JALIL EMMANUEL

MODÉLISATION D'UN OUTIL D'ACQUISITION DE CONNAISSANCES DESTINÉ À L'ENSEIGNANT

Mémoire présente
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de maître ès sciences (M.Sc.)

DÉPARTEMENT D'INFORMATIQUE ET DE GÉNIE LOGICIEL
FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2007

Résumé

Ce mémoire de maîtrise s'inscrit dans le domaine de l'ingénierie de connaissances et plus précisément dans une tâche de construction de contenu pédagogique pour un système éducationnel. Jusqu'ici, divers outils de présentation, d'échange et de stockage des informations ont été développés pour favoriser l'apprentissage de l'étudiant dans les systèmes éducationnels. Cependant, à notre connaissance, peu d'outils visent à minimiser l'effort de l'enseignant dans la construction de contenu pédagogique. L'objectif de notre projet est d'approfondir la notion d'acquisition des connaissances pour aider l'enseignant à construire les connaissances associées à un système éducationnel. Ce projet s'insère dans le cadre de la conception d'un système informatique, disponible en ligne, pour l'apprentissage humain à partir d'exemples. Un agent d'acquisition est proposé pour entrer des exemples et générer automatiquement des documents électroniques représentant ces exemples. Cet agent a été développé et permet de créer des patrons représentant la structure d'un exemple donné. Une interface d'acquisition est générée à partir d'un patron pour permettre la saisie des données qui vont servir à générer l'exemple. Cet exemple sera stocké dans un fichier XML, puis présenté à l'apprenant par le système éducationnel. La technologie XML se prête bien à ces tâches de gestion des connaissances, car elle permet d'inclure une logique de traitement des informations et d'automatiser les traitements, et ce de façon indépendante de la plate-forme utilisée. L'enseignant dispose ainsi d'un outil lui permettant d'ajouter des exemples d'exercices résolus d'une manière plus conviviale et plus flexible, peu importe le contenu de l'exemple.

Remerciements

Je tiens tout d'abord à témoigner ma reconnaissance à ma directrice de recherche, Madame Laurence Capus, ainsi qu'à ma co-directrice, Madame Nicole Tourigny, qui m'ont soutenu et aidé tout au long de mon programme de maîtrise.

Je suis honoré que Monsieur André Gamache, professeur associé au Département d'informatique et de génie logiciel de l'Université Laval, et Monsieur Luc Lamontagne, professeur au Département d'informatique et de génie logiciel de l'Université Laval, aient accepté d'être examinateurs pour mon mémoire de maîtrise.

Je remercie par la même occasion, tous les membres du laboratoire ERICAE, anciens et actuels, qui m'ont aidé de près ou de loin, dans mes travaux de recherche.

Enfin, je remercie profondément mes parents et mes frères qui n'ont jamais cessé de me soutenir et de m'encourager dans mes études.

Table des matières

Liste des tableaux

Liste des figures

1 Chapitre 1 : Introduction

L'acquisition des connaissances peut être vue comme un processus de capture des besoins, de recueil ou encore d'élicitation des connaissances suivie d'une modélisation des connaissances. Dans ce contexte, il est nécessaire de comprendre les étapes à suivre lors de la modélisation d'un système à base de connaissances (SBC).

Nous avons été amené à approfondir la notion d'acquisition des connaissances dans le domaine de l'éducation, afin de pouvoir comprendre la modélisation des connaissances dans les systèmes d'aide à l'apprentissage humain à partir d'exemples, dans le but d'aider l'enseignant à construire les connaissances d'un tel système.

Dans un autre contexte d'exploitation des connaissances préalablement acquises (raisonnement, inférence sur les connaissances, stockage des connaissances, etc.), il est aussi intéressant de comprendre les principes de conception et d'implantation des systèmes de connaissances et de voir comment ceux-ci s'appliquent à travers un exemple concret qui les met en pratique.

La première section de ce chapitre présente nos motivations à réaliser cette étude ainsi que les résultats attendus. Ensuite, dans la deuxième section, nous décrivons les objectifs à atteindre. La démarche suivie pour y parvenir est décrite dans la troisième section. Les résultats obtenus sont présentés dans la quatrième section. Enfin, nous précisons les contributions de notre travail et la structure du rapport

1.1 Motivations

Les SBC constituent une branche particulièrement florissante de l'Intelligence Artificielle (IA). L'ingénierie des connaissances porte sur le développement de SBC. La construction d'un SBC repose sur l'hypothèse principale que les connaissances jouent un rôle majeur dans la résolution d'un problème (David *et al*. 1995). En effet, l'efficacité et la compétence d'un SBC proviennent davantage des connaissances spécifiques qu'il possède, de la logique et de la puissance de ses techniques générales de résolution de problèmes (Waterman 1986).

La construction d'un SBC nécessite en général l'acquisition de connaissances à partir de différentes sources, par exemple la documentation textuelle, le contenu de bases de données existantes du domaine, des experts, etc. Les connaissances sont le plus fréquemment acquises auprès d'experts du domaine, des personnes qui possèdent la connaissance et des habiletés relatives à leur mise en pratique. L'acquisition des connaissances est une tâche cruciale du développement d'un SBC. Elle a longtemps été, et reste encore, une tâche difficile dans la création de bases de connaissances pratiques (David *et al*. 1995).

Ce phénomène s'expliquait par le manque de théorie formelle de l'ingénierie de la connaissance (David *et al*. 1995) et plusieurs autres facteurs y contribuant, en particulier la nature des connaissances de l'expert du domaine dont il n'a pas conscience, les difficultés rencontrées lors de la communication entre l'expert et l'ingénieur de la connaissance qui doit comprendre et modéliser ces connaissances et l'insuffisance de langages informatiques pour représenter ces connaissances (Puerta *et al.*1993).

Dans un contexte éducationnel, un SBC peut servir pour la présentation, l'échange et le stockage des informations dans le but de favoriser l'apprentissage de l'apprenant. Cependant, à l'heure actuelle, peu d'outils visent à minimiser le rôle de l'enseignant dans la construction et la maintenance de contenu pédagogique, surtout que, les connaissances qui sont manipulées par les systèmes informatiques ont plutôt tendance à évoluer. Ce qui peut constituer un fardeau non négligeable pour les personnes responsables de l'acquisition de ces connaissances.

En général, le développement d'un SBC demande des ressources dont le nombre varie selon la complexité du système à construire. Il implique l'emploi et le dialogue de professionnels de l'informatique et du domaine d'application, ce qui peut devenir très coûteux et difficile à réaliser. De plus, la nature dynamique de ces systèmes et le fait que ces spécifications évoluent, augmentent le risque que les connaissances contenues dans le système deviennent incomplètes et/ou incorrectes.

Une question importante se pose donc : comment aider l'enseignant à construire les connaissances d'un SBC ? Comment faire en sorte que le développement d'un SBC produise de meilleurs systèmes d'apprentissage humain, qui favorisent à la fois les activités

d’apprentissage des connaissances et facilitent les tâches des personnes qui construisent ces connaissances ?

1.2 Objectifs

Les travaux menés dans le cadre de notre mémoire visent à faciliter l'élaboration d’un système à base de connaissances graduellement extensible. Dans un contexte d’apprentissage humain à partir d’exemples, un tel système permettra de diminuer l’effort de l’enseignant lors de la construction et la maintenance de contenu pédagogique (pour le contenu des exemples, les connaissances d’apprentissage incluses dans le système, etc.).

L’objectif de ce travail est donc la modélisation d’un agent d’acquisition de connaissances destiné à l’enseignant. Cet agent joue le rôle d'un éditeur de la base de connaissances (contenu pédagogique) du SBC et a pour conséquences d'améliorer sa compétence, sa robustesse, ses capacités d'explication et de réutilisation.

Pour atteindre cet objectif, nous utilisons les résultats de recherche en acquisition de connaissances dans les SBC de manière à doter notre système des caractéristiques (robustesse, explication, réutilisabilité) des outils d'acquisition de connaissances proposés au sein des travaux récents dans l'ingénierie de la connaissance identifiés en tant que « systèmes experts de seconde génération » (SE2G) (David *et al*. 1993), (David *et al*. 1995), (Studer *et al*. 1998).

Nous proposons un modèle d’application capable d’éditer le contenu pédagogique destiné au système d’apprentissage humain. Le modèle d’application est un modèle d'agent logiciel capable d'assister l’ingénieur de la connaissance (cogniticien) et l'expert de la connaissance dans la tâche de constitution et de maintenance de la base de connaissances du système. Plus précisément, il facilitera la communication entre l’enseignant et le système éducationnel afin de lui permettre de définir et de modifier les composantes de son modèle conceptuel ainsi que les connaissances de son domaine. En bref, de faire la gestion des connaissances.

1.3 Démarche suivie

Pour réaliser notre travail, nous avons effectué tout d'abord une recherche bibliographique sur les travaux des SBC, ce qui nous a permis de comprendre le processus d'ingénierie de la connaissance, et plus particulièrement le processus d'acquisition des connaissances.

De plus, nous avons effectué une recherche sur les outils d'acquisition des connaissances existants dans le but de faire le point sur l'état de l'art dans ce domaine. La recherche a porté plus particulièrement sur les caractéristiques souhaitables d'un outil d'acquisition des connaissances. Nous avons identifié les caractéristiques et les fonctionnalités de l'interface nécessaires pour communiquer avec l'utilisateur.

Nous avons par la suite développé un cadre d'analyse présentant une problématique sur l'acquisition des connaissances et différentes méthodes avec leur capacité à satisfaire les besoins pour résoudre le problème posé.

Pour mener l'analyse, la conception et la réalisation de notre outil, nous avons utilisé la méthode de développement de SBC CommonKADS (Schreiber *et al*. 2000). Le choix de cette méthode repose sur le fait qu'elle a permis de synthétiser de nombreuses idées pertinentes et elle est devenue un point de référence et un standard pour le développement des SBC. Le développement de plusieurs modèles selon cette méthode correspond à plusieurs perspectives de la résolution du problème dans son contexte d'application.

Selon CommonKADS, seuls les modèles nécessaires pour atteindre les objectifs du projet doivent être développés entièrement. Pour notre étude, nous avons modélisé les connaissances sur notre domaine d'application et réalisé entièrement les modèles du domaine, des tâches, des agents, de communication et de conception.

Afin de valider notre travail de modélisation et de prototypage, nous avons réalisé un prototype d'application dédié à la tâche d'acquisition des connaissances du système informatique éducationnel qui est disponible en ligne.

Dans notre démarche, nous avons considéré une approche visant à intégrer de manière explicite le modèle conceptuel dans le système exécutable. Ainsi, il nous est possible de modifier les spécifications du SBC, donc d'enrichir et de maintenir son contenu tout au long

de son utilisation, en bénéficiant des avantages d'une approche par prototypage, ce qui réalise sa nature extensible.

1.4 Résultats obtenus

Les résultats de notre travail sont le modèle de notre outil d'acquisition et son prototype. Grâce à notre outil, le système qui l'intègre est capable d'accepter les nouvelles connaissances d'une manière simple et conviviale. Ainsi, l'ingénieur de la connaissance (cogniticien) lors du développement du système et l'expert tout au long de son utilisation peuvent, au fur et à mesure que l'expertise évolue et, chacun du point de vue qui l'intéresse et selon son rôle, mettre à jour le contenu de la base de connaissances du système. Cette maintenance concerne à la fois la définition de concepts de domaine et leurs relations, la décomposition et l'ordonnancement de tâches.

La mise en œuvre du prototype a été réalisée dans un environnement orienté objet qui utilise le langage de programmation Java et qui permet de développer des interfaces conviviales et flexibles. Nous considérons qu'un outil visuel d'acquisition de connaissances, utilisé en grande partie par l'expert lui-même, peut faciliter ses tâches. Cependant il est important que l'outil ne demande à l'utilisateur que les connaissances dont il a réellement besoin pour la réalisation de sa tâche pour éviter le risque que l'utilisateur ne se désintéresse de l'outil.

Nous avons également utilisé la technologie XML car elle permet d'inclure une logique de traitement des informations et d'automatiser les traitements, et ce d'une manière indépendante de la plate-forme utilisée. Ainsi le SBC peut réagir de façon appropriée aux actions de l'utilisateur.

1.5 Contribution

La principale contribution de notre travail est l'élaboration d'une approche de modélisation cognitive pour le développement d'un outil d'acquisition de connaissances destiné à l'enseignant. Cette approche vise à intégrer d'une manière explicite le modèle conceptuel de l'outil dans son système exécutable, dans le but d'améliorer son extensibilité. Pour cela,

nous avons approfondi la notion d'acquisition des connaissances pour aider l'enseignant dans la construction de contenu pédagogique. Nous avons expérimenté cette approche dans la réalisation d'un prototype.

1.6 Structure du mémoire

Une synthèse des résultats de recherche sur l'ingénierie des connaissances et les SBC est présentée dans le chapitre 2. Nous présentons les caractéristiques des SBC selon plusieurs perspectives, notamment la manière d'organiser le développement de tels systèmes. Nous précisons le procédé d'acquisition des connaissances à travers les caractéristiques partagées de différentes méthodes de l'ingénierie de la connaissance. La compréhension de ce processus joue un rôle important dans la démarche de modélisation de notre outil. Ensuite, les caractéristiques des outils d'acquisition des connaissances sont présentées afin de mettre en évidence l'état de l'art dans ce domaine.

Dans le chapitre 3, nous décrivons une problématique sur l'acquisition des connaissances plus précisément dans le cadre des systèmes d'aide à l'apprentissage humain. Après avoir précisé la problématique identifiée sur l'acquisition des connaissances, nous présentons le cadre méthodologique que nous adoptons pour mener notre analyse.

Le chapitre 4 est consacré à la description des éléments pertinents de notre méthode d'analyse. Nous y présentons en détail les différents modèles d'analyse que nous avons élaborés à l'aide du cade de modélisation de CommonKADS (Schreiber *et al*. 2000) pour l'application que nous voulons construire. Certains modèles seront plus détaillés que d'autres en fonction de leur importance dans notre projet.

Dans le chapitre 5, nous élaborons le modèle de conception et nous présentons les résultats du prototypage. Nous décrivons ensuite les diverses fonctionnalités qui seront implémentées dans le futur système. Nous discutons de différents aspects (fonctionnel, comportement et physique) qu'il est possible de distinguer dans chacun des constituants du modèle. Enfin, le prototype est présenté avec ses interfaces illustrant les diverses possibilités.

Dans le chapitre 6, nous présentons l'expérimentation menée sur notre outil avec les résultats obtenus. De plus nous dégageons et discutons les avantages et les inconvénients de notre outil avant de terminer par une discussion sur des perspectives d'extension dans le but d'améliorer l'outil d'acquisition des connaissances proposé.

Nous concluons dans le chapitre 7 en rappelant les travaux réalisés dans le cadre de ce mémoire, puis nous énonçons quelques avenues de recherche permettant de les étendre.

2 Chapitre 2 : État de l'art

Depuis les débuts de l'intelligence artificielle, les chercheurs ont essayé de rendre les machines capables d'apprendre de façon autonome, de penser et de raisonner à la manière des humains, de communiquer entre elles. Ainsi les SBC ont été mis au point pour que la machine soit capable d'inférer sur des connaissances, tout comme le ferait un humain. Mais pour le faire, il faut avoir acquis les connaissances au préalable. Le processus d'acquisition des connaissances est une tâche délicate, notamment à cause des sources qui sont souvent nombreuses et variées. Il faut une gestion appropriée des connaissances et de leurs rôles dans le système. Il faut aussi définir une ontologie associée à ces sources de connaissances de façon à permettre une bonne et même compréhension de celles-ci ainsi qu'une utilisation efficace.

Il est donc important de trouver une bonne façon pour recueillir ces connaissances et les insérer dans le SBC. Ce nouveau besoin propre à l'intelligence artificielle a fait émergé une nouvelle discipline sous-jacente à l'intelligence artificielle, l'ingénierie des connaissances (IC). L'IC est une discipline qui permet notamment de modéliser des problèmes pour lesquels les seules connaissances dont nous disposons sont de nature linguistique ou cognitive. Plus formellement, l'IC se définit comme étant l'étude des concepts, des méthodes et des techniques permettant de modéliser et/ou d'acquérir les connaissances pour des systèmes réalisant ou aidant des humains à réaliser des tâches se formalisant a priori peu ou pas (Charlet *et al.* 1999). Il revient donc à l'ingénieur de la connaissance de modéliser le système pour que celui-ci soit le mieux adapté possible et apte à répondre aux besoins de l'utilisateur tant sur la sémantique d'utilisation que sur l'acquisition des connaissances (Van Heist *et al*. 1997).

Dans ce chapitre, nous verrons comment les SBC et les méthodologies reliées à l'ingénierie des connaissances ont évolué. Nous verrons comment les chercheurs ont réussi à élaborer une méthode pour créer un vocabulaire commun aux intervenants d'un domaine particulier. Cette nouvelle manière de qualifier des mots forcera l'ingénieur de la connaissance à redéfinir sa technique d'acquisition des connaissances. Il devra donc se doter d'outils pour l'assister dans la réalisation de cette tâche.

2.1 Évolution des SBC et acquisition des connaissances

Des systèmes qui utilisent des techniques de l'intelligence artificielle (IA) pour effectuer des tâches réservées généralement aux humains sont appelés des systèmes de connaissances (SBC), ou encore des systèmes experts (SE) (Tourigny 1990). Ces systèmes sont appliqués à divers domaines (informatique, médecine, transport, etc.).

On distingue aujourd'hui deux générations de SBC (David *et al*, 1993). Dans les systèmes experts de première génération (SE1G), les connaissances étaient représentées le plus souvent de façon uniforme par des règles de la forme « Si Condition(s) alors Action(s) », ce qui permettait d'exprimer les connaissances sous forme déclarative.

Cette approche accélérait le développement des SBC. Cependant elle a conduit à des systèmes ayant un faible niveau d'abstraction des connaissances. De plus les SE1G étaient difficiles à modifier, peu réutilisables et transférables d'un environnement à l'autre.

Les travaux menés pendant les dix dernières années pour pallier ces difficultés ont conduit aux systèmes experts de seconde génération (SE2G). Ces travaux ont permis de poser les bases d'une ingénierie des connaissances basée sur le paradigme de modélisation. Avec les SE2G, la distinction est faite entre le type des connaissances et les modes de représentation utilisés pour les manipuler, ce qui offre un meilleur niveau d'abstraction (David *et al*, 1993).

Dans les années 80, construire un système expert était un travail plutôt artisanal relevant davantage du bricolage que d'une discipline scientifique. En fait, la plupart des systèmes experts étaient construits de façon « ad hoc ». Malheureusement, ces systèmes experts ont abouti à un constat d'échec. Cet échec était dû à la rigidité de leur système de règles, mais aussi à l'erreur qui avait été commise de penser que toute la connaissance accumulée par un expert pouvait être transférable sous forme de règles logiques. Le résultat obtenu était un ensemble de systèmes trop rigides, difficiles à utiliser dans des cas réels et presque impossibles à mettre à jour sans détruire la cohérence de la base de connaissances.

Les ingénieurs de la connaissance ont réalisé que cette méthodologie n'était pas adaptée aux besoins réels des utilisateurs. Ils ont donc tenté d'élaborer de meilleures méthodes qui

leur permettraient d'encadrer le processus de construction de ces systèmes (par exemple CommonKADS). Ce type d'ingénierie ne vise pas à proposer des processus normés de conception des systèmes, mais des cadres méthodologiques et technologiques structurants.

Depuis, l'ingénierie des connaissances n'est donc plus vue comme une activité de transfert d'expertise, mais comme une activité de construction de modèles (David *et al*. 1995) représentant les perspectives de résolution du problème à résoudre dans son contexte. Pour parvenir à ce but, plusieurs méthodes ont été proposées. Certaines méthodes sont dites « descendantes » ou « dirigées par les modèles », comme KADS et d'autres sont dites « ascendantes » qui partent des données, particulièrement des textes pour construire un modèle du domaine (Studer *et al*. 1998).

La méthode KADS amène l'ingénieur de la connaissance à construire un modèle conceptuel de l'expertise composé de quatre niveaux : domaine, inférence, tâche et stratégie. Les recherches ont tout d'abord été moins portées sur la couche du domaine mais se sont dirigées plutôt vers la résolution de problèmes. Par la suite, les chercheurs se sont rendus compte des difficultés à construire, échanger et partager les connaissances du domaine, connaissances qui contiennent un ensemble de termes et d'assertions à propos du domaine.

Une partie importante de la couche du domaine correspond à l'ontologie. La définition de l'ontologie ainsi que la manière de la construire sont des problèmes très étudiés mais pas encore bien maîtrisés. Avant de poursuivre notre présentation, il nous a semblé approprié pour la compréhension du lecteur d'expliciter la notion d'ontologie, ce qui fait l'objet de la section suivante.

2.2 Ontologie

Dans cette section, nous présentons la notion d'ontologie par rapport aux différentes définitions rencontrées dans la littérature.

Le terme « Ontologie » est synonyme de métaphysique générale et réfère à la « spéculation sur l'être en tant qu'être, sur l'être en soi ». Le terme « ontologie » désigne la partie de la métaphysique qui s'applique à l'être en tant qu'être, indépendamment de ses

déterminations particulières (Petit Robert 1, dictionnaire alphabétique et analogique de la langue française).

Une ontologie peut être définie comme étant : « L'ensemble des objets reconnus comme existants dans le domaine. Construire une ontologie, c'est aussi décider de la manière d'être et d'exister des objets. » (Gruber 1993). Une ontologie contient les termes qui font référence aux catégories du domaine ainsi que les termes qui se rapportent aux relations entre ces termes. L'ontologie définit donc le vocabulaire pour décrire un domaine donné.

En général, les définitions de l'ontologie données en IA évitent de référer à la réalité, mais elles utilisent plutôt les termes comme « représentation » et « conceptualisation ». En effet, une ontologie, pour un ensemble de connaissances concernant une tâche particulière ou un domaine, décrit une classification de concepts pour cette tâche ou domaine, laquelle définit l'interprétation sémantique de ces connaissances. Selon Van Heijst (1997), une ontologie est l'ensemble des distinctions porteuses de sens pour un agent et est affectée par le domaine et la tâche particulière pour laquelle elle est prévue. Cela amène Van Heijst à définir quatre classes d'ontologies :

- l'ontologie du domaine qui exprime les conceptualisations propres à un domaine (les ontologies du domaine sont très souvent subsumées par une ontologie générique) ;

- l'ontologie applicative qui contient les connaissances de la couche du domaine nécessaires à une application particulière ;

- l'ontologie générique qui contient les connaissances valables dans les concepts de cause, les états, les processus, etc. ;

- l'ontologie de représentation qui contient la conceptualisation des primitives de représentation des connaissances. (Les autres ontologies sont décrites avec des concepts d'une telle ontologie).

Une distinction entre ces différents types d'ontologies est essentielle pour bien structurer l'information et identifier ce à quoi elle servira. Les ingénieurs de la connaissance ont

beaucoup travaillé sur l'aspect de la modularité des composantes dans le but d'augmenter la réutilisation des connaissances. De cette façon, ils diminuent le temps de développement des systèmes. Les connaissances réutilisées pourront donc faciliter la tâche d'acquisition des connaissances des experts. Par conséquent, pour bien utiliser et acquérir ces connaissances, il faudra développer des outils en conséquence. Dès le début des années 1990, les ingénieurs de la connaissance se sont concentrés sur l'édition et l'acquisition des différentes ontologies.

2.3 Acquisition des connaissances

Cette section porte sur une analyse détaillée du processus d'acquisition des connaissances, ainsi que la manière générale d'organiser le processus de développement des SBC. Les méthodes récentes proposent la construction d'un modèle conceptuel qui joue un rôle central dans le cycle de vie des SBC. Dans les approches actuelles, l'acquisition des connaissances est présentée comme un processus de modélisation, plutôt qu'un processus de transfert (comme dans les SE1G). Le modèle conceptuel produit doit posséder un niveau d'abstraction élevé, appelé niveau des connaissances (Newell 1982).

Dans cette section, nous rappelons ce qu'est un modèle et précisons les critères auxquels un bon modèle doit satisfaire. Puis, nous décrivons le modèle du processus d'acquisition des connaissances en nous basant sur les considérations précédentes. Nous discutons ensuite des caractéristiques des outils d'acquisition des connaissances et présentons un aperçu historique du développement des outils d'acquisition des connaissances, avec certains outils décrits dans la littérature.

2.4 Méthodes d'acquisition de connaissances

Les méthodes récentes de l'ingénierie des connaissances fournissent un cadre pour organiser le processus de développement d'un SBC en permettant une description à un plus haut niveau d'abstraction.

La construction d'un SBC passe par les étapes suivantes :

- modélisation : construire le modèle conceptuel (MC) ;

- acquisition : structurer le recueil des connaissances en s'appuyant sur le MC ;
- conception : définir les spécifications du système à réaliser à partir du MC ;
- implémentation : réaliser le système exécutable en s'appuyant sur les spécifications détaillées.

Le MC contient une description abstraite des connaissances utilisées pour résoudre un problème donné. Les tâches, les méthodes et les concepts du domaine y sont explicités. Ce modèle permet de décrire le système et sert à structurer la constitution de la base de connaissances du système.

La construction du MC joue un rôle important dans le processus de réalisation d'un SBC. Elle représente le point de départ pour la conception détaillée et l'implémentation du système. Pour la constitution de la base de connaissances, le modèle conceptuel est instancié sur le domaine considéré. II s'agit donc d'un processus d'élicitation dirigé par un modèle. Le modèle est donc ici le résultat d'une démarche descendante qui vise à abstraire le comportement des experts.

Dans les approches descendantes, la construction du modèle est vue comme une activité de sélection. Le modèle est choisi parmi une liste de modèles existants (approche par restriction des rôles) ou composé à partir de blocs sélectionnés dans une bibliothèque (approche par tâches génériques) ou encore par combinaison de ces deux approches.

2.5 Outils d'acquisition de connaissances

Dans ce chapitre nous présentons Protégé (http://protege.stanford.edu/), un logiciel qui permet de construire des outils personnalisés pour l'acquisition de connaissances. Nous avons retenu ce choix parce que Protégé est un atelier logiciel qui intègre plusieurs plugiciels (plug-ins) capables de mettre en pratique de façon concrète les différents aspects de l'IC. Pour cette raison, nous nous limiterons à décrire l'outil Protégé qui nous apparaît plus complet au lieu de présenter différents outils.

Pour répondre à un besoin de plus en plus grandissant de pouvoir créer des outils d'acquisition de connaissances, une équipe de l'Université de Stanford a décidé d'élaborer un logiciel qui accomplirait cette tâche : Protégé. D'abord construit pour générer des outils personnalisés pour l'acquisition de connaissances, cet atelier de développement logiciel permet maintenant de construire des SBC (Gennari *et al*. 2002). Les auteurs ont été motivés par la nécessité de réduire le goulot d'étranglement créé dans l'acquisition de connaissances et leur objectif était de minimiser la tâche de l'ingénieur de la connaissance lors de la construction de bases de connaissances (Gennari *et al*. 2002).

À l'origine, Protégé était une petite application destinée à construire des outils d'acquisition de connaissances spécialisés dans la planification médicale. Le système a ensuite connu différentes évolutions vers un environnement flexible et robuste de développement de SBC.

L'évolution de Protégé s'est faite en quatre grandes phases durant une quinzaine d'années de développement. La première version, Protégé-I, était une application limitée au domaine de la planification médicale et comprenait une seule méthode de résolution de problèmes, ESPR ou *episodic skeletal-plan refinement method*, (Tu *et al*. 1989), basée sur l'instanciation et l'amélioration progressive des plans squelettiques pour des problèmes spécifiques. C*et al*gorithme de résolution des problèmes était intégré directement dans le système. Les utilisateurs fournissaient les données à l'exécution. La deuxième version, Protégé-II, incorporait plus d'une méthode de résolution de problèmes, séparées du système. Ainsi on pouvait appliquer différentes méthodes de résolution de problèmes à différentes bases de connaissances. La troisième version, Protégé/Win, intégrait de façon dynamique un ensemble d'outils indépendants pour construire le SBC. Dans cette version, l'accent avait été mis sur la facilité d'intégration des outils dans le système global et sur les notions de modularité et de réutilisation. Cette version était conçue pour une communauté limitée d'usagers. Enfin, la dernière version, Protégé-2000, inclut le modèle de connaissances OKBC (Open Knowledge-Base Connectivity) et une interface de programmation (API) qui ont rendu l'outil plus extensible et flexible.

2.5.1 Évaluation d'un outil d'acquisition de connaissances : PROTÉGÉ

Dans cette section nous évaluons l'outil Protégé, présenté brièvement à la section précédente, dans le but de mieux connaître les outils d'acquisition de connaissances. Notre évaluation porte sur la modélisation et la représentation des connaissances, ainsi que sur leur acquisition et leur sauvegarde. Nous verrons également quels sont les outils de modélisation disponibles dans l'environnement Protégé et la possibilité de les intégrer à d'autres systèmes existants.

2.5.2 Modélisation des connaissances

Pour modéliser les connaissances, les auteurs de Protégé introduisent la notion d'ontologie pour décrire de façon explicite et formelle les concepts (classes) du domaine, leurs propriétés ainsi que les relations entre ces concepts. L'ontologie et les instances des concepts constituent la base de connaissances. L'ontologie permet de définir un vocabulaire commun pour partager des mêmes connaissances. De cette façon, on peut partager une compréhension commune de la structure de l'information, permettre l'analyse et la réutilisation de la connaissance du domaine, rendre explicites les hypothèses sur le domaine et séparer la connaissance du domaine de la connaissance opérationnelle (Noy *et al*. 2000).

2.5.3 Représentation des connaissances

Avec Protégé, les connaissances sont représentées à l'aide de schémas (Noy *et al*. 2000). Les schémas permettent de représenter les types de connaissances. En plus, Protégé permet d'avoir une séparation nette entre les types de connaissances. En effet, les connaissances du domaine sont séparées des connaissances opérationnelles, et les bases de connaissances sont séparées des méthodes de résolution de problèmes.

- **Les connaissances du domaine**

Les connaissances du domaine sont les connaissances qui se rapportent au domaine d'application. Elles sont représentées comme un ensemble de concepts structurés sous forme d'une hiérarchie de classes avec leurs propriétés.

- **Les connaissances opérationnelles**

Les connaissances opérationnelles sont les méta-connaissances ou encore, les connaissances qui permettent d'agir sur les différentes connaissances du domaine (Noy *et al*, 2000). Ces méta-connaissances peuvent être des relations entre les classes, des contraintes ou des restrictions qu'on peut avoir sur les connaissances du domaine (les propriétés de classes) ou bien des méthodes de résolution de problèmes qui définissent les rôles de la connaissance.

2.5.4 Acquisition des connaissances

L'acquisition consiste en la construction des bases de connaissances par instanciation des classes identifiées dans le domaine une fois que celui-ci a été structuré. Une interface d'acquisition est automatiquement générée par Protégé pour permettre de choisir la classe, d'en créer une instance et entrer les valeurs pour ses propriétés.

2.5.5 Sauvegarde des connaissances

Grâce aux différents plugiciels (*plug-ins*) intégrables à Protégé, il est possible d'enregistrer les bases de connaissances et les ontologies selon plusieurs formats. Le format standard est le Standard Text Files. C'est un format interne de Protégé. Il permet d'enregistrer dans des documents séparés la hiérarchie des classes (structure du domaine) et les instances (bases de connaissances).

Les autres formats sont les schémas RDF (Ressource Description Framework) pour le Web sémantique, le format XML et les schémas XML, les bases de données relationnelles JDBC, etc. La sauvegarde des connaissances se fait de façon transparente pour l'utilisateur. Par exemple, les données peuvent être lues à partir d'un schéma RDF puis sauvegardées dans une base de données relationnelle.

2.5.6 Outils disponible(s) dans l'environnement Protégé

Protégé offre un grand ensemble d'outils pour la modélisation, qui sont disponibles dans son environnement pour aider le développeur dans la construction d'un SBC. Le plus important est l'éditeur d'ontologie, pour bâtir la structure des concepts du domaine.

- **L’éditeur d’ontologie**

Protégé offre un éditeur d’ontologie avec lequel on peut définir la hiérarchie de classes et créer une ontologie du domaine. Il faut commencer par déterminer le domaine et la portée de l'ontologie. Cela revient à répondre à plusieurs questions de base, dont les suivantes: Quel domaine couvrira l'ontologie ? Pourquoi veut-on utiliser l'ontologie ? Pour quels types de questions l'information dans l'ontologie devrait-elle fournir des réponses ? Qui emploiera et mettra à jour l’ontologie ?

Il est souvent intéressant de considérer l'ontologie construite par quelqu'un d'autre et vérifier s’il n’est pas possible de la raffiner et de l’étendre pour son propre domaine. La réutilisation des ontologies existantes peut être nécessaire si le système anticipé doit interagir avec d'autres applications qui utilisent des ontologies particulières. Il existe diverses bibliothèques d’ontologies (Exemples : Ontolingua, DAML ontology library). Protégé permet d’importer des ontologies déjà existantes, puis de les raffiner et de les adapter à la structure du domaine. Ainsi, l’ontologie peut être bâtie sans partir de zéro.

Ensuite, il faut établir une liste de termes relatifs à l’ontologie que l’on souhaite construire. Ce sont des termes que l’on voudrait expliquer à un utilisateur avec leurs propriétés, en mentionnant l’information à savoir.

L’étape suivante consiste à définir les classes et la hiérarchie de classes. À cette définition de classe s’ajoute la définition des propriétés des classes. Les classes se présentent sous une hiérarchie de répertoire.

Il existe trois types de hiérarchie de classes dans Protégé :

1. Descendante (*Top-Down*) : on définit les concepts les plus généraux dans le domaine et ensuite on spécialise ces concepts.

2. Ascendante (*Bottom-up*) : on commence par définir les classes les plus spécifiques au début de la hiérarchie puis on groupe ces classes dans des concepts plus généraux.

3. Combinaison : on définit les concepts principaux en premier puis on les généralise et on les spécialise de façon appropriée.

On retrouve des propriétés de type intrinsèque et de type extrinsèque. Pour une classe donnée, une propriété intrinsèque appartient à l'essence même de la classe (Exemple : l'odeur d'un vin est une propriété intrinsèque pour une classe de vin). Une propriété extrinsèque (le nom d'un vin et son terroir) n'appartient pas à l'essence de la classe.

D'autres propriétés servent à exprimer les relations entre les classes, ou l'héritage qui peut être simple ou multiple. Une classe hérite des propriétés de sa superclasse. Les propriétés héritées sont rattachées à la classe la plus générale qui la possède. Ceci a l'avantage de permettre la redéfinition de la propriété dans le cas de deux classes enfants d'une même classe mère.

- **Les autres outils disponibles**

Outre l'éditeur d'ontologie pour bâtir la structure des concepts du domaine, nous retrouvons un outil d'acquisition de connaissances, automatiquement généré à partir de l'ontologie du domaine et qui permet de construire les bases de connaissances. Un autre outil, PROMPT, permet de manipuler plusieurs ontologies. Il permet de comparer différentes versions de la même ontologie, de déplacer les schémas (frames) entre les projets, de fusionner deux ontologies et enfin d'extraire une partie d'une ontologie. Un outil de représentation graphique de la structure des domaines permet de voir les classes et les instances du domaine sous forme graphique. Des outils pour faire du raisonnement et de l'inférence sont aussi disponibles notamment les suivants :

- Algernon Tab Widget est un système d'inférence à base de règles implémenté en Java qui utilise les chaînages avant et arrière. Les bases de connaissances sont structurées à l'aide de schémas (frames).

- CLIPSTab utilise le moteur de règles de CLIPS (CLIPS Rule Engine) et fait du chaînage avant basé sur l'algorithme RETE.

- Prolog Tab Widget est une intégration de l'application "GNU Prolog for Java" à Protégé. Prolog Tab Widget permet de faire du chaînage avant et du chaînage arrière. Les relations entre les propriétés sont représentées par des faits dans Prolog.

- PSM Librarian Tab Widget est un interpréteur qui explore plusieurs bibliothèques de méthodes de résolution de problèmes et choisit la ou les méthodes les plus pertinentes pour traiter les ontologies de domaine.

Cette liste d'outils n'est pas exhaustive, mais dans notre contexte de modélisation, les outils mentionnés ci-dessus nous apparaissent les plus pertinents.

Une fois l'outil d'acquisition de connaissances généré, il est possible de l'intégrer dans une architecture où il communiquerait avec un moteur d'inférence pour raisonner sur les connaissances acquises au moyen d'un mécanisme de type plugiciel.

2.6 Une application particulière : les systèmes d'aide à l'apprentissage humain à partir d'exemples

Dans un contexte plus global, les systèmes d'aide à l'apprentissage humain s'insèrent dans le cadre d'un EIAH. Le sigle EIAH signifie « Environnement Informatique pour l'Apprentissage Humain ».

Au début de l'usage des ordinateurs, on parlait d'enseignement programmé ou encore d'« Enseignement Assisté par Ordinateur ». Le terme est devenu « Enseignement Intelligemment Assisté par Ordinateur » avec l'introduction de techniques d'Intelligence Artificielle. Par la suite, l'importance fondamentale de l'interactivité des systèmes a conduit au terme « Environnement Interactif d'Apprentissage avec Ordinateur ». Maintenant, un EIAH, au sens large du terme, désigne un environnement informatique qui intègre des agents humains (apprenant ou enseignant) et informatiques (logiciel, machine) et leurs permet d'interagir avec la possibilité d'accéder à des ressources formatives (humaines et/ou médiatisées), et ce de façon locale ou distribuée (Tchounikine, 2002).

Le domaine de recherche sur les EIAH fait intervenir plusieurs disciplines : pédagogie, didactique, psychologie cognitive, sciences de l'éducation et informatique. À cela s'ajoutent les sciences de l'information et de la communication. Les travaux de recherche en informatique sur les EIAH visent la conception de dispositifs pédagogiques informatiques prenant en compte les objectifs et contraintes liés à un apprentissage. Les problèmes qui en découlent ne relèvent pas d'un simple travail d'ingénierie. L'enseignant

joue un rôle qui n'est pas négligeable lorsqu'il s'agit de définir les spécificités d'un usage éducatif dans la réalisation d'outils adéquats supportant les tâches de l'enseignant.

À partir plusieurs fonctionnalités telles la présentation d'informations, le traitement des connaissances, la communication entre l'humain et la machine ou encore la collaboration entre humains, les EIAH permettent de répondre aux besoins de l'apprentissage (Tchounikine 2002).

Dans ce contexte, les EIAH à partir d'exemples sont un type particulier d'EIAH dédié notamment à la présentation d'exemples du domaine d'apprentissage étudié. Un tel système permet aux apprenants d'acquérir de nouvelles connaissances ou d'approfondir leurs connaissances en étudiant des exemples de résolution de problèmes. Ils peuvent étudier de plusieurs manières : observer comment le problème a été résolu, refaire le problème en vérifiant la solution ou en s'auto-expliquant la stratégie de résolution du problème.

L'auto-explication est un processus cognitif par lequel un apprenant parvient à intégrer de nouvelles données à des connaissances déjà acquises (Leclerc *et al.*1998). Il consiste principalement à réaliser des inférences à propos des données traitées de manière à créer des liens avec ces connaissances. Des recherches démontrent que les apprenants les plus habiles à réaliser ces inférences sont aussi les apprenants les plus performants (Leclerc, 1998).

Par conséquent, il apparaît intéressant d'inciter les apprenants à l'auto-explication et ainsi les supporter dans la prise en charge de leurs propres stratégies d'apprentissage. De plus, la possibilité offerte à un apprenant de présenter ses auto-explications à d'autres apprenants peut l'inciter à la collaboration. Chaque apprenant peut alors commenter les auto-explications présentées par les autres apprenants.

Dans ce chapitre, nous présentons une évolution sommaire de ces systèmes, puis nous passons en revue les tâches de l'enseignant dans la construction de tels systèmes.

2.6.1 Évolution des systèmes d'aide à l'apprentissage humain

Notre conception de l'enseignement et de la formation connaît depuis plusieurs années une profonde mutation. L'accent est mis sur les notions d'enseignement sur mesure ou à distance, de modification des rythmes d'apprentissage, de flexibilité des accès aux ressources de formation, d'auto-formation ou d'auto-apprentissage, etc.

Des environnements informatiques spécialisés dans l'aide à l'apprentissage humain ont été développés, qui mettent l'apprenant seul en face d'un système et qui réagissent au comportement individuel de cet utilisateur, en lui apportant automatiquement aide et assistance. Aujourd'hui, ces systèmes sont régulièrement utilisés et intégrés à des formations qui les utilisent comme support de formation. Si cette approche est séduisante pour l'apprenant sur le plan de l'apprentissage, en lui offrant la possibilité d'apprendre à son rythme et de collaborer avec un groupe d'acteurs humains, elle ne prend pas en compte l'aspect organisationnel de l'enseignant. En effet peu d'outils visent à diminuer l'effort requis par l'enseignant dans la construction de contenu pédagogique en prenant en compte la notion d'acquisition des connaissances pour aider l'enseignant à construire les connaissances du système éducationnel.

D'autres recherches en informatique portent sur la dimension de la formation à distance en développant les notions de "campus" et de "classe virtuels". S'appuyant sur l'expérience accumulée dans le domaine du Travail Coopératif Assisté par Ordinateur (CSCW pour le sigle anglais), ces recherches portent sur au support de la coopération pour le travail et la formation et donc au support d'une activité sociale. Elles visent à définir le rôle de chaque participant du groupe de formation, lui offrir l'accès et la gestion de ressources de communication synchrones et asynchrones. La plupart des recherches essaient de s'appuyer sur des ressources éducatives et d'impliquer des communautés d'apprenants-formateurs. De plus, elles tentent de prendre en compte l'étude des structures de dialogue propres à l'apprentissage collaboratif entre humains, l'acquisition par les utilisateurs d'une véritable culture d'auto-apprentissage leur permettant de planifier leurs apprentissages, de savoir gérer l'utilisation des ressources éducatives existantes.

2.6.2 Les tâches de l'enseignant

Dans le cadre de la conception d'un EIAH à partir d'exemples, l'un des rôles de l'enseignant est de construire le contenu pédagogique qui va servir à présenter la matière destinée à l'apprenant.

L'enseignement devrait modéliser et représenter les concepts d'apprentissage. Il devrait prendre en compte les notions d'IC pour structurer ses connaissances à enseigner selon la hiérarchie des concepts à étudier, leur importance, leur connectivité. De plus, il ne devrait pas négliger l'aspect visuel du contenu une fois présenté. Pour cela, il devrait faire appel aux sciences cognitives et éducatives. En plus de construire le contenu pédagogique, l'enseignant doit le maintenir à jour. À ces tâches s'ajoutent des tâches d'édition, de création, de modification, de sauvegarde, de suppression, bref de gestion de connaissances, autant de tâches qui présentent une certaine complexité quant à leur réalisation.

Dans le cadre d'un EIAH à partir d'exemples, un outil d'acquisition est nécessaire pour entrer les exemples et générer automatiquement les documents électroniques représentant ces exemples. L'enseignant pourra disposer ainsi d'un éditeur d'exemples lui permettant d'ajouter des exemples d'exercices résolus d'une manière plus conviviale et plus flexible.

2.7 Conclusion

Dans ce chapitre nous avons précisé certaines notions de base sur les SBC. Nous avons également présenté des résultats importants de la recherche sur les outils d'acquisition de connaissances en ce qui a trait à la nécessité d'outils logiciels pour une acquisition flexible et graduelle. Nous avons par ailleurs présenté les EIAH en précisant les tâches de l'enseignant dans la construction de tels systèmes.

Le prochain chapitre est consacré à la problématique d'acquisition de connaissances pour les EIAH et au modèle de processus d'acquisition de connaissances qui servira de cadre d'analyse pour notre étude.

3 Chapitre 3 : Problématique et cadre d'analyse

Au chapitre précédent, nous avons présenté les grandes lignes de l'évolution des SBC. Nous avons présenté également les caractéristiques des SBC selon plusieurs perspectives, notamment la manière d'organiser le développement de tels systèmes. Ensuite, nous avons précisé le processus d'acquisition des connaissances à travers les caractéristiques partagées de différentes méthodes de l'IC, afin de préciser la démarche de modélisation de notre outil. Enfin, nous avons présenté les caractéristiques des outils d'acquisition des connaissances afin de mettre en évidence l'état de l'art dans ce domaine.

Dans ce chapitre nous précisons une problématique identifiée sur l'acquisition des connaissances, plus précisément dans le cadre des EIAH. Après avoir précisé la problématique identifiée sur l'acquisition des connaissances, nous présentons le cadre méthodologique que nous avons adopté pour mener notre analyse ainsi que les éléments d'analyse et les méthodes retenues.

3.1 Problématique sur l'acquisition des connaissances

Dans le but de satisfaire différents besoins en matière de présentation, d'échange et de stockage de connaissances, diverses méthodes d'ingénierie de la connaissance peuvent être utilisées pour fournir un cadre permettant de développer un SBC.

Le développement des EIAH à partir d'exemples fait aussi appel aux divers principes et méthodes d'ingénierie des connaissances. Le processus d'acquisition des connaissances est d'autant plus important que les spécificités de l'usage éducatif que l'on veut faire des connaissances manipulées sont orientées vers l'apprenant.

Ainsi dans le cas des systèmes qui nous intéressent, c'est-à-dire les EIAH à partir d'exemples, les besoins à satisfaire sont orientés vers l'apprenant. Ces systèmes sont développés pour favoriser l'apprentissage de l'étudiant lorsqu'il étudie des exemples.

Cependant, nous avons observé que peu de ces systèmes visent à faciliter la tâche de l'enseignant dans la construction de contenu pédagogique. Il reste y a un besoin d'outils

d'acquisition de connaissances destinés à l'enseignant pour l'aider à réaliser ses tâches d'acquisition de connaissances.

Notre analyse vise à approfondir la notion d'acquisition des connaissances pour aider l'enseignant à construire les connaissances pédagogiques d'un système éducationnel, plus précisément son contenu pédagogique à sauvegarder.

3.2 Cadre d'analyse

Notre analyse consistera à approfondir la notion d'acquisition des connaissances pour aider l'enseignant à construire le contenu pédagogique de l'EIAH à partir d'exemples.

Le cadre d'analyse se situera au niveau de l'application de méthodes d'acquisition des connaissances dans le développement de systèmes éducationnels, plus particulièrement d'outils d'aide à l'acquisition d'exemples.

Dans ce cadre, notre analyse portera sur les éléments qu'il est possible de retrouver dans le cycle de développement de tels systèmes, c'est-à-dire les connaissances, la modélisation et la représentation de celles-ci. Ces éléments sont présentés en détails dans la section suivante.

3.3 Éléments d'analyse

Dans cette section, nous présentons les éléments d'analyse dans le cadre de notre problématique. Ces éléments d'analyse porteront sur les connaissances, leur modélisation et leur représentation.

3.3.1 Connaissances

Les connaissances désignent l'ensemble des connaissances manipulées dans le cadre du développement de notre application. Elles comprennent également les connaissances du domaine d'application ainsi que les connaissances qui permettent d'agir sur les connaissances du domaine. Ces connaissances sont organisées sous la forme d'une

hiérarchie de classes d'objets. Les sections qui suivent présentent la modélisation et la représentation de ces connaissances.

3.3.2 Modélisation des connaissances

Pour modéliser les connaissances, nous utiliserons la notion d'ontologie. Une ontologie est une description explicite et formelle des concepts (classes) du domaine et des relations (propriétés, attributs, restrictions, etc.) entre ces concepts. L'ontologie permet de définir un vocabulaire commun pour partager une même connaissance. De cette façon, il sera possible de partager une compréhension commune de la structure de l'information, de permettre l'analyse et la réutilisation de la connaissance du domaine, de rendre explicites les hypothèses sur le domaine et de séparer la connaissance du domaine de la connaissance opérationnelle.

La notion d'ontologie appliquée à notre projet correspond à la définition de Van Heist (1997) donnée au chapitre 2. Notre agent d'acquisition de connaissances utilisera l'ontologie dont le contenu correspond aux primitives de modélisation.

3.3.3 Représentation des connaissances

Nous représenterons les connaissances avec des patrons exprimés à l'aide de schémas. Les schémas permettent de représenter tous les types de connaissances utiles qui seront manipulées dans la future application.

De plus, nous garderons une séparation nette entre les types de connaissances. Par exemple, les connaissances du domaine seront séparées des connaissances opérationnelles. De même, les bases de connaissances seront séparées des méthodes et des stratégies de contrôle.

Les connaissances du domaine

Les connaissances du domaine sont les connaissances qui se rapportent au domaine d'application (exemple : exemple résolus, entête, énoncé, résolution, etc.). Elles seront représentées comme un ensemble de concepts structurés sous forme d'une hiérarchie de classes.

Les connaissances opérationnelles

Les connaissances opérationnelles sont les méta-connaissances ou encore les connaissances qui permettent d'opérer sur les connaissances du domaine. Ces méta-connaissances peuvent être des relations entre les classes, des contraintes ou des restrictions qu'on peut avoir sur les connaissances du domaine (par exemple, les attributs de classes) ou bien des méthodes de résolution de problèmes qui définissent les rôles de la connaissance.

3.3.4 Méthodes d'analyse

Diverses méthodes ont été développées, fournissant des cadres pour organiser les connaissances de manière à faciliter le processus de développement des SBC.

Citons comme exemple : les tâches génériques (Chandrasekaran 1986), l'approche des composantes d'expertise (Steels 1992), la méthode CommonKADS (Schreiber *et al*. 1993) et l'approche PROTÉGÉ (Musen *et al*. 1995).

Les tâches génériques

Les tâches génériques couvrent une grande classe de problèmes à travers différents domaines. Des connaissances spécifiques à la tâche sont rattachées à chaque tâche générique. De la même façon, un langage de haut niveau est défini pour chaque tâche générique au moyen de la réalisation d'un outil.

Les tâches génériques définissent des blocs élémentaires permettant, par composition, la conception de systèmes complexes. Ceci a conduit à la réalisation d'une boîte à outils qu'il est possible de voir comme une sorte d'atelier de conception de SBC.

L'approche des composantes d'expertise

L'approche des composantes d'expertise, couvre la totalité des phases de la réalisation d'un système. Dans un premier temps, il s'agit d'énumérer et d'accumuler les informations et connaissances sur le domaine qui vont par la suite être structurées et organisées au travers d'un modèle conceptuel.

Le modèle conceptuel fournit une description de l'expertise selon les quatre couches du domaine, des inférences, des tâches et des stratégies. Il est possible de construire le modèle conceptuel directement ou bien à partir d'un modèle générique que l'on peut raffiner.

Ensuite, à partir du modèle conceptuel, une succession de transformations permet de spécifier les détails d'implémentation (modèle de conception).

L'approche PROTÉGÉ

Protégé est un outil qui aide à construire des outils d'acquisition de connaissances pour des SBC dans des domaines spécifiques. La génération de l'outil d'acquisition de connaissances se fait à partir d'un ensemble de concepts structurés. L'ingénieur de la connaissance fournit la structure de concepts pour créer l'outil et l'expert du domaine utilise l'outil pour éditer la base de connaissances.

La connaissance est acquise par niveaux, et la connaissance acquise à un niveau n sert de méta-connaissances pour le niveau suivant, n+1.

3.4 Sélection d'une méthode : CommonKADS

Pour mener notre analyse, nous avons retenu l'approche CommonKADS, car elle propose un cadre de modélisation au niveau des connaissances. De plus CommonKADS est devenue un standard dans le développement de SBC. Dans l'approche CommonKADS, la modélisation se fait grâce à la construction d'une succession de modèles. Ces modèles correspondent à plusieurs perspectives de la résolution du problème dans son environnement organisationnel et son contexte d'application. Ainsi, il nous sera possible de situer notre objectif dans le contexte adéquat pour l'application, pour une modélisation optimale.

La figure 3.1 présente la suite des modèles de la méthode CommonKADS.

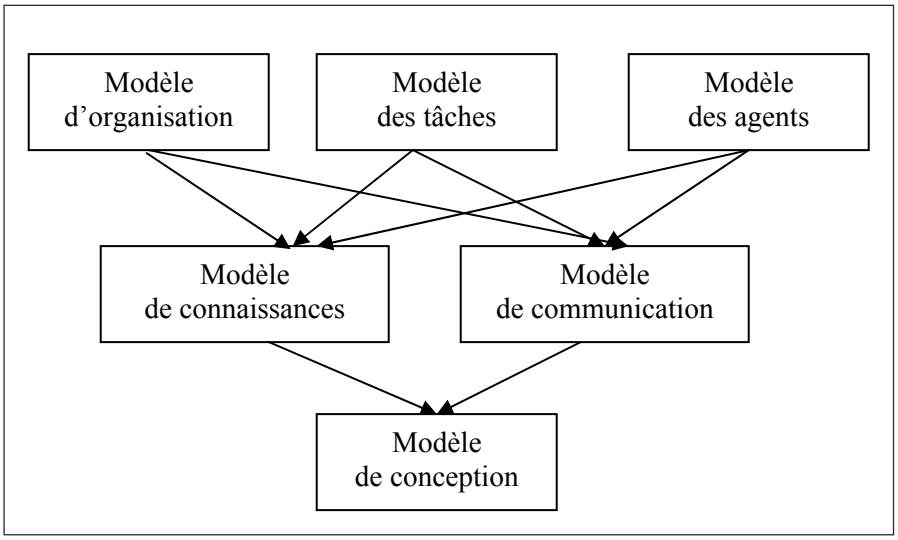


Figure 3.1 : Les modèles de CommonKADS (traduit de Schreiber *et al*. 2000)

La méthode CommonKADS comme toute autre méthode de développement logiciel consiste en un certain nombre d'éléments qu'il est possible de représenter graphiquement sous forme pyramidale. La figure 3.2 illustre la pyramide de la méthode CommonKADS.

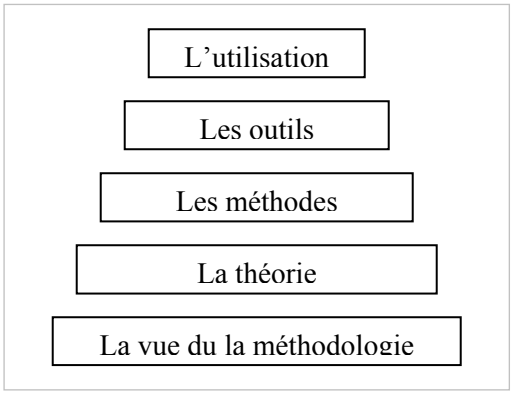


Figure 3.2 : Pyramide méthodologique (traduit de Schreiber *et al*. 2000)

Les éléments dans cette pyramide sont la vue de la méthodologie, les concepts théoriques, les méthodes et techniques qui utilisent la méthodologie et les outils d'implémentation permettant d'appliquer ces méthodes et l'étude de cas ou l'application de la méthodologie à des projets.

3.5 Conclusion

Dans ce chapitre, nous avons précisé la problématique identifiée sur l'acquisition des connaissances, plus précisément dans le cadre des EIAH. De plus, nous avons présenté le cadre méthodologique qui servira à mener notre analyse avec les différents éléments qui la composent et les méthodes retenues.

Le chapitre suivant est consacré à la présentation des éléments d'analyse pertinents, définis dans la méthode d'analyse proposée pour réaliser notre travail. Nous y définissons les différents modèles élaborés dans le cadre de la méthode CommonKADS pour l'application que nous voulons construire.

4 Chapitre 4 : Analyse

Ayant défini notre cadre méthodologique au chapitre précédent, nous allons maintenant définir les éléments pertinents de la méthode d'analyse proposée pour réaliser notre travail.

Ce chapitre présente donc en détail les différents modèles que nous avons élaborés dans le cadre de la méthode CommonKADS pour l'application que nous voulons construire. Certains modèles seront plus détaillés que d'autres en fonction de leur importance dans notre projet.

La phase d'analyse est cruciale pour le développement de systèmes d'ingénierie des connaissances (Schreiber *et al*, 2000). En suivant la méthode CommonKADS pour le développement de notre application logicielle, nous nous assurons que cette application répondra bien aux besoins de l'usage qui en sera fait.

Nous décrivons maintenant successivement les modèles suggérés par CommonKADS : modèle d'organisation, modèle des tâches, modèle des agents, modèle des connaissances et modèle de communication. Pour chaque modèle, nous rappelons sa définition telle que proposée par CommonKADS, puis nous précisons notre modèle instancié en nous basant sur cette définition.

4.1 Le modèle d'organisation

Un des objectifs du modèle d'organisation est de soutenir l'analyse des différentes composantes de l'organisation dans le but de faire ressortir des problèmes et des opportunités pour un SBC et d'établir sa faisabilité pour évaluer les impacts du SBC sur l'organisation (Schreiber *et al*, 2000).

Nous présentons dans un premier temps quelques problèmes que nous avons identifiés dans l'organisation avec leurs solutions, ainsi que le contexte organisationnel. Ensuite nous faisons une étude de faisabilité technique et économique en vue de retenir la solution la plus prometteuse.

Les problèmes identifiés pourront être vus comme des fonctions organisationnelles pour lesquelles il est possible de concevoir un SBC. Le modèle d'organisation permet d'évaluer l'impact qu'aura le futur SBC sur ces fonctions et de déterminer sa faisabilité et par la suite de présenter un aperçu du résultat de l'application dans l'organisation.

4.1.1 Le contexte organisationnel

Le contexte organisationnel est celui de l'Université Laval et plus particulièrement celui du département d'informatique et de génie logiciel dont l'une des missions est l'enseignement de l'informatique. Un des cours donnés par le département est le cours IFT-17586 Intelligence Artificielle. Dans ce contexte, l'Équipe de Recherche en Ingénierie des ConnAissancEs (ERICAE) contribue à améliorer et faciliter la réalisation de cette mission pour le cours cité, car plusieurs membres sont impliqués dans l'enseignement du cours.

Dans ce cadre, l'équipe ERICAE a développé un EIAH à base d'exemples, appelé SPHINX qui permet aux étudiants inscrits au cours d'intelligence artificielle dans leur apprentissage de la matière du cours. Le contexte organisationnel renvoie aux facteurs dans l'environnement où opère l'organisation. Ceux-ci peuvent influer sur les impacts du SBC et sur l'organisation elle-même.

D'après Schreiber *et al*. (2002), les principaux facteurs organisationnels à prendre en compte sont :

- la mission : Une des missions serait, dans la mesure où cela est possible, d'avoir de la transparence aux niveaux des procédures de réalisation des tâches ou activités d'apprentissage pour les utilisateurs (professeurs, étudiants, etc.) et aussi que ces procédures soient facilement utilisables et extensibles.

- les acteurs : Parmi les acteurs, on retrouve les experts de la connaissance (les professeurs, les assistants qui vont aider les professeurs dans leur tâche d'enseignement), les ingénieurs de la connaissance (les analystes, les concepteurs, les développeurs de l'application) et les utilisateurs finaux (les étudiants pour l'apprentissage de la matière et les professeurs pour la présentation des sujets

d'apprentissage). Pour les besoins de l'application, nous tiendrons compte des besoins des apprenants et de l'environnement logiciel.

- les ressources : Elles renvoient à l'équipement et au matériel informatique dont dispose l'organisation (ordinateurs, imprimantes, logiciels, etc.), aux connaissances et aptitudes en programmation, en modélisation ainsi qu'à son système d'information.

- les connaissances : Elles désignent l'ensemble des connaissances-clés manipulées tout au long du processus de développement du système allant de la modélisation à la conception, voire la maintenance.

4.1.2 Identification des problèmes et solutions

L'analyse de l'organisation fait ressortir les problèmes organisationnels, actuels ou anticipés, identifiés qui nécessitent d'être résolus plus ou moins rapidement ainsi que des solutions possibles à ces problèmes (Schreiber *et al*. 2000).

Rappelons que l'application que nous voulons développer devra aider à l'acquisition de connaissances dans le but de créer des exemples pour l'EIAH SPHINX, qui est un environnement éducationnel. Dans ce contexte, la création d'exemples prend beaucoup de temps du fait de la différence des exemples de par leur contenu ou leur structure. Dans l'environnement SPHINX, l'acquisition des connaissances nécessaires à la création d'un exemple présente certains inconvénients. Elle se fait présentement de façon manuelle et nécessite de passer à travers toutes les étapes de sa création même lorsqu'elles sont identiques d'un exemple à l'autre. À l'heure actuelle, le temps pour créer un exemple peut varier de une à plusieurs heures. De plus il n'y a aucune structure mise en place pour faciliter, voire accélérer le processus de création des exemples.

Ces inconvénients rendent difficile la création rapide d'exemples pour un usage urgent et même courant. Un des objectifs de l'application proposée est donc de faciliter l'acquisition des connaissances lors de la création d'exemples dans l'environnement SPHINX, et ceci en développant un système informatique d'aide à la création d'exemples.

Les problèmes identifiés, et pour lesquels l'organisation porte un intérêt, concernent l'amélioration du processus de création de contenu pédagogique. L'identification des solutions possibles a trait aux solutions qui seront élaborées pour résoudre les problèmes ou pour satisfaire les besoins actuels. Parmi ces solutions, on retrouve l'opportunité de fournir à l'organisation ou encore à l'environnement éducationnel un outil pour la création des exemples, d'automatiser le processus de création, d'offrir une formation pour familiariser les utilisateurs à l'emploi de l'outil, ainsi que l'opportunité de spécifier l'ensemble des fonctions qui peuvent l'être dans l'organisation.

4.1.3 Étude de faisabilité

L'étude de faisabilité porte sur les points économique et technique. Cette étude permet d'établir la faisabilité du projet dans son ensemble.

4.1.3.1 Étude de faisabilité économique

L'étude de faisabilité économique permet de dégager s'il y a un impact économique (Schreiber *et al*, 2000). Par exemple, est-ce que la solution sera commercialisée ou non ? Quelle valeur ajoutée apporte la solution dans l'organisation ? Est-ce qu'il y a des changements dans l'organisation ? Quels sont les coûts engendrés par la réalisation de cette solution ? Y a-t-il une ou des solutions alternatives ?

Dans le cadre de notre étude, l'outil que nous avons développé ne sera pas commercialisé, du moins pas dans un premier temps. Il le sera éventuellement plus tard s'il correspond aux critères de qualité jugés acceptables pour sa commercialisation. Actuellement, l'outil sera utilisé pour répondre aux besoins pour lesquels il a été développé, c'est-à-dire aider les professeurs dans leurs tâches d'acquisition des connaissances. Comme valeur ajoutée, les professeurs et leurs assistants disposeront d'un outil d'aide à la création d'exemples qui facilitera leur travail donc améliorera la qualité de celui-ci tout en restant convivial. L'application n'apporte aucun changement dans l'organisation au niveau de sa structure. Concernant les coûts engendrés par le développement de l'application, ceux-ci sont supportés et ne sont pas excessifs, étant donné que le développement de cette application entre dans le cadre de notre projet de maîtrise et que nous bénéficions d'une subvention

pour la réaliser. Comme solution alternative, nous aurions pu considérer les plates-formes éducationnelles présentement disponibles sur le marché. Toutefois, avec les chercheurs de l'équipe ERICAE nous n'avons pas retenu ces plates-formes parce qu'elles n'offrent pas la possibilité d'un cadre éducationnel permettant notamment les auto-explications (ICTE 2002, Espagne). Finalement, notre application sera supportée par la plate-forme de l'environnement SPHINX.

4.1.3.2 Étude de faisabilité technique

L'étude de faisabilité technique fait ressortir les caractéristiques techniques atteignables dans la réalisation de la solution, en termes de complexité des processus et des connaissances manipulées, de temps, de qualité, de validation et de performance (Schreiber *et al*. 2000).

Les connaissances que l'on manipule sont peu complexes, et les processus le sont encore moins. Certains peuvent être plus longs que d'autres parce que le choix est fait de garder un niveau de complexité le plus bas possible. Cependant, ceci ne constitue pas un frein majeur pour la performance de l'outil en termes d'exécution. La qualité de l'outil est acceptable. Les ressources logicielles et humaines nécessaires à sa réalisation sont disponibles.

La validation de l'outil devra se faire de façon conjointe entre les développeurs et les utilisateurs finaux afin de valider son comportement et s'assurer qu'il réponde bien aux besoins de l'usage qui en sera fait.

Pour résumer l'étude de faisabilité du projet, nous dirons que les ressources nécessaires pour réaliser le projet étaient disponibles en termes de temps, de personnes, de budget, d'équipement, de connaissances et de compétences. Les nouvelles connaissances et compétences acquises au cours du projet s'ajoutent à celles déjà disponibles. Les acteurs et les participants sont dynamiques, engagés et motivés et les résultats attendus sont réalisables.

4.2 Le modèle des tâches

Le modèle des tâches permet de décrire les tâches et leurs répartitions, les entrées et les sorties, les conditions préalables et les critères d'exécution, en tant que ressources et compétences nécessaires pour le système (Schreiber *et al*, 2000). Le modèle des tâches sert donc à décrire les tâches qui seront exécutées durant la phase d'enseignement par le futur système lors de la réalisation d'une fonction précise de l'organisation.

Une tâche se décrit par ses entrées requises, ses sorties désirées (atteindre un but à valeur ajoutée), les ressources qu'elle utilise, les connaissances et les compétences qu'elle requiert pour être exécutée et celles qu'elle génère. La description d'une tâche est indépendante des entités qui les réalisent. En effet à chaque tâche, il sera possible de rattacher une entité désignée par le terme agent, permettant de la réaliser, comme nous le verrons dans la section 4.3.

Le processus d'acquisition de connaissances dans notre contexte comporte quatre tâches principales qui s'inscrivent dans la tâche globale de gestion d'exemples. Ces tâches sont les suivantes : création, modification, suppression et sauvegarde d'un exemple. La figure 4.1 présente la décomposition de ces tâches.

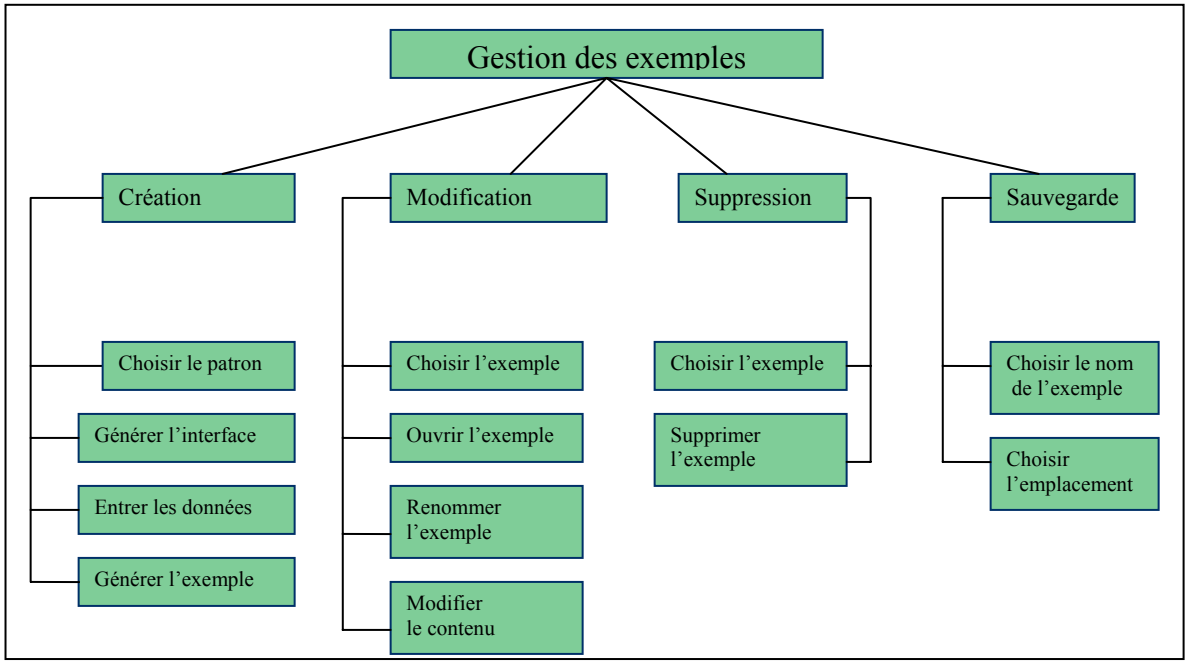


Figure 4.1 : Le modèle des tâches pour la gestion des exemples

Chacune de ces tâches se décompose en sous-tâches. Ainsi la tâche de création d'un exemple comprend une première sous-tâche « Choisir le patron », tâche de sélection d'un patron dans une base de patrons, soit celui qui correspond à la structure de l'exemple que l'utilisateur souhaite ajouter dans la base et à partir duquel l'utilisateur veut créer l'exemple. Ensuite une deuxième sous-tâche, « Générer l'interface», consiste à générer l'interface d'acquisition de l'exemple à partir du patron. La sous-tâche qui suit, « Entrer les données», consiste à saisir les données d'entrée pour la création de l'exemple. Enfin, la dernière sous-tâche, « Générer l'exemple », consiste à générer l'exemple à l'aide des données entrées.

Notons qu'une fois l'exemple généré, l'utilisateur a la possibilité de le sauvegarder s'il veut le conserver. La tâche de sauvegarde d'un exemple permet de conserver notre exemple. Cette tâche se décompose en deux sous-tâches : la tâche «Choisir le nom de l'exemple», qui consiste à nommer notre exemple et la tâche « Choisir l'emplacement», qui consiste à répertorier l'exemple. Puis, le système enregistre physiquement les données de l'exemple à l'emplacement identifié sur le disque.

La tâche de modification d'un exemple permet d'éditer le contenu des exemples. Elle se décompose en quatre sous-tâches : la tâche « Choisir l'exemple» qui consiste à identifier l'exemple que nous voulons éditer ; la tâche « Ouvrir l'exemple» qui permet de charger l'exemple et son contenu en mode édition en remplissant les champs de l'interface d'acquisition par les valeurs correspondantes contenues dans le fichier de l'exemple ; la tâche « Renommer l'exemple» et la tâche « Modifier le contenu» qui permet de changer le contenu de l'exemple.

La tâche de suppression d'un exemple permet d'éliminer un exemple. Elle se décompose en deux sous-tâches : la tâche « Choisir l'exemple» qui consiste à identifier l'exemple que l'utilisateur veut effacer et la tâche « Supprimer l'exemple» qui consiste à effacer l'exemple en question. La suppression de l'exemple se fait autant au niveau du volet d'accès des exemples que physiquement. L'exemple est supprimé de son emplacement physique sur le disque.

La figure 4.2 décrit les activités qui sont réalisées au sein de la fonction de création d'un exemple en précisant le niveau de la tâche, celui de la méthode de la tâche et celui des inférences.

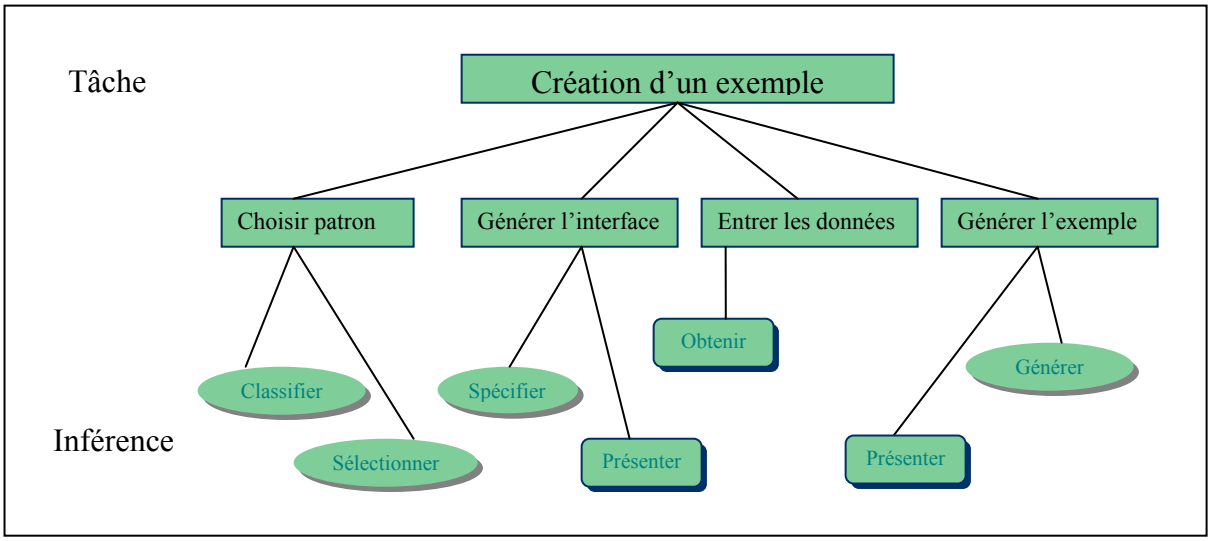


Figure 4.2 : Le diagramme des tâches pour la création d'un exemple

La méthode de la tâche représente la façon dont la tâche est effectuée. Les inférences sont éléments de raisonnement utilisées pour la méthode pour résoudre la tâche. Une même inférence peut être utilisée par plusieurs méthodes et une même méthode peut utiliser plusieurs inférences. Il est possible, de voir à travers la manière dont les inférences sont appliquées, une stratégie d'exécution de la tâche réalisée (Schreiber *et al*. 2000).

Les figures 4.3, 4.4 et 4.5 décrivent respectivement les activités qui sont réalisées au sein des fonctions de modification, de suppression et de sauvegarde d'un exemple.

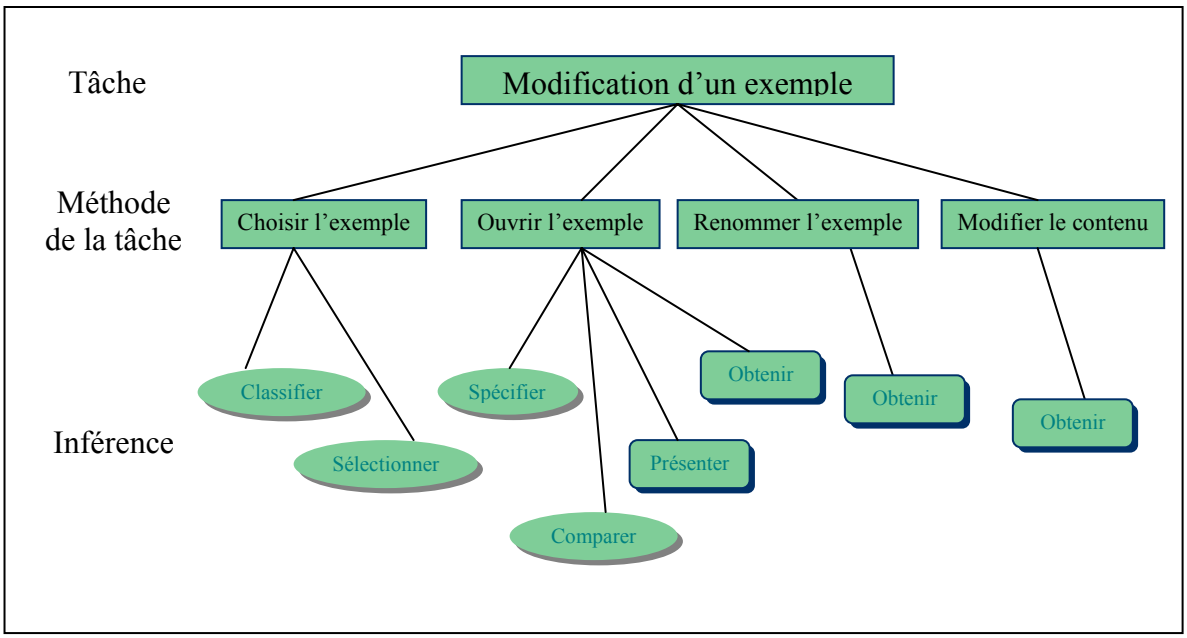


Figure 4.3 : Le diagramme des tâches pour la modification d'un exemple.

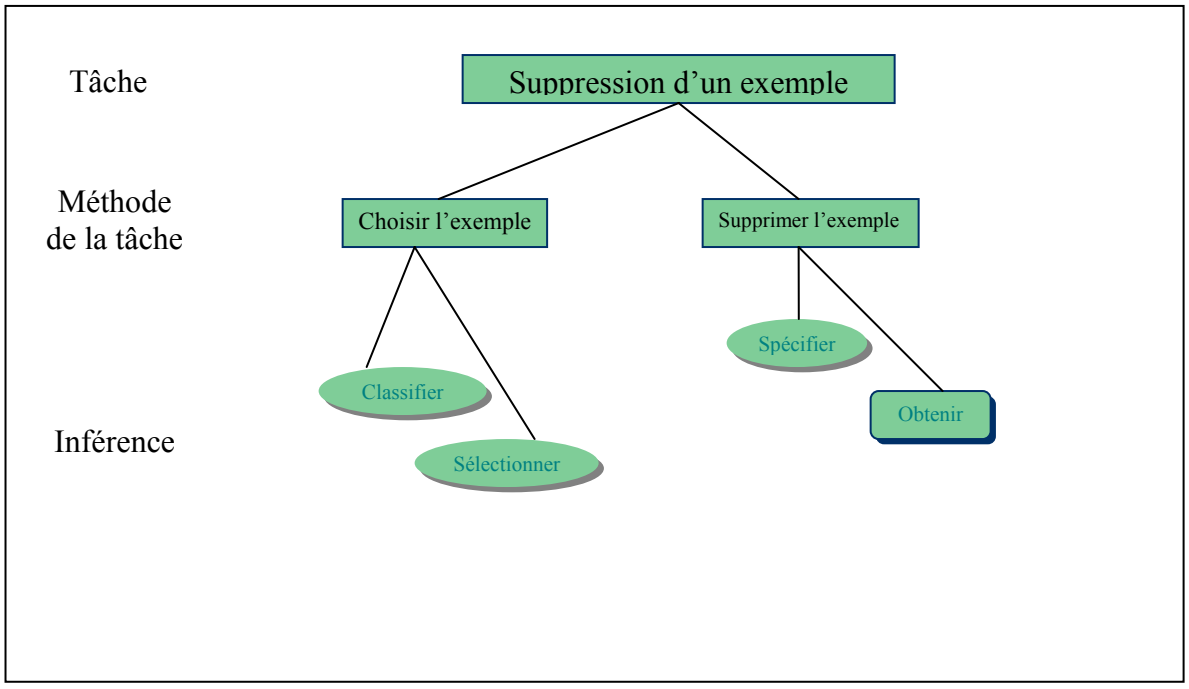


Figure 4.4 : Le diagramme des tâches pour la suppression d'un exemple.

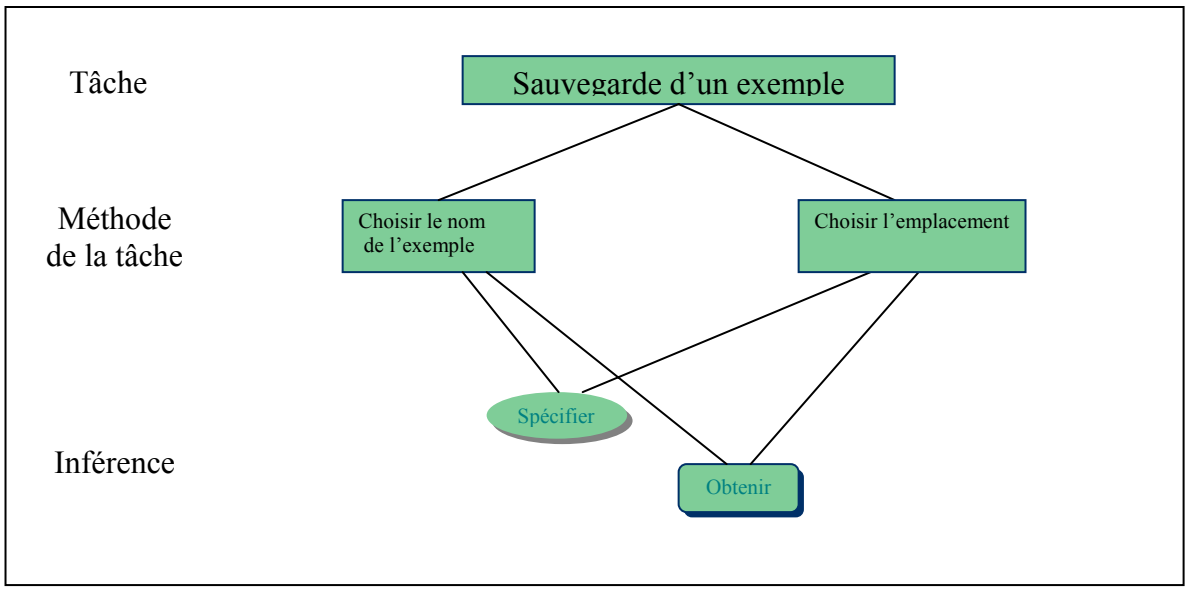


Figure 4.5 : Le diagramme des tâches pour la sauvegarde d'un exemple.

4.3 Le modèle des agents

Une tâche est exécutée par un agent qui peut être un humain ou un système d'information, ou encore n'importe quelle autre entité capable de l'effectuer. Le modèle des agents décrit les caractéristiques des agents, en particulier leurs compétences, leur autorité pour agir et les contraintes à cet égard. En outre, il énumère les liaisons entre les agents effectuant une tâche (Schreiber *et al*. 2000).

Parmi les agents dans notre modèle, on retrouve des agents humains qui jouent soit le rôle des utilisateurs, soit celui des ingénieurs de la connaissance, ou encore celui des experts de la connaissance et enfin celui des agents interface. Ces agents présentent les caractéristiques requises en termes de compétences, pour accomplir les fonctions, pendant la réalisation d'une tâche, tout en tenant compte des contraintes et des liaisons entres les différents agents.

4.4 Le modèle des connaissances

Le modèle des connaissances proposé dans la méthodologie CommonKADS permet de préciser les types, les structures et les rôles de la connaissance. Il présente le raisonnement anticipé du SBC. Il permet aussi d'avoir une description indépendante de la plate-forme du

rôle que jouent les différents éléments de la connaissance dans la résolution des problèmes, d'une manière qui est compréhensible par des humains (Schreiber *et al*. 2000).

Le modèle des connaissances permet d'avoir une vue d'ensemble pour trois catégories de connaissances : les connaissances du domaine, les connaissances de la tâche et les connaissances d'inférence. Il permet donc d'avoir une vue sur le domaine, la tâche et l'inférence.

4.4.1 La vue du domaine

La vue du domaine permet d'identifier les concepts du domaine d'application : exemple, entête, contenu, résolution, explication, etc. Il est possible de représenter sous forme graphique et textuelle ce domaine (schéma du domaine).

La figure 4.6 présente les concepts identifiés et les relations entre ces concepts. Les classes 'EXEMPLE', 'ENTÊTE', ' CONTENU', 'CONCLUSION', 'EXPLICATION', 'ÉNONCÉ', 'ÉTAPE DE RÉSOLUTION', 'RESSOURCE', 'TYPE', 'TEXTE', 'IMAGE', 'AUDIO', et 'VIDEO' représentent les concepts du domaine. Les relations représentent les propriétés des classes. Par exemple, un EXEMPLE a une ENTÊTE, un CONTENU, des EXPLICATIONs et des RESSOURCEs. Pour la classe 'CONTENU' par exemple, les propriétés sont l'ÉNONCÉ du problème, les ÉTAPEs DE RÉSOLUTION du problème posé dans l'énoncé et la CONCLUSION sur la résolution du problème.

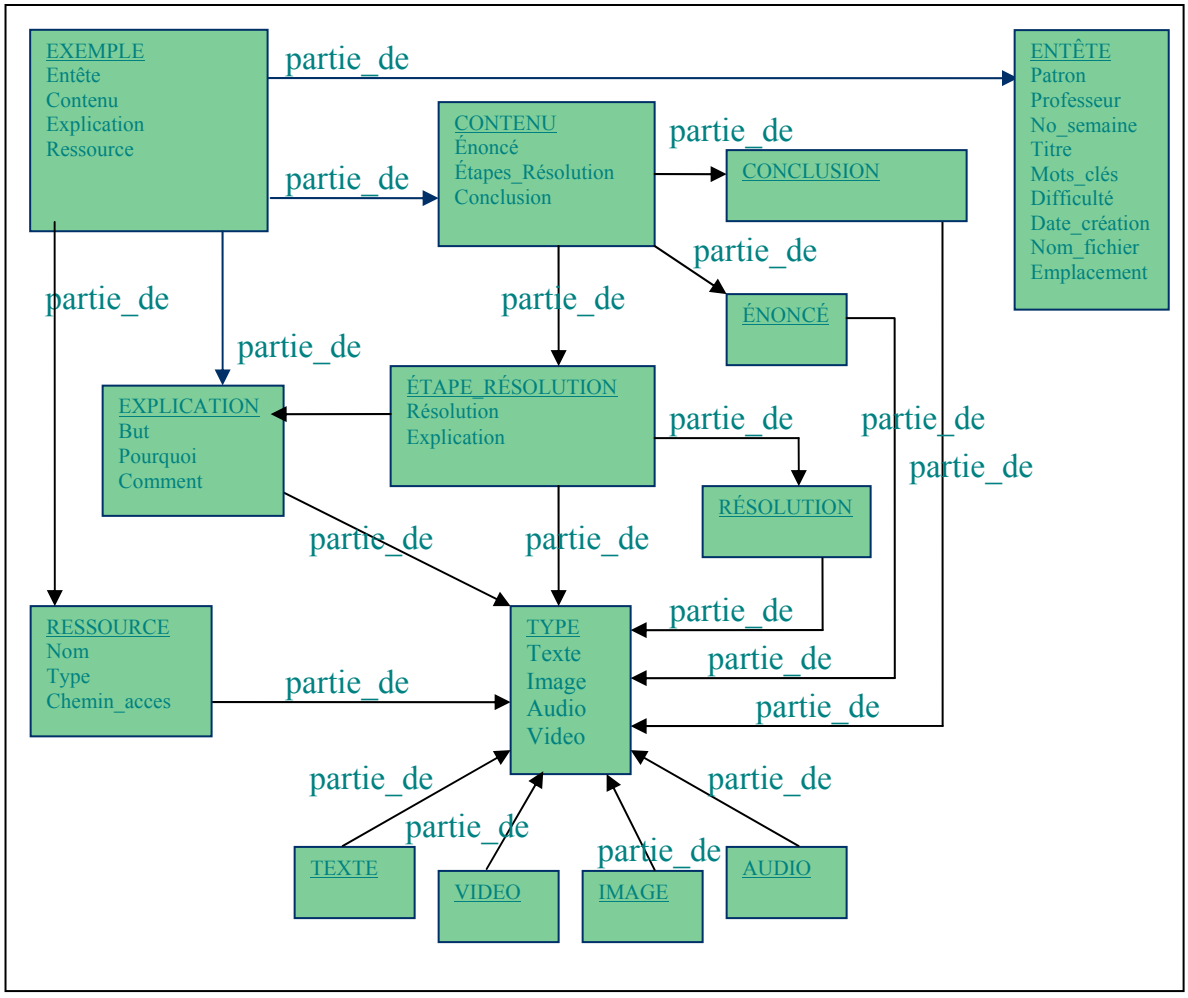


Figure 4.6 : Le modèle de connaissance de l'application.

4.4.2 La vue de la tâche

La vue de la tâche permet d'identifier les buts de l'application, en décomposant le problème à résoudre en tâches permettant de le résoudre. Les connaissances sur la tâche décrivent les buts et les stratégies employées pour atteindre ces buts. On distingue deux types de connaissances ayant un rôle prédominant : la tâche qui définit les entrées et les sorties et la méthode de la tâche qui définit comment la tâche peut être réalisée par une décomposition fonctionnelle, comment contrôler l'exécution des fonctions.

Considérant la tâche de création d'un exemple, celle-ci peut être décomposée en différentes sous-tâches. Notons que cette tâche s'inscrit elle-même dans le cadre d'une tâche plus

globale de gestion d'exemples. Le tableau qui suit présente les sous-tâches permettant d'effectuer la tâche de création d'un exemple.

Tâche : Création d'un exemple		
Sous-tâche	**Type**	**Ressources**
1. Identification du patron représentant la structure de l'exemple	classification	liste des patrons
2. Évaluation du patron	évaluation	structure de l'exemple
3. Choix du patron		
4. Génération de l'interface d'acquisition à partir du patron		
5. Saisie des données		
6. Génération de l'exemple à partir des données entrées		

Tableau 4.1 : Décomposition de la tâche de création d'un exemple

La tâche 1, Identification du patron représentant la structure de l'exemple, permet de spécifier le patron qui représente la structure de l'exemple que l'on veut créer. La tâche 2, Évaluation du patron, permet décider si le patron représente bien la structure de l'exemple. La tâche 3, Choix du patron, confirme la sélection faite du patron sélectionné. La tâche 4, Génération de l'interface d'acquisition à partir du patron, permet de produire l'interface d'acquisition correspondante à partir du patron choisi. La tâche 5, Saisie des données, permet de capter les données qui vont servir à créer l'exemple. La tâche 6, Génération de l'exemple à partir des données entrées, permet de créer l'exemple proprement dit à partir des données entrées, acquises via l'interface d'acquisition.

Les connaissances dont on dispose constituent la base de patrons. Il est possible de créer de nouveaux patrons pour enrichir cette base si aucun des patrons contenus dans la base de patrons ne représente la structure de l'exemple que l'on veut créer.

4.4.3 La vue de l'inférence

Les connaissances d'inférence établissent les étapes de base pour les inférences : poser une hypothèse (liens entre concepts) et vérifier l'inférence (identifier les tests permettant d'établir que l'ensemble des causes d'un problème observé est bien le facteur qui les engendre) (Schreiber *et al*, 2000).

La figure 4.7 présente la structure d'inférence pour la tâche de classification d'un patron. Cette structure d'inférence est tirée de la méthodologie CommonKADS. Nous l'avons adaptée pour les besoins de notre application.

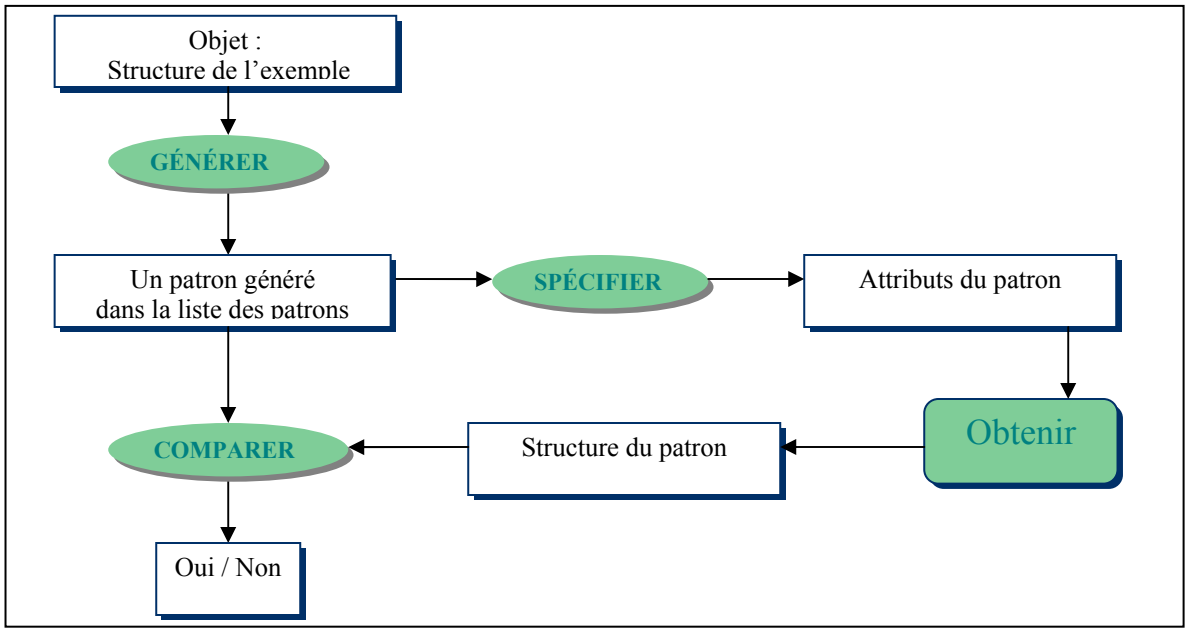


Figure 4.7 : Structure d'inférence pour la tâche de classification d'un patron.

La tâche de classification vise à associer un objet à la classe à laquelle il appartient à partir de sa description. Ainsi à partir des attributs de la structure anticipée de notre exemple, nous allons générer un patron parmi la liste des patrons disponibles.

Pour ce patron choisi, nous allons spécifier les attributs observables et à l'aide d'une fonction de transfert, nous allons pouvoir obtenir les valeurs réelles des attributs et les comparer à celles observées pour déterminer s'il y a correspondance ou pas selon la concordance des valeurs comparées.

La fonction 'Obtenir' (traduit de l'anglais « Obtain ») est une fonction de transfert standard définie dans la méthode CommonKADS. Les fonctions de transfert assurent la communication entre notre système et d'autres systèmes ou d'autres agents. Elles permettent le transfert d'une information entre l'agent responsable de la tâche et le monde extérieur (un autre système). La fonction de transfert 'Obtenir', dans le processus de raisonnement, a pour rôle d'obtenir des informations supplémentaires pour poser un diagnostic. Elle agit comme une boîte noire du modèle de connaissances dont on connaît les entrées et les sorties.

4.4.4 Base de connaissances (BC)

Notre schéma du domaine a permis de définir les différents types de connaissances (concept, relation, règle). La base de connaissances va contenir les instances de ces types. Il y a séparation nette entre la BC et le schéma du domaine. Ainsi lors de l'acquisition de connaissances, il faudra définir les types de connaissances et acquérir les connaissances par type.

Le modèle des connaissances présente les connaissances de base et le raisonnement anticipé du SBC. À ce dernier, nous ajouterons un modèle de communication pour exprimer les besoins par rapport aux interfaces avec d'autres agents. Le modèle des connaissances, ajouté au modèle de communication, donne les spécifications pour le modèle de conception. Nous définirons ce dernier modèle au chapitre 5 à l'étape de conception. Le modèle des connaissances a pour rôle de clarifier la structure d'une tâche requérant beaucoup de connaissances (knowledge-intensive task) et de définir le vocabulaire du domaine et des inférences.

4.5 Le modèle de communication

Le modèle de communication permet de modéliser de façon conceptuelle et indépendante de la plate-forme les interactions entres plusieurs agents impliqués dans une tâche (Schreiber *et al*. 2000).

Dans le contexte d'un SBC qui intègre notre application, cette dernière peut être vue comme un agent, elle appartient à la catégorie des agents interfaces. La fonction d'un agent interface consiste à établir un lien entre l'utilisateur et l'application pendant son exécution. L'interface de notre application est de type GUI (interface utilisateur graphique). Elle offre plus de convivialité que les interfaces de type texte.

La gestion de notre interface utilise la programmation par objet et par évènement. Chaque élément graphique d'une interface se représente par un objet. Lors de l'exécution de l'application, les actions de l'utilisateur génèrent des évènements destinés aux objets. Ceux-ci réagissent selon leur comportement décrit par un ensemble de fonctionnalités ou de méthodes.

La figure suivante illustre quelques unes des communications qui peuvent exister entre l'agent humain (l'utilisateur de l'application) et l'agent interface (application en exécution), notamment les communications lors de l'activité de création d'un exemple.

Pour créer un exemple, l'utilisateur choisit le patron sur lequel il veut se baser pour créer son exemple. L'interface d'acquisition est générée par l'application à partir du patron choisi. L'utilisateur entre alors les connaissances qui vont permettre de générer l'exemple. Ce dernier est généré une fois ces connaissances acquises. L'utilisateur peut alors fournir le nom de son exemple et choisir l'emplacement où il désire le stocker.

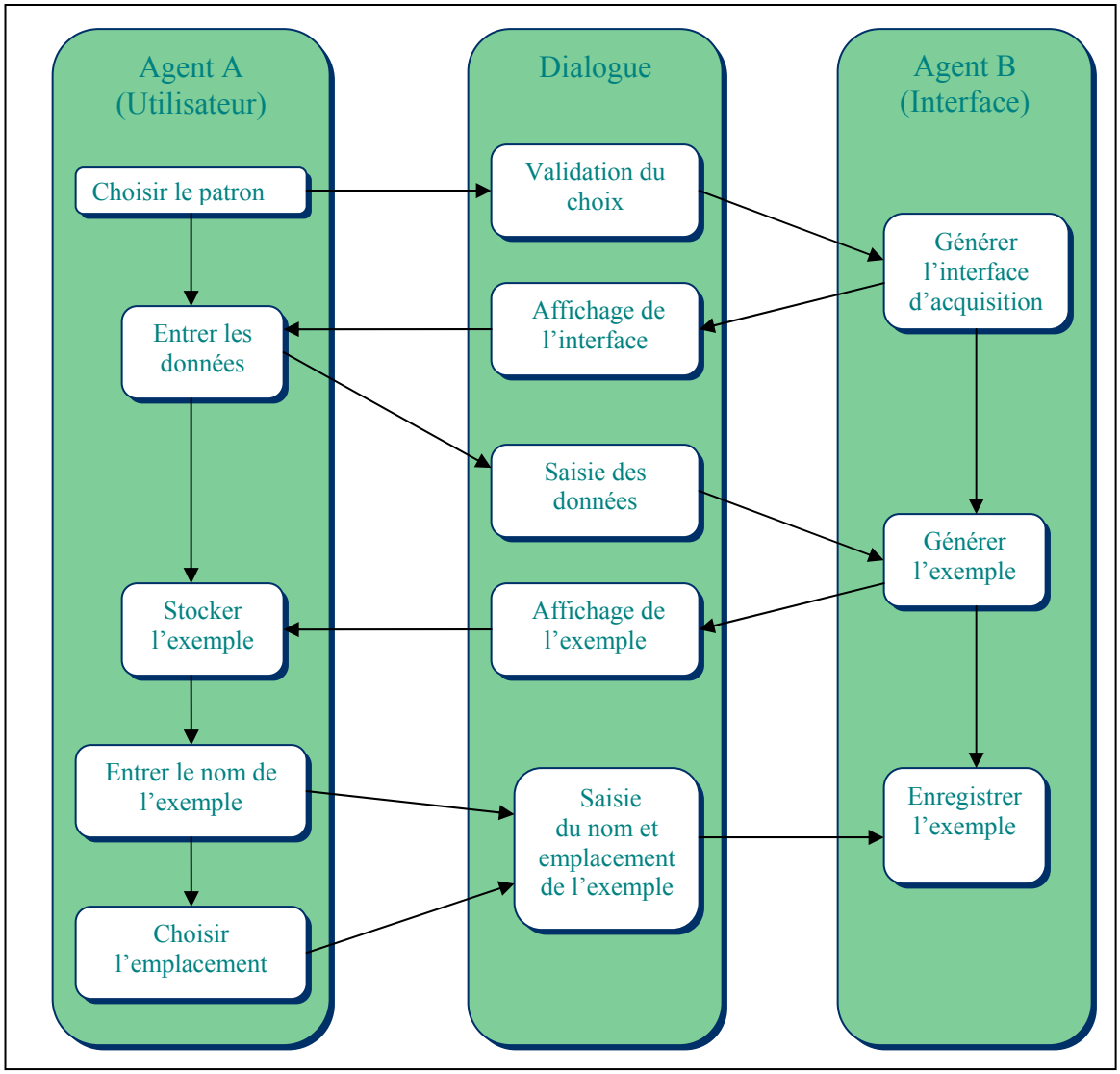


Figure 4.8 : Diagramme de communication entre les agents humain et interface lors de la création d'un exemple.

4.6 Conclusion

Dans ce chapitre nous avons présenté en détail les modèles d'organisation, des tâches, des agents, des connaissances et de communication que nous avons élaborés dans le cadre de la méthodologie CommonKADS pour l'application que nous voulons construire. Chacun des modèles a été instancié en se basant sur la définition proposée par CommonKADS. L'ensemble des modèles présentés donne les spécifications pour le modèle de conception.

Le chapitre suivant est consacré à l'élaboration du modèle de conception. Nous y discutons les aspects fonctionnels, de comportement et physiques du modèle. De plus, les résultats du prototypage sont présentés.

5 Chapitre 5 : Prototypage et conception

Dans ce chapitre, nous élaborons le modèle de conception et nous présentons les résultats du prototypage. L'élaboration du modèle de conception a pour but de décrire et de représenter en détails les diverses fonctionnalités qui seront implémentées dans le futur système. Nous discutons de différents aspects (fonctionnel, comportement et physique) qu'il est possible de distinguer dans chacun des constituants du modèle. Enfin, le prototype réalisé sera présenté avec ses interfaces et les diverses possibilités seront illustrées.

5.1 Le modèle de conception

Les modèles précédents, présentés dans la méthode CommonKADS, constituent les spécifications d'analyse requises pour le SBC. Ces spécifications sont décomposées sous différents aspects soit l'organisation, les tâches, les agents, les communications entre ceux-ci et les connaissances.

En se basant sur ces spécifications, le modèle de conception donne les spécifications techniques du système en termes d'architecture et de plate-forme d'exécution. Il spécifie aussi la manière de représenter et de construire le module logiciel ainsi que les mécanismes informatiques requis pour implanter les fonctionnalités établies dans le modèle de connaissance et le modèle de communication (Schreiber *et al*. 2000).

Le modèle de conception s'oriente davantage vers le logiciel et l'organisation interne au système. C'est comme si nous partons du domaine d'application pour nous pencher sur le système résultant.

5.1.1 Architecture fonctionnelle du système proposé

L'architecture fonctionnelle décrit la structure du système en termes de modules le composant avec la manière dont ces modules interagissent entre eux. Cette description se compose de trois éléments qui sont la décomposition du système en sous-systèmes, les interactions entre système et sous-systèmes et la décomposition des sous-systèmes en modules logiciels (Sommerville, 1995).

La conception du modèle fonctionnel se fait à partir du modèle des tâches. Rappelons que le but pratique de notre travail est d'obtenir un système graduellement extensible où l'utilisateur peut, au fur et à mesure que son expertise évolue, enrichir, voire changer les connaissances jusqu'alors acquises et utilisées par le système. Ces changements concernent à la fois la décomposition et l'ordonnancement des tâches, le comportement du système (les règles) et la description du domaine (couche du domaine).

5.1.1.1 Architecture globale du système

L'architecture globale du système est inspirée de l'architecture de référence de CommonKADS basée sur une architecture de type MVC ou Modèle-Vue-Contrôle (Goldberg, 1990) qu'il est possible de voir comme un paradigme pour la conception d'applications dans un environnement orienté objet. Nous distinguons trois systèmes principaux dans cette architecture : le modèle d'application, les vues et les structures de contrôle.

La figure 5.1 illustre l'architecture de référence de CommonKADS et ses composants.

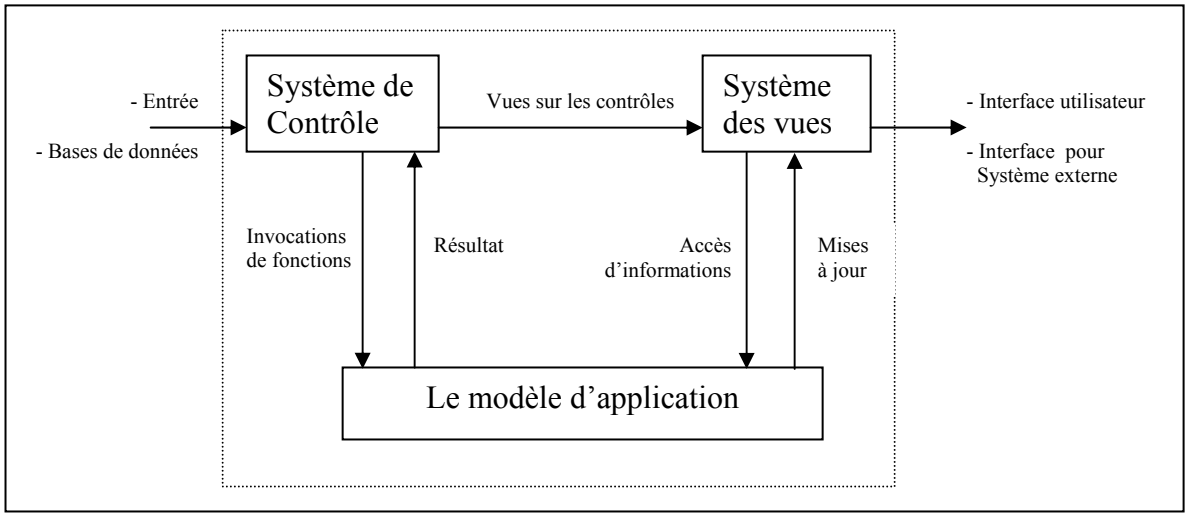


Figure 5.1 : Architecture de référence de CommonKADS (traduite de Schreiber *et al*, 2000)

Le modèle d'application spécifie les fonctionnalités de l'application et les données ou les bases de connaissances impliquées.

Le système des vues précise les vues externes sur les fonctions et les bases de connaissances de l'application. Concrètement, une vue est par exemple une vue sur le traitement d'information ou encore une vue sur la manipulation d'objets de l'application au moyen de son interface. Les vues rendent disponibles les informations de l'application aux agents externes qui peuvent être des utilisateurs humains ou encore d'autres systèmes logiciels.

Le système de contrôle représente l'unité centrale des commandes et des contrôles comme la gestion des actions et des évènements et l'activation des fonctions de l'application. Il est responsable des communications entre les sous-systèmes de l'application.

5.1.1.2 La décomposition du système en sous-systèmes

Notre système se décompose en quatre sous-systèmes. La figure 5.2 présente l'architecture du système décomposé en plusieurs modules. Nous distinguons un module responsable des interfaces qui permet à l'utilisateur d'interagir avec le système, un module intelligent qui permet d'interpréter les actions générées à travers les interfaces et d'y associer les traitements correspondants, un module de service qui assure les traitements de données dans le système, un module d'information qui assure le transfert d'informations d'un module à un autre et une base de connaissances.

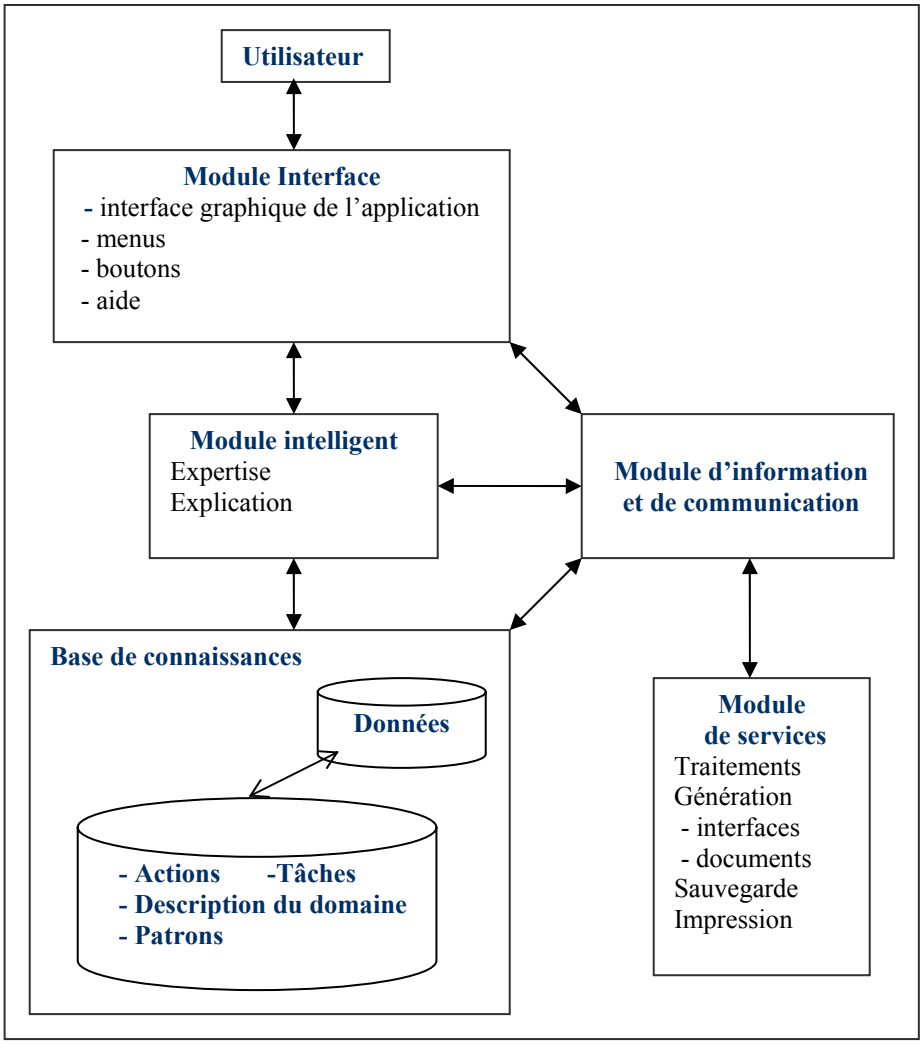


Figure 5.2 : L'architecture du système proposé

Le système possède les fonctionnalités d'un éditeur de la base de connaissances. L'ingénieur de la connaissance ainsi que l'expert utilise cette fonctionnalité selon son point de vue pour corriger et/ou enrichir le contenu de la base de connaissances du système. De plus le système est capable d'offrir une aide sur la signification des boutons et leurs actions.

5.1.1.3 Les interactions dans le système

Dans le contexte de notre application, nous retrouvons un utilisateur, un module interface, un module intelligent, un module d'information et un module de service. L'utilisateur est soit un enseignant, soit un étudiant, soit tout usager qui utilise le système. Il ne communique qu'avec le module interface. Le rôle du module interface concerne la

communication et l'assistance à l'utilisateur dans la construction et la maintenance des connaissances du système. Le module de service s'assure des procédures de traitement. Le module d'information assure la liaison entre les bases de connaissances et les autres modules.

5.1.1.4 La décomposition des sous-systèmes en modules logiciels

Nous pouvons décomposer le modèle conceptuel en divers blocs appelés « blocs fonctionnels » et qui correspondent à une unité fonctionnelle du système. L'architecture fonctionnelle donnée par ce modèle permet de représenter de façon graphique les diverses relations existant entre les blocs fonctionnels. La décomposition que nous proposons pour notre système est présentée à la figure 5.3.

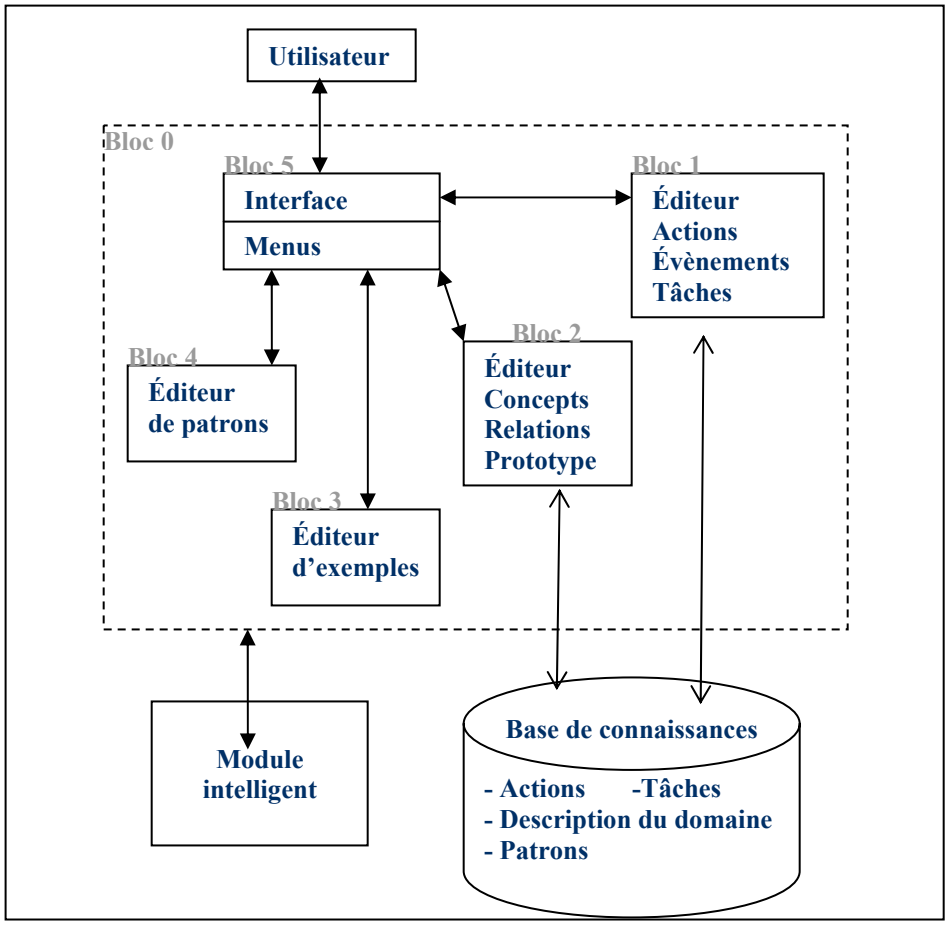


Figure 5.3 : Décomposition du système en sous-systèmes

Nous retrouvons à la figure 5.3 les sous-systèmes que nous avons présentés dans l'architecture de notre système. Chacun de ces sous-systèmes est implanté séparément par un module logiciel dédié à ses fonctionnalités et leur exécution. Ces modules sont identifiés par le terme «bloc».

À partir du menu dans l'interface principale de l'application (bloc5), l'utilisateur l'enseignant () peut soit éditer des patrons d'exemples (bloc4), soit éditer les exemples (bloc3). Ces derniers se rattachent à un domaine ou à une partie de domaine. L'enseignant a besoin de définir le domaine en question avec ses concepts et les relations entre ces concepts. Le bloc 2 lui permet alors d'éditer l'ontologie du domaine. Ensuite, l'utilisateur définit les actions, les tâches et les évènements (bloc1) devant être réalisés. Le bloc 0 est un regroupement des blocs que nous venons de présenter. Ceci facilite la communication entre les modules logiciels selon leur niveau.

5.1.2 Plate-forme d'implantation

Dans cette section, nous précisons le matériel et les outils informatiques utilisés pour implanter le système.

Le contexte de réalisation de notre système comprend plusieurs aspects. Tout d'abord, le prototype est réalisé dans un environnement orienté objet utilisant le langage de programmation Java qui permet de développer des interfaces conviviales et flexibles. Les blocs fonctionnels qui présentent les interfaces sont réalisés dans cet environnement. Ces interfaces appartiennent au type d'interfaces utilisateurs graphiques (GUI). Leurs fonctions consistent à établir un lien entre l'usager et l'application en exécution. Les détails des interfaces ont été précisés dans la section 4.5 du chapitre 4, dédiée au modèle de communication.

La programmation logique est implantée au moyen de la technologie XML. Le résultat d'un exemple créé avec notre application est un fichier XML et que cet exemple est créé à partir d'un patron représentant sa structure. Le patron est également stocké dans un fichier XML. Nous utilisons la technologie XML car elle permet d'inclure une logique de

traitement des informations et d'automatiser les traitements, et ce d'une manière indépendante de la plate-forme utilisée.

Les patrons sont définis à partir du langage XML et de classes Java en appliquant certaines restrictions imposées par la communication avec XML. Au fichier XML s'ajoute un fichier XSD qui contient la définition de données contenues dans le fichier XML d'un point de vue sémantique. Les classes Java utilisent ce fichier XSD pour interpréter les données contenues dans le fichier XML. Ceci rend plus facile la liaison de ces données à une interface.

Les composantes de l'environnement orienté objet Java permettent d'obtenir les informations contenues dans la base de données. Ces informations sont alors chargées à l'intérieur d'objets Java qui correspondent aux structures décrites pour définir le modèle des tâches.

Les classes Java sont manipulées dans tous les traitements de l'application : le chargement de la base de patrons, la génération de l'interface d'acquisition, la création de l'exemple à partir du patron, etc. Pour satisfaire les besoins en stockage d'informations, nous utilisons un espace serveur dédié à l'application. Les bases de connaissances sont créées localement sur la machine qui exécute l'application. Elles seront par la suite transférées sur l'espace serveur. La base de connaissances localisée physiquement sur le serveur est ainsi indépendante du système. La communication entre l'application et le serveur se fait au moyen d'une connexion sécurisée qui utilise les protocoles SSL.

Les structures décrites dans notre système présentent une couche de méta-modélisation servant à définir explicitement les connaissances du domaine du système projeté. La représentation explicite des caractéristiques des classes évite d'avoir des attributs implicites de classes. La relation « Sorte_de » est naturelle pour l'approche orientée objet et s'implante relativement bien avec le langage Java. Il nous est donc possible de la traiter dans un système. La relation « Partie_de », aussi importante mais particulière et qui signifie

l'appartenance à l'ensemble de départ dans un système de représentation (Sowa 1992), est cependant rarement incorporée initialement dans un système orienté objet.

Nous voulons utiliser la méta-description pour obtenir un système avec des caractéristiques qui proviennent de deux niveaux d'abstraction, un conceptuel et un opérationnel et où il existe une association entre les éléments (concepts et relations) du modèle conceptuel et les éléments du niveau opérationnel (traitements).

5.1.3 Description détaillée des modules de l'architecture

Dans cette section, chacun des sous-systèmes ou blocs identifiés dans l'architecture est décrit par un tableau de caractéristiques dont la description, les données en entrée et en sortie, la nature de l'interface pour la communication, le bloc parent et les blocs qui le composent. Le tableau 5.1 décrit le bloc B0 qui est un regroupement de certains blocs présentés dans l'architecture du système. Les tableaux décrivant les autres blocs sont présentés par la suite.

Nom du bloc : B0 – Regroupement de blocs principaux	
Type	Interface utilisateur
Description	Ce bloc est un regroupement de blocs
Entrée	N/A
Sortie	N/A
Interface externe	Nul
Sous-bloc de…	Nul
Composé des sous-blocs…	B1, B2, B3 B4 et B5

Tableau 5.1 : Bloc B0 - Regroupement de blocs dans le système

Nom du bloc : B1 – Éditeur de tâches, d'actions, d'évènements	
Type	Interface utilisateur
Description	Ce bloc permet de saisir les définitions de tâches, d'actions et d'évènements devant être réalisés par le système.
Entrée	Les descriptions des tâches, des actions et des évènements
Sortie	Les structures représentant la hiérarchie des tâches
Interface externe	Interface graphique
Sous-bloc de…	B0
Composé des sous-blocs…	Nul

Tableau 5.2 : Bloc B1 - Éditeur de tâches, d'actions, d'évènements

Nom du bloc : B2 – Éditeur d'ontologie du domaine	
Type	Interface utilisateur
Description	Ce bloc permet de saisir la définition du domaine
Entrée	Description des concepts et des relations
Sortie	Description du domaine
Interface externe	Interface graphique
Sous-bloc de…	B0
Composé des sous-blocs…	Nul

Tableau 5.3 : Éditeur d'ontologie du domaine

Nom du bloc : B3 – Édition d'exemples	
Type	Interface utilisateur
Description	Ce bloc permet de définir les exemples à partir des patrons
Entrée	Le patron
Sortie	La description de l'exemple
Interface externe	Interface graphique
Sous-bloc de…	Nul
Composé des sous-blocs…	Nul

Tableau 5.4 : Éditeur d'exemple

Nom du bloc : B4 – Édition de patron	
Type	Interface utilisateur
Description	Ce bloc permet de saisir la définition des patrons pour les futurs exemples
Entrée	La description des patrons
Sortie	Les patrons
Interface externe	Interface graphique
Sous-bloc de…	Nul
Composé des sous-blocs…	Nul

Tableau 5.5 : Éditeur de patron

Nom du bloc : B5 – Interface de communication avec l'utilisateur	
Type	Interface utilisateur
Description	Ce bloc permet d'interagir avec l'utilisateur.
Entrée	N/A
Sortie	N/A
Interface externe	Nul
Sous-bloc de…	Nul
Composé des sous-blocs…	Nul

Tableau 5.6 : Interface utilisateur

5.1.4 Application des modèles d'analyse à l'architecture

Dans cette section, nous appliquons les éléments décrits dans les modèles d'analyse (tâches, base de connaissances, traitements, etc.) à notre architecture. Nous commençons par constituer notre base de patrons. Un patron représente la structure d'un exemple mais c'est en même temps la description du domaine d'application de l'exemple. Ainsi notre définition du domaine est contenue dans la base de patrons.

La structure générale d'un exemple comprend trois parties : une partie « entête » qui contient les informations d'entête, une partie « contenu » qui contient le corps de l'exemple et une partie « ressources » qui comprend les ressources associées à l'exemple.

La partie « entête » d'un exemple pourra contenir les informations sur le nom du fichier de l'exemple, sa date de création, le thème qu'il aborde, son titre, l'auteur(e) de l'exemple, etc. Dans le contexte d'un exemple présentant un problème résolu, la partie « contenu » pourra être constituée d'une sous-partie « énoncé » qui contient l'énoncé du problème, une sous-

partie « résolution » qui contient la résolution de l'exemple et une sous-partie « conclusion » qui contient la conclusion de l'exemple.

Il serait possible de structurer la partie « résolution » pour cet exemple en y rajoutant des étapes de résolution pour résoudre le problème et des explications pour chaque étape, précisant le but que nous voulons atteindre, les raisons et la manière d'y arriver. La figure 5.4 illustre le patron que nous venons de décrire.

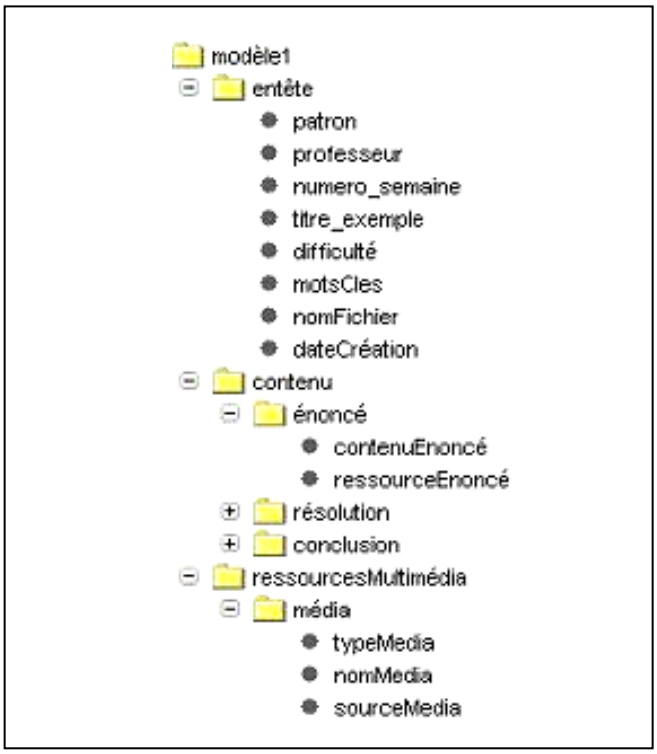


Figure 5.4 : Illustration d'un patron.

Après avoir identifié les composantes dans la structure d'un exemple, il faut préciser le type de chacune de ces composantes. Dans notre contexte, les composantes sont de type texte principalement mais peuvent aussi être de type image, son ou vidéo. Elles peuvent aussi être d'un type combiné (par exemple de type texte et image).

Maintenant que nous avons des patrons dans notre base, il nous faut définir comment procéder à l'acquisition des connaissances à partir de nos patrons. Nous allons alors définir les caractéristiques de notre interface d'acquisition et la manière de la lier au patron. Chaque composante de la structure définie dans le patron va constituer un champ pour

lequel l'usager devra entrer une valeur. Ainsi nous connaissons les champs que l'usager doit remplir. Il faut trouver la manière de présenter une interface conviviale à l'usager qui lui permet d'entrer ses valeurs.

Une fois que les données en entrée ont été acquises, nous procédons à l'interprétation de ces données, à savoir lier chacune des connaissances contenues dans la structure de l'exemple défini par le patron aux valeurs respectives entrées par l'utilisateur. Ensuite, nous pouvons générer l'exemple à proprement parler. Nous développons ainsi notre base d'exemples.

5.2 Spécification et conception du prototype

Notre système est constitué de différentes parties organisées pour assurer un ensemble de fonctions dans son environnement. Ces fonctions s'inscrivent dans le cadre de la tâche globale de gestion d'exemples. La structure fondamentale du système est définie en termes de ses constituants, interfaces, liens, comportements et contraintes. L'architecture du système est basée sur un ensemble de concepts et décrite par un ensemble de vues ou perspectives liées entre elles comme nous l'avons vu à la section 5.1.1.

À partir de l'analyse des besoins, il est possible d'établir les spécifications du système. À l'étape de conception fonctionnelle nous avons défini les spécifications fonctionnelles et non fonctionnelles du système souhaité. Les spécifications fonctionnelles renvoient aux fonctionnalités supportant les principales tâches du système (création, modification, suppression et sauvegarde d'un exemple). Les spécifications non fonctionnelles renvoient à la performance (durée d'exécution), aux ressources nécessaires, à la capacité de traitement et de communication, à l'ergonomie (Studer *et al*. 1998).

Ces spécifications nous permettent de retenir une solution fonctionnelle. La solution fonctionnelle retenue prend en compte les cas d'utilisation avec les contraintes associées, les scénarios possibles ainsi que les comportements prévus du système. À partir de cette solution, nous avons proposé une architecture sur laquelle repose l'implémentation de notre prototype.

À partir du modèle de l'architecture, nous avons bâti un plan de construction qui a permis de réaliser notre application. Les spécifications du système proposé sont les suivantes : notre outil possède des interfaces interactives. La description des données est faite en langage naturel. Le système offre une aide contextuelle pour aider à utiliser l'outil. La base d'exemples ainsi que la base de patrons évoluent dynamiquement avec le temps. Il est possible d'intégrer des images et des séquences vidéo pour créer des objets multimédias.

Concernant les besoins matériels, l'outil est autonome et fonctionne sur des machines de type PC et compatibles avec le système d'exploitation Windows 98, 2000, Millenium ou XP et ayant une machine virtuelle. La configuration minimale se compose d'un microprocesseur Pentium 3 avec 512MB de RAM, d'un disque dur de capacité minimale égale à 50 MB et d'une carte vidéo VGA.

5.3 Le prototype

Dans cette section, nous présentons les interfaces du prototype avec un exemple détaillé du processus de création d'un exemple à l'aide du système proposé.

Le système est un outil d'acquisition de connaissances. L'acquisition des connaissances est faite à partir de la structure des connaissances. L'interface d'acquisition est générée à partir de cette structure et permet d'en acquérir les connaissances.

Lors de la conception de nos interfaces, nous avons focalisé dès le début sur les utilisateurs de notre application en favorisant une conception interactive qui les fait participer. Ensuite, avec une évaluation continuelle nous avons été en mesure d'itérer sur le processus de conception des nos interfaces (Gould et Lewis 1985).

À partir des objectifs d'utilisabilité définis, nous avons développé des scénarios de tâches. Nous avons également défini les objets et les actions des interfaces. Ensuite nous avons déterminé les icônes objets et leurs représentations visuelles afin de concevoir les menus fenêtres et objets. Le design visuel a été raffiné par la suite.

Par une approche WYSIWYG (*What You See Is What You Get*), nous avons pu valider les aspects esthétique et convivial de l'interface.

Notre application système présente une interface conviviale comme le montre la figure 5.5. Les fonctionnalités du système peuvent être atteintes par les menus présents dans la barre de menus ou encore directement par les boutons de la barre de boutons pour un accès plus rapide.

Figure 5.5 : Interface principale du système

Le menu principal du système offre trois menus qui sont le menu « Fichier », le menu « Édition » et le menu « Aide ». Les figures 5.6, 5.7 et 5.8 illustrent chacun de ces menus avec les fonctionnalités offertes par chacun. Le menu « Fichier » permet de « Créer un nouveau fichier », « Ouvrir un fichier » existant pour l'éditer, « Enregistrer un fichier », « Imprimer un fichier ». Le fichier est soit un « Patron » représentant la structure des connaissances, soit un « Document » contenant les connaissances. Le menu « Édition » permet de réaliser les opérations de sélection, de copie, de coupure et de collage de texte. Le menu « Édition » permet aussi d'annuler et/ou refaire une action. Le menu « Aide » permet de visualiser l'aide sur l'application et d'en savoir plus à propos de l'application.

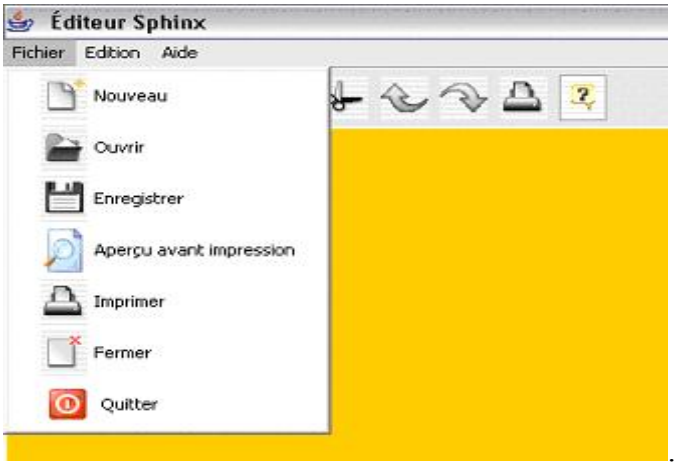


Figure 5.6 : Le menu « Fichier » de l'application

Figure 5.7 : Le menu « Edition » de l'application

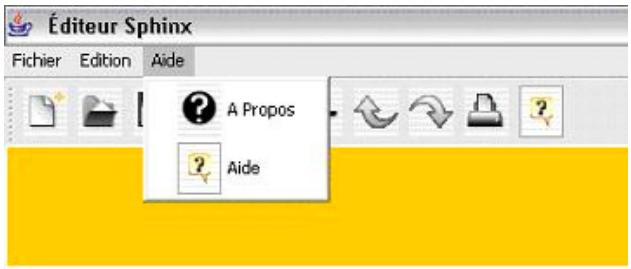


Figure 5.8 : Le menu « Aide » de l'application

Le système offre à l'usager deux types d'interfaces pour deux usages différents : l'interface de création et d'édition de patrons et l'interface de création et d'édition d'exemples. L'usager choisit le type d'interface au moment de la création ou de l'édition. Par exemple, lorsque l'usager choisit l'option « Ouvrir » dans le menu « Fichier » le système propose les deux choix possibles à savoir : ouvrir un document ou ouvrir un patron. C'est à ce moment que l'usager choisit le type d'interface. La figure 5.9 illustre ces choix.

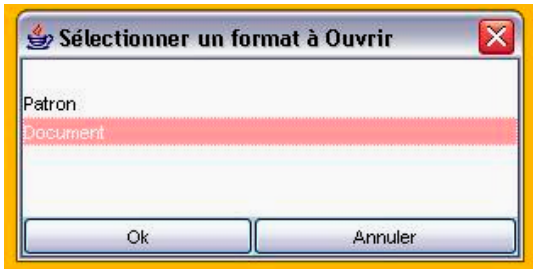


Figure 5.9 : Choix du type d'éditeur (Document ou Patron)

En mode « Patron » l'usager peut créer et éditer des patrons et ainsi enrichir sa base de patrons. Rappelons que notre système permet de créer des documents à partir de patrons représentant leur structure. En mode « Document » l'utilisateur peut alors choisir un patron dans la base de patrons et ainsi créer son document à partir de ce dernier ou encore éditer des documents existants. Il est donc nécessaire, avant de créer un exemple, de choisir un patron approprié au domaine de l'exemple ou le cas échéant de le construire.

Pour créer un nouvel exemple, l'utilisateur peut sélectionner l'option « Nouveau » dans le menu « Fichier ». Ensuite comme mentionné précédemment, l'utilisateur choisit le type « document » (figure 5.9) et valide son choix, obtenant ainsi l'interface présentée à la figure 5.10.

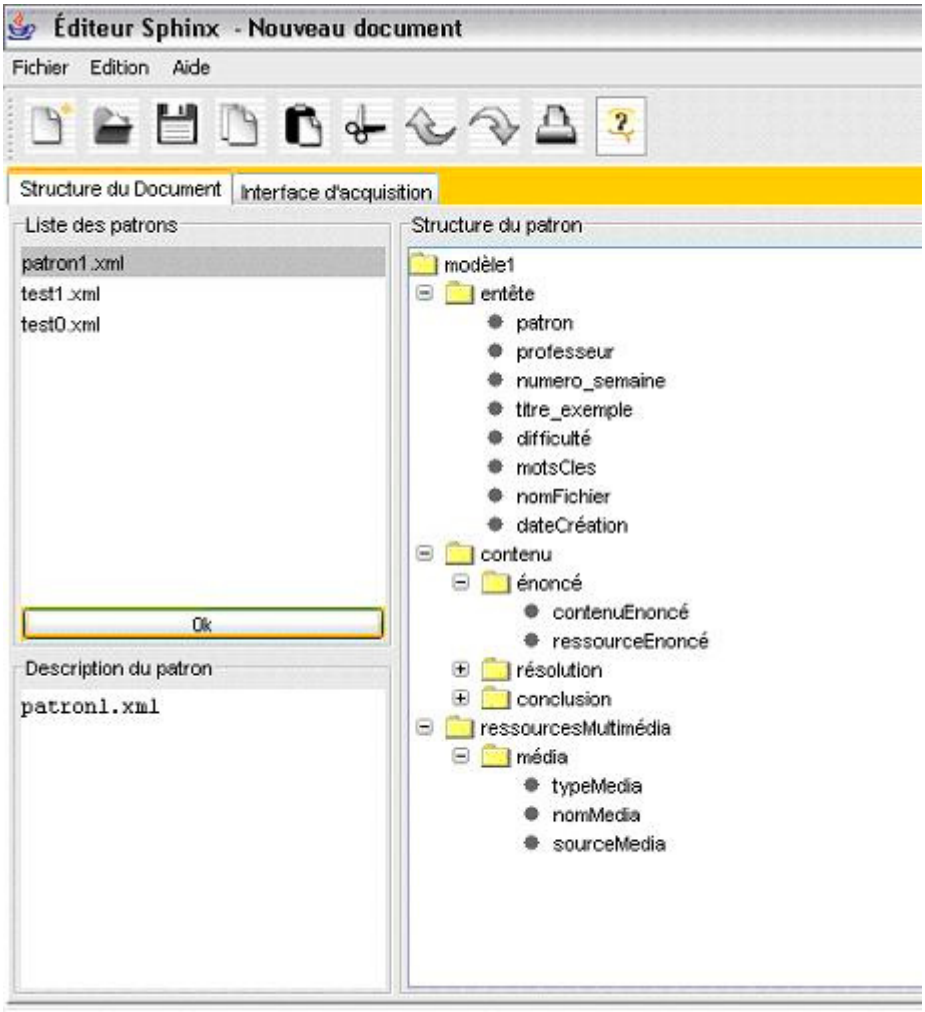


Figure 5.10 : L'interface de la structure du document

Le panneau supérieur de gauche présente la liste des patrons représentant des structures différentes. En cliquant sur un patron de la liste, la structure de ce patron s'affiche dans le panneau de droite. Il est possible de naviguer dans cette structure en cliquant sur un élément de la structure pour en développer ses composants.

Choisissons par exemple, le patron « patron1.xml » dans la liste de patrons (figure 5.10). À droite s'affiche la structure détaillée du patron avec ses éléments. Nous pouvons voir que le patron choisi se compose de trois éléments essentiels : une entête qui contient les informations d'entête sur le fichier (nom du patron, professeur, numéro de semaine, etc.), un contenu qui présente le contenu structuré du fichier et des ressources qui sont les ressources associées au document.

Une fois que le patron avec sa structure a été identifié, le choix doit être validé en cliquant sur le bouton « Ok » situé en dessous de la liste des patrons. Ceci amène à la figure 5.11a qui est en fait l'interface d'acquisition générée à partir du patron.

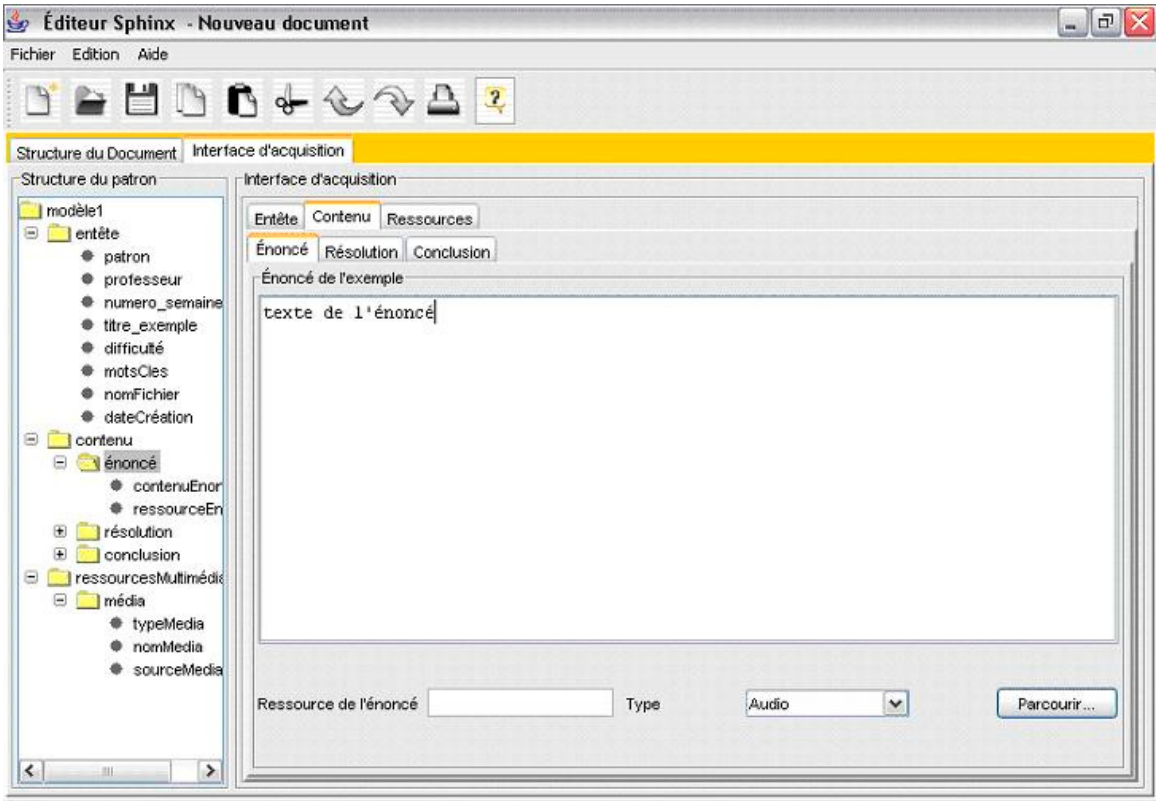


Figure 5.11a : L'interface d'acquisition de connaissances

Le panneau de gauche présente la structure du document à créer. Le panneau de droite présente l'interface d'acquisition de connaissances selon la structure de gauche. Il est possible de naviguer à travers l'interface, en cliquant sur un des onglets représentant les constituants de cette interface, pour entrer les valeurs dans les champs ou en cliquant sur un des éléments dans la structure dans le panneau de gauche.

En règle générale, un onglet correspond à un nœud identifié par l'icône représentant un répertoire et les champs correspondent aux nœuds identifiés par des ronds.

Dans certains cas où il faut plus ou moins de champs pour entrer des valeurs, des boutons « Plus » et « Moins » permettent d'ajouter ou de supprimer des champs supplémentaires de façon dynamique, comme le montre la figure 5.11b.

Le bouton « Parcourir » permet de visiter les unités de stockage de la machine à la recherche de ressources à joindre au document en cours de création.

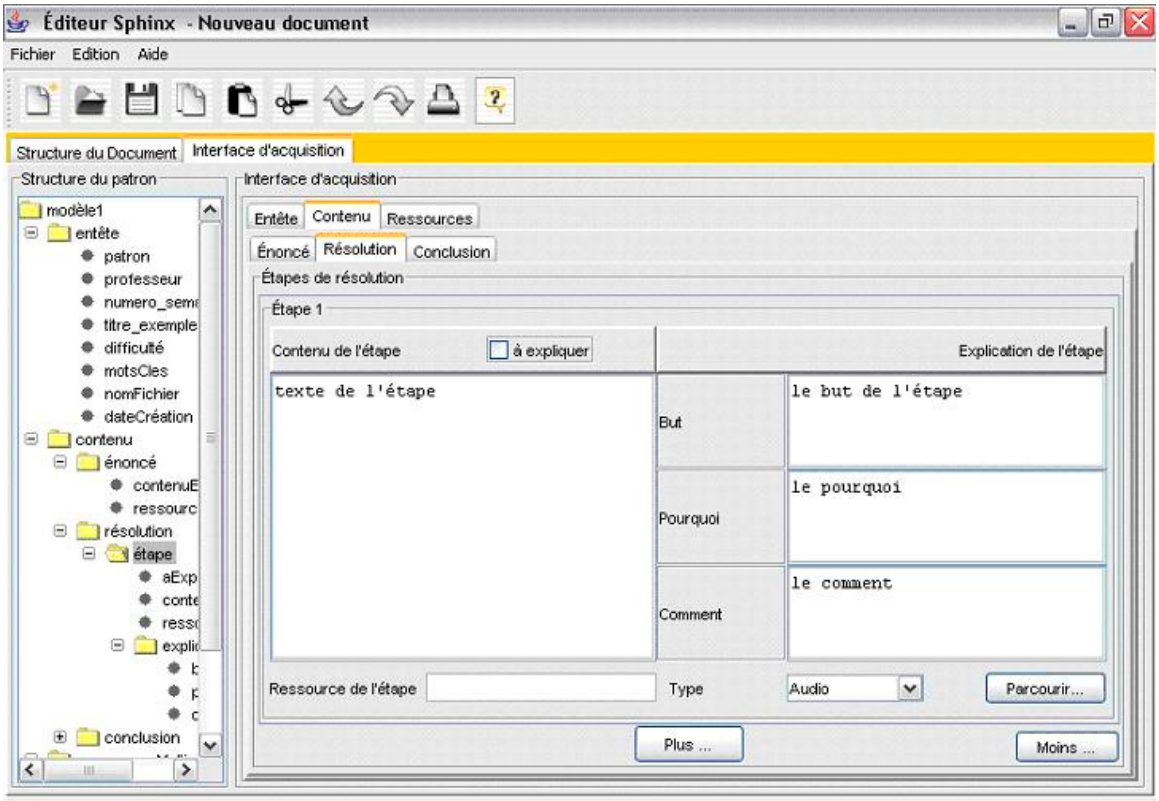


Figure 5.11b : L'interface d'acquisition de connaissances

Une fois les valeurs entrées dans les champs correspondants, le travail peut être sauvegardé. Pour cela, il suffit de choisir l'option « Enregistrer » dans le menu « Fichier ». Cette action permet de générer d'une part le fichier qui contient le travail et d'autre part de sauvegarder ce fichier à l'emplacement choisi. Ce fichier a la même structure que le patron qui a servi à le construire. En plus de la structure, il contient les connaissances entrées.

5.4 Conclusion

Dans ce chapitre, nous avons présenté en détail le modèle de conception élaboré pour notre application, que nous voulons construire. Le modèle de conception a été instancié en se basant sur sa définition telle que proposée par CommonKADS.

Nous avons également présenté le prototype réalisé avec ses interfaces et les diverses possibilités ont été illustrées.

Le chapitre suivant est consacré à l'expérimentation de notre outil. Nous y présentons les tests réalisés dans le cadre de l'expérimentation et les résultats obtenus. De plus, nous discutons des avenues possibles pour améliorer l'outil.

6 Chapitre 6 : Expérimentation, résultats et discussion

Au chapitre précédent, nous avons présenté notre prototype avec ses interfaces en illustrant les diverses possibilités de ses fonctionnalités.

L'objectif du présent chapitre est de rendre compte de l'expérimentation réalisée et des résultats obtenus. Dans un premier temps, nous allons décrire les tests que nous avons réalisés avec les différents aspects sur lesquels ils ont porté. Ensuite les résultats obtenus suite à ces tests, seront présentés.

L'expérimentation de notre outil nous permettra de valider notre travail quant aux résultats attendus et ceux obtenus. De plus nous serons en mesure de dégager les avantages et les inconvénients de notre outil. Nous terminons par une discussion sur des perspectives d'extension dans le but d'améliorer l'outil d'acquisition de connaissances proposé.

6.1 Les tests

Dans cette section, nous précisons les divers tests fonctionnels réalisés sur les composants de notre prototype. Nous avons procédé dans un premier temps à des tests unitaires. Durant cette phase de test, le prototype a été évalué pour déterminer s'il satisfaisait les spécifications requises et s'assurer qu'il était conforme à la conception détaillée.

Nous commençons par élaborer un jeu de données de tests pour notre prototype. Ensuite nous exécutons le prototype avec ce jeu, puis, nous comparons les résultats obtenus aux résultats attendus. Enfin, nous intégrons le prototype dans son environnement d'exploitation pour le tester et s'assurer qu'il se comporte comme requis dans la spécification élaborée lors de la phase d'analyse. Il s'agit de montrer le comportement et les performances du prototype dans son environnement d'exploitation réel.

6.1.1 Test des étapes de gestion d'un exemple avec le prototype

Le prototype de notre application permet de réaliser les tâches suivantes : création, modification, suppression et sauvegarde d'un exemple. Ces tâches, dont nous avons

présenté une décomposition au chapitre 4 (figure 4.1), s'inscrivent dans la tâche globale de gestion d'exemples.

Notre outil permet la création d'exemples à partir de patrons. Il permet aussi la création de patrons. Initialement, nous avons créé trois patrons pour constituer une base de patrons qui vont être déjà présents dans la base de patrons et à partir desquels il va être possible de créer des exemples. Puis en utilisant un module dédié à cette tâche, nous pouvons rajouter de nouveaux patrons pour enrichir notre base de patrons. Dans le cadre de notre travail, nous avons testé à la fois la création des exemples et la création des patrons.

Concernant les patrons, nous avons trois patrons partiels, créés initialement, en les intégrant ensemble pour obtenir un patron qui donne une structure plus polyvalente de notre exemple. De cette façon, nous pouvons tester trois patrons en même temps. La figure 6.1 illustre le patron final obtenu.

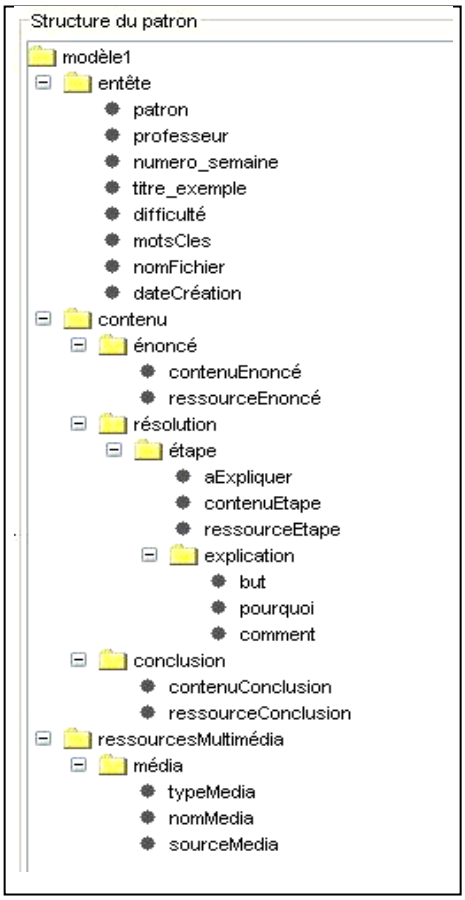


Figure 6.1 : Structure du patron

Nous commençons nos tests par l'étape de création d'un exemple, qui nous semble être la plus principale en supposant que pour faire la gestion d'exemples, il faut en disposer et dans le cas où nous n'en disposons pas, il faut en créer.

Le tableau 6.1 présente les tâches réalisées à la phase de création d'un exemple.

Numéro de la tâche	Nom de la tâche
1.	Identification du patron représentant la structure de l'exemple : spécifier le patron qui représente la structure de l'exemple que l'on veut créer.
2.	Évaluation du patron : vérifier si le patron représente bien la structure de l'exemple
3.	Choix du patron : valider le patron sélectionné.
4.	Génération de l'interface d'acquisition à partir du patron : produire l'interface d'acquisition correspondante à partir du patron choisi.
5.	Saisie des données : capter les données qui vont servir à créer l'exemple.
6.	Génération de l'exemple à partir des données entrées : créer l'exemple proprement dit à partir des données entrées, acquises via l'interface d'acquisition

Tableau 6.1 : Les tâches de création d'un exemple

À partir de l'interface de notre outil, nous générons l'interface d'acquisition correspondant au patron sélectionné. Ensuite nous saisissons les données pertinentes pour remplir les champs dans notre fichier d'exemples puis nous générons l'exemple à partir des données entrées. De cette manière, nous avons généré nos exemples avec notre outil pour tester la phase de création des exemples.

La sauvegarde des exemples dans notre espace de travail nous a permis de tester l'étape de sauvegarde des exemples. Les fichiers des exemples portent le nom que nous avons entré

lors de la sauvegarde et les exemples s'inscrivent bien dans l'emplacement physique indiqué.

Pour supprimer un exemple, nous pouvons le faire autant au niveau du volet d'accès des exemples dans l'interface ou physiquement. La suppression de l'exemple de son emplacement physique sur le disque permet de valider le test de l'étape de suppression.

Nous avons également été en mesure de modifier les exemples que nous avons créés, en les éditant à travers l'interface d'édition des exemples. Nous identifions les informations à éditer, puis nous sélectionnons les champs correspondants pour y apporter des modifications. Ces modifications sont bien prises en compte et l'affichage des exemples reflète les changements apportés. Nous testons ainsi l'étape de modification des exemples. Lorsque nous indiquons un emplacement différent de l'emplacement original, lors de la sauvegarde d'un exemple après sa modification, l'exemple est enregistré au nouvel emplacement de destination.

Nous venons de réaliser la phase de test concernant les exemples. Pour la phase de test concernant les patrons, nous avons créé des patrons en utilisant le module d'édition des patrons. À partir de l'interface du module d'édition des patrons, nous spécifions qu'il s'agit d'un nouveau patron. Nous pouvons alors créer le nœud racine du patron et lui rajouter la hiérarchie des nœuds parents et enfants avec leurs attributs. Rappelons que nos patrons sont sous la forme de fichiers XML. Pour sauvegarder les patrons, nous spécifions l'emplacement de la base de patrons. Les fichiers s'ajoutent bien à la base de patrons déjà existants. Ainsi, nous sommes en mesure de créer différentes structures d'exemples.

Pour modifier un patron, nous spécifions, au module d'édition des patrons, qu'il s'agit d'un patron déjà existant. Ensuite nous localisons ce dernier que nous ouvrons dans l'interface d'édition des patrons puis nous apportons les modifications souhaitées et enregistrons les corrections.

6.1.2 Ajout de ressources

Lors de la création d'un exemple, il est possible de lui attacher une ressource. Une ressource peut être de différents types : audio, vidéo, image. La figure 6.2 illustre l'ajout d'une ressource de type image (RH1.gif) à un exemple (puzzle8) en cours de création.

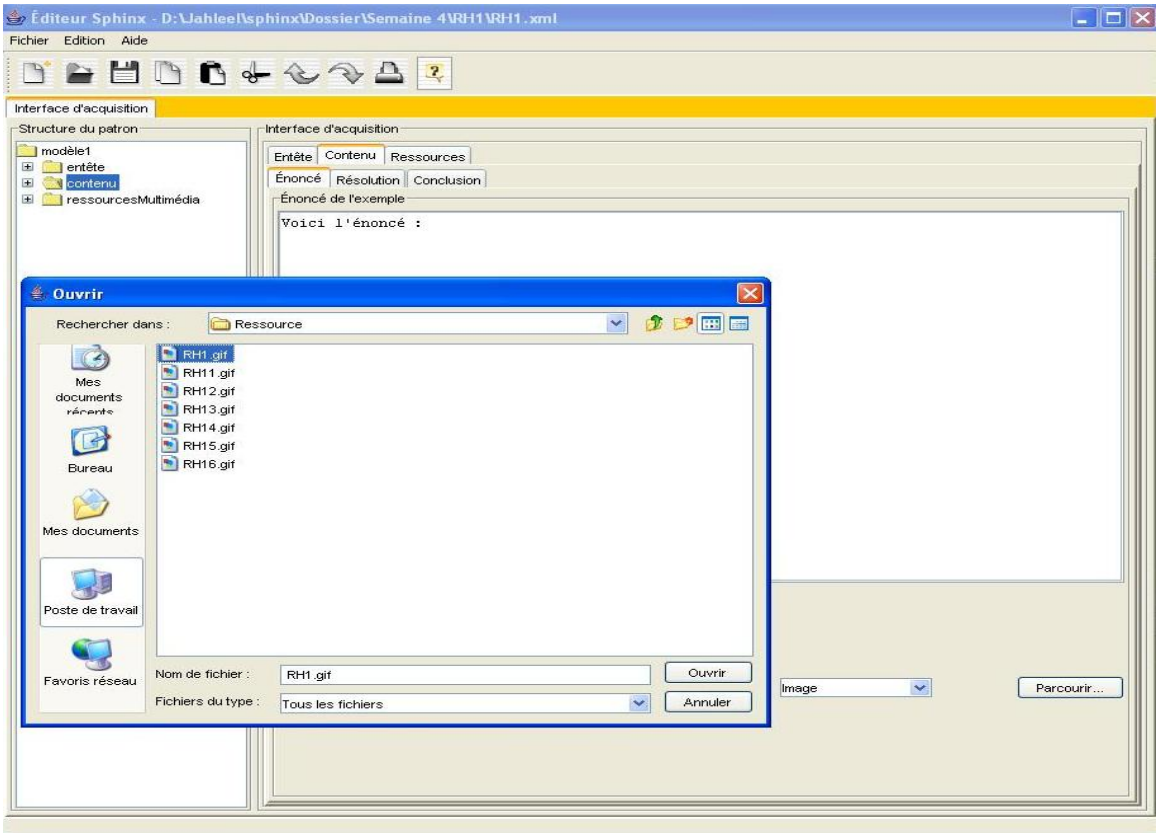


Figure 6.2 : Ajout d'une ressource de type image

Une ressource intervient à un certain moment lors de la consultation de l'exemple. Dans notre cas, une ressource peut se retrouver dans l'énoncé de l'exemple ou dans les étapes de résolution de l'exemple. Nous ajoutons donc la ressource au bon endroit où nous voulons qu'elle intervienne. Dans la figure 6.3, la ressource de type image intervient dans l'énoncé de l'exemple. Plus précisément l'énoncé de l'exemple est contenu dans l'image.

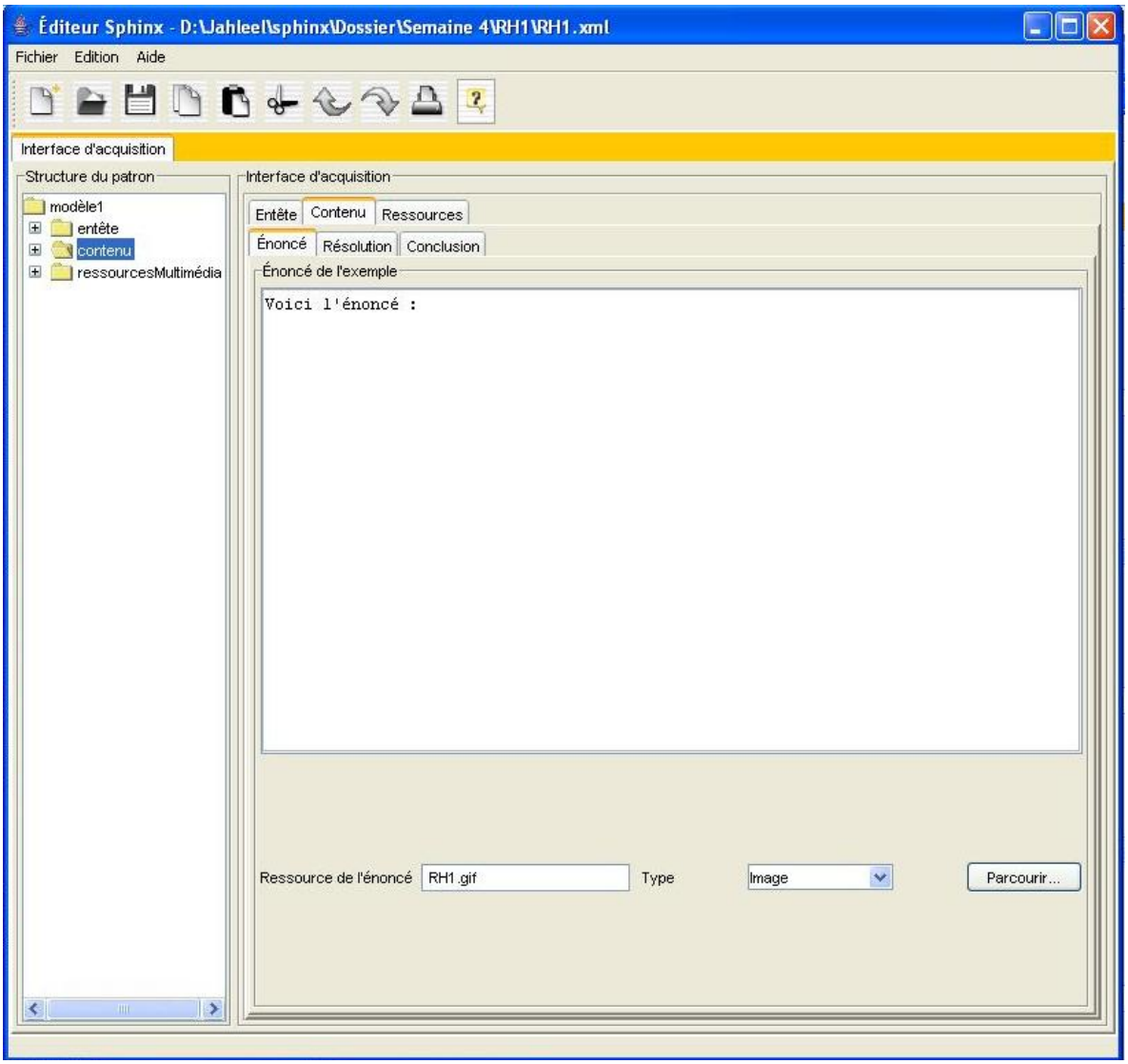


Figure 6.3 : Ressource de type image - énoncé de l'exemple

La figure 6.4 illustre le cas où la ressource intervient dans les étapes de résolution de l'exemple.

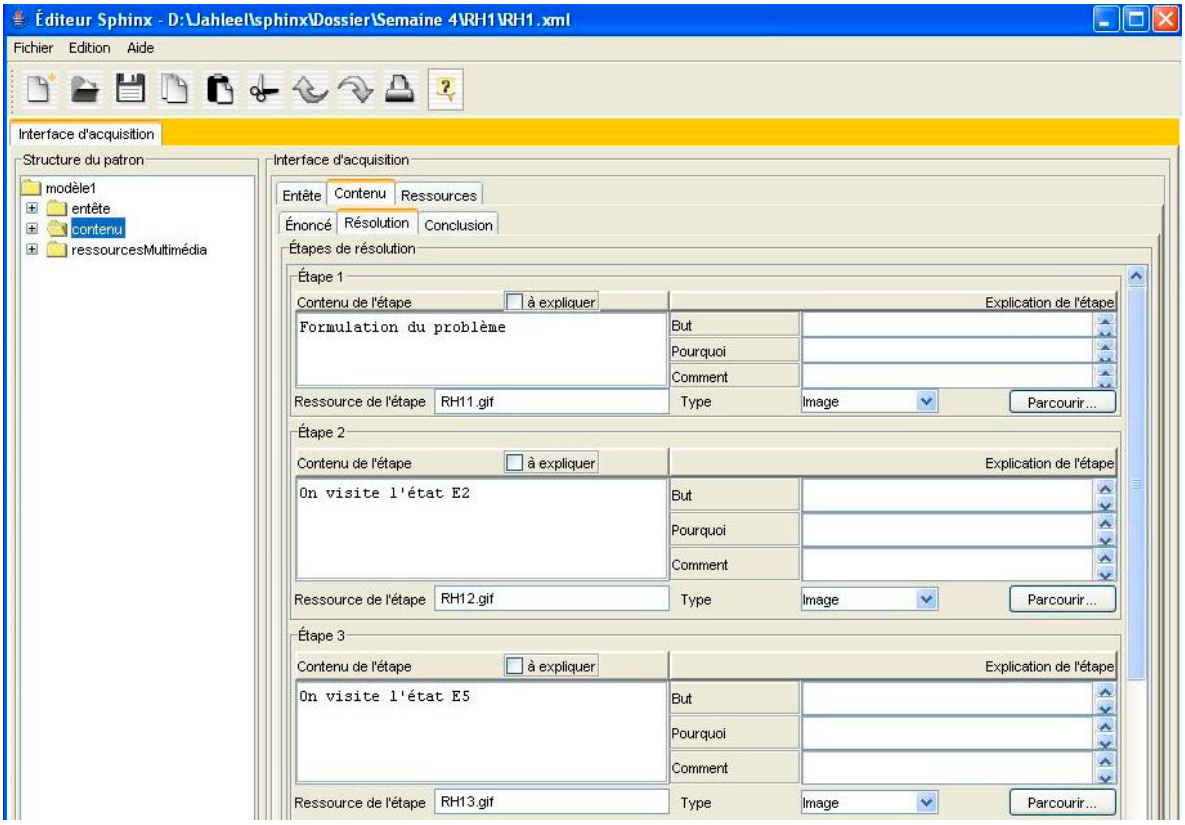


Figure 6.4 : Ressource de type image - étapes de résolution de l'exemple

6.1.3 Accès à une base de données

Nous disposons d'une base de données que nous utilisons pour stocker des informations de contrôle pour chacun des exemples que nous créons. Rappelons que les exemples créés avec notre outil sont intégrés à un système éducationnel (SPHINX) qui les utilise à des fins de présentation, d'apprentissage, d'échanges et de collaboration.

Pour chacun des exemples créés, nous inscrivons dans cette base de données les informations qui vont permettre de détecter la présence d'un nouvel exemple dans le système éducationnel, d'identifier le groupe thématique auquel il appartient, de localiser le fichier source de l'exemple et les ressources correspondantes.

La figure 6.5 illustre la structure de la table utilisée dans la base de données pour enregistrer ces informations. On retrouve, dans ces informations, le numéro de la semaine à laquelle appartient l'exemple. Une semaine aborde différents thèmes d'apprentissage. Le

numéro de semaine permet de lier l'exemple au contenu thématique abordé pendant la semaine en question. On retrouve également le nom du fichier de l'exemple, le titre de l'exemple, le niveau de difficulté de l'exemple, la liste des mots clés correspondant à l'exemple et le nombre d'étapes contenues dans la résolution de l'exemple.

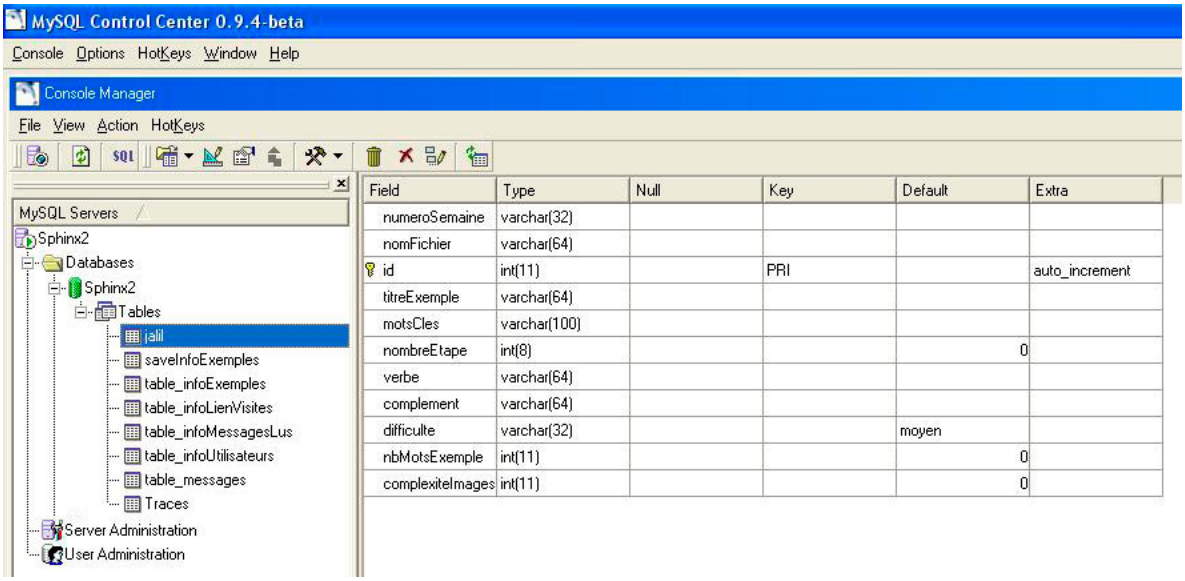
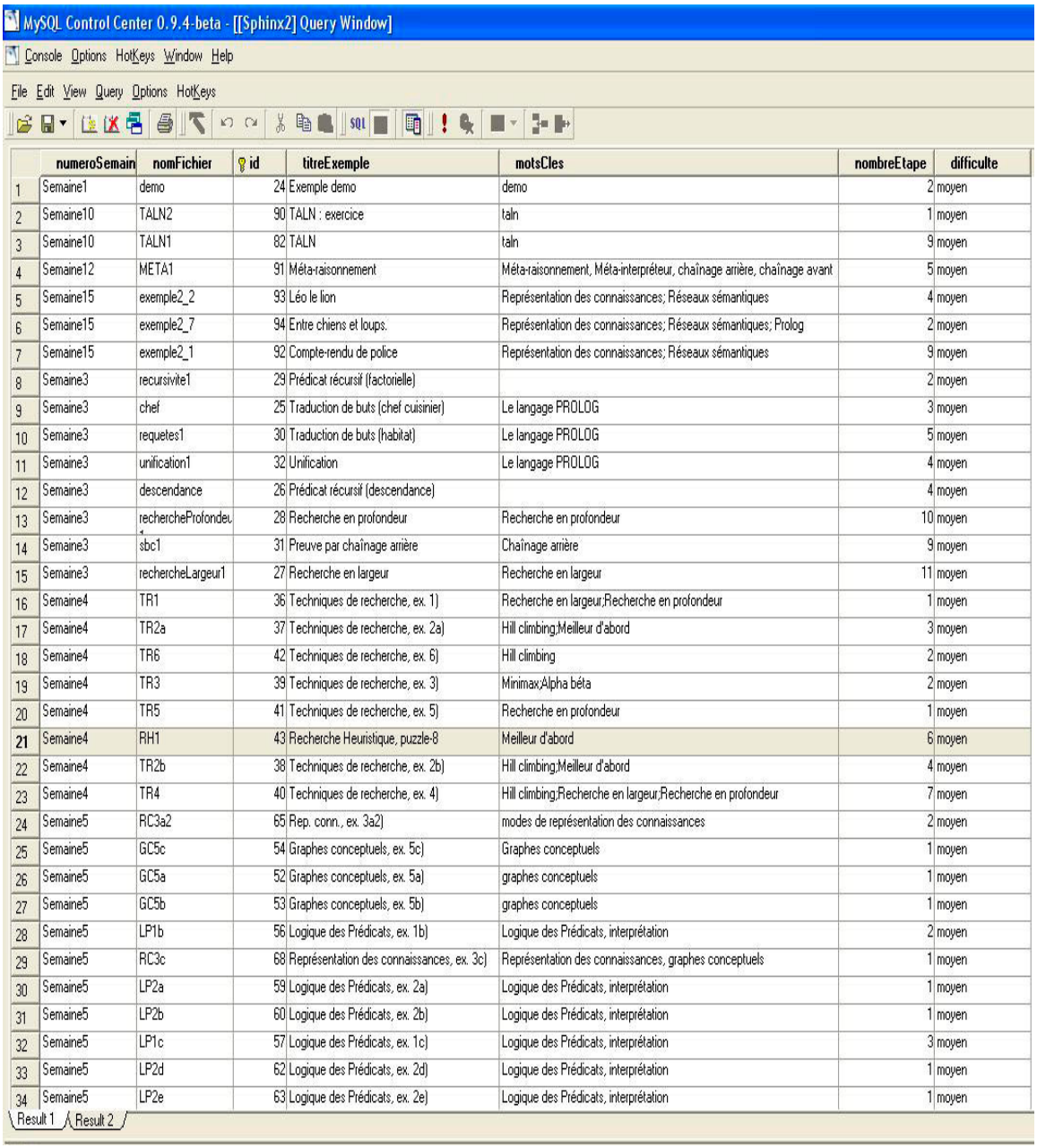


Figure 6.5 : Structure de la table contenant les informations
de contrôle des exemples dans la base de données

Le système SPHINX consulte cette table de la base de données pour extraire ces informations pertinentes qui lui permettent d'indexer efficacement les exemples avec leur contenu.

Nous avons testé d'une part l'accès à cette base de données, en y inscrivant les données pertinentes de nos exemples et en vérifiant la présence de ces données dans la base de données à l'aide d'un outil de gestion de base de données. Le système de gestion de base de donnée (SGBD) utilisé est MySQL Control Center qui nous permet de parcourir l'ensemble des tables de notre base de données et de valider l'écriture des données à l'intérieur de cellesci . La figure 6.6 illustre ce point.



MySQL Control Center 0.9.4-beta - [[Sphinx2] Query Window]

Console Options HotKeys Window Help

File Edit View Query Options HotKeys

	numeroSemain	nomFichier	id	titreExemple	motsCles	nombreEtape	difficulte
1	Semaine1	demo	24	Exemple demo	demo	2	moyen
2	Semaine10	TALN2	90	TALN : exercice	taln	1	moyen
3	Semaine10	TALN1	82	TALN	taln	9	moyen
4	Semaine12	META1	91	Méta-raisonnement	Méta-raisonnement, Méta-interpréteur, chaînage arrière, chaînage avant	5	moyen
5	Semaine15	exemple2_2	93	Léo le lion	Représentation des connaissances; Réseaux sémantiques	4	moyen
6	Semaine15	exemple2_7	94	Entre chiens et loups.	Représentation des connaissances; Réseaux sémantiques; Prolog	2	moyen
7	Semaine15	exemple2_1	92	Compte-rendu de police	Représentation des connaissances; Réseaux sémantiques	9	moyen
8	Semaine3	recursivite1	29	Prédicat récursif (factorielle)		2	moyen
9	Semaine3	chef	25	Traduction de buts (chef cuisinier)	Le langage PROLOG	3	moyen
10	Semaine3	requetes1	30	Traduction de buts (habitat)	Le langage PROLOG	5	moyen
11	Semaine3	unification1	32	Unification	Le langage PROLOG	4	moyen
12	Semaine3	descendance	26	Prédicat récursif (descendance)		4	moyen
13	Semaine3	rechercheProfondeu	28	Recherche en profondeur	Recherche en profondeur	10	moyen
14	Semaine3	sbc1	31	Preuve par chaînage arrière	Chaînage arrière	9	moyen
15	Semaine3	rechercheLargeur1	27	Recherche en largeur	Recherche en largeur	11	moyen
16	Semaine4	TR1	36	Techniques de recherche, ex. 1)	Recherche en largeur;Recherche en profondeur	1	moyen
17	Semaine4	TR2a	37	Techniques de recherche, ex. 2a)	Hill climbing;Meilleur d'abord	3	moyen
18	Semaine4	TR6	42	Techniques de recherche, ex. 6)	Hill climbing	2	moyen
19	Semaine4	TR3	39	Techniques de recherche, ex. 3)	Minimax;Alpha béta	2	moyen
20	Semaine4	TR5	41	Techniques de recherche, ex. 5)	Recherche en profondeur	1	moyen
21	Semaine4	RH1	43	Recherche Heuristique, puzzle-8	Meilleur d'abord	6	moyen
22	Semaine4	TR2b	38	Techniques de recherche, ex. 2b)	Hill climbing;Meilleur d'abord	4	moyen
23	Semaine4	TR4	40	Techniques de recherche, ex. 4)	Hill climbing;Recherche en largeur;Recherche en profondeur	7	moyen
24	Semaine5	RC3a2	65	Rep. conn., ex. 3a2)	modes de représentation des connaissances	2	moyen
25	Semaine5	GC5c	54	Graphes conceptuels, ex. 5c)	Graphes conceptuels	1	moyen
26	Semaine5	GC5a	52	Graphes conceptuels, ex. 5a)	graphes conceptuels	1	moyen
27	Semaine5	GC5b	53	Graphes conceptuels, ex. 5b)	graphes conceptuels	1	moyen
28	Semaine5	LP1b	56	Logique des Prédicats, ex. 1b)	Logique des Prédicats, interprétation	2	moyen
29	Semaine5	RC3c	68	Représentation des connaissances, ex. 3c)	Représentation des connaissances, graphes conceptuels	1	moyen
30	Semaine5	LP2a	59	Logique des Prédicats, ex. 2a)	Logique des Prédicats, interprétation	1	moyen
31	Semaine5	LP2b	60	Logique des Prédicats, ex. 2b)	Logique des Prédicats, interprétation	1	moyen
32	Semaine5	LP1c	57	Logique des Prédicats, ex. 1c)	Logique des Prédicats, interprétation	3	moyen
33	Semaine5	LP2d	62	Logique des Prédicats, ex. 2d)	Logique des Prédicats, interprétation	1	moyen
34	Semaine5	LP2e	63	Logique des Prédicats, ex. 2e)	Logique des Prédicats, interprétation	1	moyen

Result 1 Result 2

Figure 6.6 : Contenu de la table contenant les informations de contrôle des exemples dans la base de données

De plus l'affichage des exemples dans le système SPHINX à partir des informations extraites de la base de données nous a permis de valider la présence des informations et le fait que le système pouvait bel et bien y accéder.

6.1.4 Sauvegarde locale et stockage serveur

Après avoir créé nos exemples, nous les sauvegardons dans notre espace de travail local. L'espace de travail est organisé en répertoires et sous-répertoires selon une certaine hiérarchie. Nous nous sommes basé sur le découpage thématique de la matière enseignée dans le cours d'intelligence artificielle offert à l'Université Laval, pour organiser nos répertoires et créer nos exemples.

Les répertoires sont organisés par semaine. On retrouve quinze répertoires qui correspondent au nombre de semaines par session universitaire. Chacun de ces répertoires par semaine se divise en sous-répertoires. Chaque sous-répertoire correspond à un exemple créé. Chaque dossier contient un exemple et les ressources qui lui sont associées. Il est possible de créer différents exemples dans un même dossier, qui partagent les mêmes ressources. Les exemples sons stockés localement sur le disque dans un premier temps, comme l'indique la figure 6.7.

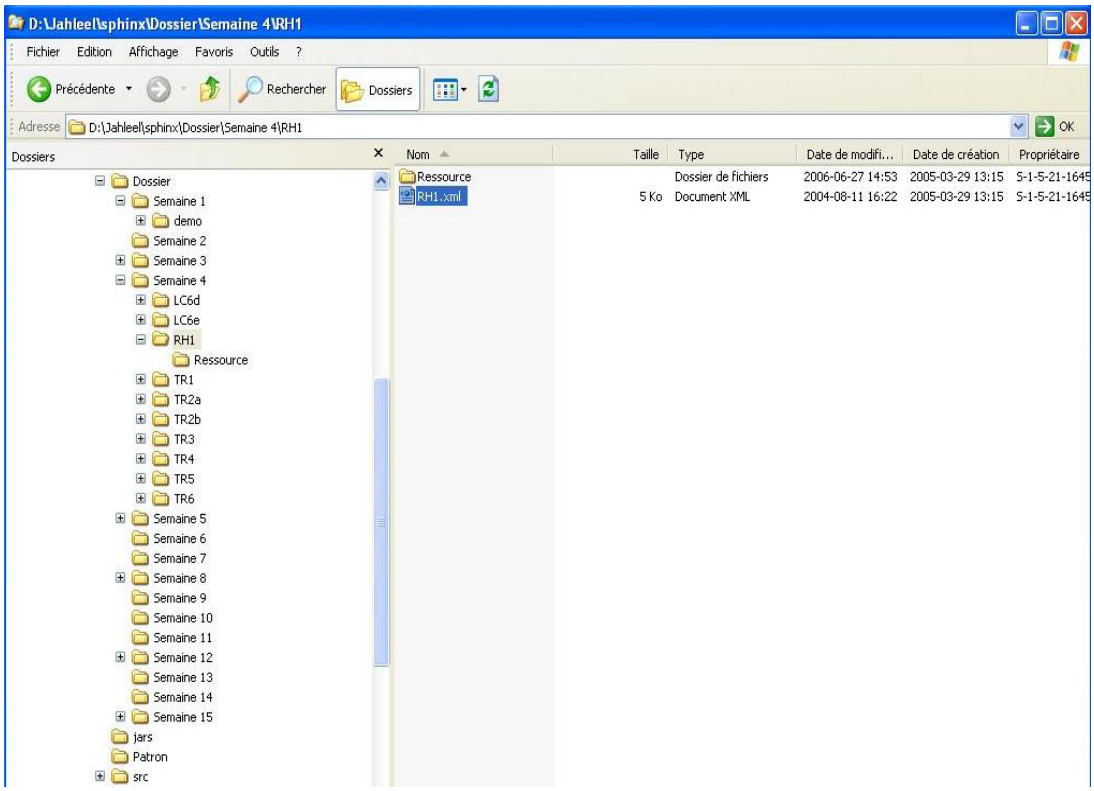


Figure 6.7 : Exemple stocké sur un disque local

Dans un deuxième temps, les exemples sont transférés depuis le disque local sur le serveur de l'application. La figure 6.8 illustre la sauvegarde des exemples sur le disque local et leur transfert sur le serveur.

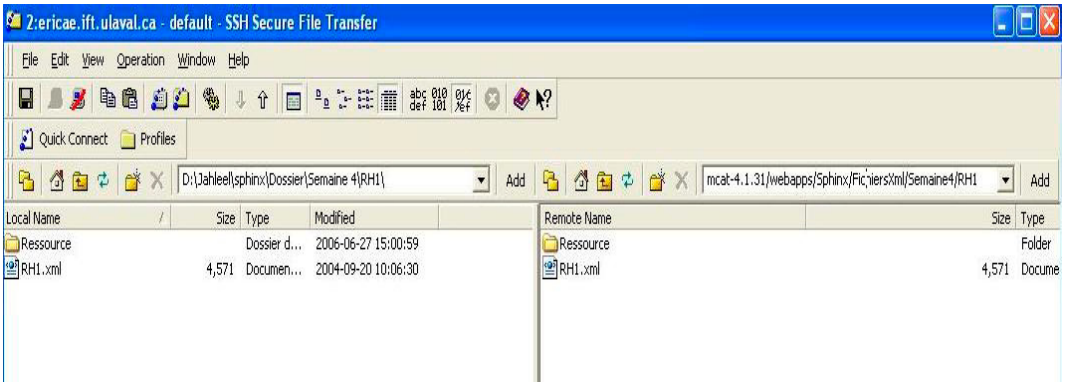


Figure 6.8 : Exemple stocké localement et sur le serveur de l'application

Nous avons exécuté les fonctionnalités de sauvegarde et de transfert, testé le stockage et le transfert des exemples. De plus, les exemples étant intégrés au système éducationnel SPHINX, leur apparition dans l'interface de ce dernier valide la présence des données sur le serveur, donc leur transfert.

6.1.5 Performance - Calcul du temps

En termes de performance, nous avons mesuré le temps d'exécution pris par notre prototype en calculant le temps moyen que nécessite la création d'un exemple depuis sa création à son transfert sur le serveur voire sa présence dans le système éducationnel SPHINX.

Nous avons tenu compte des performances de la machine utilisée pour réaliser les tâches et des spécifications en usage, de la mémoire et du langage utilisé pour développer le prototype afin d'analyser la performance de notre outil.

Nous avons également calculé le temps nécessaire pour le chargement et la mise à jour d'un exemple. Nous avons pris séparément l'exécution des fonctionnalités d'une tâche puis l'exécution de la tâche globale pour avoir des mesures de comparaison.

6.2 Les résultats obtenus

Notre outil se comporte tel que nous l'avons anticipé. En effet, avec notre prototype, nous pouvons créer des patrons et nous pouvons créer des exemples à partir de patrons existants. Les fichiers créés sont des fichiers XML bien formés avec la bonne extension.

Les patrons créés s'inscrivent dans la base de patrons à l'emplacement physique indiqué dans l'espace de travail. Lors de la création des exemples, ils se retrouvent dans la liste de patrons disponibles à partir desquels nous pouvons créer des exemples. L'interface d'acquisition générée à partir du patron permet de remplir les champs nécessaires pour la création de l'exemple. La structure des exemples créés respecte celle des patrons correspondants.

En consultant le fichier d'un exemple qui a été créé, nous pouvons valider sa structure par rapport à celle du patron utilisé pour le créer et nous pouvons aussi valider la présence des données pertinentes que nous avons saisies dans l'interface d'acquisition pour générer l'exemple. Quand l'exemple comporte une ressource de type image par exemple, celle-ci s'affiche comme il se doit. Lorsque nous éditons nos exemples, les modifications sont bel et bien prises en compte et l'affichage des exemples reflète les changements apportés.

Les exemples s'enregistrent au bon endroit dans la hiérarchie des exemples. Nous pouvons le constater en nous rendant à leur emplacement physique dans la machine. Les fichiers des exemples y sont présents et portent le nom que nous avons entré lors de la sauvegarde. Dans le cas où il y en a, les ressources se retrouvent aussi à cet emplacement dans le répertoire prévu pour les ressources. Lorsque nous indiquons un emplacement différent de l'emplacement original, lors de la sauvegarde d'un exemple après sa modification, l'exemple est déplacé à son emplacement de destination. Quand nous supprimons un exemple, il est effacé de son emplacement physique sur le disque.

L'accès à la base de données pour y inscrire les informations de contrôle relatives aux exemples se fait avec succès. En effet, nous retrouvons dans la base de données, les enregistrements correspondant à chacun de nos exemples avec les valeurs par champ d'information. La sauvegarde locale des données et leur transfert sur le serveur de l'application se déroulent sans erreur. Nous retrouvons la même hiérarchie des exemples

sur le serveur. De plus, le système, qui intègre les exemples sur le serveur, les indexe correctement avec leur contenu.

En calculant le temps moyen que nécessite la création d'un exemple et son transfert sur le serveur, l'exécution de notre prototype prend 6 à 8 minutes. Ce temps est calculé en ne considérant que les exemples qui vont être utilisés par SPHINX. Ceux-ci ont des structures similaires peu différentes. De plus ce temps n'inclut pas le temps mis à concevoir le modèle de l'exemple (environ 5 minutes), ni celui mis pour construire les ressources (temps variable selon la complexité des ressources). Enfin, ce temps est plus ou moins long selon le nombre d'étapes dans la résolution de l'exemple.

La machine que nous avons utilisée pour réaliser les tâches utilise un processeur de 3.2GHz avec 1.0 GB de mémoire. Notre machine n'est pas des plus rapides mais elle correspond au standard que l'on retrouve de nos jours en matière d'ordinateur. Nous avons testé notre application sur une machine dotée d'un processeur Pentium 3 à 733 MHz avec 512MB de RAM et son exécution n'était pas moins rapide. Nous avons adopté la configuration de cette machine comme configuration minimale pour exécuter notre outil, car elle est la machine la moins puissante sur laquelle nous avons été en mesure d'exécuter notre prototype.

Le temps nécessaire pour le chargement, la sauvegarde et le transfert des exemples est de quelques secondes (moins de 130 secondes). Bien que le langage Java soit un peu gourmand en mémoire, le prototype s'exécute de façon convenable dans un temps raisonnable.

6.3 Discussion

Dans le cadre de notre travail de recherche, notre objectif était de modéliser un outil d'acquisition d'exemples pour un système d'aide à l'apprentissage humain à partir d'exemples, puis d'implanter un prototype logiciel supportant cette méthode et qui permet à l'enseignant de faire la gestion des exemples destinés au système.

Nous avons présenté dans les sections précédentes, les tests réalisés pour valider notre prototype avec les résultats obtenus. Dans cette section nous discutons ces résultats. Nous

discutons également les avantages et les inconvénients quant à l'utilisation de l'outil, avec des perspectives d'extension dans le but de l'améliorer.

6.3.1 Les résultats obtenus correspondent-ils aux résultats attendus ?

Nos travaux visaient à faciliter l'élaboration d'un système graduellement extensible. Nous sommes intervenu plus précisément dans la phase d'acquisition de connaissances du système. Les résultats de notre travail sont le modèle de notre outil d'acquisition et le prototype de l'outil.

Dans le contexte d'un système d'apprentissage humain à partir d'exemples, notre prototype permet d'assister le rôle de l'enseignant dans la construction et la maintenance de contenu pédagogique. Dans ce sens, les résultats obtenus pendant l'expérimentation de notre outil sont concluants.

Nous souhaitions développer une application permettant d'éditer le contenu pédagogique destiné au système d'apprentissage humain. Notre prototype permet d'acquérir des exemples et d'en faire la gestion. Il permet de créer des exemples, de les éditer, de les sauvegarder et de les supprimer. Notre prototype permet également de créer des patrons d'exemples à partir desquels les exemples sont créés. Notre prototype permet donc d'éditer le contenu pédagogique d'un EIAH.

L'enseignant peut, à travers notre application, faire la gestion du contenu pédagogique dans le système éducationnel. Ainsi, notre prototype facilite la présentation, l'échange et le stockage de connaissances dans le but de favoriser l'apprentissage de l'apprenant.

Nous souhaitions un outil convivial, facile d'utilisation. Nous avons doté notre outil d'interfaces interactives et conviviales. De plus, l'outil offre une aide contextuelle pour aider à son utilisation. Nous avons évalué les interfaces de notre application, en considérant les critères d'évaluation suivants : la durée d'apprentissage de fonctions spécifiques, la rapidité dans l'exécution des tâches, le taux d'erreurs utilisateurs, la facilité de rétention dans le temps et enfin la satisfaction subjective des utilisateurs. Ceci nous a permis de répondre à des questions d'utilisabilité du genre « est-ce que l'utilisateur sera

capable de répondre plus vite aux besoins d'urgence pour un exemple illustrant une notion nouvelle d'apprentissage »?

Pour évaluer les interfaces, nous distinguons deux types d'évaluation. L'évaluation formative, effectuée durant le processus de conception et de développement sur une version incomplète de l'application et l'évaluation sommative effectuée à la fin du développement sur une version complète de l'application.

Parmi les méthodes possibles pour évaluer l'utilisabilité, nous retrouvons les méthodes analytiques qui consistent en une simulation d'exécution des activités de l'utilisateur sans l'implication de ce dernier et les méthodes empiriques méthodes basées sur l'observation des attitudes et des comportements en situation d'utilisation de l'application évaluée.

Dans notre contexte nous avons retenu l'évaluation formative et la méthode analytique pour évaluer nos interfaces. Les interfaces de notre application ont été présentées au chapitre 5 à la section 5.2, réservée à la description du prototype. Les fonctionnalités du système peuvent être atteintes par les menus présents dans la barre de menus ou encore directement par les boutons de la barre de boutons pour un accès plus rapide.

Durant son exécution, notre outil ne demande à l'utilisateur que les connaissances dont il a réellement besoin pour la réalisation de sa tâche, ceci, pour éviter le risque que l'utilisateur ne s'en désintéresse. La technologie XML se prête bien à notre application. Grâce à elle, nous avons été en mesure d'inclure une logique de traitement des informations et d'automatiser les traitements. Ainsi, nous avons pu atteindre notre objectif de transparence pour les utilisateurs. De plus, les traitements restent indépendants de la plate-forme utilisée, ce qui ajoute à la flexibilité de notre outil.

Au vu des résultats obtenus, nous pouvons affirmer que ceux-ci correspondent aux résultats attendus. Cependant certains objectifs, d'un point de vue qualitatif n'ont pas été complètement réalisés. Nous discutons de ces aspects au chapitre 7 à la section 7.3 portant sur les améliorations possibles.

Cependant, pour atteindre nos objectifs, nous avons rencontré des difficultés qui nous ont amené à faire des choix de modélisation et de conception. Dans la section qui suit nous présentons les problèmes rencontrés.

6.3.2 Problèmes rencontrés

Les problèmes rencontrés tout au long de notre travail de recherche interviennent autant à l'étape de modélisation qu'à celle de conception de notre application.

Les problèmes rencontrés à l'étape de modélisation concernent la modélisation et la représentation des connaissances de notre domaine d'application. Initialement, nous avons modélisé notre domaine pour définir la hiérarchie des classes et représenter tous les types de relations entre les classes. Cependant certaines classes d'objets se retrouvant à différents niveaux dans notre hiérarchie auraient nécessité des relations d'héritage multiple. L'héritage multiple n'est pas nécessairement le meilleur choix de représentation parce qu'il n'est pas toujours facile à implanter. Dans notre cas, le langage orienté objet Java, utilisé pour développer l'application, ne permettait pas d'implanter l'héritage multiple. Nous n'étions donc pas en mesure d'implanter ces classes. Il nous a donc fallu redéfinir la hiérarchie des classes et représenter tous les types de relations entre les classes en prenant soin d'éviter d'avoir de l'héritage multiple et ainsi assurer le bon fonctionnement du système. C'est là l'importance des choix dans la modélisation et l'analyse détaillée du domaine.

6.3.3 Structure des exemples

La structure des fichiers que nous manipulons est celle d'un fichier XML qui utilise des balises pour décrire des données avec leur contenu. Si un fichier XML comporte des balises dont le contenu est vide, alors ce fichier XML présente des erreurs lors de son affichage. Au niveau de notre application, les balises présentes dans nos fichiers XML représentent des classes avec leurs attributs. Le fait d'avoir une classe qui n'est pas instanciée, ou alors d'avoir une classe avec un attribut nul, conduit à un fichier XML avec des balises dont le contenu est vide. Nous avons dû éviter cette situation pour que notre système fonctionne bien.

6.3.4 Gestion des interfaces

Concernant les interfaces, nous avons éprouvé des difficultés à définir le modèle de navigation de notre application. Nous obtenions par endroits des interfaces qui n'étaient pas stables et les composants ne se disposaient pas conformément à la configuration qui avait été prévue. Après essais et erreurs et avec l'aide de la documentation sur la programmation des interfaces, nous sommes arrivé à produire des interfaces conviviales et flexibles.

6.3.5 Gestion des accents

Le langage de développement que nous manipulons utilise la langue anglaise par défaut, dans la spécification des normes qu'il utilise et des fonctionnalités qu'il offre. Comme nous voulons construire une application qui utilise la langue française par défaut, nous avons rencontré quelques incompatibilités notamment au niveau des accents.

Avantages

Notre outil a l'avantage majeur de servir à ce pour quoi il avait été destiné : permettre à l'enseignant de faire la gestion du contenu pédagogique dans un système d'apprentissage humain. Ainsi, nous pouvons croyons que notre outil sert à quelque chose, ce qui en fait sa qualité première.

De plus, notre prototype permet de diminuer l'effort de l'enseignant dans la construction et la maintenance de contenu pédagogique. Ce dernier n'a plus à construire ses exemples directement dans le système. Il dispose désormais d'un outil visuel d'acquisition de connaissances, qui permet de créer des exemples, de les éditer, de les sauvegarder et de les supprimer de façon conviviale. Bref, l'enseignant peut acquérir les exemples et en faire la gestion plus facilement.

Notre prototype facilite également la présentation, l'échange et le stockage de connaissances dans le but de favoriser l'apprentissage de l'apprenant. Grâce à notre outil, le système qui l'intègre est capable d'accepter les nouvelles connaissances ou celles mises à jour, d'une manière simple et conviviale. Ainsi, l'ingénieur de la connaissance (cogniticien) lors du développement du système et l'expert tout au long de son utilisation peuvent, au fur et à mesure que l'expertise évolue et, chacun du point de vue qui l'intéresse et selon son

rôle, mettre à jour le contenu de la base de connaissances du système. Cette maintenance concerne à la fois la définition de concepts dans le domaine et leurs relations, la décomposition et l'ordonnancement de tâches.

Les interfaces conviviales et flexibles de l'outil et son aspect visuel permettent de faciliter la réalisation des tâches de l'enseignant lorsque ce dernier exécute notre prototype. De plus le comportement de notre outil pendant son exécution entraîne l'usager et le garde intéressé tout le long. En effet, notre outil ne demande à l'usager que les connaissances dont il a réellement besoin pour la réalisation de ses tâches pour éviter le risque que celui-ci ne se désintéresse.

L'utilisation de la technologie XML donne l'avantage de pouvoir inclure une logique de traitement des informations et d'automatiser les traitements, et ce d'une manière indépendante de la plate-forme utilisée.

Limites

Bien que notre application comporte les avantages que nous venons d'énoncer, il présente cependant certaines limites que nous retrouvons au niveau des fonctionnalités. Par exemple, la base de patrons disponibles reste limitée. En effet, dans un premier temps, nous avons considéré les différents patrons possibles en nous basant sur l'application immédiate de notre outil à son domaine. Ainsi nous disposons de trois patrons disponibles dans la base de patrons.

Il est possible de créer de nouveaux patrons. Cependant, le module de création des patrons n'étant pas complètement intégré, l'ajout de nouveaux patrons dans la base de patrons ne se fait pas tout à fait automatiquement. Nous pourrions corriger facilement cette limite en intégrant le module de création de patrons.

Une autre limite se rapporte à la localisation et l'emplacement physique sur le disque du serveur sur lequel se fait le transfert des fichiers générés après leur sauvegarde. En ce moment, chaque début de session durant laquelle l'application est utilisée, il faut un nouvel espace de travail. La structure du répertoire de travail sur le serveur nécessite une copie de

sauvegarde du dossier des fichiers de la session précédente pour libérer l'espace de travail actuel.

Ces actions se font manuellement parce que l'emplacement du répertoire de travail sur le serveur est fixe et qu'aucune procédure n'a été mise en place pour tenir compte d'une migration éventuelle de ces fichiers.

6.4 Conclusion

Dans ce chapitre, nous avons présenté l'expérimentation réalisée avec notre outil. Nous avons présenté les tests réalisés ainsi que les résultats obtenus. De plus, nous avons discuté des avantages et des limites observés ainsi que des avenues possibles pour améliorer l'outil. Ce chapitre termine la présentation de nos travaux dans le cadre de notre étude.

Le chapitre suivant fait la synthèse de notre travail de recherche en rappelant la problématique sur laquelle nous nous sommes penché ainsi que la solution retenue. De plus les améliorations possibles sont proposées pour l'outil ainsi que des travaux futurs pour étendre nos travaux de recherche.

7 Chapitre 7 : Conclusion

Dans un contexte d'ingénierie des connaissances utilisant le paradigme de modélisation, d'acquisition et d'exploitation des connaissances, nous avons été amené à approfondir la notion d'acquisition de connaissances dans un domaine éducationnel. Ceci nous a conduit dans un premier temps à comprendre les étapes de modélisation des SBC. Ensuite, dans un deuxième temps, nous avons mis en pratique notre compréhension de la modélisation des SBC.

Notre étude a porté, précisément, sur les EIAH à partir d'exemples. L'objectif initial de notre travail était de proposer une approche qui permettait de développer graduellement un SBC en lui assurant une nature extensible. De plus cette approche devait permettre d'aider l'enseignant à construire les connaissances du système considéré.

Les systèmes d'aide à l'apprentissage humain à partir d'exemples sont un type particulier d'EIAH. Dédiés notamment à la présentation d'exemples relatifs à un domaine d'étude ou à un thème d'apprentissage précis, ils permettent aux apprenants d'acquérir de nouvelles connaissances ou d'approfondir leurs connaissances sur la matière, en s'auto-expliquant les stratégies de résolution des problèmes posés dans les exemples.

Les systèmes d'aide à l'apprentissage humain permettent de répondre aux besoins de l'apprentissage à partir de présentation d'informations, de traitement des connaissances, de communication entre l'humain et la machine ou de collaboration entre humains (Tchounikine 2002).

Dans ce contexte, l'application que nous voulions développer, et qui est le résultat de nos travaux de recherche, visait à permettre la gestion des connaissances et à assurer le traitement des informations afin de faciliter la communication entre l'enseignant et le système. Nous avons proposé un modèle d'outil, assistant à l'acquisition des connaissances, qui permet à l'enseignant de définir et de modifier les composantes de son modèle conceptuel ainsi que les connaissances de son domaine. L'intégration du modèle conceptuel dans le système exécutable permet de réaliser notre objectif de départ.

Dans ce chapitre nous résumerons les travaux réalisés dans le cadre de ce mémoire, puis nous énoncerons quelques avenues permettant de les étendre.

7.1 Rappel de la problématique

Notre problématique visait à faciliter le rôle de l'enseignant dans la construction de contenu pédagogique dans le cadre de systèmes d'aide à l'apprentissage humain à partir d'exemples. Développés pour favoriser l'apprentissage de l'apprenant, ces systèmes sont le plus souvent orientés vers l'apprenant. Nous nous proposions d'approfondir la notion d'acquisition des connaissances pour aider l'enseignant à construire les connaissances du système éducationnel, aspect pas ou peu abordé dans les travaux de recherche. Nous avons voulu mettre à profit les résultats des travaux de recherche en acquisition des connaissances des SBC afin de les appliquer aux systèmes d'aide à l'apprentissage humain.

Dans le chapitre 2, nous avons précisé certaines notions sur les SBC et avec leurs caractéristiques et leur architecture de base. Nous avons également discuté les différentes caractéristiques du procédé d'acquisition des connaissances dont nous avons tenu compte pendant la modélisation et la réalisation de notre outil.

Ensuite, nous avons traité certains aspects d'évolution des SBC en précisant les difficultés rencontrées par les méthodes de construction des premiers SBC (nature extensible, réutilisabilité) et le niveau des connaissances qui permet aux méthodes actuelles de dépasser ces difficultés. Ceci nous a permis de justifier notre motivation en ce qui concerne le caractère extensible d'un SBC.

Nous avons présenté différentes méthodes d'ingénierie qui traitent du processus de développement des SBC. Nous avons détaillé le processus d'acquisition des connaissances de manière à faire ressortir une façon générale d'organiser ce processus. L'accent a été mis sur le processus d'acquisition et les principes de modélisation au niveau des connaissances et au niveau opérationnel.

Dans le troisième chapitre, nous avons décrit la problématique sur l'acquisition des connaissances plus précisément dans le cadre des systèmes d'aide à l'apprentissage humain Après avoir précisé la problématique identifiée sur l'acquisition des connaissances, nous

avons présenté le cadre méthodologique que nous avons adopté pour mener notre analyse avec les éléments et les méthodes constitutifs. Notre analyse des méthodes d'ingénierie au chapitre 2 a permis de justifier notre choix de la méthode CommonKADS qui a constitué le cadre d'analyse de notre démarche.

Dans le quatrième chapitre, nous nous sommes consacré à la description des éléments pertinents de notre méthode d'analyse. Plus précisément, nous avons présenté en détail les différents modèles que nous avons élaborés à l'aide du formalisme de la méthode CommonKADS pour l'application que nous voulions construire. Par exemple, nous avons décrit, le modèle de tâches qui est une spécification du processus d'acquisition des connaissances, le modèle du domaine qui définit les structures manipulées par les tâches que nous avons décrites, etc. Nous avons décrit un modèle du processus d'acquisition des connaissances basé sur le paradigme de l'acquisition de connaissances guidée par un modèle. Ceci a eu pour but de permettre la construction d'un outil qui assiste l'enseignant dans sa tâche de définition des connaissances requises par le système pour créer un exemple. Dans ce chapitre, l'accent a été mis sur les modèles qui se rapportent à l'analyse.

Dans le cinquième chapitre, nous avons élaboré différents aspects de conception et de réalisation d'un prototype de notre application. Nous avons d'abord présenté le modèle de conception. Par la suite, les résultats du prototypage ont été illustrés. Nous avons proposé une architecture fonctionnelle pour notre système en décrivant la plate-forme sur laquelle ce dernier a été implanté. Ensuite, nous avons décrit en détail les modules qui composent l'architecture. L'application des modèles d'analyse à notre architecture nous a permis d'obtenir une spécification de notre outil à partir de laquelle nous avons pu concevoir un premier prototype. Enfin, le prototype a été présenté avec ses interfaces en illustrant les diverses fonctionnalités qui ont été implémentées dans le système.

Enfin dans le sixième chapitre, nous avons présenté différents éléments constituant l'expérimentation que nous avons faite de notre outil. Cette expérimentation a permis de valider notre travail et de dégager les avantages et les inconvénients de l'outil. À la fin du chapitre nous avons discuté des perspectives d'extension dans le but d'améliorer l'outil d'acquisition des connaissances proposé.

7.2 Rappel de la solution

Notre objectif était de développer une méthode d'acquisition de connaissances pour un système d'aide à l'apprentissage humain à partir d'exemples, puis d'implanter un prototype logiciel supportant cette méthode et qui permet à l'enseignant de faire la gestion des exemples destinés au système.

Dans le but de nous faire une première idée de notre solution, nous nous sommes consacré à l'étude des travaux de recherche portant sur les outils d'acquisition des connaissances. De là, nous avions une meilleure compréhension du processus d'acquisition des connaissances et de la notion d'intégration des modèles dans les outils.

En effet, nous avons compris l'importance d'exprimer les modèles abstraits d'une manière explicite et de les rendre disponibles dans le système final. Nous avons alors orienté nos travaux vers une approche visant à intégrer le contenu de notre modèle conceptuel dans le système exécutable. Ainsi, il serait possible de mettre à jour la spécification du système grâce à la communication entre les niveaux conceptuel et exécutable, ce qui réalisera sa nature évolutive.

L'architecture de notre outil a montré qu'il représente un environnement de développement du système d'acquisition. Notre outil possède également les fonctionnalités d'un éditeur de BC. L'enseignant peut maintenir ou enrichir les informations contenues dans la BC, d'une manière conviviale. Les modifications apportées peuvent s'appliquer à différentes facettes du contenu de la BC, notamment la description des tâches du SBC, de son domaine et de son comportement. Une fois le noyau du système établi, il est possible d'utiliser le SBC et de le faire évoluer sans avoir besoin d'un nouveau cycle de développement. Grâce à l'intégration d'un contenu correspondant à un haut niveau d'abstraction dans le système exécutable, nous avons atteint le but de la modélisation et de l'expérimentation effectuées dans le cadre de notre travail.

7.3 Améliorations possibles, travaux futurs

Cette section présente diverses avenues pour des travaux futurs en décrivant les améliorations possibles dans le cadre de l'expérimentation présentée dans ce mémoire.

Considérant que le travail de modélisation de notre outil n'est pas terminé, nous proposons d'abord un premier travail pour le compléter. Ce travail concerne la réalisation de primitives de modélisation applicables aux inférences. Par exemple, ces primitives pourraient permettre de créer les composantes réalisant les inférences de base qui peuvent être à disposition et avec lesquelles notre outil sera plus complet. Les inférences seront alors réalisées par un ensemble des règles traitant des faits chargés dynamiquement. De cette manière, nous obtiendrons des inférences encore plus génériques.

Ces inférences serviront à résoudre les méthodes utilisées pour réaliser les tâches de l'outil. En étant plus génériques et compte tenu du fait que plusieurs inférences peuvent servir à résoudre une même méthode, il sera possible d'améliorer la stratégie de résolution de la tâche réalisée à travers la manière dont les inférences seront appliquées.

Une deuxième orientation possible est l'utilisation d'un agent dans l'intégration de bases de données existantes. Grâce à l'aspect de méta-modélisation, notre approche peut prendre en considération l'existence d'une masse de données dont les modèles se chevauchent mais couvrent un même domaine. Ainsi notre application pourrait exploiter la richesse contenue dans les données pour constituer une mémoire de l'organisation afin d'aligner les différentes perspectives de modélisation à celles de l'expert selon les buts d'utilisation de données. La possibilité de définir le modèle au niveau des connaissances, que notre outil offre, dans ce contexte peut s'avérer très utile.

Un autre travail vise à intégrer entièrement le module de création de patrons à notre outil. Ainsi il sera possible d'y accéder depuis le menu contextuel de notre application dans le but d'offrir une utilisation encore plus conviviale.

7.4 Mot de la fin

Dans ce dernier chapitre du mémoire, nous avons rappelé notre but qui était dans un premier temps d'approfondir la notion d'acquisition de connaissances dans un domaine éducationnel puis dans un deuxième temps de proposer une approche qui permettait de développer graduellement un SBC en lui assurant une nature extensible et évolutive.

Nous avons résumé le contenu du mémoire. Nous avons fait un rappel de la problématique sur laquelle a porté notre attention dans ce mémoire. Également, un rappel de la démarche entreprise pour proposer notre solution a été fait, avec les résultats obtenus.

Ensuite, nous avons proposé des orientations possibles pour des travaux futurs permettant de poursuivre la modélisation et l'expérimentation présentées dans ce mémoire. Ces orientations visent à compléter le contenu de notre application, étendre l'application de notre outil et améliorer le caractère convivial de son utilisation.

Bibliographie

Altman, R.B., Weiser, B., et Noller, H.F. (1994). *Constraint satisfaction techniques for modeling large complexes: Application to central domain of the 16s ribosomal subunit*. Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, Stanford, CA, pp. 10-18.

Breuker, J.A., et Van de Velde W. (1994). *The CommonKADS Library for Expertise Modelling*. Amsterdam: IOS Press, the Netherlands.

Capus, L., Potvin, B., Tourigny, N. (2002). *Retour d'expérience sur l'enseignement de l'intelligence artificielle : vers un environnement d'apprentissage Web*. TICE 2002, Lyon 13-15 novembre, p. 29-38.

Card, S.K., Moran, T.P., Newell A. (1980). *The keystroke-level model for user performance with interactive systems*, CACM Vol 23, pp. 396-410.

Card, S.K., Moran, T.P., Newell A. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Ass. NJ.

Chandrasekaran, B. (1986). *Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design*, IEEE Expert 1(3). Amsterdam: IOS Press, pp. 23-30.

Charlet, J., Zacklad M., Kassel G. et Bourigault D. (1999) *Ingénierie des connaissances. Évolutions récentes et nouveaux défis*, Collection technique et scientifique des télécommunications, Eyrolles, Paris.

David, J.M., Krivine, J.P. Simmons, R. (1993). *Second Generation Expert Systems*, Springer-Verlag, Heidelberg/Berlin, pp. 273-298.

Dzida, W. (1989). *The development of ergonomic standard*. ACM SIGCHI Bulletin, Vol. 20, 3, 35-43.

Eriksson, H., Shahar, Y., Tu, S.W., Puerta, A.R., Musen, M.A. (1995). *Task modeling with reusable problem-solving methods*. Artificial Intelligence, 79(2), pp. 293-326.

Fensel, D. (1994). *A comparison of languages which operationalize et formalize KADS models of expertise*. The Knowledge Engineering Review, 9(2), pp. 105-146.

Gennari, J.H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson H., Noy N. F. et Tu S. W. (2002). *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development*. Technical Report. Stanford, CA.

Gennari, J.H., Grosso, W. Musen, M.A. (1998). *A method-description language: An initial ontology with examples*. Proceedings of 11th Workshop on Knowledge Acquisition, Modeling, and Management 1998. (KAW'98), Banff, Alberta, Canada, 18-23.

Gil, Y., Melz, E. (1996). *Explicit representations of problem-solving strategies to support knowledge acquisition*. Proceedings of the Thirteenth National Conference on ArtificialIntelligence. Menlo Park, CA. AAAI Press/MIT Press. pp. 469-476.

Gould, J.D., Lewis, C. (1985). *Designing for usability: Key Principles and What Designers Think*. CACM, 28(3), pp. 300-311.

Goldberg, A. (1990). *Information Models, Views, and Controllers*. Dr. Dobb's Journal. July.

Grosso, W. E., Eriksson H., Fergerson R. W., Gennari J. H., Tu S. W., Musen M. A. (1999). *Knowledge Modeling at the Millennium (The Design and Evolution of Protege-2000*). Proceedings of the 12th workshop on knowledge acquisition, modeling and management (KAW'99), Banff, Alberta, Canada, 16-21 Octobre.

Gruber, T. R. (1993). *Toward principles for the design of ontologies used for knowledge sharing. Presented at the Padua workshop on Formal Ontology*, International Journal of Human-Computer Studies, March 1993, Vol. 43, Issues 4-5, November 1995, pp. 907-928.

Leclerc, S., de Maisonneuve, S., Châteauvert, J. (1998) *Inciter les sourds à l'auto-explication*, Lecture et surdité : visions actuelles et nouvelles perspectives, C. Dubuisson et D. Daigle (éd.), Montréal : Les Éditions Logiques, p. 321-334.

Marcus, S. (1988). *Preliminary work toward a knowledge-acquisition tool*. Automatic Knowledge Acquisition for Expert Systems, Boston, MA: Kluwer Academic Publishers, pp. 201-24.

Musen, M.A., Gennari, J.H., Park, J. H. (1998). *Mappings for Reuse in Knowledge-Based Systems*. Proceedings of 11th Workshop on Knowledge Acquisition, Modeling, and Management 1998. (KAW'98), Banff, Alberta, Canada, 18-23 Avril.

Musen, M. A., Fergerson, R. W., Grosso, W. E., Noy, N. F., Crubezy, M., Gennari J. H. (2000) *Component-Based Support for Building Knowledge-Acquisition Systems*. Conference on Intelligent Information Processing (IIP 2000) of the International Federation for Information Processing World Computer Congress (WCC 2000), Beijing.

Musen, M.A, Gennari J.H., Eriksson H., Tu, S.W. Puerta, A.R. (1995). *Protégé-II: Computer support for development of intelligent systems from libraries of components*. Proceedings of Medinfo '95, Vancouver, pp. 766-770.

Musen, M.A., Gennari, J.H., Tu, S.W. Rothenfluh, T.E. (1994). *Mapping domains to methods in support of reuse*. International Journal of Human-Computer Studies 41(3) pp. 399-424.

Newell A. (1982). *The Knowledge Level*, Artificial Intelligence, vol. 18, n° 1, 1982, 87-127.

Nielsen, J., et Molich, R. (1990). *Improving a Human-Computer Dialogue*. CACM, 33(3), pp. 338-348.

Noy, N. F., Fergerson, R. W., Musen, M. A. (2000). *The knowledge model of Protege-2000: Combining interoperability and flexibility*. 2nd International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France, pp. 17-32.

Puerta, R., Tu, S. W., Musen, M. A. Ford K.M. Bradshaw J.M., (1993). *Knowledge Acquisition as a Modeling Activity*. Technical report. John Wiley, New York. pp. 129-152.

Rothenfluh, T.E., Gennari, J.H., Eriksson, H., Puerta, A.R., Tu, S.W. Musen, M.A. (1996). *Reusable ontologies, knowledge-acquisition tools, and performance systems: PROTÉGÉ-II solutions to Sisyphus-2*. International Journal of Human-Computer Studies, 44(3-4), pp. 303-332.

Runkel, J.T., Birmingham, W.P. (1994). *Separation of knowledge: a key to reusability.* Gaines, B.R. Musen, M. A., Proceedings of the 8th Bank of Knowledge Acquisition for Knowledge-Based Systems Workshop Canada, Alberta, SRDG Publications, University of Calgary, pp. 361-369.

Schreiber, G., Wielinga, B.J, De Hoog, R., Akkernians, FI. Van de Velde, W. (1994). *CommonKADS : A Comprehensive Methodology for KBS development*, IEEE Expert, Vol.9, No.6, pp. 28-29.

Schreiber, G., Wielinga, B. Breuker. J.A. (1993). KADS *A Principled Approach of Knowledge-Based System Development*, Academic Press, London, pp. 119-149.

Schreiber, G., Wielinga, B.J., Akkermans, H., Van de Velde, W. Anjewierden, A. (1994). *CML: The CommonKADS Conceptual Modeting Language*. 8th European Knowledge Acquisition Workshop EKAW'94 Hoegaarden, numéro 867, Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, Heidetberg/Berlin, pp.2-25.

Shadbolt, N.R. Wielinga, B.J. (1990). *Knowledge-based knowledge acquisition: the next generation of support tools*. Current Trends in Knowledge Acquisition, 105 Press, Amsterdam, pp. 313-338.

Shahar, Y., Miksch, S. Johnson, P. (1997). *A Task-Specific Ontology for the Application and Critiquing of Time-Oriented Clinical Guidelines*. Artificial Intelligence in Medicine, Proceedings of 11th Conference on Artificial Intelligence in Medicine, Europe, AIME 97 Grenoble, Springer-Verlag, I-leidelberg/Berlin, pp. 51-61.

Shaw, M.L.G. Gaines, B.R. (1987). *An interactive knowledge elicitation technique using personal construct technology*. Kidd, AL., éditeur, Knowledge Acquisition for Expert Systems: Practice Handbook, Plenum Press, New York.

Shneiderman, B. (1998). *Designing the User Interface: Stratégies four Effective Human-Computer Interaction*, Addison-Wesley.

Simian, G. Tourigny, N. (1996). *Conception et architecture des systèmes experts*, Notes de Cours, Université Laval, Québec.

Sowa, F. J. (1992). *Conceptual Graphs*, Technical Report on the IRDS Conceptual Schema ANSI.

Spider, R., Benjamins, R. Fensel, D. (1998). *Knowledge Engineering Principles and Methods*. Data and Knowledge Engineering, Vol.25, pp. l61-197.

Steels,L., Schreiber, G. Van de Velde, W. (1994). *A Future for Knowledge Acquisition*, 8th European Knowledge Acquisition Workshop EKAW 94 Hoegaarden, numéro 867, Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, Heidelberg/Berlin, pp. 394-413.

Steels L. (1993) *The Componential Framework and its Role in Reusability*, DAV 93, pp. 273-298.

Studer, R., Fensel, D. (1999) *Knowledge Acquisition, Modeling and Management:* 11th European Workshop, EKAW'99: Proceedings Springer-Verlag

Studer R., Benjamins V.R. Fensel D. (1998) *Knowledge Engineering: Principles and Methods*, Data and Knowledge Engineering, vol. 25, 1998, 161-197.

Stylianou, C.A., Madey, R.G. Smith, D.R. (1992). *Selection Criteria for Expert System* Shells A Socio-Technical framework, Communication of the ACM, Octobre, Vol.35, pp. 30-49.

Swartout, R.W. Moore, D.J. *Explanation in Second Generation Expert Systems*. Second Generation Expert Systems, Springer-Verlag, Heidelberg/Berlin, 1993, pp. 543-545.

Tansley, D.S.W. Haybail, C.C. (1993). Knowledge-*Based System Analysis and Design: a KADS developer's handbook*, Prentice-Hall, New York.

Tchounikine, P. (2002). *Pour une ingénierie des EIAH*. Assises du GDR I3-5, Décembre.

Terpstra, P., Van Heist, G., Shadbolt, N. Wielinga, B. (1993). *Knowledge Acquisition process Support through Generalised Directive Models*. Second Generation Expert Systems, Springer-Verlag, Heidelberg/Berlin, pp. 3-24.

Tu, S. W., Kahn, M. G. Musen, M. A. (1989). *Episodic skeletal-plan refinement based on temporal data*. Communications of the ACM, 32(12): 1439-1455.

Tourigny, N. (1990*). Méthodologie de développement des systèmes à base de connaissances*. Mémoire de maîtrise, Département d'informatique, Université Laval, Québec.

Van de Velde, W. (1993). *Issues in Knowledge Level Modelling*. Second Generation Expert Systems, Springer-Verlag, Heidelberg1'Berlin, pp. 211-231.

Van Heist, G., Schreiber, G. Wielinga, B.J. (1997*). Using Explicit Ontologies in KBS Development, international Journal of Human Computer Studies*, 46(2-3), 1997, pp. 183-292.

Van Heist, G., Terpstra, P., Wielinga, B.J. Shadbolt, N.R. (1992). *Using generalised directive models in knowledge acquisition*. Current Developments in Knowledge Acquisition, 61st European Knowledge Acquisition Workshop EKAW'92, Heidelberg/Berlin, pp. 112-132.

Van Heist, G. Schreiber, G., (1994). *Ontology Based Knowledge Acquisition*, European Knowledge Acquisition Workshop EKAW94 Hoegaarden, numéro 867, Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, Heidelberg!Berlin, pp. 178-199.

Vicat, C., Busac, A. Ganascia, .J.G. (1993). *CERISE, A Cyclic Approach for Knowledge Acquisition.* Knowledge Acquisition for Knowledge-Based Systems, 7 European Knowledge Acquisition Worlcshop EKAW 93 Toulouse, numéro 723, Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, Heidelberg/Berlin, pp.237-255.

Vogel, C. (1990). KOD, *une méthode pour le recueil et la modélisation des connaissances*. Cours pour les dixièmes Journées Internationales sur les systèmes experts et leurs applications, Avignon.

Waterman, D. A. (1986). *A Guide to Expert Systems*, Addison-Wesley, New York.

Wielinga, B.I. Breuker, T.A. (1985). *Interpretation of verbal data for knowledge acquisition*. Advances in Artificial Intelligence, Elsevier Science Publisher B.V. North Holland, pp. 312-328.