. Ainsi, au lieu de se déplacer à chaque fois à l'école pour effectuer des modifications, nous n'aurions qu'à changer les fichiers de la base. De cette manière, les apprenants disposeraient toujours de la dernière version possible des agents.

Au final, la base de données postgresql contient donc les croyances des agents

d'analyse et de connaissance ainsi que les fichiers contenant leurs règles d'inférences. Cela permettrait également de faciliter la sauvegarde des éléments du modèle de l'apprenant.

4.2. Méta-comportement

Tout agent de PAGMA possède un ensemble de comportements lui permettant de réaliser sa tâche. Pour faciliter la conception et l'implémentation de ces comportements, nous avons choisi de doter chaque type d'agent d'un méta-comportement, c'est-à-dire d'un comportement qui gèrera les autres comportements. Il permet de contrôler les différentes actions et communications de l'agent. Celui-ci, en fonction des messages qu'il recevra de la part des autres agents ou en fonction de certaines perceptions ou croyances, décidera d'appliquer tel ou tel comportement spécifique permettant de réaliser telle tâche. Il s'agit en quelque sorte d'une « gare de triage » des comportements des agents. Le méta-comportement dispose de règles du type :

Si je reçois un message *m* de type *t* en provenance de l'agent *a* **alors** déclencher tel comportement.

Le méta-comportement est un comportement qui s'exécute indéfiniment, c'est-à-dire qui tourne en boucle jusqu'à ce qu'un événement se produise (message, croyance, …). Ce méta-comportement est une classe Java qui étend la classe CyclicBehaviour de Jade, comportement spécial qui tourne en boucle. Un exemple de méta-comportement en Jade peut être les suivant :

```
class Metacomportement extends CyclicBehaviour {
      ...
      ACLMessage message = myAgent.receive();
      ...
      if (message.getPerformative() == ACLMessage.REQUEST) {
            addBehaviour(new OneShotBehaviour() {
                  public void action() {
                        traiterDemande(message.getContent());
                  }
            }
```

```
        }
        ...
}
```

Nous définissons ici une classe Metacomportement étendant la classe CyclicBehaviour de Jade. Dans cet exemple volontairement simple et incomplet pour ne pas alourdir l'exposé, le méta-comportement attend un message de la part de n'importe quel agent. Ensuite, si ce message est un message demandant l'exécution d'une action (performative REQUEST), on déclenchera le comportement visant à répondre à cette de demande. Ce méta-comportement déclenche donc les autres comportements de l'agent suivant les évènements. Nous ne détaillerons pas davantage les méta-comportements des agents qui ont été réalisés uniquement dans un but de programmation.

4.3. Classe AgentDrools

Les agents d'analyse et de connaissance étendent la classe AgentDrools que nous avons réalisée. Le diagramme de classe UML suivant permet de détailler la composition de cette classe AgentDrools :

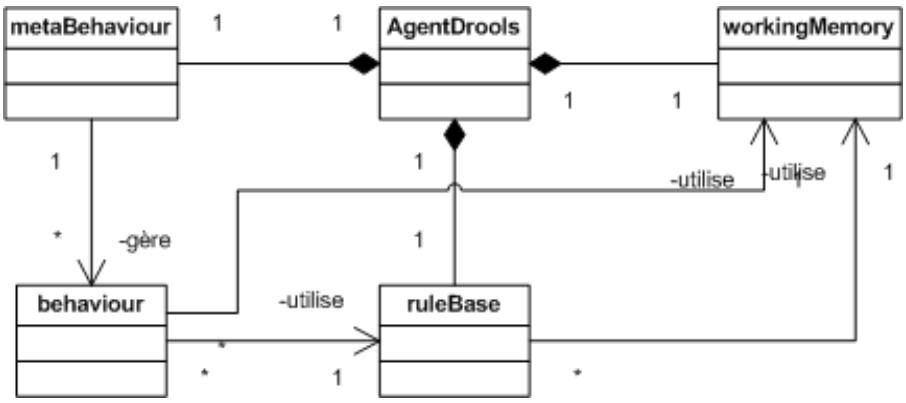

Figure 5.3 : Diagramme UML de la classe AgentDrools

Un AgentDrools est tout simplement un agent Jade auquel nous avons intégré le moteur d'inférences Drools. Pour cela, un AgentDrools utilise une working memory et possède une rule base. Comme tout agent Jade, il dispose également

d'un ensemble de comportements que nous avons choisi de faire gérer par le méta-comportement vu précédemment.

Cette classe a été réalisée pour simplifier le développement des agents d'analyse et de connaissance. Ainsi, chacun de ces agents disposera de méthodes prédéfinies pour charger sa base de croyances et sa base de règles d'inférences à partir de la base de données.

4.4. Initialisation des agents

Au démarrage du prototype, chaque agent s'initialise de différentes manières. Tous les agents s'enregistrent par les services qu'ils rendent. Les agents d'analyse et de connaissance, quant à eux, en plus, vont récupérer leurs croyances ainsi que leurs règles d'inférences stockées dans la base de données postgresql. Dans la suite, nous détaillons cette initialisation.

4.4.1. Enregistrement des agents

Comme nous l'avons dit au chapitre 3, nous avons décidé d'intégrer un mécanisme de service directory à AGMA pour que chaque agent le constituant puisse se trouver plus facilement. Ainsi, les agents de communication ont besoin de savoir quels agents d'analyse ils doivent contacter pour leur faire parvenir le sous-compte-rendu le concernant. Les agents d'analyse, eux, doivent pouvoir déterminer à quels agents de connaissance ils doivent envoyer les hypothèses qu'ils ont émis sur l'état de savoir de l'apprenant. Pour cela, chaque agent qui a besoin de contacter un autre agent le recherche par le service qu'il rend, en consultant le service directory, c'est-à-dire les page jaunes. Pour réaliser ce service de pages jaunes, la plateforme Jade dispose d'un élément appelé Directory Facilitator ou DF qui fonctionne suivant le même principe : un agent qui rend un service s'enregistre auprès du DF et un agent qui a besoin d'un service particulier peut consulter le DF pour savoir quel agent le rend. Dans le cas de PAGMA, les agents utilisent le DF pour rechercher à qui envoyer soit le sous-compte-rendu pour les agents de communication, soit les hypothèses pour les agents d'analyse.

A son lancement, chaque agent de PAGMA s'enregistre donc auprès du DF. Celui-

ci est effectué à l'initialisation de l'agent dans la méthode *setup()*. L'enregistrement auprès du DF se réalise avec Jade de la manière suivante :

```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sd1 = new ServiceDescription();
sd1.setName("Service 1");
sd1.setType("Type 1");
dfd.addServices(sd1);
ServiceDescription sd2 = new ServiceDescription();
sd2.setName("Service 2");
sd2.setType("Type 2");
dfd.addServices(sd2);
...
DFService.register(this, dfd);
```

L'objet dfd correspond à la description de l'agent désigné par son identifiant (fonction getAID()). Cette description contient tous les services que l'agent peut rendre (objets sd1 et sd2). Chaque service possède un nom et un type. Sur l'exemple précédent, l'agent est capable de rendre deux services sd1 et sd2. Le service sd1 porte le nom Service 1 et est de type Type 1. Le service sd2 porte le nom Service 2 et est de type Type 2. Une fois la description de l'agent remplie, nous l'enregistrons auprès du DF (DFService.register).

Au sein de PAGMA, chaque type d'agent s'enregistre de la manière suivante :

Type d'agent	Nom du service	Type du service
Agent de communication	numéro	communication
Agent d'analyse	identifiant SD	analyse
Agent de connaissance	<connaissance>-<type de connaissance>-<instance de connaissance>	connaissance

Le type de service correspond tout simplement au type d'agent : communication pour les agents de communication, analyse pour les agents d'analyse, connaissance pour les agents de connaissance.

Les agents de communication portent comme nom un numéro pour les distinguer des autres agents de communication. En effet, les agents rendent tous le même service et aucun d'entre eux n'a de rôle particulier. En fait, ce numéro n'a que peu d'importance car ce que recherchent les autres agents quand ils souhaitent utiliser les services d'un agent de communication, c'est uniquement le premier qui est disponible.

Les agents d'analyse portent comme nom l'identifiant de la situation didactique qu'ils ont en charge d'analyser. En effet, comme nous l'avons déjà dit, chaque situation didactique porte un identifiant unique qui permet de la distinguer des autres.

Le nom des agents de connaissance est un nom composé de la façon suivante : <connaissance>-<type de connaissance>-<instance de connaissance>

<*connaissance*> correspond bien entendu à la connaissance du domaine gérée : objet, concept ou stratégie. <*type de connaissance*> correspond à une connaissance particulière. Par exemple, lettre ou mot. <*instance de connaissance*> n'est utilisé que dans le cadre des objets. En effet, contrairement aux concepts et aux stratégies, un objet porte sur une instance particulière. Par exemple, on trouve l'objet lettre a ou l'objet lettre b, qui ne renvoient pas à la même lettre et donc pas au même agent. Ce champ n'est pas rempli dans le cadre des concepts et des stratégies. Donnons un exemple pour clarifier la manière de nommer les services : Concept-Lettre correspondra au concept de lettre alors que Objet-Mot-Nicolas représentera l'objet mot Nicolas.

Nous verrons comment un agent accède à ce DF par la suite sur un exemple.

4.4.2. Récupération des croyances et règles d'inférence

A son lancement, en plus de s'enregistrer auprès du DF, chaque agent d'analyse et chaque agent de connaissance va rechercher ses croyances et règles d'inférences dans la base de données postgresql. Ces opérations de récupération sont également réalisées lors de l'initialisation de chaque agent, c'est-à-dire dans la méthode *setup()* de l'agent.

Les croyances sont stockées dans la base de données sous forme relationnelle, c'est-à-dire dans des tables de la base de données. Or, la working memory ne contenant que des objets, il convient donc pour chaque agent de traduire les données stockées de manière relationnelle sous forme d'objets. Pour cela, les agents vont utiliser Hibernate pour aller récupérer les données dans la base et les convertir en objets. Ensuite, ces objets seront ajoutés à la working memory de l'agent. Pour réaliser ce travail Hibernate utilise des fichiers de mapping, c'est-à-dire des fichiers permettant de faire la correpondance entre les champs des tables de la base de données et les attributs des différents objets qui seront intégrés à la working memory. Nous ne détaillerons pas davantage ce mécanisme qui serait hors du champ de ce document.

Les règles d'inférences sont également stockées dans cette même base de données mais d'une manière différente. En effet, chaque agent dispose d'un fichier contenant l'ensemble de ses règles d'inférences. Et c'est ce fichier qui est stocké au sein de la base de données mais de manière binaire. Au sein de postgresql, nous disposons d'une table qui associe à chaque agent, le fichier de règles d'inférences correspondant. Cette table possède la définition suivante :

```
CREATE TABLE (
        nom_agent           text
        regles_inferences   oid
);
```

Cette table contient tout simplement deux champs : le premier correspond au nom de l'agent et est de type chaine de caractère et le deuxième correspond au fichier de règles d'inférences de l'agent et est une référence au fichier stocké dans la base de données.

Chaque agent, pour récupérer son fichier de règles d'inférences se connecte directement à la base postgresql, sans passer par Hibernate, en utilisant le driver JDBC fourni avec la base de données. Nous rappelons qu'un driver JDBC (jdbc) permet de se connecter à une base de données directement à partir de Java. Là encore, nous ne rentrerons pas plus en détail dans le mécanisme.

4.5. Arrêt des agents

Comme nous l'avons dit, l'arrêt des agents en Jade correspond à la méthode takeDown().

Quand un agent s'arrête, celui-ci doit se désenregistrer du DF pour indiquer qu'il ne sera plus disponible pour les autres agents. Il le réalise grâce à la commande suivante :

DFService.deregister(this);

Les agents utilisant une working memory doivent, en plus, aller stocker les croyances présentes dans celle-ci dans la base de données postgresql. Pour cela, ils utiliseront Hibernate.

Après avoir présenté PAGMA dans ses grandes lignes, voyons un peu plus en détail l'implémentation de chacun de ses agents. On supposera dans la suite que *message* est un ACLMessage, c'est-à-dire un message ACL de Jade.

5. Les agents de communication

Dans son rôle de modélisation de l'apprenant, l'agent de communication est en attente d'un message lui indiquant la disponibilité d'un nouveau compte-rendu. Une fois ce message reçu, il déclenchera un comportement visant à aller chercher ce compte-rendu XML dans la base dbXML. Une fois ce compte-rendu XML récupéré, un autre comportement visant à exploiter ce compte-rendu sera déclenché. L'agent parcourra le fichier XML pour le découper en morceaux ; chaque morceau correspondant à une situation didactique et qui sera envoyé à l'agent d'analyse en charge de celle-ci.

5.1. Récupération du compte-rendu

Les fichiers XML des comptes-rendus sont stockés dans la base XML dbXML les uns à la suite des autres. Chaque fichier s'identifie par son nom. Celui-ci est de la forme :

<prénom de l'apprenant>_<date du compte-rendu>_<heure du compte rendu>.xml

où <prénom de l'apprenant> correspond au nom de l'apprenant ayant effectué la séquence de travail, <date du compte-rendu> correspond à la date à laquelle la séquence a eu lieu et <heure du compte rendu> correspond à l'heure à laquelle a été enregistré le compte-rendu. En effet, on suppose qu'il peut y avoir plusieurs comptes-rendus pour un même apprenant dans une journée ; ceux-ci se distinguant par l'heure de l'enregistrement.

Quand l'apprenant a terminé la session de travail que le système avait prévu, AGSSDI va stocker le compte-rendu construit pendant cette session dans la base dbXML. La forme du fichier XML d'une session est détaillée page 144. Une fois ce compte-rendu stocké dans la base dbXML, AGSSDI va en informer AGMA grâce à un message envoyé au médiateur. Le médiateur va envoyer au premier AC disponible l'information qu'un nouveau compte-rendu est disponible. AC recevra donc le message suivant en provenance du médiateur :

```
(inform
      :sender Médiateur
      :receiver AGMA
      :content "nouveauCR_morgane_20-09-2009_15-35-25.xml"
      :language Text)
```

Le contenu du message n'est en fait qu'une simple chaine de caractères de la forme : nouveauCR_<nom du fichier>. La première partie du contenu sert à indiquer qu'un nouveau compte-rendu est disponible alors que la seconde partie indique sous quel nom celui-ci a été stocké dans la base dbXML.

Ce message est détecté par le méta-comportement d'AC de la manière suivante :

```
if (message.getPerformative() == ACLMessage.INFORM &&
message.getContent().startsWith("nouveauCR")) {
...
```

Cela signifie que si la performative du message est INFORM et que le contenu de celui-ci commence par nouveauCR, alors l'agent va lancer un comportement qui

va aller récupérer le fichier de nom <nom du fichier> en se connectant à la base dbXML. Nous ne contrôlons pas le nom de l'agent qui a envoyé le message car celui-ci est nécessairement le médiateur. Le principe de ce comportement consiste simplement à se connecter à la base dbXML via le driver associé, puis à récupérer le fichier et à le stocker dans un objet File permettant ainsi le stockage du fichier pour pouvoir l'exploiter.

Ensuite, le comportement visant à exploiter ce compte-rendu sera exécuté. Le principe de celui-ci consiste à construire un vecteur d'objets *SituationDidactique* à partir de l'objet File en utilisant la bibliothèque XML dom4j. Un objet *SituationDidactique* est défini de la manière suivante :

```
public class SituationDidactique {
      private int identifiant_sd;
      private File sous_compte_rendu;
      ...
      public File getSousCompteRendu() {
            return sous_compte_rendu;
      }
}
```

Nous rappelons qu'un sous-compte-rendu se trouve entre les balises <sdt numero=...> et </sdt> au sein du compte-rendu de la séquence de travail. L'attribut numero de la balise correspond à l'identifiant de la situation didactique au sein du système. Pour un sous-compte-rendu, l'agent va donc construire un objet *SituationDidactique* en initialisant l'attribut *identifiant_sd* avec la valeur associée à l'attribut numero de la balise <sdt numero=...> et en initialisant l'attribut *sous_compte_rendu* avec le contenu XML compris entre les balises <sdt numero=...> et </sdt>. Il construira un objet pour chaque situation didactique de la séquence.

5.2. Envoi des messages

L'agent va ensuite parcourir le vecteur précédemment construit, puis pour chaque objet *SituationDidactique*, il va interroger le DF pour avoir l'identifiant de l'agent en charge d'analyser la situation didactique représentée dans l'objet par la valeur

de l'attribut *identifiant_sd*. Il enverra ensuite un message à cet agent pour lui demander d'analyser le sous-compte-rendu présent dans l'attribut *sous_compte_rendu* de l'objet.

On supposera qu'un objet *SituationDidactique* correspond à la variable *situation.* Pour interroger le DF, l'agent exécutera de la requête suivante :

```
DFAgentDescription template = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType("analyse");
sd.setName(situation.identifiant_sd);
template.addServices(sd);
try {
      DFAgentDescription[] result = DFService.search(myAgent,
template);
      agent = result[0].getName();
}
```

Pour faciliter la compréhension, on supposera qu'un seul agent peut répondre à un service et on supposera également que l'on trouve toujours le service. L'interrogation du DF est relativement simple : nous décrivons le modèle du service que nous recherchons puis nous lançons la recherche. Comme nous l'avons vu au paragraphe 4.4.1., le service rendu par un agent d'analyse est enregistré auprès du DF sous le nom de l'identifiant de la situation didactique dont il a la charge et sous le type analyse. C'est donc ce que nous devons rechercher ici. C'est pour cela que la description du service recherché porte ici sur le type analyse et sur le nom correspondant à la situation didactique contenue dans le compte-rendu. La variable agent, après la recherche, contient l'identifiant de l'agent remplissant ce service.

A partir de cette recherche, on construira le message ACL contenant la partie du compte-rendu :

```
ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
msg.addReceiver(agent);
msg.setContentObject(situation.getSousCompteRendu());
myAgent.msg(msg);
```

Le message construit porte ici la performative REQUEST demandant à l'agent qui recevra le message d'exécuter une action et sera envoyé à l'agent précédemment trouvé lors de la recherche. Le contenu de celui-ci correspondra au sous-compte-rendu XML. En effet, la méthode *setContentObject()* permet de transmettre directement le fichier.

L'agent fera la même chose pour toutes les situations didactiques présentes dans la séquence, c'est-à-dire pour chaque objet *SituationDidactique.*

6. Les agents d'analyse

Un agent d'analyse attend de recevoir une demande d'analyse d'un compte-rendu de la part d'un agent de communication. Dès qu'il la recevra, il convertira celui-ci en en ensemble de croyances qu'il ajoutera à sa working memory. Puis, il appliquera ses règles de pré-analyse puis ses règles d'analyse pour émettre les hypothèses quant à l'état de savoir de l'apprenant.

La réception de ces différents messages est traité par le méta-comportement. Nous avons déjà montré à quoi ressemblait la réception d'un message quand nous avons parlé des agents de communication, nous n'y reviendrons donc pas.

6.1. Conversion du compte-rendu

Le fichier XML reçu par l'agent est intégré directement au contenu du message qu'il reçoit. Il lui suffit donc tout simplement de regarder ce contenu pour récupérer le fichier XML dans un objet File :

```
...
File compte_rendu = message.getContent();
...
```

A partir de cet objet File et en utilisant la bibliothèque dom4j, l'agent convertira ce fichier XML en croyances objets qui seront intégrées à sa working memory. Les croyances définies page 151 sous forme Prolog sont représentées par les classes :

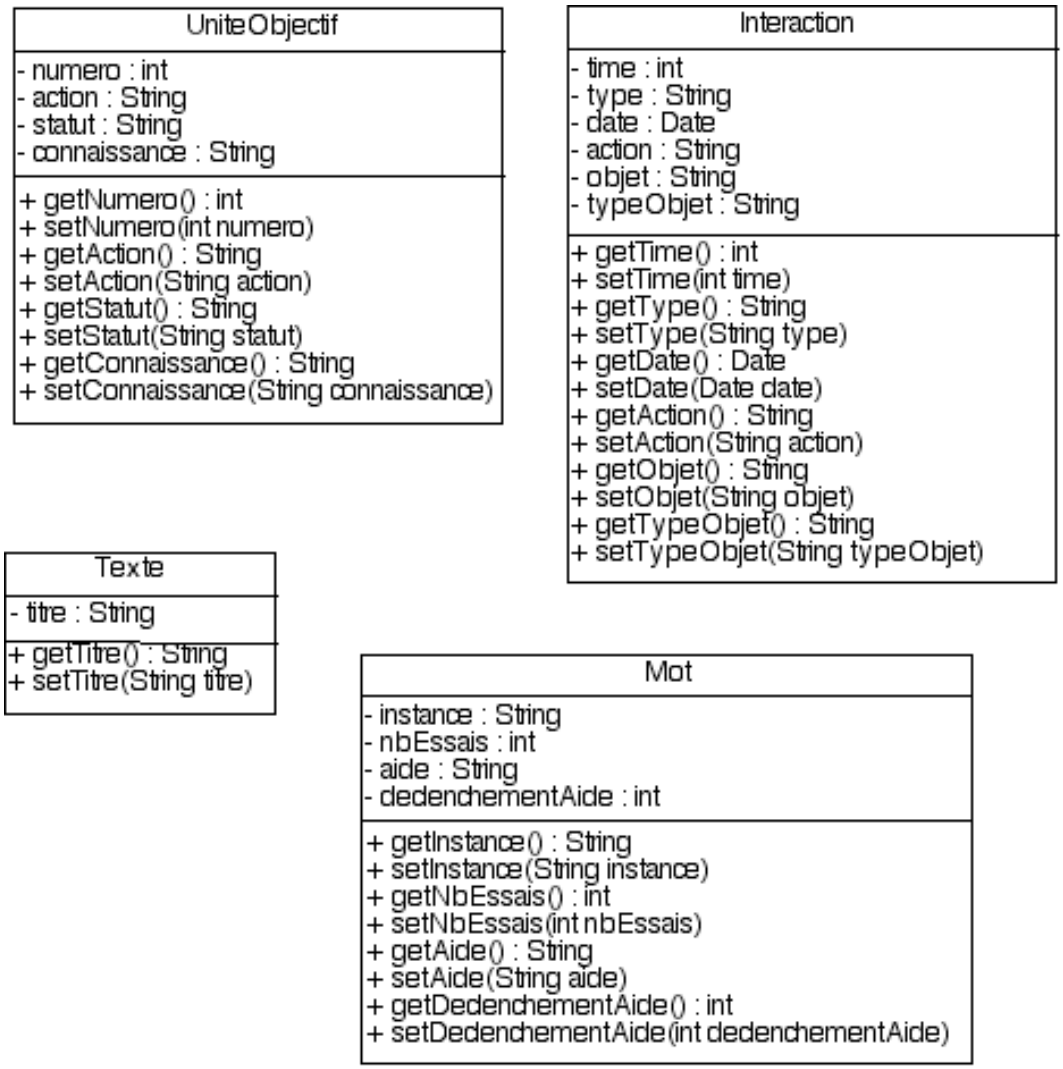

Figure 5.4 : Exemples de classes intégrées à la working memory

Comme les paramètres sont de taille variable en fonction du type de paramètre, nous avons choisi de définir une classe par type de paramètre. Par exemple, nous avons réalisé une classe Texte et une autre classe Mot.

Le fichier XML est parcouru et chaque partie est convertie en un objet particulier : les balises <unitéObjectif><unitéObjectif> sont converties en objets *UniteObjectif*, les balises <parametre></parametre> sont converties en objets correspondant au type du paramètre, par exemple, *Texte* ou *Mot* et les balises <interaction></interaction> sont converties en objets *Interaction*. La conversion est réalisée par un mécanisme d'introspection que nous ne détaillerons pas. Ce

mécanisme permet de créer dynamiquement des objets Java. Ces objets sont ensuite ajoutés à la working memory grâce à la commande :

```
workingMemory.insert(o);
```

o correspondant à n'importe quel objet devant être ajouté à la working memory.

Une fois tous ces objets ajoutés à la working memory, les règles de pré-analyse sont lancées.

6.2. Règles de pré-analyse

Les règles de pré-analyse utilisent les croyances intégrées à la working memory et ajoutent de nouvelles croyances à celle-ci. Les règles sont écrites en Drools dans des fichiers textes ; fichiers, comme nous l'avons dit, que nous avons chargés à partir de la base postgresql à l'initialisation de l'agent. L'exemple de la règle de pré-analyse de la page 155 sera traduite avec Drools sous la forme :

```
rule "compterNombreUtilisationAide"
when
      $motsTrouves : ArrayList() from collect(Interaction(type ==
"apprenant", action == "clic", objet == "bouton", typeObjet ==
"aide"))
then
      insert(new NombreFoisUtilisationAide($motsTrouves.size()));
end
```

Cette règle s'appelle compterNombreUtilisationAide. Elle utilise les objets *Interaction* dans sa prémisse. En fait, elle construit la liste de toutes les interactions, ou plutôt de tous les objets *Interaction*, répondant aux critères souhaités : l'interaction est de type apprenant, l'action est un clic sur un objet bouton de type aide. Cette liste est stockée dans la variable Drools $motsTrouves. Une fois cette liste construite la règle va exécuter la partie then, en l'occurence elle va ajouter un nouvel objet à la working memory contenant le nombre de fois où l'apprenant à utilisé l'aide.

Une fois toutes les règles de pré-analyse exécutées, les règles d'analyse sont lancées.

6.3. Règles d'analyse

Les règles d'analyse utilisent également les croyances intégrées à la working memory. A la différence des règles de pré-analyse qui ajoutent des nouvelles croyances à la working memory, les règles d'analyse vont remplir un vecteur *hypotheses* qui contiendra toutes les hypothèses (objets *Hypothese*) émises au cours de l'analyse. Ce vecteur est défini lors du chargement de la working memory dans la méthode setup() de la façon suivante :

```
hypotheses = new ArrayList<Hypothese>();
workingMemory.setGlobal("hypotheses", hypotheses);
```

Nous définissons ici le vecteur d'objets *Hypothese*. Puis, nous le déclarons auprès de la working memory pour que celle-ci puisse l'utiliser. Une hypothèse est représentée par l'objet *Hypothese* :

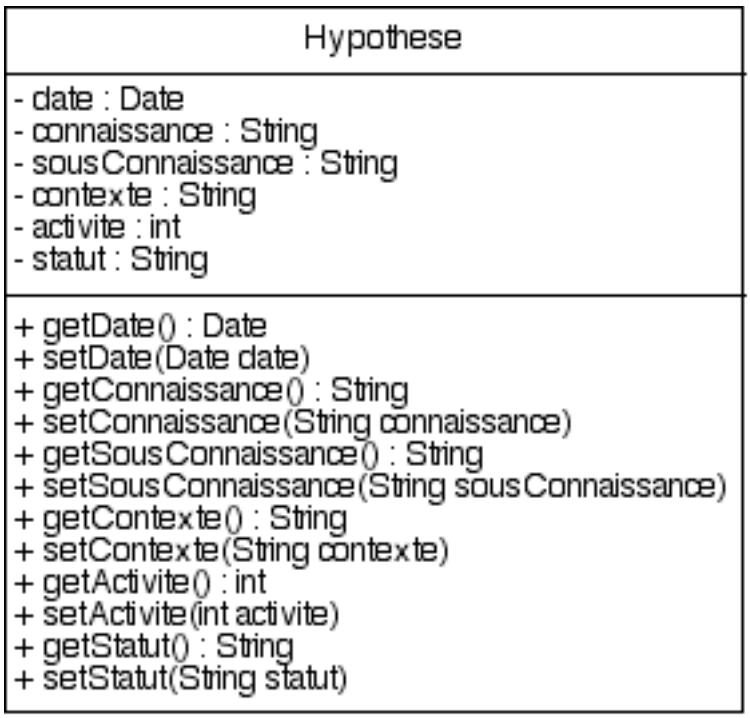

Figure 5.5 : Objet hypothèse utilisé par AA

Chaque règle d'analyse va donc ajouter un nouvel objet *Hypothese* au vecteur *hypotheses*. Celui-ci doit également être déclaré dans le fichier des règles d'inférences Drools pour pouvoir être utilisé. La variable globale *identifiant_sd* est déclarée en même temps que le vecteur *hypotheses*. Celle-ci permet de rendre accessible aux règles l'identifiant de la situation didactique sur laquelle elles travaillent. L'exemple de la règle d'analyse « regleAnalyse1 » de la page 158 sera traduite avec Drools sous la forme :

```
global java.util.List hypotheses;
global int identifiant_sd;

rule "regleAnalyse1"
when
      NombreMotsTrouves($nbMots : nombre)
      $nbmots >= 1
      NombreFoisUtilisationAide($nbFois : nombre)
      $nbFois == 0
then
      hypotheses.add(new Hypothese(new Date(), "stratégie indices
contextuels", "", "", identifiant_sd,"supposee"))
end
```

Cette règle utilise dans sa prémisse des objets ajoutés par les règles de pré-analyse : NombreMotsTrouves et NombreFoisUtilisationAide. Nous n'avons pas représenté le constructeur de l'objet *Hypothese* sur la Figure 5.5, mais celui prend en compte tous les attributs de l'objet. Il n'existe dans la working memory qu'une seule instance des objets NombreMotsTrouves et NombreFoisUtilisationAide, ce qui fait qu'aucun conflit n'est possible. La prémisse de cette règle stocke dans un premier temps le nombre de mots trouvés dans la variable Drools $nbMots, puis si ce nombre est supérieur ou égal à 1 alors on ira rechercher le nombre de fois où l'aide a été utilisée dans la variable Drools $nbFois et puis si celui-ci est égal à 0 alors on ajoutera l'hypothèse. Ce qui signifie donc que si l'apprenant a trouvé au moins un mot sans aide alors on exécute la conclusion de la règle. Celle-ci ajoute un nouvel objet *Hypothese* au vecteur hypotheses. Certains attributs ne sont pas remplis car ceux-ci n'ont pas de valeur pour la connaissance concernée.

Une fois toutes les règles d'analyse exécutées, le vecteur *hypotheses* sera ensuite parcouru pour envoyer chaque objet *Hypothese* à l'agent de connaissance en ayant besoin. Là encore, à chaque fois, l'agent consultera le DF pour savoir quel agent gère quelle connaissance. L'interrogation se fera comme pour les agents de communication sauf que le type de service recherché sera « connaissance » et que le nom du service dépendra de l'attribut *connaissance* de le l'objet *Hypothese* considéré.

L'objet *Hypothese* sera ensuite intégré au message ACL via la méthode *setContentObject()* que nous avons vue pour les agents d'analyse. L'objet sera donc directement transmis aux agents de connaissance sous forme binaire. Ensuite, le message sera bien entendu envoyé à l'agent de connaissance gérant le type de connaissance de l'objet *Hypothese*.

7. Les agents de connaissance

Un agent de connaissance attend de recevoir soit une hypothèse de la part d'un agent d'analyse soit une requête de la part d'un agent de communication. La réception de ces différents messages est traitée par le méta-comportement. A la réception d'une nouvelle hypothèse, celle-ci est convertie en croyance de l'agent puis ajoutée à sa working memory. Ensuite, l'agent lance ses différentes règles de synthèse pour, éventuellement, découvrir de nouvelles croyances à partir de la synthèse de toutes celles dont il dispose. Nous rappelons également que l'agent a chargé ses anciennes croyances à son initialisation en allant les récupérer dans la base de données postgresql via Hibernate.

Comme nous l'avons vu au chapitre précédent, il existe plusieurs niveaux de détail possibles. Au lieu de créer autant d'objets *Croyance* que de niveaux de détail, nous avons préféré ne représenter qu'un seul objet *Croyance* et lui adjoindre un attribut niveau de détail. Cela permettra d'utiliser les règles Drools de manière beaucoup plus simple. De plus, l'agent, pour répondre à une éventuelle requête, n'aura qu'à regarder le niveau de détail correspondant à cette requête.

Une croyance au sein d'un agent de connaissance est donc représentée par le diagramme suivant :

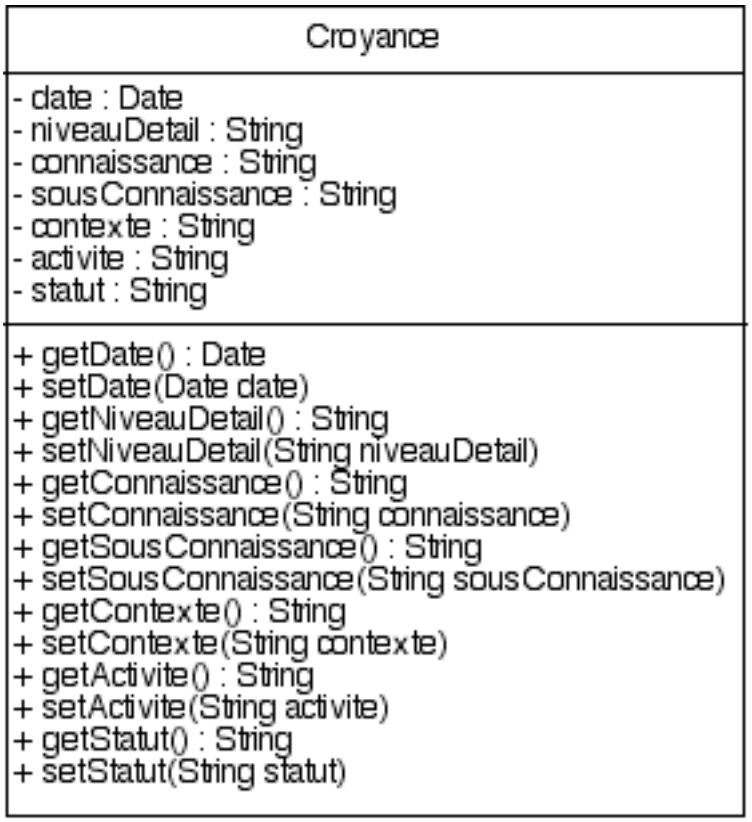

Figure 5.6 : Objet Croyance intégré à la working memory

7.1. Règles de synthèse

Après que l'agent ait converti les hypothèses qu'il a reçues et qu'il les ait ajoutées à sa working memory, il lance ses différentes règles de synthèse pour découvrir éventuellement de nouvelles croyances. Les règles de synthèse sont représentées elles aussi par des règles Drools. L'exemple de la règle de synthèse « regleSyntheseConnaissance1 » de la page 170 sera traduite en Drools de la façon suivante :

```
rule "regleSyntheseConnaissance1"
when
      $ssConAcquises : ArrayList(size == 3) collect
(Croyance(niveauDetail == "sous-connaissance", statut ==
"acquis"))
```

```
then
      insert(new Croyance("acquis"));
end
```

Cette règle construira dans sa prémisse la liste de toutes les croyances de niveau de détail « sous-connaissance » étant acquise, c'est-à-dire de statut égal à acquis. Puis, si toutes les sous-connaissances sont acquises (size == 3), alors on ajoutera à la working memory un nouvel objet *Croyance* indiquant ainsi que la connaissance gérée par l'agent est acquise. Nous ne l'avons pas noté sur la Figure 5.6 pour ne pas alourdir le diagramme mais il existe un constructeur pour l'objet *Croyance* pour chaque niveau de détail.

Chaque règle de synthèse ajoutera de la même manière que la règle précédente une nouvelle croyance à la working memory, si bien sûr, elle est applicable.

8. Interfaces graphiques du prototype

Pour conclure cette rapide présentation du prototype que nous avons développé, nous terminerons en présentant les deux principales interfaces graphiques que l'on rencontre au sein de PAGMA : la première permet de visualiser le fonctionnement de l'agent pendant la simulation de la disponibilité d'un nouveau compte-rendu et la deuxième permet à l'utilisateur de visualiser le modèle de l'apprenant disponible.

Ces différentes interfaces graphiques ont été réalisées, toujours dans le but de respecter la philosophie d'intégration à la plateforme Java, en utilisant la bibliothèque graphique fournie par Java : Swing. En effet, utiliser une autre bibliothèque comme SWT aurait nécessité l'installation de cette bibliothèque sur chaque ordinateur.

La partie de l'écran « Console Jade » située en bas de chaque interface permet de visualiser tous les événements qui se produisent au sein de la plateforme Jade pour voir si, par exemple, il n'y a pas de problème de lancement des différents agents.

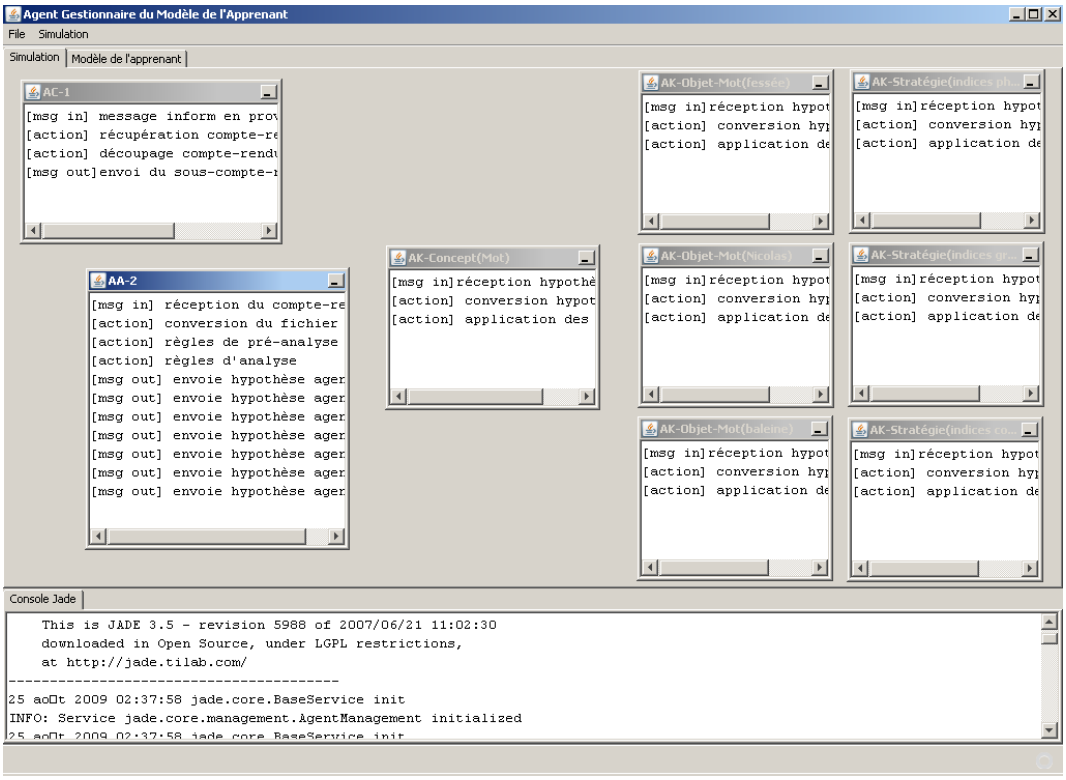

Figure 5.7 : Ecran permettant de visualiser la simulation

Cet écran affiche les différents agents lancés au cours de la simulation de la disponibilité d'un compte-rendu. Ceux-ci apparaissent au fur et à mesure de leur utilisation. Sur l'écran ci-dessus nous voyons que les agents présentés au chapitre précédent pour la situation didactique « identification de mots en contexte » sont tous lancés lors de la simulation de la disponibilité d'un compte-rendu ne comportant que cette situation didactique.

L'écran de visualisation d'un agent peut afficher trois type de messages :

1. [msg in] indiquant que l'agent a reçu un message,
2. [msg out] indiquant que l'agent a envoyé un message,
3. [action] indiquant que l'agent a effectué une action, par exemple émettre une hypothèse.

Nous avons volontairement limité le nombre de messages s'affichant sur chaque écran d'un agent de la simulation pour ne pas rendre illisible la lecture. Néanmoins, ce niveau de détail peut bien entendu se modifier très facilement pour, par exemple, voir quelles règles de pré-analyse, d'analyse ou encore de

synthèse ont été appliquées.

Le deuxième écran d'interface que l'on trouve au sein de PAGMA concerne la visualisation du modèle de l'apprenant.

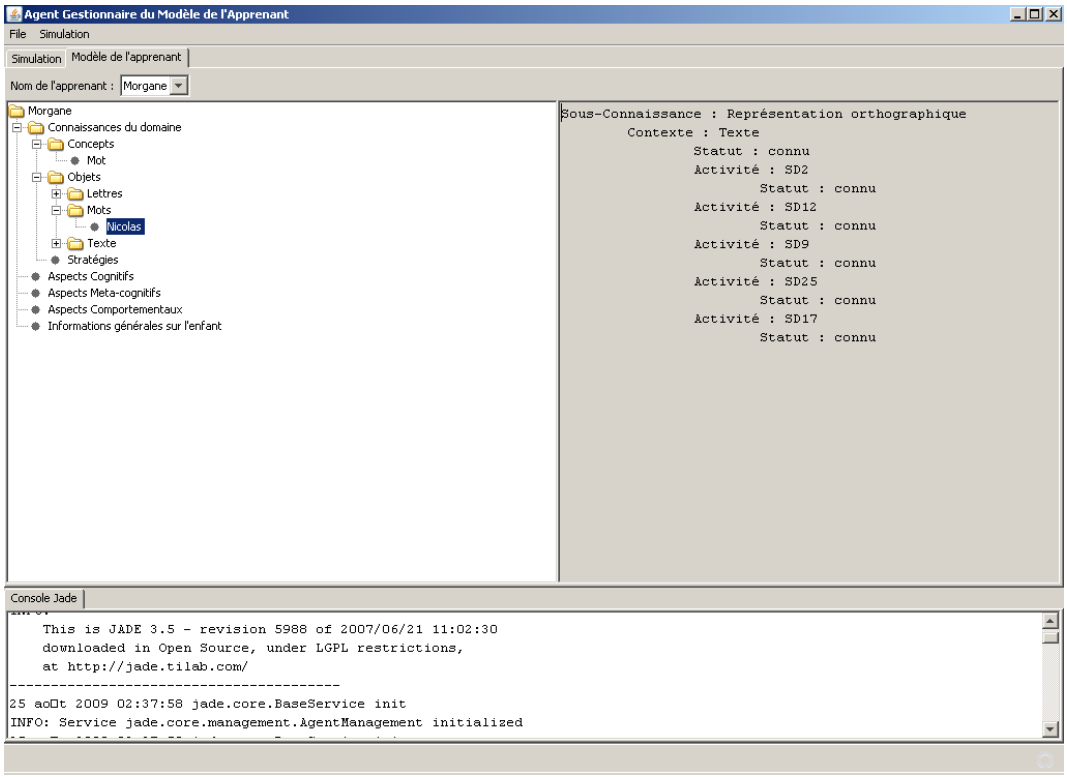

Figure 5.8 : Ecran permettant de visualiser le modèle de l'apprenant

Cet écran est divisé en deux parties :

- la partie droite qui permet de naviguer dans les différentes connaissances du modèle de l'apprenant au sein des cinq catégories évoquées au chapitre 2,
- la partie gauche qui permet de visualiser le détail de la connaissance sélectionnée.

Sur l'écran ci-dessus, nous voyons le détail de la connaissance du domaine : Objet->Mot->Nicolas. Nous voyons par exemple, que la sous-connaissance « représentation orthographique » est connue dans le contexte « Texte ». La représentation du détail d'une connaissance est organisée de manière arborescente correspondant aux différents niveaux de détail évoqués aux chapitre 2 et 4.

9. Conclusion

PAGMA a été développé dans le but de tester les différents outils envisageables pour les développements futurs d'AGMA. Un peu à la manière du programme Apollo, PAGMA est donc le premier prototype d'AGMA ouvrant la voie à ses successeurs. Celui-ci a été réalisé en faisant avec l'existant de l'environnement multi-agents AMICAL, à la fois en terme d'agents réalisés que de situations didactiques disponibles.

Notre prototype utilise donc pour la plupart de ses composants, des éléments s'intégrant parfaitement à la plateforme Java. Ceci, bien entendu, permet de faciliter considérablement les développements. PAGMA regroupe un ensemble important de technologies : un environnement multi-agents Jade où certains agents disposent d'une capacité de raisonnement grâce à Drools ; d'une base de données XML permettant de stocker les comptes-rendus pour de futures utilisations ou bien pour permettre de les visualiser par des outils extérieurs développés par exemple grâce à des interfaces Web ; d'un stockage de données via le couple Hibernate-Postgresql.

Nous avons montré comment ces différents outils pouvaient cohabiter et donc pouvaient constituer une base saine et solide pour un développement plus poussé d'AGMA. Les bases des développements ont été réalisées. Il resterait maintenant à compléter en intégrant d'autres agents d'analyse et de connaissance dans ce prototype.

Nous allons donc maintenant conclure ce document en faisant le bilan de ce travail et en présentant différentes perspectives possibles quant à l'évolution de celui-ci.

CONCLUSIONS ET PERSPECTIVES

1. Bilan

L'objectif de notre travail était de proposer une architecture pour l'agent gestionnaire du modèle de l'apprenant dans l'environnement AMICAL et c'est ce que nous avons réalisé au cours de ce travail. Nous avons proposé une architecture modulaire d'AGMA : le module communication, le module analyse et le module modèle. Essentiellement pour des raisons de rapidité d'exécution, nous avons choisi de représenter chaque module lui-même comme un système multi-agents. Nous avons vu que le module communication était peuplé d'agents de communication, que le module analyse était peuplé d'agents d'analyse et enfin que le module modèle était, quant à lui, peuplé d'agents de connaissance. Les agents de communication ont en charge la communication entre le reste de l'environnement AMICAL et les modules d'AGMA. Les agents d'analyse ont pour travail d'émettre toutes les hypothèses possibles quant à l'état de savoir de l'apprenant sur la situation didactique dont ils ont la charge, à partir du compte-rendu envoyé par un agent de communication. Enfin, les agents de connaissance ont pour charge de maintenir à jour une connaissance du modèle de l'apprenant. Comme nous l'avons vu au cours de ce document, cette architecture est également facilement évolutive en ajoutant de nouveaux agents d'analyse correspondant aux situations didactiques développées et en ajoutant les agents de connaissances permettant d'intégrer de nouvelles informations au modèle.

D'un point de vue du modèle de l'apprenant, nous avons adopté une nouvelle approche qui n'avait pas été explorée dans ce sens : représenter le modèle de l'apprenant lui-même comme un système multi-agents. Ainsi, le modèle de l'apprenant est composé d'agents de connaissance où chacun a en charge une petite partie du modèle. Le modèle est donc passer d'une perspective statique avec les approches existantes à une perspective dynamique où le modèle de l'apprenant se

maintient tout seul à jour.

Malheureusement, la complexité du domaine abordé et l'ampleur de la tâche nous a conduit à nous limiter à quelques connaissances du modèle de l'apprenant : les connaissances du domaine. De plus, ce travail a nécessité de nombreuses réflexions sur de nombreux domaines comme l'apprentissage de la lecture, la modélisation de l'apprenant ou bien encore les agents et ceci n'a pas toujours été évident.

Comme nous l'avons dit à plusieurs reprises, le fait que de nombreux travaux soient encore en cours aussi bien en ce qui concerne la conception et la réalisation des autres agents de l'environnement qu'en ce qui concerne la réalisation d'autres situations didactiques, nous a considérablement limité dans le développement d'un prototype. Nous avons dû essentiellement nous concentrer sur le choix des différents outils ainsi que sur la pertinence et la souplesse d'utilisation des outils retenus. De fait, il a été tout simplement impossible d'envisager de faire une évaluation de notre prototype autre que de juger les différents outils retenus pour le réaliser. Néanmoins, le prototype que nous avons réalisé a permis de poser les bases pour de futurs développements.

2. Perspectives

Nous pouvons envisager plusieurs perspectives possibles pour étendre ce travail.

2.1. Prise en compte d'autres informations dans notre modèle de l'apprenant

Comme nous l'avons expliqué, nous n'avons considéré que les connaissances du domaine avec leurs trois types de connaissances. Dans la suite de nos travaux, il faudrait prendre en compte de manière plus précise les autres entrées du modèle de l'apprenant. Pour cela, il conviendra de travailler encore avec les experts en apprentissage de la lecture afin de détailler davantage les situations didactiques existantes sur les autres entrées de notre modèle : aspects cognitifs, méta-cognitifs et comportementaux.

D'un point de vue informatique, cela reviendra à ajouter de nombreuses autres règles d'analyse. Celles-ci devront bien entendu être intégrées aux agents d'analyse déjà existant pour que ceux-ci puissent émettre des hypothèses sur les autres entrées du modèle. Cela reviendra aussi à concevoir les différents agents de connaissance ayant en charge ces nouvelles connaissances.

2.2. Développement d'autres situations didactiques

Toutes les situations didactiques de l'environnement AMICAL n'ont pas été réalisées. Ceci en raison du nombre de travaux encore en cours au sein de l'équipe et de la complexité de la tâche. Les situations didactiques réalisées concernent le tout début de l'apprentissage de la lecture. Il conviendrait donc de poursuivre le travail de réalisation des autres situations didactiques du système.

D'un point de vue informatique, cela reviendra à concevoir de nouveaux agents d'analyse capables d'analyser ces situations didactiques. Pour cela, il faudra, toujours en collaboration avec les experts en apprentissage de la lecture, recueillir les différentes règles d'analyse permettant d'émettre des hypothèses sur chaque situation didactique et les intégrer aux agents.

2.3. Niveaux de synthèse entre connaissances du modèle de l'apprenant

Nous avons vu qu'il existait différents niveaux de synthèse pour une connaissance du modèle de l'apprenant et nous nous sommes limités à ceux-là. Néanmoins, il conviendrait également de réfléchir à l'existence probable de niveaux de synthèse entre les connaissances du modèle. Par exemple, entre les objets et les concepts.

D'un point de vue des agents de connaissance, cela reviendrait à s'interroger sur les relations que ces agents devraient avoir entre eux.

2.4. Travail avec plusieurs enfants

Le fait de travailler avec plusieurs enfants pourrait être utile et apporter de

nouvelles connaissances. En effet, les modèles de l'apprenant étant stockés dans une base de données, on pourrait imaginer exploiter cette base pour découvrir par exemple de nouvelles règles d'analyse. A cet effet, nous pourrions envisager d'utiliser des techniques de datamining ou d'apprentissage automatique.

2.5. Une nouvelle architecture pour AGMA ?

Au cours de nos réflexions, il nous est apparu qu'une autre architecture pourrait être envisageable pour AGMA. Celle-ci pourrait être une architecture composée uniquement d'agents de connaissance. Néanmoins, ces agents devraient intégrer une intelligence plus importante que ceux de l'architecture proposée. Il faudrait intégrer au sein de chaque agent des capacités d'analyse des comptes-rendus en ce qui concerne la connaissance gérée. Ceci serait envisageable à partir du moment où l'on disposerait de l'ensemble des situations didactiques de l'environnement AMICAL à partir desquelles on pourrait déterminer et généraliser des principes d'analyse. Ces principes pourraient être ensuite utilisés pour développer la capacité d'analyse des comptes-rendus des différents agents de connaissance.

BIBLIOGRAPHIE

(Akhras et Self, 1996)
Akhras F., Self J., From the process of instruction to the process of learning: Constructivist Implications for the design of intelligent learning environments, Proceedings of EuroAIED, Lisbon, Portugal, pp 9-15, 1996.

(Anderson, 1983)
Anderson J.R., The Architecture of Cognition, Harvard University Press, Cambridge, Mass, 1983.

(Aussenac, 1989)
Aussenac N., Conception d'une méthodologie et d'un outil d'acquisition de connaissances expertes, Thèse de doctorat, Université Paul Sabatier, Toulouse, 1989.

(Azmatally, 1997)
Azmatally S., Approche multi-agents : contribution au développement d'un environnement d'aide à l'apprentissage de la lecture, Mémoire, Laboratoire de Recherche sur le Langage, Université Blaise Pascal, Clermont-Ferrand II, 1997.

(Baker et Lund, 1997)
Baker M.J., Lund K., Promoting reflective interactions in a computer-supported collaborative learning environment, Journal of Computer Assisted Learning, No. 13, pp 175-193, 1997.

(Barr et Feigenbaum, 1981)
Barr A., Feigenbaum E.A., The handbook of artificial intelligence, New-York: Addison-Wesley, Vol. 1, 1981.

(Bauer et al., 1993)
Bauer M., Biundo S., Dengler D., Koehler J., Paul G., PHI, a Logic-Based Tool

for Intelligent Help Systems, Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambéry, France, pp 460-466, 1993.

(Becker et Naïm, 1999)
Becker A., Naïm P., Les réseaux bayésiens, modèles graphiques de connaissances, Eyrolles, Paris, 1999.

(Bellifemine et al., 2007)
Bellifemine F.L., Caire G., Greenwood D., Developing Multi-Agent Systems with JADE, Wiley, 2007.

(Bigus et Bigus, 2001)
Bigus J.P., Bigus Jennifer, Constructing Intelligent Agents Using Java, Wiley, 2001.

(Billard et al., 2007)
Billard C. et al., Résultats préliminaires d'une étude épidémiologique transversale des apprentissages en lecture, orthographe et calcul au CE1, INSERM, 2007.

(Blanchard al., 1987)
Blanchard J.S., Mason G.E., Daniel D., Computer applications in reading, International Reading Association (Newark, Del), 1987.

(Bratko, 2001)
Bratko I., Prolog Programming for Artificial Intelligence, Pearson, Addison Wesley, third edition, 2001.

(Brecht et al., 1989)
Brecht B.J., MacCalla G.I., Greer J.E., Jones M., Planning the Content of Instruction, proceedings of Artificial Intelligence and Education, pp 32-41, 1989.

(Brown et Burton, 1978)
Brown J., Burton R., Diagnostic models for procedural bugs in basic mathematical skills, Cognitive Science 2, pp 155-192, 1978.

(Bruillard, 1997)
Bruillard E., Les machines à enseigner, Editions Hermès, 1997.

(Burton et Brown, 1982)
Burton R.R., Brown J.S., An investigation of computer coaching for informal learning activities, dans Sleeman D., Brown J.S. (dir.), Intelligent Tutoring Systems, Academic Press, Londres, pp 79-98, 1982.

(Chaïb-draa et al., 2001)
Chaïb-draa B., Jarras I., Moulin B., Agents et systèmes multi-agents, Principes et architecture des SMA, Edité par J-P Briot et Y. Demazeau, 2001.

(Chaïb-draa, 2000)
Chaïb-draa B., Agents et systèmes multi-agents, cours gradué du département d'informatique, Université Laval, Québec, Canada, 2000.

(Chambreuil et al., 2000)
Chambreuil M., Bussapapach P., Fynn J., Didactic situations as mutlifacetted theoretical objects, 5th International Conference on Intelligent Tutoring Systems, Montréal, Canada, 2000.

(Cherkaoui, 1996)
Cherkaoui C., La planification didactique : une contribution au développement d'un planificateur didactique dynamique sous le modèle de tableau noir, Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand II, 1996.

(Cialdea, 1992)
Cialdea M., Meta-Reasoning and Student Modelling, New Directions for Intelligent Tutoring Systems, Springer-Verlag, Berlin, pp 71-90, 1992.

(Cleder, 2002)
Cleder C., Planification didactique et construction de l'objectif d'une session de travail individualisée : modélisation des connaissances et du raisonnement mis en jeu, Thèse de doctorat, Université Blaise Pascal. Clermont-Ferrand II, 2002.

(Conati et al., 2002)
Conati C., Gertner A., VanLehn K., Using Bayesian networks to manage uncertainty in student modelling, Journal of User Modeling and User-Adapted Interaction, Vol. 12, pp 371-417, 2002.

(Danna, 1997)
Danna F., Modélisation de l'apprenant dans un logiciel d'EIAO - Application à un tutoriel intelligent dédié aux composés anglais, Thèse de doctorat, Université de Rennes I, 1997.

(DARPA, 1993)
by The DARPA Knowledge Sharing Initiative External Interfaces Working Group, with major contributions from Tim Finin (co-chair), University of Maryland ; Jay Weber (co-chair),Enterprise Integration Technologies ; Gio Wiederhold (former co-chair),Stanford University ; Michael Genesereth,Stanford University ; Richard Fritzson,Donald McKay,Paramax Systems ; James McGuire,Richard Pelavin,Lockheed AI Center ; Stuart Shapiro,SUNY Buffalo ; Chris Beck,University of Toronto., Specification of the KQML Agent-Communication Language -- plus example agent policies and architectures, http://www.cs.umbc.edu/kqml/papers/kqmlspec.pdf, 1993.

(dbxml)
Site web dbXML, http://www.oracle.com/technology/products/berkeley-db/xml/index.html.

(Demazeau et Müller, 1991)
Demazeau Y., Müller J.P., From Reactive to Intentional Agents, Decentralized Artificial Intelligence, Editors, North Holland, Vol. 2, 1991.

(Despres et Leroux, 2003)
Despres C., Leroux P., Tutorat synchrone en formation à distance, Actes de conférence EIAH 2003, dans Desmoulins C., Marquet P., Bouhineau D. (dir), Strasbourg, France, pp 139-150, 2003.

(dom4j)
Site web dom4j, http://www.dom4j.org/.

(drools)
Site web Drools, http://jboss.org/drools/.

(Ecaille et Magnan, 2002)

Ecaille J., Magnan A., L'apprentissage de la lecture, Armand Colin, Paris, 2002.

(Erceau et Ferber, 1991)
Erceau J., Ferber J., L'intelligence artificielle distribuée, Recherche 233, Vol. 22, 1991.

(Ermine, 2000)
Ermine J.L., Capitaliser et partager les connaissances avec la méthode MKSM, In Traité IC2 Information, Communication et Commande, Volume Capitalisation des Connaissances, Hermès, 2000.

(Fayol et al., 1992)
Fayol M., Gombert J.E., Lecocq P., Sprenger-Charolles L., Zagar D., M. Fayol, Jean Emile Gombert, P. Lecocq, L. Sprenger-Charolles, D. Zagar, PUF, Paris, 1992.

(Ferber, 1995)
Ferber J., Les systèmes multi-agents: vers une intelligence collective, Inter-Edition, Paris, 1995.

(FIPA, 2002)
FIPA, FIPA ACL Message Structure Specification, http://www.fipa.org/specs/fipa00061/index.html, 2002.

(FIPA, 2002b)
FIPA, FIPA Contract Net Interaction Protocol Specification, http://www.fipa.org/specs/fipa00029/, 2002.

(FIPA, 2002c)
FIPA, FIPA Abstract Architecture Specification, http://www.fipa.org/specs/fipa00001/index.html, 2002.

(Fragne et al., 2005)
Fragne D., Faqir R., Sayouri N., A Learner model in a French reading tutor, proceedings of ICCE 2005 (13th International Conference on Computers in Education), 2005.

(Fragne et al., 2007)
Fragne D., Sayouri N., Mazladi H., The Architecture of a Learner Model Manager Agent, proceedings of ICAI 2007 (International Conference on Artificial Intelligence), 2007.

(Fragne, 2007)
Fragne D., A learner model manager agent in an agent-based reading tutor, proceedings of ISA 2007 (IADIS International Conference Intelligent Systems and Agents), 2007.

(Gaguet, 2001)
Gaguet L., Attitudes mentales et planification en intelligence artificielle : modélisation d'un agent rationnel dans un environnement multi-agents, Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand II, 2001.

(Goigoux et Cèbe, 2006)
Goigoux R., Cèbe S., Apprendre à lire à l'école , Tout ce qu'il faut savoir, Retz Eds, , 2006.

(Goldstein, 1982)
Goldstein I.P., The genetic graph : a representation for the evaluation of procedural knoledge, Intelligent Tutoring Systems, Academic Press, New York, pp 51-78, 1982.

(Gombert et al., 2000)
Gombert J.E., Colé P., Fayol M., Goigoux R., Mousty P., Valdois S., Enseigner la lecture au cycle 2, Nathan Université, 2000.

(Hartley et Sleeman, 1973)
Hartley J.R., Sleeman D.H., Towards More Intelligent Teaching Systems, In International Journal of Man-Machine Studies, Vol. 5, No. 2, pp 245-236, 1973.

(Haton et al., 1991)
Haton J.P., Bouzid N., Charpillet F., Haton M., Léasri H., Marquis P., Mondot T., Napoli A., Le raisonnement en intelligence artificielle. Modèles, techniques et architectures pour les systèmes à bases de connaissances, InterEditions, 1991.

(hibernate)
Site web Hibernate, https://www.hibernate.org/.

(Holt et al., 1993)
Holt P., Dubs, Jones M., et Greer J., The State of Student Modelling, Student Modelling: Key to Individualized Instruction, J. Greer & G. McCalla (Eds.), NATO Conference on Student Modelling, 1993.

(Iglesias et al., 1998)
Iglesias C. A., Garrijo M., Gonzalez J., and Velasco J. R., Analysis and Design of multiagent systems using MAS-CommonKADS, Proceedings of the Fourth InternationalWorkshop on Agent Theories, Architectures and Languages (ATAL), Springer-Verlag, LNCS 1365, pp 313-328, 1998.

(Ikeda et al., 1989)
Ikeda M., Mizoguchi R., Kakusho O., Student Model Description Language SMDL and Student Model Inference System SMIS, IEICE, pp 112-120, 1989.

(jade)
Site web Jade, http://jade.tilab.com/.

(java)
Site web Java, http://java.sun.com/.

(jdbc)
Site web JDBC, http://java.sun.com/javase/technologies/database/.

(Just et Carpenter, 1987)
Just M.A., Carpenter P.A., The psychology of reading and language comprehension, Just, M.A. & Carpenter, P.A. (Eds.), Newton, MA: Allyn and Bacon, 1987.

(Kass, 1987)
Kass, R, The Role of User Modelling in Intelligent Tutoring System, Moore School, Université de Pennsylvanie, 1987.

(Lotin et al., 1999)
Lotin P., Steck L., Chambreuil M., Outils fonctionnels modulaires pour l'étude et le développement d'environnements tutoriels, Sciences et Techniques Éducatives, Vol. 6, 1999.

(Mahmoud, 1997)
Mahmoud A., Architecture agent-objet pour les interfaces apprenant-machine : Vers une méthodologie de conception et un système d'aide à la construction, Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand II, 1997.

(Mislevy et Gitomer, 1996)
Mislevy R., Gitomer D., The role of probability-based inference in an intelligent tutoring system, User Modeling and User-Adapted Interaction, Vol. 5, pp 253-282, 1966.

(Mostow et al., 1993)
Mostow J., Hauptmann A.G., Chase L.L., Roth S., Towards a Reading Coach that Listens: Automated Detection of Oral Reading Errors, proceedings of the Eleventh National Conference on Artificial Intelligence, pp 392-397, 1993.

(Müller, 1999)
Müller J.P., The right agent architecture to do the right thing, In Müller J.P., Singh M.P., Rao A.S. (Ed.), Intelligent Agent V - Agent Theories, Architectures and Languages, Springer, pp. 211-226, 1999.

(Mullis et al., 2007)
Ina V.S. Mullis, Michael O. Martin, Ann M. Kennedy, Pierre Foy, IEA's Progress in International Reading Literacy Study in Primary School in 40 Countries, TIMSS & PIRLS International Study Center, Boston College, 2007.

(Nicaud et Vivet, 1988)
Nicaud J.F., Vivet M., Les tuteurs intelligents, réalisations et tendances de recherche, Revue Technique et Science Informatiques, 1988.

(Nwana, 1996)
Nwana H.S., Software Agents: An Overview, Knowledge Engineering Review,

Cambridge University Press, Vol. 11, No. 3, pp 1-40, 1996.

(O'Shea et Self, 1983)
O'Shea T., Self J., Learning and teaching with computers, Artificial Intelligence in Education, Harvester Press, 1983.

(Ohlsson et Langley, 1987)
Ohlsson S., Langley P., Identifying solution paths in cognitive diagnosis, Learning Issues for Intelligent Tutoring Systems, Springer-Verlag, Berlin, pp 42-62, 1987.

(Ohlsson, 1992)
Ohlsson S., Constraint-Based Student Modelling, Artificial Intelligence in Education, Vol. 3, pp 429-447, 1992.

(Olson et Wise, 1992)
Olson R.K., Wise B.W., Reading on the computer with orthographic and speech feedback. An overview of the Colorado remediation project, Reading and Writing : An Interdisciplinary Journal, 4, pp 107-114, 1992.

(ONL, 2005)
ONL, Rapport de l'ONL et de l'Inspection Générale sur l'apprentissage de la lecture à l'école primaire, http://onl.inrp.fr/ONL/publications/publi2005/onl_2005.pdf, 2005.

(Paiva et John, 1994)
Paiva A., John A., A Learner Model Reason Maintenance System, European Conference on Artificial Intelligence, pp 193-196, 1994.

(Paiva, 1996)
Paiva A., Communicating with Learner Modelling Agents, ITS 96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITS, 1996.

(Pearl, 1988)
Pearl J., Probabilistic Reasoning in Intelligent System, Morgan Kauffman, 1988.

(postgresql)
Site web Postgresql, http://www.postgresql.org/

(Py et Hibou, 2006)
Py D., Hibou M., Représentation des connaissances de l'apprenant, Environnements Informatiques pour l'Apprentissage Humain, Grandbastien M., Labat J.-M. (ed.), Traité IC2 Information Commande Communication, Edité par Hermès, 2006.

(Py, 1998)
Py D., Quelques méthodes d'intelligence artificielle pour la modélisation de l'élève, Sciences et Techniques Educatives, Vol. 5, pp 123-140, 1998.

(Py, 2001)
Py D., Environnements interactifs d'apprentissage et démonstration en géométrie, Thèse de doctorat, habilitation à diriger des recherches, Université de Rennes I, 2001.

(Quanquin, 2000)
Quanquin V., Le choix du texte : problématique et application dans le cadre d'un environnement informatique d'aide à l'enseignement et l'apprentissage de la lecture en cours préparatoire, Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand II, 2000.

(Ragnemalm, 1996)
Ragnemalm E.L., Student diagnosis in practice; Bridging a gap, User Modelling and User Adapted Interaction, Vol. 5, No. 2, pp 93-116, 1996.

(Rao et Georgeff, 1995)
Rao A.S., Georgeff M.P., BDI-agents: From Theory to Practice, Proceedings of the First International Conference on Multiagent Systems, 1995.

(Rumelhart, 1985)
Rumelhart D., Toward an interactive model of reading, Theoretical Models and Processes of Reading, 3rd edition, Newark, Delaware: International Reading Association., In H. Singer, & R. B. Ruddell (Eds.), pp 722-750, 1985.

(Russell et Norvig, 2003)
Russell S. J., Norvig P., Artificial Intelligence: a Modern Approach, Prentice Hall,

2nd edition, 2003.

(Schreiber et al., 1999)
Schreiber G., Akkermans H., Anjewierden A., De Hoog R., Shadbolt N., Van De Velde, Wielinga B., Knowledge Engineering and Management - The CommonKADS Methodology, MIT Press, 1999.

(Searl, 1969)
Searl J., Speech Acts, Cambridge University Press, 1969.

(Seidenberg et McClelland, 1989)
Seidenberg M.S., McClelland J.L., A Distributed, Developmental Model of Word Recognition and Naming, Psychological Review, 1989.

(Self, 1987)
Self J., Student Models: What Use Are They?, Actes de IFIP/TC3, pp 73-85, 1987.

(Self, 1988)
Self J., Bypassing the intractable problem of student modelling, Proceedings of International Conference on Intelligent Tutoring Systems, ITS'1988, Gauthier, G. Frasson, C. (Eds), Springer-Verlag, pp 18-24, 1988.

(Self, 1992)
Self J., Cognitive Diagnosis for Tutoring Systems, 10th European Conference on Artificial Intelligence, ECAI'92, pp 699-703, 1992.

(Self, 1994)
Self J., The role of student models in learning environments, Transactions of the Institute of Electronics, Information and Communication Engineers, 1994.

(Sleeman et Brown, 1982)
Sleeman D., Brown J.S., Introduction: Intelligent Tutoring Systems, In Intelligent Tutoring Systems, Orlando, Florida: Academic Press, Inc, pp 1-10, 1982.

(Sprenger-Charolles, 1993)
Sprenger-Charolles L., Procédures de traitement de l'information écrite utilisées par des lecteurs/scripteurs francophones en début d'apprentissage : Examen à

partir de l'analyse d'un corpus d'erreurs, Etudes de Linguistique Appliquée, pp 70-83, 1993.

(Sterling et Shapiro, 1994)
Sterling L., Shapiro E., The Art of Prolog, The MIT Press, second edition, 1994.

(Swartz, 1992)
Swartz M.L., Introduction, Actes des journées Intelligent Tutoring Systems for Foreign Language Learning - The Bridge to International Communication, Springer-Verlag, pp 1 - 6, 1992.

(VanLehn, 1988)
VanLehn K., Towards a theory of impasse-driven learning, Learning Issues for Intelligent Tutoring Systems, H. Mandl & A. Lesgold (Eds.), New York, NY: Springer, pp 19-41, 1988.

(Vicente et Pain, 2002)
Vicente A., Pain H., Informing the Detection of the Student's Motivational State: an Empirical Study, proceedings of Intelligent Tutoring Systems 2002, LNCS 2363, Springer-Verlag Berlin, pp 933-943, 2002.

(Webb et al., 2001)
Webb G.I., Pazzani M.J., Billsus D., Machine Learning for User Modelling, User Modelling and User-Adapted Interaction, Netherlands: Kluwer Academic Publishers, No. 11, 2001.

(Wenger, 1987)
Wenger E., Artificial Intelligence and Tutoring Systems, Morgan Kaufmann Publishers, 1987.

(Wooldridge et al., 2000)
M. Wooldridge, N. R. Jennings, and D. Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, Journal of Autonomous Agents and Multi Agent Systems, Vol. 3, No. 3, pp. 285-312, 2000.

(Wooldridge et Jennings, 1995)
Wooldridge M., Jennings N.R., Intelligent Agents: Theory and Practice,

Knowledge Engineering Review, 1995.