Université de Montréal

Développement d'un curriculum et d'un modèle de l'apprenant orientés vers un domaine multidisciplinaire : l'informatique quantique

par
Sébastien Gambs

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures en vue de l'obtention du grade de Maîtrise ès Science (M.Sc.) en informatique

Août, 2003

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé :

Développement d'un curriculum et d'un modèle de l'apprenant multidisciplinaires orientés vers un domaine multidisciplinaire :
l'informatique quantique

présenté par :

Sébastien Gambs

a été évalué par un jury composé des personnes suivantes :

Petko Valchev
président-rapporteur

Esma Aïmeur
directrice de recherche

Gilles Brassard
codirecteur

Alain Tapp
membre du jury

Résumé

L'informatique quantique est la science qui étudie les principes de la mécanique quantique à des fins de traitement de l'information. C'est un domaine encore jeune mais très prometteur, qui permet de réaliser certains exploits hors de portée de l'informatique classique. Un des problèmes importants que rencontre ce domaine est que malgré sa popularité grandissante, il compte encore relativement peu de spécialistes. Un système tutoriel intelligent (STI) enseignant ce domaine pourrait offrir une solution partielle à ce problème en permettant aux personnes n'ayant pas d'expert sous la main de se familiariser quand même avec le domaine.

C'est le but du projet QUANTI d'arriver à terme à la construction d'un tel système. Dans ce mémoire, on s'intéressera à la construction des deux premières composantes fondamentales du STI que sont le curriculum et le modèle de l'apprenant. On verra en particulier la méthode d'acquisition développée pour gérer l'inhérente multidisciplinarité de l'informatique quantique. On cherchera aussi à évaluer HAKE, un nouvel algorithme d'élicitation, et finalement on utilisera un algorithme d'apprentissage machine du nom de CLARISSE pour proposer une alternative à l'initialisation du modèle de l'apprenant.

Mots clés : système tutoriel intelligent, informatique quantique, multidisciplinarité, curriculum, acquisition des connaissances, algorithme d'élicitation, initialisation du modèle de l'apprenant.

Abstract

Quantum Information Processing is the science studying the principles of quantum mechanics for information processing purposes. It is a relatively young but promising field, which can realize wonders out of reach of classical computing. Despite its evergrowing poularity, this field has a major problem that it still has very few specialists. By enabling even people who don't have an expert close to them to get familiar with the domain, an intelligent tutoring system (ITS) teaching this domain could offer a partial solution to this problem.

It's the purpose of the QUANTI project to finally build such a system. In this thesis, we will be interested in the construction of the first two fundamentals components of the ITS that are the curriculum and the student model. In particular, we will look at the knowledge acquisition method developed in order to deal with the inherent multidisciplinarity of Quantum Information Processing. We will also try to evaluate a novel elicitation algorithm called HAKE, and finally we will use a machine learning algorithm called CLARISSE as a tool for the initialization of the student model.

Keywords : Intelligent Tutoring System, Quantum Information Processing, Multidisciplinarity, Curriculum, Knowledge Acquisition, Elicitation Algorithm, Initialization of the Student Model.

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et des abréviations

STI : Système Tutoriel Intelligent.

Qubit : abréviation de Quantum Bit (ou bit quantique en français).

CLARISSE : CLuster And Rules ISSuEd.

HAKE : Hybrid Algorithm for Knowledge Elicitation.

Aux personnes à qui je tenais et qui ne sont plus là, mais surtout à celles que j'ai encore la chance d'avoir.

Remerciements

Avant tout j'aimerais remercier mes directeurs de recherche, Esma Aïmeur et Gilles Brassard, pour l'aide et le soutien qu'ils m'ont apportés pendant ma maîtrise, et aussi pour toutes les discussions intéressantes que nous avons eu l'occasion d'avoir durant cette période ainsi que toutes les idées échangées. De par leurs connaissances, et même leurs caractères, je pense qu'ils sont très complémentaires et que leur association forme une synergie exceptionnelle.

Je voudrais ensuite remercier toutes mes camarades à la fois du laboratoire Héron, et à la fois du laboratoire d'informatique théorique et quantique (LITQ). Je suis très heureux d'avoir pu à la fois vivre dans deux laboratoires qui ont des domaines de compétence aussi différents, ça a été pour moi une expérience à la fois passionnante et très enrichissante. L'ambiance et la façon de travailler sont très différentes dans les deux laboratoires, mais elles ont en commun d'être très intéressantes et agréables dans les deux cas. En particulier je voudrais remercier Serge, Maryam, Meriem et Manu pour leur amitié et leur soutien.

Enfin j'aimerai remercier bien sûr ma famille ainsi que mes amis proches, et en particulier mes potes d'enfance qui ont une très grande importance à mes yeux. Merci simplement pour tout ce qu'ils sont et ce qu'ils ont fait depuis le début.

Chapitre 1 : Introduction

L'informatique quantique constitue un nouveau paradigme de traitement de l'information qui s'appuie, non pas sur les lois de la physique « classique », mais sur les principes de la mécanique quantique. C'est une science encore jeune, mais qui est en train de connaître un intérêt croissant depuis quelques années, et pour laquelle il y a encore relativement peu d'experts. Les systèmes tutoriels intelligents (ou STI) quant à eux appartiennent à un domaine un peu plus mature, et s'intéressent à la création de logiciels pouvant adapter son enseignement aux besoins d'un étudiant à l'aide de techniques issues de l'intelligence artificielle. De leur rencontre est né le projet QUANTI et l'idée de créer un STI portant sur l'informatique quantique. Un avantage direct d'avoir un tel système à disposition serait de pouvoir faciliter la découverte de cette matière en permettant un accès à un plus grand nombre de personnes. Ceci s'ajoute au bénéfice habituel d'un STI qui est de pouvoir être utilisé en complément d'un cours pour aider un étudiant à progresser dans sa compréhension de la matière.

Une spécificité importante de l'informatique quantique est son aspect multidisciplinaire. En effet, l'informatique quantique se situe au carrefour des quatre disciplines que sont l'informatique, les mathématiques, la physique et la chimie. Cette multidisciplinarité apporte une certaine richesse que n'ont pas d'autres domaines plus « classiques », en permettant par exemple l'adaptation de l'enseignement du STI en fonction du background disciplinaire de l'étudiant, mais cela peut aussi compliquer certaines tâches comme celle de l'acquisition des connaissances.

Voici comment s'organise ce mémoire. Tout d'abord, le Chapitre 2 introduira la notion de système tutoriel intelligent plus en détails, en explicitant notamment les composants essentiels du STI que sont le curriculum, le modèle de l'apprenant et le tuteur.

Dans le Chapitre 3, on regardera dans un premier temps quelles sont les méthodes d'acquisition des connaissances existantes pouvant être utilisées pour créer une base de

connaissances dans un contexte général, puis on cherchera à voir plus spécifiquement ce qui se fait du côté des STI.

Un survol de l'informatique quantique sera présenté dans le Chapitre 4. Suivront dans ce même chapitre, une explication des motivations ayant conduit à la création du projet QUANTI et une brève description de celui-ci.

Ensuite, dans le Chapitre 5, on présentera la structure du curriculum ainsi que la méthode d'acquisition des connaissances développée dans le cadre de QUANTI. Cette méthode a été conçue en vue de gérer le côté multidisciplinaire de l'informatique quantique.

Dans le Chapitre 6, on explorera un peu plus loin la piste de l'acquisition des connaissances en évaluant HAKE, un nouvel algorithme d'élicitation. HAKE est un hybride entre les variantes pour l'élicitation des algorithmes de profondeur d'abord et largeur d'abord. On cherchera ici à comparer HAKE à ses deux cousins proches afin de déterminer objectivement son efficacité.

Le chapitre 7 s'intéressera à un problème fondamental et difficile du domaine des STI, à savoir celui de l'initialisation du modèle de l'apprenant. On cherchera à découvrir, à l'aide de l'algorithme de catégorisation CLARISSE, les catégories naturelles d'étudiants existants en informatique quantique. Le but est d'utiliser cette information afin de classer un nouvel étudiant dans sa catégorie, et d'initialiser ensuite son modèle de l'apprenant en conséquence.

Finalement, on conclura en ouvrant quelques pistes possibles pour la suite du projet QUANTI et en résumant le travail apporté par ce mémoire.

Chapitre 2 : État de l'art des systèmes tutoriels intelligents

2.1. Définition

On peut définir un *système tutoriel intelligent* ou *STI*, comme un logiciel assurant l'enseignement d'une matière et utilisant des techniques d'intelligence artificielle afin de pouvoir adapter son enseignement aux besoins de chaque étudiant. Le but visé étant que l'enseignement individualisé délivré par le STI contribue à améliorer les performances de l'apprenant [Bloom 84] soit en étant utilisé comme complément à un cours traditionnel, soit en lui permettant d'obtenir une meilleure compréhension du domaine comparé à un apprenant n'ayant pas suivi le cours traditionnel.

Les systèmes tutoriels intelligents se situent au confluent des trois disciplines que sont *l'éducation*, *l'informatique* et *la psychologie* (cf. Figure 1). Puisqu'il s'agit d'enseignement, il est évident que les théories issues du domaine de l'éducation ont ici une place primordiale. L'informatique, quand à elle est utilisée comme vecteur d'enseignement, c'est à travers elle que la connaissance sera véhiculée. Enfin, l'apport de la psychologie est essentiel lorsqu'il s'agit d'essayer de comprendre comment fonctionne l'esprit d'un apprenant, et qu'il faille émettre des hypothèses quant à ses connaissances ou ses modes de raisonnement.

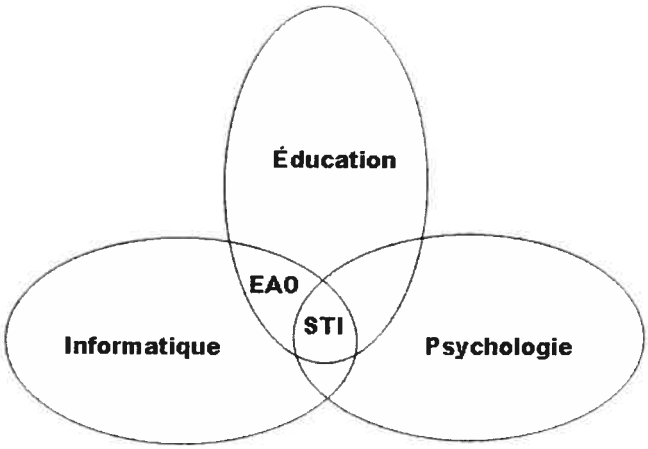

Figure 1 : Au confluent de trois disciplines.

2.2. Historique

Quand on se penche sur l'histoire occidentale mais en se concentrant sur l'aspect éducation, on remarque qu'à quelques exceptions près il n'y a pas eu de grandes révolutions durant ces deux derniers millénaires. L'une de ces exceptions étant l'instauration de l'école laïque, gratuite et obligatoire pour tous avec la loi Jules Ferry à la fin du XIX^{ième} siècle en France ou encore la loi de l'instruction publique obligatoire en 1943 au Québec. En facilitant ainsi l'accès à l'éducation à un grand nombre de personnes, elles ont ainsi contribué à faire basculer le schéma typique d'apprentissage d'un précepteur vers *son* pupille à une situation du type un professeur vers *ses* élèves. Ensuite au cours du siècle dernier, ce schéma a relativement peu évolué, même si les interactions entre le professeur et les élèves ont augmenté en intensité au fur et à mesure que la « distance académique » les séparant s'amenuisait. Il n'en demeure pas moins que la méthode consistant à faire un enseignement d'une personne à une *seule* autre personne reste la plus efficace, même si elle est aussi la plus onéreuse, car elle permet d'offrir un enseignement plus adapté à l'étudiant. En effet, il semble évident qu'un enseignant ayant un seul pupille aura beaucoup plus de temps pour l'écouter et l'aider que s'il devait s'occuper de vingt élèves en même temps.

L'utilisation de l'informatique comme support de l'enseignement est une autre révolution marquante. L'un des apports des STI a été de refaire basculer le paradigme d'apprentissage de type « un vers plusieurs » à un paradigme de type « un vers un », et de pouvoir généraliser cette méthode à un grand nombre de personnes.

La naissance de *l'enseignement assisté par ordinateur* ou *EAO* remonte aux années 60. Les premiers systèmes d'EAO étaient rudimentaires et les sessions d'apprentissage statiques et identiques d'un apprenant à l'autre. Une session d'apprentissage se résumait à une présentation de la matière, suivie ensuite d'une évaluation de l'apprenant souvent sous la forme d'une succession de *questions à choix multiples* (ou *QCM*). Dépendamment de ses résultats à l'évaluation, l'apprenant est ensuite aiguillonné vers d'autres séquences

d'apprentissage (aussi appelé *frames* [Minsky 75]). Le cheminement de l'apprenant était donc prédéterminé et programmé à même le système.

L'EAO évolue vers l'EIAO (ou *enseignement intelligemment assisté par ordinateur*) au début des années 70, avec l'intégration pour la première fois de techniques provenant de l'intelligence artificielle [Frasson et Gauthier 90], comme l'utilisation de réseaux sémantiques pour représenter les connaissances. SCHOLAR [Carbonell 70], un logiciel enseignant la géographie de l'Amérique du Sud, est un des tous premiers exemples de système de cette génération. Le terme STI [Sleeman et Brown 82] est sémantiquement identique à celui d'EIAO.

2.3. Architecture et composants

Pour l'instant il n'existe encore aucun véritable consensus sur ce que devrait être l'architecture d'un STI (cf. Figure 2), mais en général tout le monde dans le domaine s'accorde sur le fait qu'elle devrait comporter les modules essentiels suivants :

-Un *curriculum*, qui est la base de connaissances contenant la matière du domaine à enseigner.

-Un *modèle de l'apprenant*, qui cherchera à représenter le plus fidèlement possible l'état des connaissances d'un étudiant.

-Un *module pédagogique* ou *tuteur*, qui est en charge d'enseigner la matière à l'étudiant en utilisant différentes stratégies pédagogiques.

L'architecture possède aussi souvent les deux éléments suivants, qui sans être d'une importance aussi cruciale que les modules précédents, n'en restent pas moins essentiels :

-Un *planificateur* dont la tâche est de définir le contenu et la durée des sessions d'apprentissage. Il se sert pour cela des informations contenues dans le curriculum ainsi

que dans le modèle de l'apprenant afin de pouvoir sélectionner un contenu pour une session d'apprentissage qui correspond aux besoins de l'étudiant. Une fois cette sélection effectuée, elle sera transmise au tuteur qui lui sera en charge de l'enseigner.

-Une *interface* qui va gérer les interactions entre l'apprenant-humain et l'enseignant-ordinateur.

À noter que, dans certaines architectures, le tuteur et le planificateur sont regroupés au sein du même module au lieu d'en constituer deux séparés, si l'on considère que déterminer la succession des séances d'apprentissage relève de l'expertise pédagogique. De plus, même si ces différents modules sont souvent séparés conceptuellement dans l'esprit des développeurs d'un STI, il arrive souvent qu'au niveau de la programmation ils ne le soient pas explicitement.

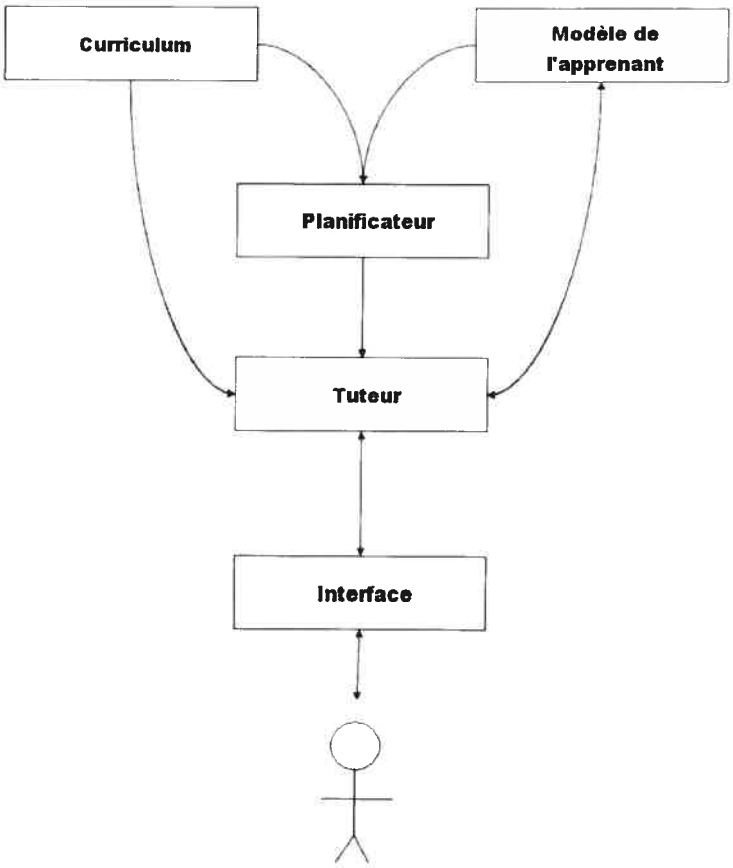

Figure 2 : Architecture d'un système tutoriel intelligent.

On peut établir une analogie très naturelle entre un système tutoriel intelligent et un enseignant, tel que par exemple un professeur d'université. Afin d'être un bon professeur une personne doit réunir trois caractéristiques. Elle doit très bien connaître la matière qu'elle enseigne, pouvoir identifier les forces et faiblesses de ses étudiants et aussi être un bon pédagogue pour transmettre son savoir de façon efficace. Pour un STI, c'est la même chose. Sans une bonne connaissance du domaine, le STI n'aura de toute façon rien de valable à enseigner à l'étudiant (cf. curriculum). S'il est incapable de se faire une idée de l'état des connaissances d'un étudiant, il lui sera alors impossible de pouvoir adapter son enseignement en fonction des besoins spécifiques de cet étudiant (cf. modèle de l'apprenant). S'il ne possède pas de stratégies pédagogiques appropriées, il ne sera pas capable de transmettre ses connaissances efficacement à l'apprenant (cf. tuteur). Un STI possédant toutes ces caractéristiques serait un STI idéal mais, de même que tous les professeurs ne sont pas forcément parfaits et ne possèdent donc pas forcément toutes ces caractéristiques, les STI actuels sont encore loin d'être « idéaux ». Souvent, les STI actuels excellent dans une ou deux de ces caractéristiques et contiennent un modèle simplifié pour implémenter les autres.

2.3.1. Le curriculum

Le but premier du curriculum est de permettre, sous une forme ou une autre, de *représenter* et de stocker les connaissances du domaine. C'est un composant primordial, sans lui il n'y aurait rien à enseigner de toute façon. Il est cependant possible d'aller au-delà de ce but premier en fournissant au curriculum la capacité de *raisonner* sur le domaine. Ainsi, plutôt que de se contenter de fournir des connaissances statiques au curriculum, on peut aussi le doter de la possibilité de raisonner directement sur le domaine et donc de résoudre des problèmes non prévus explicitement à l'avance.

La capacité de réfléchir sur le domaine peut par exemple être encodée dans le système sous forme de *règles*, comme cela se fait pour les systèmes experts. Un des tous premiers systèmes construits sur ce modèle fut GUIDON, qui fut créé directement à partir du

système expert MYCIN dont il utilise les règles de production pour représenter le domaine médical [Clancey 83]. Le principal avantage de l'utilisation de règles est de pouvoir allier en même temps, sans aucun coût supplémentaire, une représentation des connaissances du domaine et une manière de raisonner sur ce domaine. Ce mode de représentation est très bien adapté pour les connaissances *procédurales* mais se prête par contre très mal à la représentation de connaissances *conceptuelles*.

Une représentation plus intuitive du domaine que par règles, est celle par *graphes* et *réseaux*. Les *réseaux sémantiques*, les *graphes conceptuels* et les *frames* appartiennent à ce type de représentation. Dans ce mode de représentation, les *nœuds* du graphe représentent les *connaissances* alors que les *arêtes* représentent les *relations* entre ces connaissances. Les connaissances représentées dans les nœuds peuvent être de plusieurs types : *fait*, *concept*, *procédure* ou *principe* [Merrill 94, Murray 97]. Les relations entre les connaissances peuvent aussi être de différents types : *spécialisation*, *généralisation*, *composition*, *analogie*, etc.

Un curriculum *multi-couches* est une extension naturelle de la représentation sous forme de graphes et de réseaux. Dans ce mode de représentation, plusieurs couches (liens horizontaux) sont reliées entre elles par des liens verticaux au sein d'un même curriculum. Les couches présentes peuvent être de différents types. Par exemple, dans CREAM [Nkambou 96], qui a été le modèle de curriculum développé pour le projet SAFARI, trois types de réseaux coexistent au sein du même curriculum et fusionnent pour former un réseau global appelé CKTN (pour Curriculum Knowledge Transition Network). Les trois types de réseaux sont : *capacités*, *objectifs pédagogiques* et *ressources didactiques*. Le principal intérêt des réseaux multi-couches est de pouvoir combiner au sein d'un seul curriculum plusieurs types de connaissance, ainsi que d'autres éléments liés à l'enseignement à l'instar des objectifs pédagogiques, ou encore à l'évaluation tel que les ressources didactiques.

La plupart des approches en intelligence artificielle sont en général orientées vers la façon de *représenter* les connaissances plutôt que par le *contenu* même des

connaissances, ceci incluant toutes les approches décrites précédemment. Malgré tout, il existe des méthodes qui s'intéressent plus au contenu qu'à la représentation; c'est le cas des *ontologies*. On peut trouver beaucoup de définitions différentes et non équivalentes du mot ontologie dépendamment de ce que l'on veut désigner et du contexte. Parmi les sept définitions énoncées par [Mizoguchi 01], on en retiendra deux. D'abord, celle provenant de la philosophie où ce mot signifie *théorie de l'existence* et qui essaye d'expliquer ce qui existe dans le monde et comment ce dernier est organisé en introduisant un système de catégories pour représenter les choses et leurs relations intrinsèques. Ensuite, celle qui correspond le plus au domaine des STI, et qui dit qu'une ontologie consiste en des *concepts* comportant des définitions qui sont organisées hiérarchiquement, des *relations* entre ces concepts (allant plus loin que des relations du type *is-a* et *part-of*), et des *axiomes* qui formalisent les définitions et les relations. L'approche ontologique se libère donc de savoir comment la connaissance va être utilisée plus tard, en se focalisant plutôt sur le contenu de la connaissance elle-même et ses propriétés. De cette approche résulte deux principaux avantages. D'une part, la connaissance du domaine est représentée d'une manière beaucoup plus profonde et poussée avec une ontologie qu'avec les autres formes de représentations « traditionnelles ». D'autre part, il devient relativement facile de partager ou réutiliser les connaissances. Une ontologie peut être utilisée soit directement en tant que curriculum dans un STI, soit comme brique de construction pour créer la base de connaissances du système.

Lors de la construction d'un modèle de curriculum, un compromis doit souvent être fait entre la *profondeur de représentation* du modèle et sa *généricité*. En général, plus un modèle permet de représenter finement un domaine précis, plus cela le rend non générique et donc difficilement applicable tel quel à d'autres domaines. De plus, pour l'instant une modélisation très fine a seulement pu être réalisée pour des domaines de taille restreinte, pas encore pour des domaines de taille importante. Un domaine de taille restreinte serait typiquement un domaine où il serait possible pour un seul (ou quelques) individu(s) d'énumérer tous les concepts appartenant à ce domaine en un temps raisonnable, ainsi que de préciser les relations entre ces concepts.

2.3.2. Le modèle de l'apprenant

Réussir à modéliser les connaissances de l'apprenant de manière fine et précise reste un des problèmes majeurs des STI [Self 88]. Il est de toute façon impossible de savoir exactement les connaissances de l'apprenant puisque cela supposerait qu'on puisse directement aller regarder à l'intérieur de son cerveau. Il est même difficile de simplement chercher à approximer les connaissances de l'apprenant car la *bande passante* existante entre l'apprenant et le système tutoriel intelligent est en général étroite. Les informations dont on dispose sur l'apprenant sont limitées à celles qui peuvent être déduites des interactions entre l'interface et le système, ainsi qu'aux connaissances préalables qu'on a de lui. Certaines expériences ont été faites afin de tenter d'élargir cette bande passante, tel que par exemple le suivi du regard pour déterminer sur quel région de l'écran se focalise l'attention de l'apprenant [Gluck et *al* 00], mais ce type d'expérience reste encore marginal à l'heure actuelle. Le but d'un bon modèle de l'apprenant va être d'essayer de modéliser le plus fidèlement possible les connaissances de l'apprenant, tout en étant conscient qu'il s'agit là seulement d'une approximation de ce qui se passe réellement dans le cerveau de l'apprenant.

De même que les connaissances de l'apprenant vont évoluer au fur et à mesure de l'apprentissage, le modèle de l'apprenant qui reflète ses connaissances doit accompagner cette évolution et donc ne pas être statique.

Voici les types de services qu'on pourrait attendre d'un modèle de l'apprenant [Dufour 99] :

-*Informer* sur l'état des connaissances de l'apprenant.

-*Évaluer* le niveau global de connaissances de l'apprenant.

-*Choisir* le prochain sujet à enseigner à l'apprenant en regardant quels sont ses lacunes et les sujets qu'il ne maîtrise pas, et en tenant compte du plan global d'apprentissage (cf. planificateur).

-*Expliciter* à l'apprenant les erreurs qu'il a commises, comment il les a réalisées et comment les corriger (cf. curriculum, tuteur).

-*Prédire* le comportement de l'apprenant dans une certaine situation d'apprentissage [Murray et VanLehn 00].
-Permettre à l'apprenant de *scruter* lui-même le modèle, et aussi d'exprimer son avis sur ce qu'il pense maîtriser ou non (développement des capacités de *métacognition* [Gama 01]).

Toutes ces fonctionnalités sont loin d'être toujours implémentées, et même si c'est le cas elles ne sont pas forcément prises en charge directement par le modèle de l'apprenant. Par exemple, le choix du prochain sujet à enseigner est en général laissé à la charge du planificateur qui, à partir des données présentes dans le modèle de l'apprenant et le curriculum, va déterminer le prochain cours. De même, des fonctionnalités comme la prédiction du comportement de l'apprenant ou la possibilité de laisser l'apprenant scruter et interagir avec le modèle, sont encore relativement sophistiquées et peu fréquentes dans les STI actuels.

De toutes les approches de modélisation existantes, la méthode la plus simple et la plus directe est le modèle par *recouvrement* (ou *overlay* model) qui a été inventé par Carr et utilisé pour la première fois dans le système WUSOR II [Carr et Goldstein 77]. Dans ce modèle, on considère les connaissances de l'apprenant comme étant un sous-ensemble des connaissances de l'expert. Pour chaque connaissance formant le domaine, on va chercher à savoir lesquelles sont maîtrisées ou non par l'apprenant. Au fur et à mesure que l'apprenant progresse, ses connaissances se rapprochent de plus en plus de celles d'un expert jusqu'à idéalement les égaler. À cause de la ressemblance très forte entre la manière d'appréhender les connaissances de l'apprenant et la manière de représenter les connaissances du domaine, le modèle par recouvrement dérive souvent sa structure de celle du curriculum. Il est cependant possible d'étendre ce modèle de base en ne regardant pas seulement si l'apprenant maîtrise ou non les connaissances du domaine mais en y ajoutant d'autres informations, comme les connaissances erronées qu'il possède.

Parmi les méthodes de modélisation qui prennent justement en compte les connaissances erronées de l'apprenant, on peut trouver celles se basant sur la théorie des erreurs [Burton 82]. Dans ce type de modèle, appelé *buggy* model, on ne cherche pas seulement à déterminer ce que l'apprenant sait ou ne sait pas, mais aussi à détecter et corriger les erreurs qu'il peut commettre durant l'apprentissage. Le système doit donc être en mesure de détecter une erreur réalisée par l'apprenant, ainsi que de comprendre comment il a raisonné pour arriver à cette erreur. Une fois l'erreur détectée, le système doit aussi être en mesure d'expliciter l'erreur afin que l'apprenant puisse la comprendre et la corriger. Il est possible de gérer les erreurs de plusieurs manières : soit on constitue une librairie d'erreurs énumérant toutes les erreurs auxquelles on aura pu penser; soit on cherche à reconstruire l'erreur en partant des actions effectuées par l'apprenant; soit au contraire, on part d'une erreur observée et on essaye de déterminer quelles sont les raisons qui ont pu conduire à cette erreur. La première approche a comme principal défaut de n'être réalisable que pour des domaines de taille restreinte où il est facile d'imaginer toutes les erreurs pouvant survenir. À l'inverse, les deux autres approches ont l'avantage de pouvoir s'occuper d'erreurs non prévues à la base par le système, mais par contre la taille de l'espace de recherche dans ce cas conduit souvent à une explosion combinatoire devant être endiguée grâce à l'utilisation d'heuristiques. Des systèmes combinant plusieurs approches peuvent être créés, par exemple en gardant en mémoire les erreurs les plus fréquentes à partir desquelles on pourra générer de nouvelles erreurs non prévues explicitement au départ.

Du fait du problème de la bande passante réduite entre l'humain et la machine décrit précédemment, l'information dont le STI dispose sur l'étudiant est au mieux incomplète, et au pire erronée ou incertaine. Différentes solutions issues de l'intelligence artificielle ont été utilisées afin de pouvoir traiter avec cette *incertitude* telles que les *croyances* [Self 88] ou la *logique floue*. Dans un modèle basé sur les croyances, on ne se contente plus de savoir ce que l'apprenant connaît ou non, mais aussi quelle est la confiance que l'on peut avoir en cette information. Le système pourrait par exemple estimer que l'apprenant maîtrise un concept, tout en étant conscient que sa confiance en cette information est faible, car provenant d'une source peu fiable. Au contraire, si le système a pu interroger

directement l'apprenant sur ce sujet (par exemple avec un questionnaire), sa confiance en cette information devrait être relativement élevée. Il est possible de combiner les croyances de l'étudiant avec celles du système dans un même modèle, comme cela peut se faire avec les modèles de l'apprenant autorisant l'utilisation de capacités de métacognition et permettant l'inspection par un étudiant de son propre modèle de l'apprenant. La logique floue est un autre formalisme pouvant être utilisé pour traiter l'incertitude. Dans la logique floue, un objet n'est pas restreint à appartenir (ou ne pas appartenir) de façon absolue à un ensemble comme c'est le cas dans la logique classique avec les valeurs booléennes *vrai* ou *faux*, mais on dit qu'il *appartient à un certain degré à cet ensemble*. On modélise généralement ce degré d'appartenance par une valeur comprise entre 0 et 1, 1 correspondant à la valeur booléenne vrai et 0 à faux. Cela permet une plus grande flexibilité dans la représentation de l'apprenant. Ainsi, au lieu d'appartenir à un seul niveau de compétence pour une compétence donnée, l'apprenant peut appartenir à divers degrés à plusieurs niveaux de compétence.

En plus de pouvoir traiter l'incertitude de l'information concernant l'apprenant, il faut aussi pouvoir traiter son *incomplétude* [Conati et *al* 02]. En effet, il est irréaliste de penser que l'on va pouvoir interroger de manière exhaustive l'apprenant sur toutes les parties du curriculum. Pour remédier à cela, il faut pouvoir propager l'information dans le modèle de l'apprenant sans quoi celui-ci ressemblera à un gruyère, et ne permettra donc pas d'estimer de manière réaliste quelle doit être la prochaine leçon à enseigner à l'apprenant. Pour cela, il faut que le système soit capable de réaliser des *inférences* sur les parties du curriculum où il n'aura pas pu interroger l'apprenant directement. Supposer que l'apprenant maîtrise les prérequis d'un concept si le système a pu observer que l'étudiant connaît ce concept, ou encore estimer que l'étudiant ait une chance de pouvoir généraliser un exemple parce qu'il comprend cet exemple, sont des mécanismes d'inférence très simples pouvant être utilisés assez facilement. Il est possible d'enchaîner des inférences en cascade les unes à la suite des autres, il faut cependant noter qu'à chaque nouvelle inférence l'information transmise devient de plus en plus incertaine, car elle s'éloigne de l'observation directe de ce que l'apprenant connaît. Toute information

reçue concernant l'apprenant est susceptible d'être utilisée pour mettre à jour le modèle de l'apprenant, et revoir ce que le système croit savoir des connaissances de celui-ci.

En plus de représenter les connaissances de l'apprenant concernant le domaine, le modèle peut aussi mémoriser d'autres informations tels que le *profil affectif* de l'étudiant ou son *état émotionnel*. Le profil affectif garde en mémoire les *préférences* en terme d'apprentissage de l'étudiant ainsi que ses *modes de raisonnement* comme par exemple s'il est plutôt visuel ou textuel, s'il a besoin d'être guidé ou au contraire préfère explorer librement... L'état émotionnel reflète les *émotions* actuelles de l'apprenant tel que la joie, la colère, l'intérêt, la frustration... Ces deux types de données influent sur le processus d'apprentissage et permettent une modélisation plus fidèle quand ils sont pris en compte, notamment au cours d'un mécanisme inférentiel mettant à jour le modèle de l'apprenant. C'est ainsi qu'une même série d'actions effectuées par deux apprenants distincts bien que possédant des connaissances similaires au départ, peut avoir une signification différente pour chacun de ces apprenants dépendamment de leurs modes de raisonnement, de leurs préférences d'apprentissage et de leurs états émotionnels. L'état émotionnel d'un apprenant peut être amené à varier très fortement au cours d'une séance d'apprentissage, passant par exemple de la curiosité lors de la présentation d'un problème, à la frustration si le problème lui semble à un moment insoluble, et finalement la joie et la fierté si l'apprenant réussit à le résoudre. Le profil affectif peut aussi être conduit à évoluer, mais cela se fait beaucoup plus lentement que l'état émotionnel du fait que dans ce dernier, l'information soit plutôt de type *court terme* alors qu'elle est plutôt de type *long terme* pour le profil affectif.

Lors de la mise au point d'un modèle de l'apprenant survient la question du niveau de granularité souhaitée. A l'extrême de vouloir simplement connaître ce que l'apprenant sait ou ne sait pas sans aucune finesse répond l'opposé qui serait d'enregistrer toutes les actions faites par l'apprenant aussi insignifiantes soient-elles. Pour répondre à cette problématique, il faut séparer les données qui vont réellement être utilisées, et qui sont donc importantes, de celles qui ne le sont pas. Rien ne sert de garder la trace de chaque action de l'apprenant si on ne se sert pas de cette information [Self 88], à moins bien sûr

de projeter de s'en servir *à posteriori* à des fins d'analyse et pour de possibles raffinements du système. Dans le cas du modèle par recouvrement, cette question est en partie résolue directement, car dans ce cas la granularité du modèle de l'apprenant est aussi fine ou aussi grossière que celle du curriculum.

2.3.3. Le planificateur

Le rôle du planificateur est de déterminer pour un apprenant particulier dans quel ordre et de quelle manière vont être enseignés les différents sujets qui composent la matière [Lê et *al* 98]. Afin de réaliser cela, le planificateur se base sur les informations contenues dans le curriculum et le modèle de l'apprenant. Plusieurs niveaux différents de planification existent dépendamment de la granularité du curriculum et du modèle de l'apprenant. En général, on distingue au moins deux niveaux de planification : celui qui détermine le *cheminement global* à travers la matière enseignée et celui qui génère un *plan pour un cours ou une session d'apprentissage*. La planification du cheminement dans la matière consiste à déterminer l'ordre dans lequel on va enseigner les différents sujets formant cette matière (niveau d'abstraction élevé). La planification d'un cours (ou d'une session d'apprentissage) consiste à définir quelle va être la *séquence d'activités* à réaliser pour enseigner un sujet spécifique (niveau d'abstraction bas) en se basant sur des facteurs tels que la durée de la session, le rythme et le style d'apprentissage de l'apprenant ou encore ses préférences. Le plan de session délivré par le planificateur peut-être *statique*, s'il ne change pas au cours de la session, ou *dynamique*, s'il peut évoluer durant la session en fonction des actions de l'apprenant. Une alternative possible est que le planificateur génère un plan de session sous la forme d'un arbre comportant plusieurs chemins d'apprentissage, et où chaque embranchement représente les différentes conséquences possibles suite à une certaine action de l'apprenant. Par exemple, si on pouvait évaluer l'apprenant sur un concept particulier et ensuite choisir de passer au concept suivant s'il passe l'évaluation avec succès, ou au contraire présenter le même concept sous une autre forme s'il échoue, ces deux actions correspondraient à deux chemins d'apprentissage différents qui sont les conséquences de l'évaluation. Deux approches sont possibles pour modéliser les objectifs pédagogiques dont va se servir le planificateur pour établir le plan

de session : soit on choisit d'inclure cette information *dans le curriculum* d'une façon ou d'une autre, soit on décide que le planificateur est *indépendant du curriculum* (et aussi du modèle de l'apprenant) et que, bien que partant de celui-ci pour établir un plan de session, les règles de planification ne doivent pas être y encodées directement. Cependant, dans tous les cas la structure du curriculum reste primordiale pour la tâche de planification, même lorsque le planificateur est indépendant du domaine, car les relations définies entre les éléments du curriculum, comme celles de prérequis ou de composition, servent de base à la construction des plans de session. Une fois un plan de session établi, le planificateur passera la main au tuteur pour l'enseignement du cours proprement dit.

2.3.4. Le tuteur

Afin de l'aider à dispenser son enseignement, un STI dispose de plusieurs *stratégies pédagogiques*, parfois aussi appelées *stratégies tutorielles*. Le but de ces stratégies est d'aider à la transmission efficace de la connaissance à l'apprenant. Le tuteur a pour tâche d'appliquer ces stratégies pédagogiques durant l'enseignement. Il existe deux types de stratégies pédagogiques : celles qui sont spécifiques au domaine enseigné et celles qui sont indépendantes dudit domaine. Parmi ces dernières on peut citer [Aïmeur et Frasson 96]:

-Le *tutorat classique* où l'interaction se fait entre un tuteur simulé et un apprenant humain.

-Le *co-apprenant* où le tuteur est remplacé par un apprenant simulé de même niveau que l'apprenant humain [Chan 95].

-Le *compagnon* dans lequel coexistent trois acteurs : un tuteur simulé, un co-apprenant simulé et un apprenant humain [Chan et Baskin 88, Chan 95].

-Le *perturbateur* qui ressemble fortement au compagnon quant au nombre et types des acteurs, mais dans lequel le co-apprenant simulé cherche à induire en erreur l'apprenant

humain à certains moments clés, afin qu'il remette en cause ses connaissances et ses croyances [Aïmeur 96]. Cette stratégie pédagogique s'appuie sur la théorie psychologique de la *dissonance cognitive*.

-Le *tutorat inversé* où les rôles sont bousculés puisque c'est l'apprenant humain qui joue le rôle du tuteur et qui doit expliciter ses connaissances à un apprenant simulé [Chan 95].

Il est possible de faire des variantes sur ces stratégies pédagogiques, par exemple en multipliant le nombre de co-apprenants simulés dans le cas de la stratégie du compagnon ou du perturbateur.

Certaines de ces stratégies sont basées sur des théories de psychologie cognitive reconnues, alors que d'autres sont le résultat de tests et d'observations et sont donc plus empiriques. Parmi les théories de la cognition utilisées intensivement dans le domaine des STI, on retrouve celles issues de psychologues de l'éducation tels que Bloom, Gagné et Merrill ou celles développées par des informaticiens comme Anderson. La *taxonomie de Bloom* [Bloom 56] classifie les niveaux de comportements intellectuels qui sont importants dans l'apprentissage. Gagné quant à lui a identifié cinq types de connaissances et d'habiletés et a proposé qu'à chaque type de connaissance soit associée une méthode d'instruction différente [Gagné 85]. Il a aussi défini quels sont les neuf *échelons* à franchir lors d'une situation d'apprentissage et là aussi, il a avancé l'idée qu'à chaque échelon de l'apprentissage devrait correspondre une stratégie pédagogique différente. Merrill a aussi développé sa propre classification [Merrill 94], qui est basée à la fois sur le niveau de performance de l'apprenant et le type du contenu. La théorie *Adaptive Control of Thought* (ou *ACT**) de Anderson décrit quelles doivent être les actions d'un tuteur dépendamment des situations d'apprentissage pouvant survenir [Anderson 83]. Cependant, il est important de noter que les STI offrent beaucoup de nouvelles possibilitées en termes d'apprentissage par rapport aux méthodes d'enseignement traditionnelles, et qu'il serait dommage de vouloir se restreindre seulement à des théories classiques alors qu'il y a tant de nouvelles pistes à explorer.

Un STI possédant une bonne *expertise pédagogique* aura à sa disposition plusieurs de ces stratégies pédagogiques, et pourra passer de l'une à l'autre dépendamment de leur efficacité sur l'apprenant ainsi que du contexte d'apprentissage. On appelle *méta-stratégie*, le mécanisme permettant de choisir une stratégie pédagogique parmi plusieurs dépendamment des besoins de la situation [Beck et Woolf 00]. Ce mécanisme peut être attribué au planificateur, laissant alors comme rôle au tuteur l'application de la stratégie pédagogique et non pas sa détermination.

Une question importante est de savoir à quelle fréquence durant l'apprentissage le système doit intervenir. Si le système intervient trop fréquemment, l'apprenant va être rapidement agacé et donc dérangé dans son apprentissage [Kashibara et *al* 94]. À l'inverse, si le système intervient trop peu, l'apprenant risque d'avoir l'impression de rester seul face à lui-même [Anderson 93]. Par exemple, doit-on corriger l'apprenant immédiatement à chaque erreur qu'il commet, ou doit on le laisser poursuivre pendant un certain temps sur la voie qu'il a choisie même si celle-ci comporte des erreurs? En plus d'avoir une fréquence d'intervention appropriée, le niveau d'aide apporté par le système doit lui aussi être adapté aux besoins de l'étudiant. Si l'apprenant a commis une petite erreur, il ne faut pas l'enterrer sous une tonne d'explications; par contre, s'il a peu ou pas compris le sujet, il faut pouvoir lui fournir des explications détaillées.

Toutes les stratégies pédagogiques présentées précédemment s'appliquent à une situation d'apprentissage *individuel*. Depuis quelques années, de plus en plus de STI privilégiant l'apprentissage *collectif* ou *collaboratif* ont été développés [Dillenbourg 99]. Dans ces systèmes, l'étudiant n'est plus seul face à l'ordinateur mais se trouve au milieu d'autres étudiants dans une *classe virtuelle*. Ce contexte d'apprentissage permet aux étudiants de s'entraider, et décharge donc le système d'une partie du travail d'aide et d'explications. Ce type d'apprentissage génère tout de même deux grands problèmes dus au fait qu'il y a maintenant plusieurs et non pas un seul apprenant. D'abord, comment mettre à jour le modèle de l'apprenant quand l'apprentissage est collaboratif ? Ensuite, comment choisir les prochains sujets qui vont être enseignés ? Belvedere [Suthers et *al* 95], où les

étudiants sont engagés dans des discussions critiques portant sur des sujets touchants à la science ou à la loi, est un exemple d'un tel système.

2.3.5. L'interface

L'interface homme-machine du STI joue elle-aussi un rôle primordial puisque c'est à travers elle que tout échange d'information entre l'ordinateur et l'apprenant passe. Il est essentiel que cette interface soit faite pour faciliter au maximum l'interaction entre l'homme et la machine, et aussi qu'elle ne soit pas rébarbative afin de maintenir la motivation de l'apprenant. L'aspect visuel de l'interface a énormément évolué au fur et à mesure des progrès technologiques en informatique. De textuelle au départ, elle est passée à des formes beaucoup plus graphiques jusqu'à finalement incorporer la réalité virtuelle dans certains systèmes actuels permettant ainsi une immersion quasi-totale dans l'environnement (voir environnements virtuels dans la section suivante). L'utilisation d'une interface graphique, ainsi que l'intégration des technologies multimédia comme la vidéo, a rendu plus conviviale et attrayante la transmission de l'information. Cependant, on a souvent eu tendance à user et abuser à outrance de ces technologies sans réellement se préoccuper de l'influence de ces moyens de communication sur le processus d'apprentissage [Scaife et Rogers 96].

Au départ l'interface graphique a été considérée comme une structure certes attrayante pour véhiculer l'information, mais aussi relativement fixe et rigide. De nos jours, la tendance est plutôt à ce que l'interface elle-même puisse se modifier pour correspondre aux besoins et aux préférences de l'utilisateur. L'important effort qui a été fait au niveau de la modélisation de l'apprenant dans les STI permet ainsi d'utiliser l'information récoltée pour pouvoir adapter l'interface de façon efficace. Certaines interfaces actuelles permettent ainsi de fournir plusieurs *vues* (ou représentations) du domaine et de naviguer entre elles, ce qui facilite la compréhension du domaine en offrant plusieurs points de vue [Scaife et Rogers 96].

2.4. Les différents types de systèmes tutoriels intelligents

Une classification naturelle possible pour catégoriser les STI prend en compte comme critère principal le type de domaine enseigné par le STI. On cherchera à différencier les STI qui cherchent plutôt à entraîner l'étudiant à résoudre des situations qui pourront se présenter à lui dans le monde *réel*, de ceux qui enseignent des connaissances *abstraites*. En dehors des deux grandes familles évoquées ci-dessus, d'autres approches aussi considérés souvent comme faisant partie du domaine des STI bien que moins « classiques » existent, tels que les *environnements virtuels* ou les *hypermédias adaptatifs*.

Les *environnements virtuels* sont des systèmes qui utilisent la *réalité virtuelle* afin de simuler un monde dans lequel l'utilisateur pourra se plonger et avec lequel il pourra interagir. L'immersion peut être plus ou moins totale, selon qu'elle se réalise à l'aide d'équipements tels qu'un casque et des gants sensoriels ou non. Alors que les STI traditionnels servent plutôt en général à enseigner des connaissances abstraites, les environnements virtuels permettent au contraire l'apprentissage de tâches plus concrètes. Ils permettent aussi d'impliquer beaucoup plus l'étudiant dans le processus d'apprentissage, puisque ce sont ses actions qui vont modifier le monde, conduisant ainsi à une augmentation de sa motivation. On peut distinguer trois types d'environnements virtuels [Saldiàs et *al* 99] : les *micromondes,* les *environnements d'exercice et de pratique* et les *environnements de jeux*. Dans les micromondes, l'étudiant est placé dans un environnement qu'il est libre d'explorer comme il l'entend. Il peut manipuler les objets et observer leurs propriétés. De cette manière il apprend par lui-même plutôt qu'en étant guidé. SimForest [Murray et *al* 03] est un excellent et typique exemple de micro-monde. SimForest cherche à faire comprendre comment peut fonctionner et vivre une forêt en encourageant les apprenants à jouer avec un simulateur et observer ce qui se produit. Les environnements d'exercice et de pratique diffèrent des micromondes sur le fait que cette fois-ci, l'étudiant au lieu d'explorer le monde est plutôt confronté à une situation virtuelle où il doit résoudre certains problèmes à l'aide de ses connaissances. Il existe souvent plusieurs façons de résoudre le même problème, parfois il y en a autant qu'il y a

d'utilisateurs. Ce type d'environnement permet le renforcement de l'apprentissage des concepts de base tout en encourageant la créativité. Enfin dans les environnements de jeux, l'étudiant se retrouve avec un but ludique à atteindre, et pour le compléter il lui est permis de faire toutes les actions autorisées dans le cadre des règles du jeu. Ici, ce sont les capacités stratégiques de l'étudiant qui sont principalement développées. Un des principaux points forts des environnements virtuels est de pouvoir solliciter en même temps plusieurs sens tels que la vue, l'ouïe, le toucher ou encore le sens de la représentation spatiale. Un monde virtuel peut être façonné pour correspondre aux besoins de l'étudiant, et ainsi agir sur les sens de manière à favoriser le mieux possible l'apprentissage. L'utilisation de la réalité virtuelle permet aussi d'explorer des mondes normalement inaccessibles aux êtres humains, tels que le cœur des atomes ou celui des étoiles, ou encore de simuler des phénomènes qui seraient impossibles ou trop dangereux à réaliser dans le monde réel. Elle autorise aussi une approche beaucoup plus intuitive de la navigation à travers le système, touchant ainsi une plus large audience car elle requiert moins d'expérience préalable.

La réalité virtuelle a pourtant ses limitations. D'abord, elle est encore dans son enfance et le développement d'un environnement virtuel reste un processus très long et complexe, de même l'équipement nécessaire à une immersion totale, tels que le casque ou les gants, coûte encore relativement cher. De plus, la réalité virtuelle seule ne peut pas complètement remplacer l'expérience du contact avec le monde réel qui permet de réaliser totalement le processus d'apprentissage. Pour l'instant on trouve encore assez peu de projets mariant la réalité virtuelle avec les STI, on peut cependant citer l'exemple de STEVE [Johnson et *al* 98]. STEVE est un tuteur virtuel dont le but est d'assister l'étudiant durant l'apprentissage. Il cohabite dans le monde virtuel avec l'étudiant et est capable de lui montrer comment effectuer certaines tâches ou encore de répondre à certaines de ses questions. Il a été utilisé pour l'enseignement de tâches réalisables sur des navires de la marine américaine, telle que par exemple la manipulation des moteurs.

Les *hypermédias adaptatifs* [Brusilovsky 01], situés au confluent des hypermédias et de la modélisation de l'utilisateur, sont des cousins proches des STI. Le but des hypermédias

adaptatifs est de pouvoir dépasser les limites des hypermédias classiques en adaptant le contenu et la présentation de la matière en fonction des besoins de l'utilisateur. En ce sens, les hypermédias adaptatifs sont très semblables aux STI, d'ailleurs les avancées faites dans la modélisation de l'apprenant dans le domaine des STI ont souvent été reprises par les hypermédias adaptatifs et vice-versa. Par contre, les hypermédias adaptatifs mettent beaucoup moins l'emphase sur la modélisation de la connaissance du domaine et sur l'aspect tutoriel (exception faite de la présentation). De manière générale et dans le domaine de l'éducation en particulier, les hypermédias adaptatifs ont surtout connu des avancées importantes depuis 1996 avec l'essor d'Internet et le développement des STI en ligne tels que ELM-ART [Brusilovski et *al* 96a] et InterBook [Brusilovski et *al* 96b]. L'adaptation au niveau du contenu peut se faire par exemple par l'ajout, la suppression ou même la modification des lignes d'un texte devant être présenté à l'utilisateur. Au niveau de la présentation on utilisera du texte, ou bien des images ou encore de la vidéo dépendamment des préférences et/ou du style d'apprentissage de l'utilisateur. Cette adaptation peut aussi être beaucoup plus subtile, par exemple en influant sur la présentation des liens que l'utilisateur utilise pour naviguer dans le système. On pourra faire un tri sur ces liens en fonction d'un critère x lié à notre connaissance de l'utilisateur, ou encore mettre en évidence certains liens avec une couleur codifiée, ou même supprimer ou ajouter des liens.

2.5. Réalités et perspectives

Un des principaux défis qui attend les STI en ce début de millénaire est le passage du laboratoire et de l'expérimentation à petite échelle au déploiement à plus grande échelle. Les STI ont été un terrain très fertile pour tester et étudier des théories issues de la psychologie ou de la pédagogie mais il est encore très rare que les STI dépassent le stade du prototype pour aller se mesurer au monde réel [Koedinger et Anderson 97]. Le problème de la modélisation de l'apprenant reste lui aussi un des grands challenges actuels des STI. En effet même si de nombreuses approches ont été proposées sur la manière de modéliser un apprenant, aucune n'a réussi pour l'instant à faire l'unanimité.

Pour résumer en caricaturant, on pourrait dire que pour l'instant les STI n'en sont qu'à leurs balbutiements et n'ont pas encore atteint leur maturité complète. De nombreuses expérimentations ont été faites et de nombreux prototypes construits, mais le passage au monde réel et aux tests à grande échelle n'a pas encore été franchi. On pourrait par exemple comparer la situation actuelle des STI à celle des systèmes experts dans les années 70 et au début des années 80.

Chapitre 3 : L'acquisition des connaissances et les STI

3.1 Présentation de l'acquisition des connaissances dans un contexte général

L'*acquisition des connaissances* est la branche de l'intelligence artificielle qui s'occupe du développement de méthodes, de logiciels et d'outils permettant la construction de bases de connaissances. C'est un domaine qui a pris de l'essor avec la nécessité de créer des bases de connaissances de taille importante lors du développement des systèmes experts. On distingue généralement trois catégories de méthodes d'acquisition des connaissances : les techniques d'*élicitation directe*, les techniques *assistées par ordinateur* et les techniques utilisant l'*apprentissage machine*.

Le but des techniques d'élicitation directe est d'acquérir les connaissances d'un expert et de découvrir les méthodes de résolution de problèmes qu'il emploie à l'intérieur d'un domaine particulier [Boose 89, Cooke 94]. Ces techniques requièrent une forme d'interaction entre l'expert et un *ingénieur de la connaissance*. Généralement on retrouve les trois étapes suivantes dans le processus d'acquisition des connaissances de ces méthodes : *élicitation* auprès de l'expert, *interprétation* des données collectées afin d'inférer les connaissances sous-jacentes et le processus de raisonnement, et finalement *création d'un modèle conceptuel* modélisant la connaissance de l'expert. Parmi les techniques d'élicitation directe, on peut trouver l'interview non structurée [Hoffman 87], l'interview structurée [Merton et *al* 56], l'observation [Grabowski 88], l'analyse de protocoles [Newell et Simon 72] et l'analyse de la grille de répertoire [Kelly 55]. Ces méthodes trouvent souvent leur origine en psychologie. D'autres méthodes ont aussi été développées pour permettre l'intégration des données collectées auprès de multiples experts (et les conflits qui en résultent) [Gaines 86, Mittal et Dym 85] tels que le brainstorming, la technique du groupe nominal, la technique Delphi ou la prise de décision par consensus.

Alors que les techniques d'élicitation directe exigent une forme d'interaction entre deux humains (l'expert et l'ingénieur de la connaissance), les techniques assistées par

ordinateur permettent à l'expert d'expliciter ses connaissances directement à la machine. Ainsi, c'est l'ordinateur qui jouera ici le rôle d'ingénieur de la connaissance. Les techniques assistées par ordinateur peuvent elles-mêmes être subdivisées en deux catégories : les *langages d'ingénierie de la connaissance* et les *systèmes d'aide à la construction de bases de connaissance*. Les langages d'ingénierie de la connaissance offrent un formalisme permettant à l'expert de représenter ses connaissances sous une forme structurée, ainsi que certains mécanismes limités de raisonnement tel que le chaînage arrière. Cependant, ils requièrent une expérience minimale dans le domaine de la programmation de la part de l'expert, ainsi que l'apprentissage du langage qui sera utilisé pour représenter les connaissances. Les systèmes d'aide à la construction de bases de connaissance, eux en revanche utilisent un modèle conceptuel pour interagir avec l'utilisateur permettant ainsi de cacher la complexité et la non-familiarité du modèle symbolique à partir duquel la base de connaissances est réellement construite (comme les réseaux sémantiques, les règles, les frames, etc.). Les systèmes d'aide à la construction de bases de connaissance sont aussi connus sous le nom d'*outils d'acquisition des connaissances* [Kim et Gil 02]. Comme exemples de tels systèmes on peut citer AQUINAS [Boose 85, Boose et *al* 89], KADS [Breuker et Wielinga 89], MORE [Kahn et *al* 85] et CommonKADS [Schreiber et *al* 99, Schreiber 01]. Des outils utilisant des langages visuels, pour aider l'expert à structurer ses connaissances sous la forme d'un graphe [Clark et *al* 01], ont aussi été développés tel que l'outil de création de réseaux sémantiques Kdraw [Gaines 91] ou encore l'outil de création de cartes conceptuelles Kmap [Gaines et Shaw 93].

Les techniques d'acquisition des connaissances utilisant l'apprentissage machine [Tecuci et Kodratoff 95, Webb et *al* 99] vont encore un pas plus loin que celles assistées par ordinateur, puisqu'ici il ne s'agit plus seulement de faire jouer le rôle d'ingénieur de la connaissance à l'ordinateur, mais aussi en partie celui de l'expert en automatisant une partie importante du processus d'acquisition des connaissances. Ces techniques emploient des algorithmes d'apprentissage machine afin d'inférer à partir d'exemples les règles et principes qui gouvernent le monde. L'expert humain n'est cependant pas totalement absent, puisqu'il intervient durant une phase de raffinement afin de vérifier et corriger si

nécessaire les inductions faites par le système. Parmi les algorithmes d'apprentissage machine ayant été appliqués avec succès à l'acquisition des connaissances, on peut citer la programmation logique [Gaines 96], les algorithmes génétiques [De Jong *et al* 93, Wetjers et Paredis 02], l'apprentissage analytique [Pazzani et Brunk 93] et les réseaux neuronaux [Krishnan et *al* 99, Zhou et Chen 02].

En dehors des trois catégories principales décrites précédemment, il existe des logiciels appelés *méta-outils* n'entrant dans aucune de ces catégories. Les méta-outils peuvent être utilisés sur demande pour créer des outils d'acquisition des connaissances adaptés à un domaine particulier [Gennari *et al* 03]. Ils peuvent être vus comme des outils d'acquisition des connaissances pouvant servir à créer d'autres outils d'acquisition des connaissances plus spécialisés.

3.2. L'acquisition des connaissances dans le cadre des STI

Les méthodes d'acquisition des connaissances disponibles dans le commerce, tels que par exemple ceux utilisés pour l'enseignement assisté par ordinateur traditionnel, ou encore pour les systèmes multimédias, manquent en général de la profondeur et de la sophistication nécessaires pour construire des systèmes tutoriels intelligents intéressants. Ces systèmes permettent en général de produire assez facilement du matériel interactif qui sera visuellement attrayant, mais en contrepartie la représentation de la matière enseignée y est souvent superficielle, et on y trouve peu ou pas de stratégies pédagogiques. Pour remédier à cela, les chercheurs dans le domaine des STI ont voulu développer leurs propres outils d'acquisition des connaissances adaptés à leurs propres besoins. On pourra trouver une excellente et très complète revue de l'état de l'art de ce domaine dans un article de Tom Murray [Murray 99]. Dans cet article, Murray observe que l'acquisition des connaissances appliquée aux STI a connu un essor important depuis 1994 et que de nombreuses approches différentes ont été explorées, mais sans pour autant que l'une d'elles ne réussisse à se placer en modèle standard pour l'instant. Les outils d'acquisition des connaissances pour les STI sont encore au stade de la recherche et ont souvent été développés par des équipes universitaires pour des projets spécifiques.

Malgré quelques succès limités, aucun de ces outils n'a réellement franchi l'étape d'utilisation à grande échelle et peu ont effectué le passage au domaine commercial. Le but premier de la recherche d'outils d'acquisition efficaces est bien sûr de réduire l'effort nécessaire à la création du STI que ce soit en terme de temps, coût financier ou autres ressources. Murray estime qu'il faut normalement environ 300 heures de travail pour produire 1 heure de cours avec les outils d'acquisition traditionnels, soit un ratio de 300:1. Certains outils d'acquisition des connaissances affirment leur ambition de ramener ce ratio à 20:1, voir même à 10:1, mais pour l'instant le meilleur résultat observé parmi le peu d'études sérieuses disponibles est un ratio de 100:1 avec le système KAFITS [Murray et Woolf 90], le prédécesseur d'EON [Murray 98].

Il est important de remarquer que le problème de l'acquisition des connaissances dans les STI, contrairement à celui d'une base de connaissances traditionnelle, est *multiple*. En effet, dans un STI il ne s'agit pas seulement d'acquérir la connaissance du domaine sous la forme d'un curriculum, mais aussi de définir les stratégies tutorielles qui vont être utilisées, ainsi que de préciser la structure du modèle de l'apprenant sans parler du design de l'interface graphique. De part sa multiplicité, la tâche de création d'un STI est donc plus complexe et ardue que celle d'une simple base de connaissances. Certains outils d'acquisition des connaissances prennent parfois la décision d'encoder plus ou moins directement à l'avance dans le système les stratégies tutorielles, le modèle de l'apprenant ou encore l'interface, laissant ainsi plus ou moins de degrés de liberté à l'auteur qui construira le STI. Cela étant dit, on se focalisera pour la suite surtout sur les outils servant à la création du curriculum puisque c'est sur cette tâche que s'attelle en partie le travail décrit dans ce mémoire. Néanmoins, même quand on s'intéresse principalement à la modélisation de la matière du domaine, l'aspect enseignement n'est jamais totalement absent puisque la façon dont on structure la matière influence fortement comment cette matière va être sélectionnée et agencée par le planificateur pour former un cours. La plupart des outils ne différencient pas la phase modélisation de la connaissance de celle de la définition des objectifs pédagogiques. Certains systèmes comme IRIS [Arruarte *et al* 97] ou CREAM-Tools [Nkambou et *al* 03] font néanmoins exception en permettant une définition séparée de la partie connaissance et de la partie pédagogie.

Pour les outils d'acquisition, le *choix du mode de représentation des connaissances* va fortement influencer le type de connaissances qui pourra être stocké dans le curriculum et donc enseigné par le STI. Ainsi des systèmes comme Demonstr8 [Blessing 97] ou D3-Trainer [Reinhardt 97], qui utilisent des règles pour représenter les connaissances, peuvent facilement modéliser des connaissances procédurales ou des faits. Par contre, ils sont mal adaptés à la modélisation des connaissances conceptuelles. Certains systèmes tel que DNA [Shute et *al* 98] restent néanmoins capables de modéliser plusieurs types de connaissances même s'ils utilisent un seul mode de représentation. Dans DNA, les connaissances, que ce soit des faits, des procédures ou des concepts sont toutes représentées sous forme de réseaux sémantiques. Ces différents réseaux sont reliés entre eux par des liens précisant les relations entre les différents types de réseaux constituant ainsi un réseau plus important. XAIDA [Hsieh *et al* 99] a choisi une autre voie, celle de mélanger plusieurs modes de représentation des connaissances. XAIDA est spécialisé dans la modélisation du fonctionnement d'appareils telle que celle des machines industrielles par exemple. Il utilise des réseaux sémantiques pour représenter les caractéristiques physiques et les procédures d'opération et de maintenance, couplés avec des schémas de raisonnement causal pour expliciter les phénomènes apparaissant durant une procédure et des arbres de faute pour détailler les problèmes pouvant éventuellement survenir.

La *granularité* de la représentation des connaissances peut varier très fortement suivant les systèmes. Pour les outils tels que RIDES [Munro et *al* 97] ou SIMQUEST [Van Joolingen et de Jong 03] servant à la création de STI cherchant à simuler ou modéliser une situation, un appareil, voir un monde, il est souvent important que la connaissance soit représentée avec la plus grande finesse et la plus grande précision possibles pour que la simulation soit réaliste. Plus la connaissance est modélisée finement, plus le STI pourra potentiellement profiter de cette richesse pour donner des explications détaillées. Une modélisation grossière de la connaissance limite forcément la capacité du tuteur plus tard à pouvoir donner des explications. Cependant, une modélisation fine se paye par un

travail plus astreignant et plus coûteux en temps de la part de l'expert qu'une modélisation grossière.

La *visualisation des connaissances* constitue aussi un aspect important des outils d'acquisition des connaissances. Certains systèmes comme EON, RIDES ou CREAM-Tools intègrent directement des facilités pour pouvoir visualiser et éditer graphiquement la structure de représentation des connaissances. Ces outils permettent de visualiser facilement les relations entre les éléments du curriculum, permettant aussi d'avoir une vue d'ensemble sur la structure du curriculum. La représentation des connaissances dans ces systèmes peut être hiérarchique ou non. Si la connaissance est organisée sur plusieurs niveaux, ces outils ont l'avantage de pouvoir permettre de zoomer et de se déplacer instinctivement entre les niveaux.

Reste aussi à définir *à qui va s'adresser* l'outil d'acquisition des connaissances. En particulier, est-ce qu'il est nécessaire que l'expert ait une expérience dans la programmation ou d'autres domaines, est-ce l'utilisateur doit subir un entraînement avant de pouvoir utiliser le système efficacement, ou encore est-ce que l'auteur type sera n'importe quel professeur « standard » ou plutôt quelqu'un ayant une certaine familiarité avec le monde des STI ? Plus un système est facile à apprendre et/ou à utiliser, plus il est possible de faire participer des personnes au processus de création du STI, réduisant ainsi du même coup le temps nécessaire à sa mise au point. XAIDA et REDEEM [Major *et al* 97] par exemple ont été conçus dans l'optique de pouvoir être utilisés par des personnes ayant eu peu d'entraînement alors que d'autres logiciels comme EON ou RIDES supposent que les auteurs qui les utiliseront seront relativement expérimentés.

Enfin reste à considérer quelle est la *généricité* de l'outil d'acquisition, à savoir à quel point l'outil est efficace et approprié pour pouvoir construire des curriculum/STI pour des domaines qui sont différents du domaine initial. En général, moins un outil est générique, plus il a été construit pour cibler un type de domaine particulier et sera donc d'autant moins applicable à d'autres domaines. La question de savoir si un outil doit être générique ou non reste encore très controversée. Certes, il peut sembler avantageux qu'un

système soit le plus générique possible, mais en même temps cela implique qu'un surcroît de libertés soit accordé à l'auteur, augmentant ainsi les chances pour lui d'entrer d'une façon incorrecte des informations qui seront plus tard mal interprétées par le système. On peut donc parfois avoir intérêt à spécialiser l'outil d'acquisition des connaissances afin d'éviter ce type de problème même si cela signifie contraindre en partie l'auteur.

Chapitre 4 : L'informatique quantique : un domaine multidisciplinaire

4.1. Survol de l'informatique quantique

Habituellement, lorsque l'on définit l'informatique comme « la science liée au traitement de l'information », on sous-entend implicitement que ce traitement de l'information s'effectue en suivant les lois de la physique *classique*. L'informatique *quantique* [Nielsen et Chuang 00] s'intéresse elle aussi au traitement de l'information, mais en se basant cette fois sur les principes de la mécanique quantique. Les principes de la mécanique quantique qui s'appliquent lorsque l'on travaille à l'échelle de l'atome, et qui servent à expliquer le comportement des particules élémentaires, sont très différentes de ceux de la physique classique qui servent à décrire des phénomènes macroscopiques. Certains de ces principes peuvent sembler tout à fait contre-intuitifs au premier abord.

L'énoncé de la loi de Moore [Moore 65] nous affirme la chose suivante : « Le nombre de transistors pour un circuit de même taille double en moyenne tous les 12 à 24 mois. »
Cette loi surprenante a toujours été vérifiée empiriquement pour l'instant depuis 1965. Au fur et à mesure que la taille des composants diminue et que la transition s'effectue du macroscopique vers le microscopique, on commence à ressentir de plus en plus les effets du monde quantique. Si la loi de Moore reste toujours valide jusqu'en 2020, les puces auront alors atteints un ordre de grandeur proche de celui des atomes et des molécules, et seront donc totalement soumis aux effets quantiques. Il ne faut pas considérer cela comme une limite absolue et une restriction, mais plutôt chercher à voir ce que cela permet de réaliser qui est hors de portée de l'informatique classique. C'est Richard Feynman en 1982, qui le premier a émis l'idée d'utiliser un appareil basé sur les principes de la mécanique quantique afin de pouvoir simuler n'importe quel système physique [Feynman 82]. Un peu plus tard, David Deutsch avança l'hypothèse que la disponibilité d'un ordinateur quantique permettrait de réaliser certains calculs plus rapidement qu'il ne serait possible avec un ordinateur classique [Deutsch 85]. Dans ce même papier, Deutsch définit le premier modèle de calcul quantique, celui de la *machine de Turing quantique universelle*. Cette notion sera revue et corrigée plus tard par Bernstein et Vazirani

[Bernstein et Vazirani 97]. En 1989, Deutsch propose un autre modèle, celui des *circuits quantiques* [Deutsch 89]. Andrew Yao démontrera en 1993 l'équivalence entre les deux modèles [Yao 93]. Le formalisme le plus couramment utilisé pour décrire le fonctionnement d'un algorithme quantique reste toujours celui des circuits à l'heure actuelle.

L'unité de base de l'information quantique est le *qubit* (abréviation de « quantum bit ») et constitue la contrepartie quantique du bit classique. Alors qu'un bit peut seulement se trouver dans l'état 0 ou l'état 1 à un moment donné, il est possible pour un qubit d'être dans une *superposition* cohérente de ces 2 états. Physiquement on peut imaginer cela comme un électron qui se situerait sur 2 couches électroniques en même temps, ou encore un photon qui se trouverait à 2 endroits à la fois. Le qubit est dans chacun de ces états avec une certaine *amplitude* (et non pas une certaine probabilité). L'amplitude de chaque état possible est représentée sous la forme d'un nombre complexe. C'est la mesure qui va forcer le qubit à choisir entre ces 2 états. Chaque état va être observé avec une *probabilité* égale à la norme de son amplitude élevée au carré. Les probabilités d'observer chaque état individuellement somment bien sûr à 1. De plus l'acte de mesure est *irréversible*, voulant dire qu'une fois qu'un qubit a été mesuré dans un certain état il *est* maintenant dans cet état. Tout le reste de l'information qui était contenue dans le qubit avant la mesure est irrémédiablement perdu. Il est cependant possible de *calculer* en utilisant le qubit et de n'effectuer l'acte de mesure qu'à la toute fin du calcul. En fait en informatique quantique la mesure est la seule étape irréversible, tout le reste du calcul pouvant se faire de manière totalement *réversible* [Bennett 73]. L'information quantique est donc particulièrement fragile et précieuse. Elle est aussi très différente de sa contrepartie classique, en particulier alors qu'on peut *lire* et *copier* un bit classique sans restriction, il est *impossible* de mesurer (et donc d'obtenir de l'information) un qubit sans le *perturber*, mais aussi de pouvoir le *cloner* parfaitement et donc aussi de pouvoir le *diffuser*. Ces dernières propriétés de l'information quantique, qui à première vue peuvent sembler négatives, servent de fondations à la cryptographie quantique. Le nombre d'états pouvant être en superposition dans un registre est exponentiel par rapport au nombre de qubits. Alors que classiquement un registre de n bits se trouve dans un état parmi les 2^n

possibles, quantiquement un registre de *n* qubits peut être en superposition de ces 2^n états. D'autres ressources quantiques existent qui n'ont aucun équivalent classique, comme l'*interférence* et l'*intrication*.

Il est possible d'imaginer un calcul sur un registre quantique comme un calcul qui s'effectuerait sur un nombre exponentiel de chemins logiques en *parallèle*. Chaque opération quantique, telle que l'action d'une porte quantique, constitue un embranchement d'où partent 2^n chemins, n étant le nombre de qubits sur lequel l'opération quantique agit. Chaque chemin est suivi avec une certaine amplitude. Classiquement ces calculs devraient être effectués de façon séquentielle par un processeur unique ou par 2^n processeurs travaillant en parallèle. Plusieurs chemins de calcul peuvent mener au même état. L'amplitude d'un état à un moment particulier du processus de calcul est déterminée en regardant tous les chemins qui peuvent mener à cet état et en faisant la somme des amplitudes de chacun de ces chemins. On calcule l'amplitude d'un chemin particulier de calcul en multipliant les amplitudes des « branches » qui le composent. C'est maintenant qu'il est important de se souvenir que les amplitudes sont des nombres complexes, l'amplitude d'un chemin de calcul à la fin sera donc aussi un nombre complexe pourrait être un nombre réel négatif. Lorsqu'on fait la somme des amplitudes de chacun des chemins de calcul menant à un état, il est possible que ces valeurs se consolident entre elles si par exemple toutes les amplitudes pointent dans la même direction, ou encore s'annulent entre elles si par exemple toutes les amplitudes des chemins sont réelles et ont la même norme mais qu'exactement la moitié de ces chemins sont des nombres positifs et que l'autre moitié est formée de nombres négatifs. Ce phénomène appelé *interférence* est *constructif* lorsque les chemins se consolident entre eux, et *destructif* lorsque les chemins s'annulent entre eux. Lors d'un calcul, il y a toujours aussi bien de l'interférence destructive que de l'interférence constructive qui prend place.

Einstein, Podolsky et Rosen avaient les premiers imaginés dès 1935 une « expérience de pensée » qui, impliquait, selon eux, que la mécanique quantique était incomplète [Einstein et *al* 35]. Bien qu'à l'époque leur idée était plutôt de souligner la bizarrerie de

la mécanique quantique, cette « expérience » montrait un aspect essentiel et unique de la mécanique quantique, celui de l'*intrication* (aussi appelée *enchevêtrement*). Lorsque deux particules sont intriquées, il est impossible de définir l'état de l'une des deux particules sans tenir compte en même temps de l'état de l'autre. Autrement dit, l'état d'une particule est corrélée avec l'état de l'autre particule, mais d'une façon telle qu'il est impossible de fournir une explication valide classique à cette corrélation. Pour expliquer cette corrélation de manière complète, il faut se baser sur les principes de la mécanique quantique. Bell en 1964 a imaginé une expérience qui utilisait des paires de particules intriquées (ou paire EPR) [Bell 64] et qui, si elle pouvait être réalisée, démontrerait la validité de la mécanique quantique. Cette expérience fut réalisée par Alain Aspect en 1982 [Aspect et *al* 82].

En utilisant ces ressources mises à notre disposition par la mécanique quantique, il est possible de réaliser des prouesses qui sont hors de portée de l'informatique classique.

L'algorithme de Shor [Shor 97] permet de résoudre un problème considéré comme difficile en cryptographie classique, celui de la *factorisation des grands nombres*. Classiquement il n'existe pas pour l'instant d'algorithme efficace connu pour résoudre ce problème, le meilleur algorithme disponible requiert un temps *exponentiel* alors que l'algorithme de Shor fonctionne en temps *polynomial*. Il utilise le fait que chercher les facteurs d'un grand nombre est équivalent à trouver la période d'une fonction périodique qui est injective sur sa période. La plupart des systèmes de cryptographie utilisés actuellement sur Internet tel que RSA par exemple ont leur sécurité basée sur la difficulté du problème de factorisation ou d'autres problèmes équivalents, leur sécurité est donc menacée directement par l'existence de l'algorithme de Shor.

L'autre algorithme fondamental de l'informatique quantique est l'algorithme de Grover [Grover 96]. Il s'attaque au problème de la *recherche d'un élément spécifique dans une liste non ordonnée* de taille n. Classiquement la solution naïve, qui consiste à simplement regarder les éléments de la liste un par un au hasard jusqu'à ce que l'on tombe sur l'élément recherché, est aussi la plus efficace. Cette méthode requiert en moyenne $n/2$

accès à la liste. Avec l'algorithme de Grover, on peut trouver l'élément recherché en un temps approximatif de l'ordre de $\sqrt{n}$, le gain obtenu est donc *quadratique*.

Cependant, ce que l'informatique quantique prend d'un côté avec l'algorithme de Shor, elle le redonne de l'autre avec la *cryptographie quantique*. La cryptographie quantique [Bennett et Brassard 84] permet à deux individus, que nous appelerons Alice et Bob, de communiquer de manière totalement confidentielle, même s'ils étaient observés par un espion disposant d'une puissance de calcul illimitée. Classiquement il a été démontré par Shannon [Shannon 48] que cela ne serait possible que si les deux personnes partageaient à l'avance une clé secrète au moins aussi longue que le message. Le protocole de distribution quantique de clés BB84 permet justement à Alice et Bob de générer cette clé secrète commune, sans possibilité aucune pour un éventuel espion d'avoir aucune information sur cette clé. Une fois qu'ils partagent cette clé secrète, Alice et Bob n'ont plus qu'à utiliser le protocole classique bien connu du « masque jetable » (ou « one time pad » en anglais) [Shannon 48] pour pouvoir communiquer en toute confidentialité. L'implémentation du protocole de distribution quantique de clés a même dépassé le stade du prototype puisqu'il est maintenant possible d'acheter un système plug-and-play fonctionnant par fibres optiques auprès de la compagnie id Quantique. Cette compagnie a été fondée par une équipe du groupe de physique appliquée de l'Université de Genève dirigée par Nicolas Gisin, et son existence illustre le fait que la cryptographie quantique a maintenant franchi les murs du laboratoire pour atteindre celui des applications commerciales. Ce système plug-and-play a été testé sur un parcours de 67 km de long passant sous le lac Léman et reliant les villes de Genève et Lausanne [Stucki et *al* 02].

Une des plus belles applications de l'informatique quantique est la *téléportation quantique* [Bennett et *al* 93]. Si Alice et Bob partagent préalablement une paire d'états intriqués, il est possible pour Alice de transmettre un qubit dans un état inconnu à Bob en utilisant simplement deux bits de communication classique. À la fin du processus Alice ne possède plus le système quantique, ce dernier se retrouve entre les mains de Bob d'où le terme de « téléportation ». Il est important de noter que la téléportation ne s'effectue

pas à une vitesse plus rapide que celle de la lumière, puisque tant que Bob n'a pas reçu les deux bits de communication envoyés par Alice, la téléportation n'est pas complète.

Il existe de nombreuses autres applications pour lesquelles le passage au quantique permet de gagner comparativement au monde classique, c'est par exemple le cas du *calcul distribué quantique* [Buhrman et *al* 98]. Ainsi si l'intrication mentionnée précédemment ne permet pas de communication plus rapide que la vitesse de la lumière entre deux personnes (ce qui aurait pour effet de violer la théorie de la relativité), elle peut néanmoins leur permettre d'établir des corrélations entre eux de manière quasi-instantanée, chose impossible classiquement. Elle peut aussi permettre de réduire la communication nécessaire pour réaliser une tâche distribuée, voire même pour certaines tâches de totalement s'en passer comme dans le cas de la *pseudo-télépathie* [Brassard et *al* 99].

Il faut malgré tout émettre un bémol. En effet, il existe un frein important au développement de l'informatique quantique, celui de la difficulté de construire physiquement un tel système. Le premier problème vient du fait que pour construire un ordinateur quantique, il faut travailler non plus au niveau macroscopique mais à l'échelle de l'atome. Il est beaucoup plus ardu de manipuler un atome qui représenterait un qubit que cela ne l'est par exemple de changer la valeur d'un bit sur un disque dur. En pratique tout système pouvant se trouver en superposition de deux ou plusieurs états est un candidat potentiel en tant que brique de base pour la construction d'un ordinateur quantique. Les couches électroniques distinctes d'un atome, la polarisation d'un photon ou le spin d'un électron en sont des exemples. D'autres propriétés sont néanmoins aussi nécessaires : pouvoir initialiser l'état du système, avoir un ensemble « universel » d'opérations pouvant être réalisées sur ce système et pouvoir mesurer de manière spécifique les qubits du système [DiVincenzo 00]. S'ajoute à cela l'action d'un phénomène typiquement quantique, celui de la *décohérence*; un système quantique qui est en superposition aura tendance à vouloir aller vers un état plus stable à cause des interactions entre le système quantique et l'environnement. On peut s'imaginer l'action de l'environnement comme étant celle de mesurer un petit peu en permanence le système

et d'en extraire un peu d'information. La décohérence peut complètement fausser le résultat d'un calcul quantique. Le temps typique de décohérence d'un système quantique dépend du support qui a été choisi pour l'implémentation. En général on cherchera à trouver des supports ayant un temps de décohérence long en particulier comparé à la vitesse de l'horloge interne de l'ordinateur quantique. Pour prévenir l'effet de la décohérence, il faut essayer d'isoler le plus possible le système quantique de toute interaction avec l'environnement. Plus il est possible d'isoler le système de l'environnement, moins le système décohère rapidement. Malheureusement, en pratique il faut de toute façon prendre pour acquis que la décohérence va venir corrompre et influencer le calcul à un moment ou à un autre. Il existe cependant des techniques permettant de contrôler l'accumulation et la propagation d'erreurs comme les *codes correcteurs d'erreurs* [Steane 01] ou encore le *calcul tolérant aux erreurs* [Gottesman 98].

En pratique, le prototype le plus avancé d'ordinateur quantique construit à l'heure actuelle comporte seulement 7 qubits [Vandersypen et *al* 01]. Il été mis au point par Isaac Chuang et son équipe à IBM Almaden. Malgré le fait qu'on soit très loin encore d'avoir un ordinateur quantique de taille raisonnable, il faut cependant relativiser ce pessimisme. En principe, il serait possible, si on disposait d'un ordinateur quantique capable de travailler sur un millier de qubits, d'effectuer certains calculs ne pouvant pas être réalisés avec un ordinateur classique de la taille de l'univers. Du reste certaines applications comme la cryptographie quantique ne requièrent pas un ordinateur de cette taille et peuvent déjà être réalisées avec la technologie actuelle. À la question de savoir quand on aura vraiment un ordinateur quantique posé sur notre bureau, la réponse varie de quelques années pour les plus optimistes à jamais pour les plus pessimistes. Fondamentalement, il n'existe aucune loi physique connue s'opposant à la mise au point d'un ordinateur quantique.

L'auteur de ce mémoire pense que l'informatique quantique est une science fascinante, prometteuse et qui est en plein développement. Si jamais il devait essayer de convaincre un jeune chercheur de cela, et le motiver pour s'orienter vers ce domaine, plusieurs

arguments pourraient être avancés. Parmi les arguments qui sont les plus pragmatiques (et pas forcément les meilleurs), il pourrait citer le fait que si la loi de Moore continue à se vérifier, on atteindra le mur quantique en 2020, et qu'il vaut mieux s'y préparer à l'avance plutôt que d'attendre au dernier moment et de foncer dedans tête baissée. Un autre argument pragmatique pourrait être que c'est un domaine qui est particulièrement attrayant au niveau des moyens de recherche et des subventions depuis quelques années en conséquence de la découverte d'algorithmes tels que celui de Shor. L'auteur de ce mémoire pense néanmoins qu'il y a d'autres raisons plus importantes de s'intéresser à ce domaine. Premièrement, l'informatique quantique offre un nouveau paradigme de traitement de l'information qui ouvre de nouvelles possibilités et repousse certaines limites de l'informatique classique. Pour tout sous-domaine compris dans l'informatique classique, il serait naturel de se poser la question de savoir si celui-ci pourrait bénéficier d'un passage au quantique. Par exemple en intelligence artificielle, on pourrait se demander si l'utilisation du quantique permettrait que l'ordinateur puisse arriver à un niveau d'apprentissage et de compréhension plus proche de celui de l'humain qu'on ne le peut actuellement avec les méthodes purement classiques. Deuxièmement, les progrès et les avancées en informatique quantique pourraient permettre d'améliorer notre compréhension de la théorie de l'information et aussi peut-être celle du fonctionnement du monde physique en général. Finalement, l'informatique quantique est un domaine encore jeune et bien que disposant déjà de quelques résultats intéressants du point de vue théorique, il reste encore de nombreuses choses à faire et à découvrir, en particulier quant à l'élaboration de nouveaux algorithmes quantiques.

4.2. Pourquoi développer un système tutoriel intelligent pour l'informatique quantique ?

Il existe plusieurs raisons justifiant le développement d'un système tutoriel intelligent pour l'informatique quantique. Premièrement, l'informatique quantique est un domaine jeune mais qui prend une importance grandissante depuis une dizaine d'années, avec la découverte en 1994 de l'algorithme de Shor qui permettrait de briser RSA et plus tard, en 1996 avec l'apparition de l'algorithme de Grover. C'est donc un domaine qui compte encore relativement peu d'experts pour l'instant, mais pour lesquels il y a une demande

importante et croissante. L'existence d'un système tutoriel intelligent enseignant l'informatique quantique pourrait aider à faire découvrir ce domaine auprès des personnes n'ayant pas un expert du domaine sous la main. Ceci s'ajoute bien sûr à l'avantage traditionnel d'un STI de pouvoir être utilisé en complément d'un cours ou d'une leçon pour pouvoir aider l'apprenant à réviser et progresser dans la matière.

La principale originalité au fait de choisir l'informatique quantique, plutôt qu'un autre domaine plus traditionnel comme les mathématiques ou l'histoire, réside dans la nature fondamentale de ce domaine. L'informatique quantique est ce qu'on appelle un domaine *multidisciplinaire*. Il est situé à la croisée de l'*informatique*, des *mathématiques*, de la *physique* et de la *chimie*. L'informatique car il s'agit d'un nouveau paradigme de traitement de l'information, les mathématiques pour le formalisme qui permet de décrire ce paradigme, la physique pour les principes de la mécanique quantique qui explicitent la réalité physique derrière ce paradigme et aussi pour la plupart des implémentations physiques, et enfin la chimie pour certaines implémentations telle que la résonance magnétique nucléaire. Cette multidisciplinarité est une richesse mais constitue aussi un challenge intéressant au niveau du développement d'un système tutoriel intelligent, tant pour l'acquisition des connaissances auprès des experts que pour l'enseignement plus tard. Le côté multidisciplinaire apporte donc une dimension intéressante de plus comparé au STI enseignant un domaine traditionnel. Tout d'abord durant la phase de construction du curriculum, il faudra gérer la nature multidisciplinaire de la connaissance, alors que plus tard on pourra utiliser cet aspect multidisciplinaire pour essayer d'adapter de façon flexible l'enseignement aux apprenants venant de domaines différents. Ainsi l'enseignement délivré par le STI pourrait être différent, suivant par exemple que l'étudiant vienne d'une maîtrise en mathématiques, ou plutôt qu'il soit en train de passer un doctorat en physique. Un STI a l'avantage de pouvoir contourner le problème que rencontre les manuels traditionnels, à savoir délivrer un enseignement statique et relativement ciblé. Dans le cas de l'informatique quantique par exemple, le livre de Chuang et Nielsen [Chuang et Nielsen 01] qui constitue la référence à l'heure actuelle est écrit d'une façon telle qu'il est plutôt destiné à des physiciens au premier abord. Un STI

multidisciplinaire pourrait passer sur certaines notions et en approfondir d'autres en tenant compte du background des étudiants.

De plus, on pourrait imaginer utiliser le formalisme développé pour l'enseignement d'autres domaines multidisciplinaires tels que la bioinformatique, qui se situe à la jonction de la biologie, de l'informatique et des mathématiques, ou encore le commerce électronique, qui se trouve au confluent du commerce, de l'informatique et du droit. En poussant le raisonnement à l'extrême, ou pourrait même penser créer un système tutoriel intelligent enseignant le domaine des systèmes tutoriels intelligents, qui lui aussi est un domaine multidisciplinaire…

4.3. Le projet QUANTI

Le projet QUANTI vise à la mise au point d'un système tutoriel intelligent pour l'enseignement de l'informatique quantique. Il est le fruit d'un travail conjoint entre Esma Aïmeur du laboratoire de systèmes tutoriels intelligents (HÉRON) et Gilles Brassard du laboratoire d'informatique théorique et quantique (LITQ) de l'université de Montréal. L'auteur de ce mémoire est dans le projet depuis son début. Benoît Fusade et Emmanuel Blanchard ont travaillé sur le projet à son début pour aider à la mise au point et la construction de l'outil. Actuellement, il y a une collaboration en cours avec Michel Toulouse et Nivedita Ramchand Kadaba de l'université du Manitoba afin d'aboutir à la création d'un éditeur de circuits quantiques orienté vers la pédagogie. Contrairement aux éditeurs qui existent déjà à l'heure actuelle, son but serait plus d'aider l'utilisateur à comprendre ce qui se passe à l'intérieur du circuit que de simuler un calcul quantique avec le maximum d'efficacité.

Chapitre 5 : L'acquisition des connaissances dans QUANTI

Ce chapitre présente l'outil d'acquisition des connaissances qui a été développé pour QUANTI. Le but de cet outil est de permettre à plusieurs experts, possiblement séparés par une longue distance, de collaborer ensemble à la construction d'une base de connaissances multidisciplinaire. Cette base de connaissances ayant pour fonction principale de servir de matière pour l'enseignement délivré par le système tutoriel intelligent. À noter que les travaux résumés dans ce chapitre ont conduit à la publication de deux articles [Aïmeur et *al* 01a, 01b] auxquels on pourra se référer pour plus de détails.

5.1. Représentation des connaissances

Pour l'outil d'acquisition des connaissances développé pour QUANTI, le choix a été fait de séparer la phase de *définition de la connaissance* de celle de *structuration de la connaissance*. Cela signifie que, dans un premier temps, on demandera seulement aux experts d'exprimer leurs connaissances sur le domaine sans chercher à spécifier les relations qui relient ces connaissances entre elles. Les relations entre ces connaissances seront précisées lors d'une seconde phase par les experts. On appelle *dictionnaire* la base de connaissances issue de la phase de définition de la connaissance, et *curriculum* la base de connaissances plus raffinée issue de la phase de structuration de la connaissance.

5.1.1. Dictionnaire

On pourrait définir le dictionnaire (cf. Figure 3) comme stockant la connaissance du domaine dans son état *brut*, sans aucune classification ou structuration autres que celles par défaut qu'est l'ordre alphabétique. Le dictionnaire contient le *vocabulaire* de base du domaine. Chaque *mot* du dictionnaire représente un *élément de base* du domaine. Il est en outre possible d'associer une ou plusieurs *définitions* à un même mot, et comme le domaine est multidisciplinaire chaque définition est typée par un *sous-domaine* de l'informatique quantique. Par exemple, le mot *entropie* possède trois définitions très différentes, suivant qu'on se réfère à son sens en physique ou à celui qu'il a en

mathématiques et en informatique. Par sous-domaine, on entend les quatre sous-domaines de l'informatique quantique que sont l'informatique (ici « classique »), les mathématiques, la physique et la chimie, plus le « sous-domaine général » qui signifie dans notre cas l'informatique quantique elle-même. Dans le « sous-domaine général » se trouve tous les mots (ou plutôt les définitions) appartenant spécifiquement à l'informatique quantique, et n'ayant aucune existence propre dans un des sous-domaines. Entrent dans cette catégorie de mots, l'algorithme de Shor, le protocole BB84, etc. qui sont des notions typiquement et exclusivement quantiques.

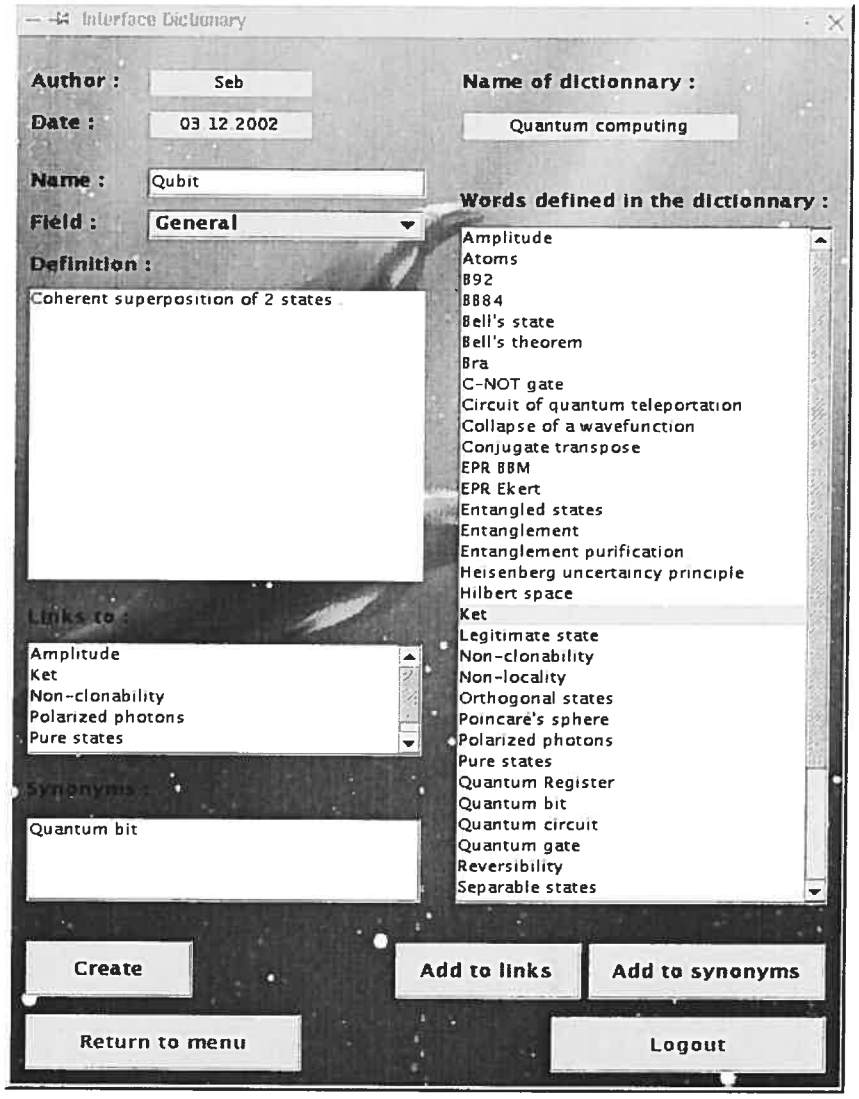

Figure 3 : Interface du dictionnaire.

Chaque mot du dictionnaire peut être relié à des *synonymes* et des *notions proches*. Dans le contexte de QUANTI, on fait référence au sens usuel du mot synonyme, à savoir un mot (ou une expression) partageant la même signification qu'un autre mot. Par exemple, en informatique quantique le mot qubit et l'expression « bit quantique » sont sémantiquement équivalents. Les notions proches servent à souligner une connexion entre le mot actuel et d'autres mots. Par exemple, il pourrait sembler logique que la téléportation ait comme notion proche Star Trek. L'intérêt principal de définir des notions proches et des synonymes, est de pouvoir utiliser *a posteriori* ces informations lors de la création du curriculum pour faciliter la mise en place de liens éventuels entre un mot (et donc un élément du domaine) et d'autres mots. Ainsi, lors de la construction du curriculum, les synonymes et notions proches d'un mot seront mis en avant et portés à l'attention de l'auteur lors de la sélection de ce mot.

Les autres informations stockées pour un mot, sont la date de création de ce mot ainsi que le nom de l'auteur primaire, c'est-à-dire la personne qui a inséré le mot dans le dictionnaire. Pour chaque définition, on garde aussi en mémoire en plus de la définition elle-même, le sous-domaine auquel elle appartient ainsi que le nom de l'auteur de cette définition.

5.1.2. Curriculum

Le curriculum représente la connaissance sous sa forme *structurée*. Alors que dans le dictionnaire, on se contente de définir les éléments de base du domaine, dans le curriculum on va préciser les relations intrinsèques entre ces éléments. La manière de structurer la connaissance sous forme du curriculum à partir d'un dictionnaire n'est pas unique. De même que deux personnes peuvent avoir deux conceptions différentes d'une même matière (par exemple deux professeurs), il est possible d'imaginer construire deux curriculums (relativement) différents à partir du même dictionnaire. La connaissance à la base reste la même, car il s'agit du même dictionnaire, mais les deux curriculums différents incarnent *deux façons différentes de voir et de penser la matière*, et donc de la structurer et de l'agencer, et ultimement de l'enseigner (cf. planificateur).

Le mode de représentation choisi pour le curriculum est celui des *réseaux sémantiques* [Sowa 84, 00]. Pour rappel, les réseaux sémantiques (cf. Figure 4) sont des graphes dont les nœuds sont des *morceaux de connaissance*, et les arêtes symbolisent les *relations* entre ces nœuds. Dans notre cas, les nœuds formant les réseaux sémantiques renvoient chacun directement à une entrée du dictionnaire. Le curriculum est organisé de manière *hiérarchique*. Au sommet de la hiérarchie trône le réseau des *concepts*. Les concepts constituent les éléments les plus *généraux* du domaine. Les relations entre les concepts peuvent être de type *prérequis*, *composition*, ... Lorsqu'on descend d'un niveau, chaque concept se développe à son tour en un réseau sémantique dont le concept devient le nœud *racine*. Les nœuds dans ces réseaux sémantiques peuvent être de quatre types : *concept*, *composant*, *caractéristique* ou *exemple*. Un composant représente un élément de base du domaine dont le sens contribue à définir le sens d'un autre composant ou d'un concept. Une caractéristique symbolise, comme son nom l'indique, une caractéristique ou une propriété qui, soit sert à définir un concept ou un composant, soit découle d'une autre caractéristique. Un exemple sert à illustrer concrètement une notion abstraite de type concept ou composant.

Les liens entre les nœuds peuvent être de plusieurs types : *prérequis*, *composition*, *caractéristique*, *association*, *exemple*, *non-exemple*, *faux-ami*, *exception*, *analogie* ou *exclusion*. Les liens sont toujours *orientés*, sauf les liens comme analogie ou exclusion qui peuvent être considérés comme étant bidirectionnels. Dans chaque graphe chaque nœud a donc en général un nœud parent, sauf le nœud racine, car normalement aucun nœud ne devrait se retrouver seul en train de dériver. Il est cependant possible pour un nœud d'être pointé par plusieurs types de liens. Par exemple, un nœud donné pourrait servir d'exemple pour un concept et de non-exemple pour un autre.

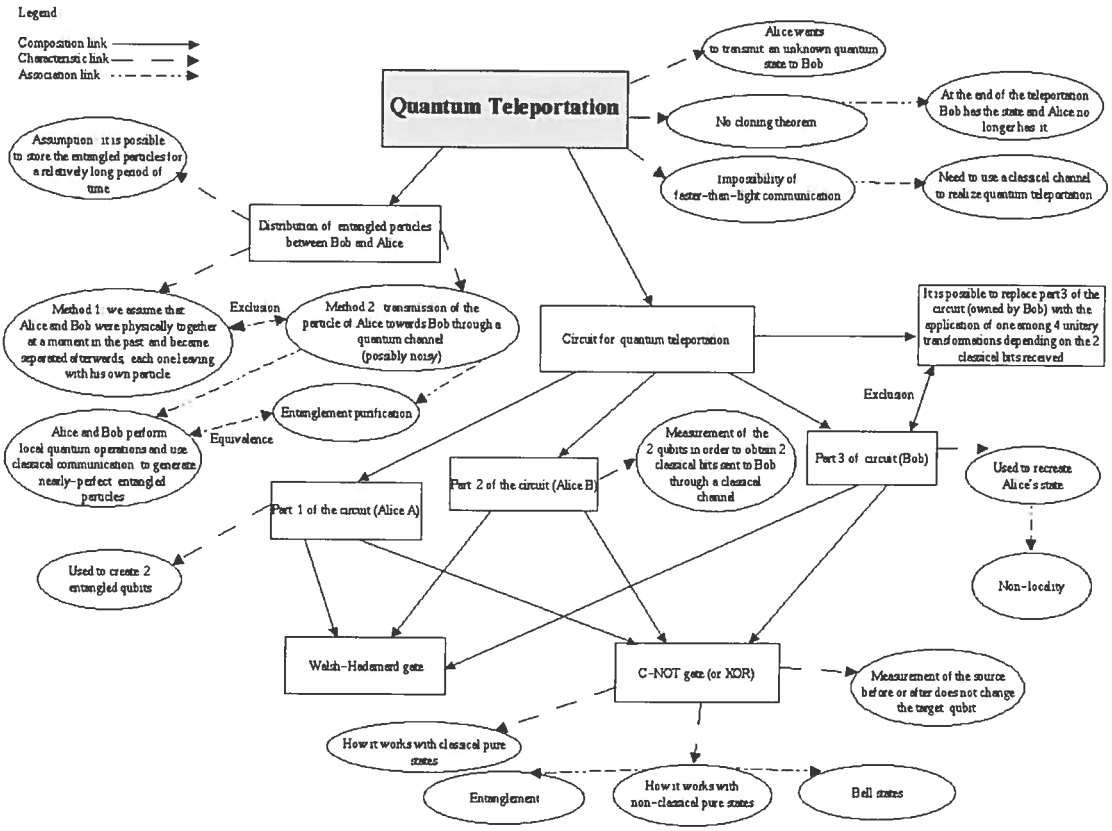

Figure 4 : Réseau sémantique de la téléportation quantique.

5.1.3. Élicitation des connaissances

Afin de créer un réseau sémantique représentant un concept dans le curriculum, l'auteur a le choix entre deux modes d'élicitation: le *mode libre* et le *mode guidé*. Dans le mode libre, l'auteur est invité à choisir les mots du dictionnaire qu'il veut insérer en tant que nœuds dans le réseau sémantique, puis à préciser les liens entre ces nœuds dans l'ordre qu'il désire. L'auteur est donc laissé libre de faire ce qui lui plaît sans aucune contrainte.

Le mode guidé contraste avec le mode libre, dans le sens qu'ici on va chercher à guider l'auteur lors de l'élicitation des connaissances pour l'aider à structurer ses connaissances, contrairement au mode libre où il est livré à lui-même. Pour le guider, on va utiliser un *algorithme d'élicitation*. Dans le cas de QUANTI, l'algorithme d'élicitation qui a été développé dans le cadre de l'outil d'acquisition s'appelle HAKE (Hybrid Algorithm for Knowledge Elicitation). Il s'agit d'un nouvel algorithme, mais qui est un hybride entre

les méthodes d'élicitation en profondeur d'abord et en largeur d'abord qui sont plus « classiques ». Une évaluation de l'algorithme HAKE ainsi que plus de détails sur la notion d'algorithme d'élicitation pourront être trouvés dans le chapitre suivant.

5.2. Architecture du système

Comme précisé dans le chapitre précédent, le nombre d'experts en informatique quantique est encore relativement peu élevé dans le monde, ce qui peut poser un problème au niveau de l'acquisition des connaissances. Pour pallier à ce problème et permettre à un maximum d'experts de pouvoir contribuer à la construction de la base de connaissances, le choix a été fait de développer un outil d'acquisition des connaissances fonctionnant *à distance* et permettant à plusieurs auteurs de *collaborer* pour créer la base de connaissances.

5.2.1. Architecture client-serveur

L'architecture de l'outil d'acquisition des connaissances (cf. Figure 5) est de type *client-serveur*. En clair, cela signifie que la base de connaissances proprement dite est physiquement stockée au niveau du *serveur*, alors que les experts peuvent se connecter sur le système de n'importe où dans le monde en utilisant le logiciel *client* et un réseau de communication comme Internet.

Avant de pouvoir rentrer dans le système, un auteur doit d'abord *s'identifier* à travers son login et son mot de passe. Les profils des auteurs sont contenus dans une base de données existante au niveau du serveur. Si une nouvelle personne veut s'ajouter à la liste des auteurs existants, il faudra qu'il passe par l'administrateur du système pour s'enregistrer, avant d'avoir le droit de modifier la base de connaissances. Le choix d'autoriser ou non une personne à devenir un auteur est du ressort de l'administrateur, et est donc forcément en partie subjectif. Cette façon de procéder a été préférée à d'autres solutions plus ouvertes, comme par exemple la possibilité pour n'importe quelle personne d'être un auteur comme c'est le cas dans les wikis, car cela permet un accès plus *sécurisé* à la base de connaissances. Évidemment, cela réduit le nombre potentiel d'auteurs pouvant

participer à la création de la base de connaissances, mais comme le curriculum va être utilisé comme la base du cours du STI, il est important que les sources dont sont tirées les connaissances (c'est-à-dire les auteurs) soient *fiables*.

Chaque auteur est considéré comme *compétent* dans un ou plusieurs domaines d'expertise. À noter qu'un auteur n'est pas limité à donner des définitions seulement dans le (ou les) domaine(s) où il est compétent, mais que lors d'un *conflit* éventuel entre deux auteurs à propos d'une définition, si l'un des deux est compétent dans le domaine auquel appartient la définition et l'autre pas, cela donne plus de poids à la définition de celui qui est compétent. On se réfèrera à la Section 5.2.2. pour avoir plus de détails sur le protocole de résolution de conflits. À noter que les domaines de compétence d'un auteur sont définis au moment de son enregistrement en tant que nouvel auteur.

Pour résumer, les bases de données présentes sur le serveur sont :
-La *base de données des profils auteurs* mentionnée plus haut.
-La *base de connaissances contenant les dictionnaires*.
-La *base de connaissances contenant les curriculums*.
-La *base des ressources* qui contient les *ressources didactiques* comme les simulations, les démonstrations, les exercices, les problèmes, etc... Les ressources peuvent être sous forme de textes, de pages HTML, de vidéos, de sons, etc...
-L'*entrepôt de données* garde l'historique de toutes les actions effectuées dans le système. Il sera possiblement utilisé ultérieurement à des fins de forage de données.

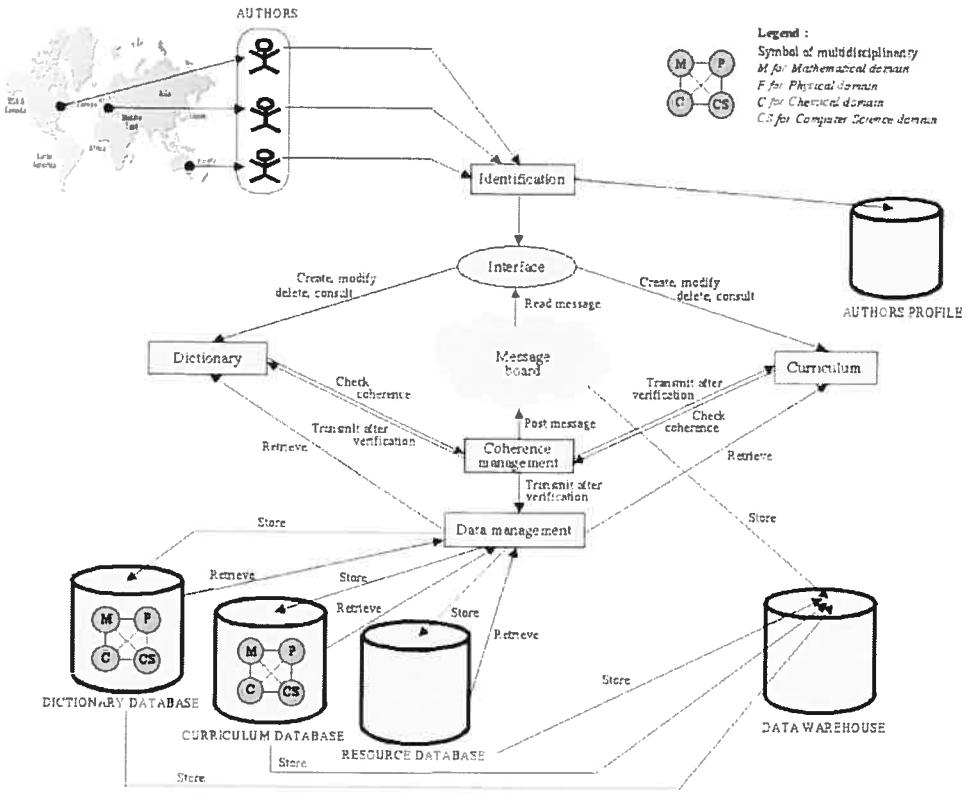

Figure 5 : Architecture du système.

5.2.2. Communication entre les experts du domaine et gestion des conflits

Comme l'architecture est de type client-serveur et que deux auteurs, participant potentiellement à la création du même curriculum, peuvent être situés à des milliers de kilomètres l'un de l'autre, il est important qu'ils disposent d'un *moyen de communication* entre eux. Pour mettre en place cette communication, un *tableau de messages* (ou *messageboard* en anglais) a été implémenté directement à même le système (cf. Figure 6). En pratique, ce tableau de messages fonctionne comme une boîte aux lettres électronique classique. Elle autorise l'envoi et la réception de messages, permettant ainsi l'échange d'informations, et idéalement surtout de points de vue, de manière *asynchrone* entre les auteurs. Il est évident que le système pourrait retirer des bénéfices de la disponibilité d'un mode de communication *synchrone*, comme un mode de dialogue en

ligne par exemple qui permettrait à plusieurs auteurs d'échanger en direct leurs points de vues, mais pour l'instant cette possibilité n'est pas encore implémentée.

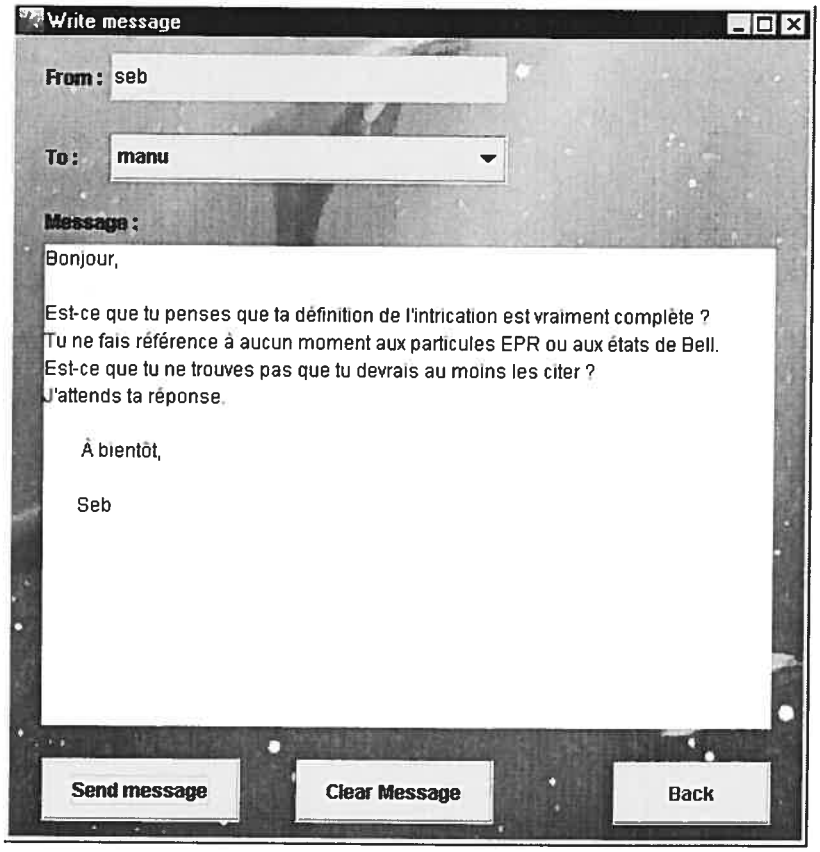

Figure 6 : Exemple de message pouvant être envoyé avec le messageboard.

Le premier moyen pour régler d'éventuels conflits entre deux auteurs reste bien sûr la communication. C'est d'ailleurs en grande partie pour cette raison que le tableau de messages existe. Malgré tout, il est normal que parfois deux auteurs n'arrivent pas à s'entendre sur une définition commune pour un mot et dans ce cas c'est au système que revient le rôle d'*arbitre*. Le protocole utilisé pour QUANTI utilise le fait que les auteurs ont des domaines de compétence pour, en cas de litige, favoriser ceux dont la définition correspond à leurs domaines de compétence. Le protocole en lui-même est assez simple et intuitif. Imaginons qu'un auteur souhaite modifier une entrée créée par un autre auteur. C'est ce qui arrive typiquement si un auteur est en désaccord avec la définition d'un autre auteur. Dans un premier temps, il va demander la permission à cet auteur à travers le tableau de messages. Quatre cas peuvent alors survenir :

1. Si les deux auteurs s'entendent sur la modification, elle est implémentée.

2. Si les deux auteurs sont du même domaine d'expertise mais qu'ils n'arrivent pas à se mettre d'accord sur la modification, les deux opinions sont sauvegardées par le système. Dans ce cas, chacune des deux définitions est étiquettée par le nom de son auteur.

3. Si les deux auteurs sont de deux disciplines différentes, et que la définition est clairement du domaine de l'un d'entre eux, le système donne priorité à l'auteur de la discipline correspondante. Dans ce cas, l'opinion de l'auteur non spécialiste ne sera pas prise en compte par le système.

4. Si les deux auteurs sont de disciplines différentes et que le concept appartient aux deux disciplines, alors le système prend en compte les deux opinions et les garde en mémoire.

Ce protocole ressemble au protocole de construction collaborative de base de connaissances du nom de CO4 [Euzenat 95].

Chapitre 6 : Évaluation d'un nouvel algorithme d'élicitation

Dans ce chapitre on va présenter un nouvel algorithme d'élicitation développé dans le cadre de QUANTI. On va aussi chercher à évaluer sa pertinence et son efficacité par rapport à d'autres algorithmes d'élicitation. Ce nouvel algorithme d'élicitation s'appelle HAKE et est en fait un hybride entre deux approches plus classiques que sont profondeur d'abord et largeur d'abord. Deux expérimentations furent successivement menées afin de chercher à évaluer l'algorithme, la seconde cherchant à apprendre des limites et des défauts de la première. On pourra trouver plus de détails sur les deux expériences dans un article publié dans le cadre d'un workshop de la conférence IJCAI 03 [Aïmeur et *al* 03].

6.1. Qu'est ce qu'un algorithme d'élicitation et quel est son but?

Le but principal de l'acquisition des connaissances est de permettre un transfert de connaissances extraites d'un humain vers un ordinateur. Afin de faciliter un tel transfert, il faut fournir à l'humain une façon d'exprimer son savoir qui soit pour lui « la plus naturelle et la plus intuitive possible ». Par « (la plus) naturelle et (la plus) intuitive (possible) », on veut simplement dire une manière de faire qui ressemble le plus possible à la façon dont l'expert organise ses connaissances dans son esprit. On appellera *algorithme d'élicitation*, un algorithme essayant de tendre vers ce but et dont la raison d'être est d'aider un expert à expliciter et structurer ses connaissances. On peut supposer que l'expert a déjà un réseau sémantique représentant la connaissance dans son esprit, même s'il n'en est pas forcément conscient. L'idée (ou plutôt l'espoir) est qu'un algorithme d'élicitation proche du comportement mental de l'expert devrait rendre la tâche de transmission des connaissances plus efficace et plus fiable.

Dans ce chapitre, on regardera uniquement des algorithmes d'élicitation s'occupant de l'acquisition des connaissances dans le cas de réseaux sémantiques, ce qui est pertinent pour QUANTI puisqu'il s'agit du mode de représentation des connaissances utilisé pour le curriculum.

6.2. Variantes de profondeur d'abord et largeur d'abord comme algorithmes d'élicitation

Les approches en *profondeur d'abord* et *largeur d'abord* pour l'élicitation des connaissances sont directement inspirées de leurs cousins utilisés pour la recherche dans un graphe. Cependant il faut noter qu'ici, *on ne cherche pas à explorer* un graphe qui existe déjà, mais plutôt qu'on veut assister l'expert pour créer le graphe. L'algorithme utilisé va *influencer l'ordre* dans lequel les nœuds et les arêtes vont être créés. Ceci contraste avec une élicitation non-guidée où l'ordre d'entrée des nœuds et des arêtes est laissé totalement au choix de l'expert. Pour des raisons de simplicité, les algorithmes vont être illustrés en utilisant des arbres, cependant les expériences réelles sont faites à partir de graphes plus compliqués, comme décrit à la Section 6.4.1.1.

En général, les techniques d'élicitation peuvent être utilisées pour créer des nœuds (objets) en plus de servir à définir des relations entre lesdits nœuds. Dans notre cas, ce n'est pas exactement ce qui se passe. La différence dans QUANTI est que l'ensemble des objets a déjà été défini au préalable lors de la création du dictionnaire. Les utilisateurs n'ont donc pas besoin de rentrer manuellement les nœuds, mais simplement de typer ces nœuds et de créer les arêtes entre les nœuds.

6.2.1. Profondeur d'abord

Avec l'approche en profondeur d'abord, l'expert commence la création du graphe à partir du nœud *racine* en spécifiant un lien entre ce nœud racine et un de ses nœuds fils; ensuite l'attention se redirige *directement* sur cet enfant sans attendre que l'expert ait entré tous les enfants du nœud racine. Le processus d'élicitation continue ensuite de la même manière : à chaque fois qu'un lien est créé entre un nœud et un de ses enfants, l'élicitation continue directement avec le nœud enfant. Quand le processus d'élicitation atteint un cul-de-sac (par exemple une *feuille*), l'algorithme revient sur ses pas en retournant au nœud parent, puis continue l'élicitation. Les nombres figurant sur les arêtes de la Figure 7 illustrent l'ordre dans lequel un expert qui utiliserait l'approche en profondeur d'abord pourrait éliciter un graphe explicitant un écosystème. Sur la Figure 7, la présence d'une

arête entre deux nœuds symbolise le fait qu'un être vivant (représenté par un nœud) puisse être mangé par un autre être vivant.

Un humain créant un graphe en profondeur d'abord peut être vu comme quelqu'un suivant le fil de ses idées au fur et à mesure qu'elles arrivent. Il commence par penser à un objet A, puis ensuite il imagine une association entre cet objet A et un autre objet B. Maintenant, il cherche à trouver une association entre cet objet B et un autre objet, et cela continue ainsi jusqu'à qu'il soit incapable d'associer l'objet qu'il a actuellement en tête avec un autre objet. Quand ce stade est atteint, l'expert remonte à l'objet précédent, faisant ainsi ce qu'on appelle du *backtracking*, et reprend maintenant le processus. L'avantage de cette technique est que l'expert peut suivre une idée jusqu'à la fin sans aucun délai: en particulier, il n'a pas à attendre d'avoir énuméré tous les descendants d'un nœud avant de pouvoir continuer l'élicitation avec le nœud suivant. De plus, il n'y a aucun risque pour l'expert de se perdre en chemin puisqu'il n'y a jamais aucun *saut conceptuel* (voir Section 6.2.2.) entre deux nœuds consécutifs durant le processus d'élicitation. En effet, la distance entre deux nœuds consécutivement élicités en terme de nombre d'arêtes est toujours de un. Le principal inconvénient de l'approche en profondeur d'abord est que, le temps que l'auteur revienne au nœud racine après avoir élicité tout le sous-graphe du premier fils, la liste de tous les descendants qu'il avait dans son esprit au début du processus ne soit plus très fraîche dans sa mémoire.

L'approche en profondeur d'abord pour l'élicitation est donnée plus formellement ci-dessous en pseudo-code. À noter qu'on dit d'un nœud qu'il est *visité*, si l'expert l'a au moins une fois considéré en créant un lien vers lui. La racine est considérée comme visitée dès le départ, même si aucun lien en provenance d'un autre nœud ne pointe vers elle.

Processus initialisation_profondeur(*racine*)
pour chaque *mot* dans le Dictionnaire **faire**
    marquer le *mot* comme non visité
demander à l'expert de choisir le type du nœud *racine*
marquer *racine* comme ayant été visité
appeler élicitation_profondeur(*racine*)

Processus élicitation_profondeur(*node*)
répéter
    demander à l'expert de sélectionner une entrée *succ* du Dictionnaire sur laquelle *node* pointera
    demander à l'expert de choisir le type de l'arête allant de *node* à *succ*
    si *succ* n'a pas été visité **alors**
        demander à l'expert de choisir le type du nœud *succ*
        marquer *succ* comme ayant été visité
        appeler élicitation_profondeur(*succ*)
jusqu'à ce que l'expert ne souhaite plus sélectionner un autre successeur pour *node*

Pseudo-code de l'algorithme d'élicitation en profondeur d'abord

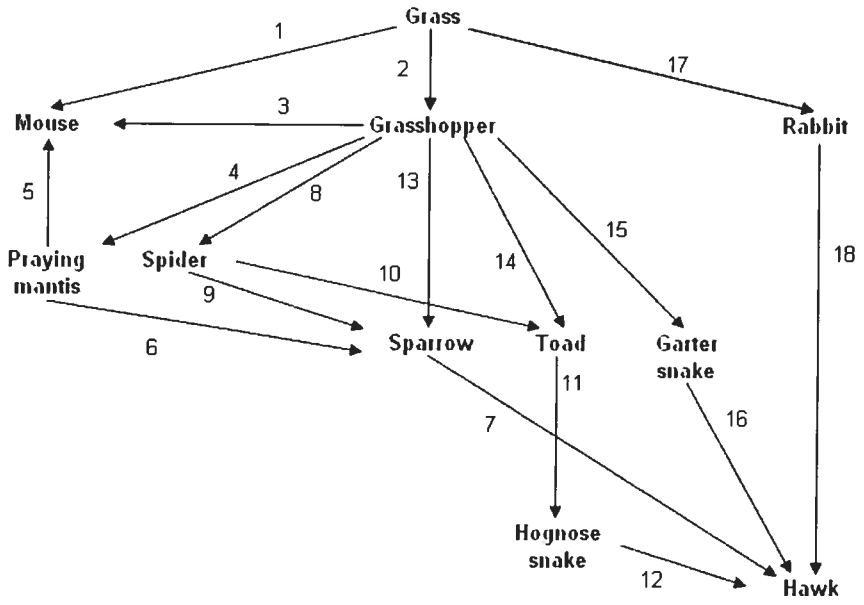

Figure 7 : Illustration d'une élicitation en profondeur d'abord.

6.2.2. Largeur d'abord

En utilisant l'approche en largeur d'abord, qui est illustrée par la Figure 8 avec le même exemple que précédemment, l'expert commence depuis le nœud racine et spécifie tous ses enfants en créant les arêtes correspondantes. La racine est considérée comme se trouvant au niveau 0 alors que ses enfants se situent au niveau 1. Ensuite, l'expert considère les nœuds du niveau 1 un par un, et il crée tous les liens entre eux et *leurs* descendants; ces derniers étant maintenant considérés comme appartenant au niveau 2. Le

processus d'élicitation continue de la même manière, niveau par niveau, jusqu'à ce que le graphe entier ait été élicité.

Le principal avantage de la méthode en largeur d'abord, est qu'elle permet à l'expert de se focaliser sur un noeud particulier et de donner toutes les relations entre ce nœud et tous les autres nœuds avant de continuer. Ainsi, l'expert n'aura pas à revenir plus tard sur ce nœud et n'aura donc eu besoin de se concentrer sur lui qu'une seule fois. Durant le processus d'élicitation, un *saut conceptuel* apparaît lorsque la distance entre deux nœuds successivement élicités est grande. Plus la distance entre ces deux nœuds est grande, plus grand est le risque que l'expert puisse être confus ou retardé suite à ce saut conceptuel. En effet, l'expert pourrait ne pas comprendre la relation entre les deux nœuds, ni pourquoi ils se suivent dans l'ordre d'élicitation de l'algorithme. Un des inconvénients de l'approche en largeur d'abord est que si le graphe est profond, il y aura forcément de temps en temps des sauts conceptuels importants entre deux nœuds successivement élicités. En fait, plus le graphe est profond, plus grands et plus nombreux seront les sauts conceptuels. Ainsi, il pourrait devenir difficile pour l'expert de réussir à garder en tête l'aspect général du graphe.

```
Processus élicitation_largeur(racine)
pour chaque mot dans le Dictionnaire faire
        marquer le mot comme non visité
initialiser file comme une file vide
demander à l'expert de choisir le type du nœud racine
marquer racine comme ayant été visité
ajouter racine à file
répéter
        soit node le premier élément de file
        enlever node de file
        répéter
                demander à l'expert de sélectionner un entrée succ du Dictionnaire sur
                laquelle node pointera
                si succ n'a pas été visité alors
                        demander à l'expert de choisir le type du nœud succ
                        marquer succ comme ayant été visité
                        ajouter succ à file
                demander à l'expert de choisir le type de l'arête allant de node à succ
        jusqu'à ce que l'expert ne souhaite plus sélectionner un autre successeur pour
        node
jusqu'à ce que file soit vide
```

Pseudo-code de l'algorithme d'élicitation en largeur d'abord

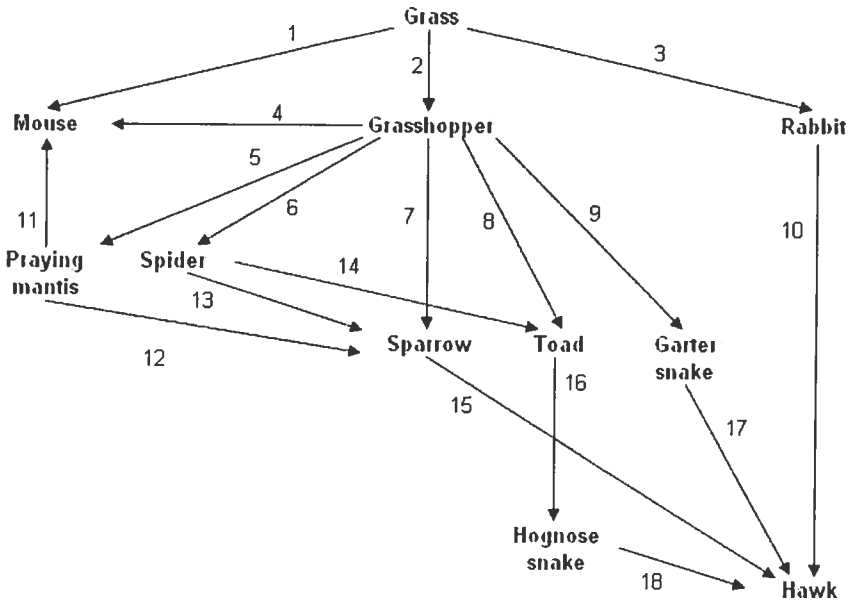

Figure 8 : Illustration d'une élicitation en largeur d'abord.

6.3. Présentation de HAKE

Le nouvel algorithme d'élicitation développé dans le cadre de l'outil d'acquisition des connaissances de QUANTI est un *hybride* entre les approches en profondeur d'abord et en largeur d'abord, d'où son nom HAKE (Hybrid Algorithm for Knowledge Elicitation). Cet algorithme a été inventé par Esma Aïmeur et Gilles Brassard. Leur but en imaginant cet algorithme était de réussir à combiner les points forts des deux approches tout en évitant leurs faiblesses. Il est important de rappeler une fois de plus que le but ici n'est pas d'explorer un graphe qui existe déjà mais bien de guider l'expert dans la création de ce graphe. HAKE commence son élicitation de la même manière que *largeur d'abord* : l'expert part du nœud racine et spécifie tous ses enfants en créant les liens correspondants. Mais ensuite, l'attention se focalise sur le premier enfant, comme pour *profondeur d'abord*, et on requiert de l'expert qu'il continue ainsi récursivement en élicitant entièrement le sous-graphe accessible depuis ce fils, en commençant avec l'élicitation de tous ses descendants. Quand le processus atteint un cul-de-sac parce que le premier sous-graphe a été complété, un retour en arrière sera effectué et le focus se place sur le second fils de la racine, et de nouveau le deuxième sous-graphe sera élicité en

entier avant de revenir encore en arrière et de passer au fils suivant de la racine. Ce processus est répété jusqu'à ce que le focus ait été placé successivement sur chaque enfant de la racine, et que le graphe entier ait été élicité.

Pour comprendre le pseudo-code ci-dessous détaillant le fonctionnement de HAKE, on aura besoin de définir une nouvelle notion. On dira d'un nœud qu'il est *élicité* si toutes les relations partant de ce nœud et allant vers d'autres nœuds ont été définies. Encore une fois, c'est l'exemple de la chaîne alimentaire qui sera utilisé pour illustrer HAKE dans la Figure 9.

```
Processus initialisation_HAKE(racine)
pour chaque mot dans le Dictionnaire faire
        marquer le mot comme non visité et non élicité
demander à l'expert de choisir le type du nœud racine
marquer racine comme ayant été visité
appeler élicitation_HAKE(racine)

Processus élicitation_HAKE(node)
initialiser file comme une file vide
{ sélection en largeur d'abord de tous les successeurs immédiats de node }
répéter
        demander à l'expert de sélectionner un entrée succ du Dictionnaire sur laquelle
        node pointera
        si succ n'a pas été visité alors
                demander à l'expert de choisir le type du nœud succ
                marquer succ comme ayant été visité
        si succ n'a pas été élicité alors
                ajouter succ à file
        demander à l'expert de choisir le type de l'arête allant de node à succ
jusqu'à ce que l'expert ne souhaite plus sélectionner un autre successeur pour node
marquer node comme ayant été élicité
{ élicitation en profondeur d'abord de ces successeurs }
tant que file n'est pas vide faire
        soit succ le premier élement de file
        enlever succ de file
        si succ n'a pas été élicité alors
                appeler récursivement élicitation_HAKE(succ)
```

Pseudo-code de HAKE

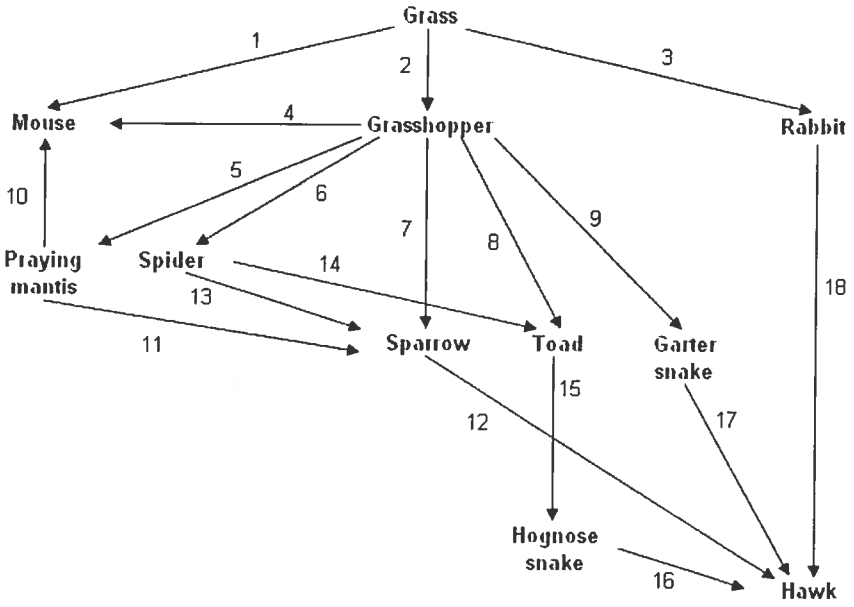

Figure 9 : Illustration d'une élicitation avec HAKE.

6.4. Évaluation de HAKE

Dans les sections précédentes, il a été expliqué pourquoi les méthodes d'élicitation en profondeur d'abord et en largeur d'abord semblent en partie inappropriées et quels sont leurs défauts présumés. Le but de HAKE est justement d'essayer de pallier à ces défauts. Reste maintenant le plus difficile à faire, c'est à dire prouver que HAKE est effectivement plus approprié que ses deux cousins. Il est impossible de donner une preuve mathématique que HAKE est meilleur que ses concurrents. En effet, alors que les algorithmes d'exploration de graphe peuvent facilement être comparés en terme d'espace, de temps, et par rapport à la qualité de la solution trouvée, les algorithmes d'élicitation ne peuvent pas être comparées aussi directement. La comparaison doit être faite *empiriquement*, en testant les algorithmes directement auprès d'humains. Deux expériences ont été menées afin de déterminer si oui ou non l'intuition que HAKE est meilleur que ses deux cousins est fondée. Ainsi une expérience préliminaire a eu lieu durant l'automne 2002, qui a été suivie en 2003 par une autre expérimentation plus élaborée.

6.4.1. Première expérimentation

6.4.1.1. Description

La première expérimentation fut conduite avec 45 volontaires en utilisant un logiciel graphique écrit en Java spécifiquement pour le cadre de l'expérience. L'évaluation a consisté en trois parties différentes : l'élicitation libre d'un graphe fourni sur papier, l'élicitation guidée du même graphe en utilisant chacun des algorithmes l'un après l'autre, et l'élicitation libre d'un autre graphe, ce graphe n'étant cette fois là pas donné explicitement contrairement à celui de la première partie. Un post-test concluait l'expérimentation. Le graphe élicité dans les deux premières parties de l'expérimentation a été le graphe de la chaîne alimentaire des Figures 7, 8 et 9. Il était fourni sur papier aux volontaires, sans les numéros bien sûr et avec quelques arêtes de plus qu'il ne porte sur les Figures 7, 8 et 9 (en fait dans cette version il contenait exactement 11 nœuds et 20 arêtes). Cet exemple, ainsi que le graphe utilisé pour la troisième partie, avaient été choisi car il s'agit des domaines pour lesquels tout le monde possède quelques connaissances. Le graphe élicité dans la troisième partie contenait 20 nœuds et 21 arêtes et portait sur la décomposition des ingrédients nécessaires à la préparation de la pizza Margarita. Toutes les actions d'un utilisateur étaient enregistrées dans un fichier de log. En outre, un assistant était toujours disponible à côté du volontaire et prêt à l'aider au cas où celui-ci aurait eu quelque question que ce soit.

6.4.1.2. Critères

Il faut se rappeler que durant la première et la troisième partie, l'utilisateur est laissé libre d'éliciter le graphe dans l'ordre qu'il souhaite. Basé sur le comportement observé, le but est de réussir à trouver lequel des trois algorithmes se rapproche le plus de la façon naturelle d'éliciter de l'utilisateur. Pour réaliser cela, il faut définir une *métrique* qui servira à mesurer la similarité entre deux ordres d'élicitation, ou plus généralement entre l'ordre observé et une stratégie d'élicitation spécifique. Supposons maintenant qu'un numéro est assigné à chaque arête du graphe. Comme tous les nœuds du graphe sont

connus à l'avance, une élicitation peut être representée comme une séquence de numéros correspondant à l'ordre chronologique de création des arêtes. On appelle *séquence d'élicitation* une telle séquence. Pour un graphe particulier, il est possible de générer de manière exhaustive toutes les séquences possibles d'élicitation pour chacun des algorithmes profondeur d'abord, largeur d'abord et HAKE. Le *nombre minimum d'échange de deux éléments* requis pour passer d'une séquence à une autre peut constituer une *mesure possible de distance* entre deux séquences d'élicitation. C'est cette mesure qui a été utilisée durant la première expérimentation pour calculer la distance séparant la séquence d'élicitation de l'utilisateur de toutes les séquences possibles correspondant à un algorithme d'élicitation spécifique; de toutes ces distances calculées, on garde *la plus petite distance* comme notre métrique. Pour résumer, pour un algorithme d'élicitation donné, on va utiliser comme métrique la plus petite distance en terme d'échange de deux éléments entre la séquence d'élicitation de l'utilisateur et toutes les séquences d'élicitation imaginables correspondant à cet algorithme. Une fois que la métrique a été calculée pour chacun des trois algorithmes, *la plus petite distance parmi ces trois* indique lequel des trois algorithmes est le plus proche de la façon naturelle d'éliciter de l'utilisateur.

L'objectif principal de la deuxième partie de l'expérimentation ainsi que du post-test était d'obtenir du feedback de la part des utilisateurs sur chacun des trois algorithmes. En particulier, les réponses aux questions du post-test servent à définir des critères *subjectifs* car basés sur l'opinion de l'utilisateur, contrastant ainsi avec les critères objectifs décrits dans le paragraphe précédent. Néanmoins les données collectées pendant la deuxième partie nous apportent aussi des critères objectifs pour comparer les algorithmes, comme les considérations de temps ou compter combien de fois l'utilisateur a fait des erreurs et de quel type elles ont été. Pour chacun des trois algorithmes, les critères suivants ont été considéré :

- Critère de *temps* : Combien de temps est requis avec cet algorithme pour éliciter le graphe ?

On obtient cette information en regardant le fichier de log.

- Critère de *compréhension* : Est-ce que l'utilisateur a compris comment fonctionne l'algorithme ?
On demande à l'utilisateur de dire si oui ou non il pense avoir compris comment fonctionne l'algorithme (question de type « oui/non »).

- Critère d'*intuitivité* : Est-ce que l'algorithme semble naturel et intuitif à l'utilisateur ?
L'utilisateur doit donner une note entre 1 et 10 reflétant combien l'algorithme lui semble naturel : 1 pour « totalement contre-nature» et 10 pour « totalement naturel ». En plus, l'utilisateur doit préciser entre les trois algorithmes lequel lui semble le plus naturel. La raison de cela étant qu'en le forçant ainsi à faire un choix, on peut départager sans ambiguïté les algorithmes même si plusieurs ont reçu la même meilleure note.

- Critère de *facilité d'utilisation* : Est-ce que l'algorithme est facile à utiliser ?
L'utilisateur doit donner une note entre 1 et 10 reflétant combien l'algorithme lui semble facile à utiliser : 1 pour « très dur » et 10 pour « très facile ».

- Critère de *concentration* : Quel effort de concentration requiert l'utilisation de l'algorithme ?
L'utilisateur doit donner une note entre 1 et 10 reflétant l'effort de concentration qu'il pense que l'algorithme requiert à utiliser : 1 pour « beaucoup de concentration » et 10 pour « très peu de concentration ». Par exemple un algorithme pourrait demander à l'utilisateur de garder en permanence en tête une large portion du graphe, ce qui dans ce cas lui demanderait beaucoup de concentration. À noter qu'il s'agit sûrement du plus subjectif de tous les critères subjectifs, car par exemple la valeur de « très peu de concentration » varie sûrement beaucoup d'un individu à un autre.

- Critère de *taux d'erreurs* : Combien de fois est-ce que l'utilisateur a commis une erreur?
Ce critère peut facilement être mesuré en regardant le fichier de log et en comptant le nombre de fois que l'utilisateur *a essayé de créer une relation qui n'existe pas* dans le

graphe qui lui a été donné sur papier, ou encore le nombre de fois où *il a oublié de créer une relation qui aurait dû exister*.

6.4.1.3. Résultats

En moyenne, une session d'évaluation avec un volontaire durait entre 45 minutes et une heure. Approximativement, la moitié du temps était utilisé pour l'élicitation des graphes; le reste du temps étant divisé entre les explications de la personne qui supervisait l'expérience, les discussions entre le participant et ledit superviseur, et finalement répondre aux questions du post-test. La Figure 10 illustre comment le temps passé a éliciter est réparti entre les trois parties de l'expérimentation, ainsi qu'entre les trois algorithmes à l'intérieur de la deuxième partie.

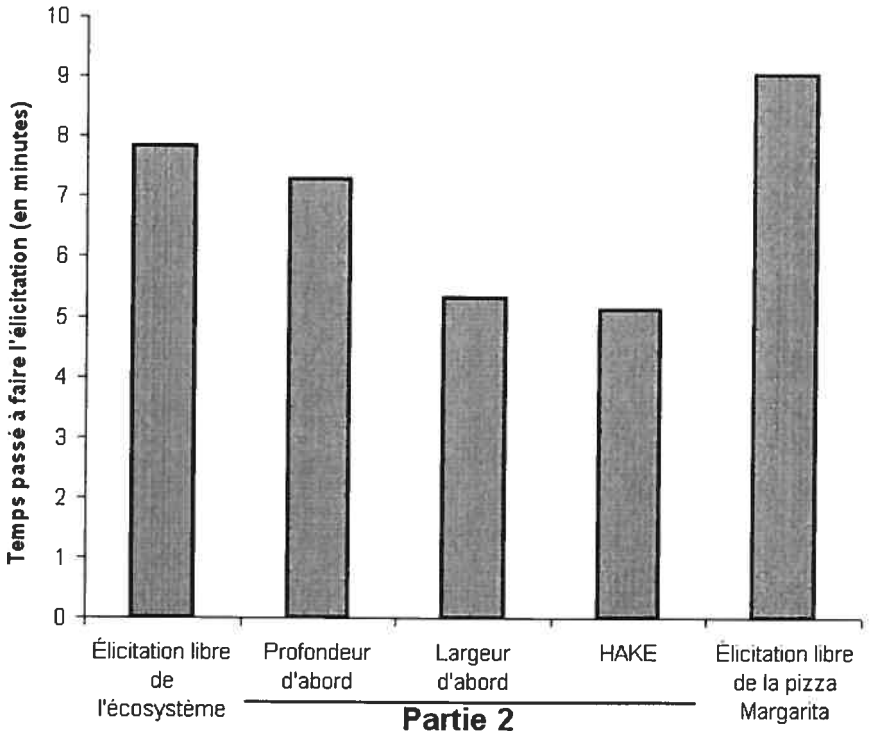

Figure 10 : Répartition du temps d'élicitation.

Au total, les participants ont passé 34 minutes et 48 secondes à éliciter des graphes durant une session. Les élicitations libres, soit la première et troisième partie, sont les parties qui ont exigées le plus de temps. Ceci est vérifié à la fois lorsque le graphe est donné explicitement à l'utilisateur (graphe de l'écosystème), et lorsqu'il ne l'est pas (graphe de la pizza Margarita). En détail, un utilisateur passait en moyenne 7 minutes et 50 secondes à éliciter librement le graphe de l'écosystème, alors que lorsqu'il était guidé par un algorithme il passait pour le même graphe 7 minutes et 17 secondes avec profondeur d'abord, 5 minutes et 30 secondes avec largeur d'abord, et finalement 5 minutes et 10 secondes avec HAKE. Être guidé par un algorithme durant le processus d'élicitation paraît donc être bénéfique à l'utilisateur, en particulier comparé à une élicitation libre. La raison principale est qu'un algorithme d'élicitation fournit à un auteur un moyen de construire le graphe de façon méthodique, lui permettant ainsi de gagner du temps. De plus, parmi les trois algorithmes, HAKE semble être celui qui permet à l'utilisateur de créer son graphe le plus rapidement, bien que largeur d'abord ne soit pas très loin derrière lui. Il faut cependant admettre que le fait que les utilisateurs ont été les plus rapides en utilisant HAKE pourrait venir d'une raison autre qu'uniquement son propre mérite. En effet, il est possible qu'il y ait eut un « effet d'apprentissage », et que les utilisateurs soient devenus de plus en plus familiers avec le système et le graphe de l'écosystème à chaque nouvelle élicitation. HAKE venant alors en dernier dans l'ordre d'utilisation des algorithmes, c'est à ce moment-là que l'utilisateur serait le plus entraîné, et donc le plus à même d'utiliser rapidement le système. À l'inverse des deux autres algorithmes, l'utilisation de profondeur d'abord n'a pas permis de gagner beaucoup de temps comparé à une élicitation libre. Bien que les deux graphes utilisés soient comparables en termes de nombre d'arêtes, la deuxième élicitation libre a requis plus de temps que la première. Compte tenu du fait que le graphe n'était pas fourni durant la deuxième élicitation, rendant ainsi celle-ci plus difficile et aussi plus réaliste, ce résultat est parfaitement explicable. Il faut aussi préciser que, bien que le nombre d'arêtes dans les deux graphes était semblable, ce n'était pas le cas pour le nombre de nœuds qui était plus important dans le cas du graphe de la pizza Margarita.

À la question de la compréhension, tous les participants ont affirmé avoir compris le fonctionnement de chacun des trois algorithmes. Compte tenu que le superviseur faisait toujours un briefing expliquant le fonctionnement d'un algorithme avant son utilisation, ce résultat n'est guère surprenant. De plus, le superviseur restait toujours à côté d'un participant durant l'élicitation au cas où celui-ci ressentirait le besoin de poser une question. Quant à la question de savoir ce qu'est un graphe dans un contexte général, tous les participants sauf un ont répondu positivement.

Comme expliqué dans la Section 6.4.1.2., il est possible de mesurer la distance entre une séquence d'élicitation qui vient naturellement à un utilisateur durant une élicitation libre et n'importe quelle autre séquence d'élicitation. Plus précisément, on va chercher à calculer le nombre minimum d'échanges de deux éléments nécessaires pour transformer une séquence naturelle d'un utilisateur en une autre séquence qui correspondrait à avoir été guidé par un algorithme. Cette distance a été calculée à la fois pour chacun des deux graphes élicités, et pour chacun des trois algorithmes d'élicitation. Les résultats de ces calculs apparaissent dans le Tableau 1. Parce que le graphe n'était pas donné explicitement dans le cas de la pizza, chaque utilisateur a en général produit un graphe différent qui reflète sa propre interprétation culinaire de la recette de la pizza Margarita. En moyenne, un graphe de pizza comportait seulement 18.05 arêtes, soit environ trois de moins que le graphe de référence imaginé au départ. À noter que lors du calcul des distances, la séquence de l'utilisateur est comparée avec les séquences guidées qui auraient produit le même graphe que l'utilisateur, et non pas celles qui auraient produit le graphe de référence.

Tableau 1 : Distance moyenne séparant la séquence d'élicitation d'un utilisateur lors d'une élicitation libre avec les séquences correspondantes des trois algorithmes d'élicitation.

	Profondeur d'abord	Largeur d'abord	HAKE
Graphe de l'écosystème (graphe donné explicitement sur papier)	7.56	6.07	5.91
Graphe de la pizza Margarita (graphe *non* donné explicitement)	6.15	4.89	4.52

Dans les deux cas, HAKE semble être l'algorithme le plus proche de la façon naturelle d'éliciter d'un utilisateur, bien que largeur d'abord ne soit pas très loin derrière non plus. Profondeur d'abord par contre semble être laissé loin en arrière par ses deux concurrents.

Le Tableau 2 contient les données collectées auprès des participants durant le post-test, quand on leur a posé des questions sur l'intuitivité, la facilité d'utilisation ainsi que l'effort de concentration requis par chaque algorithme. La première ligne indique le nombre de personnes ayant voté pour chaque algorithme comme étant le plus naturel (une personne n'ayant pas voté). Les trois lignes suivantes représentent le score moyen pour chacun de ces trois critères sur une échelle de 1 à 10. Il faut se souvenir que plus les scores sont hauts, plus cela est positif pour l'algorithme, un score de 10 représentant la perfection aux yeux de l'utilisateur.

Tableau 2 : Notes pour les critères subjectifs.

	Profondeur d'abord	Largeur d'abord	HAKE
Intuitivité (nombre)	7	24	13
Intuitivité (score)	5.82	7.60	6.91
Facilité d'utilisation	6.38	7.27	7.11
Concentration	5.38	5.91	5.83

Comme précédemment, profondeur d'abord semble être le plus mauvais choix du fait qu'il se retrouve avec les plus mauvaises notes. Cette fois cependant, largeur d'abord et HAKE ont échangés leurs positions et c'est largeur d'abord qui reçoit les meilleures notes pour les critères subjectifs. Pour ce qui est de la facilité d'utilisation et de l'effort de concentration requis, les deux algorithmes se talonnent de près avec tout de même un léger avantage à chaque fois pour largeur d'abord. Par contre en ce qui concerne l'intuitivité, largeur d'abord se place devant par un écart relativement important. Il est possible cependant que cet avantage soit une conséquence du fait que les utilisateurs soient plus familiers avec profondeur d'abord et largeur d'abord, alors que HAKE est pour eux nouveau et peu familier.

Afin de compléter et pousser un peu plus loin les résultats énoncés précédemment, deux comparaisons additionnelles ont été faites. Elles cherchent à répondre aux questions qui suivent. Est-ce l'utilisateur a suivi la même stratégie pour l'élicitation libre des deux graphes? Est-ce que l'utilisateur s'est montré cohérent entre la note donnée aux algorithmes pour leur intuitivité et la façon dont il élicite naturellement quand il est libre?

La réponse à la première question est positive pour 48% des personnes, ce qui signifie qu'approximativement la moitié des participants n'ont donc pas changé leur manière d'éliciter entre le début et la fin de l'expérimentation. Deux raisons pourraient expliquer pourquoi la moitié restante de la population a changé sa façon d'éliciter d'un graphe à l'autre. Une explication plausible serait que le fait d'avoir ou non le graphe en face de lui, change la façon dont l'utilisateur élicite. Par exemple, lorsque le graphe est donné explicitement à l'utilisateur, la disposition des noeuds et des arêtes pourrait influencer la manière d'éliciter de l'utilisateur. Une autre explication sensée serait que les utilisateurs ayant fait connaissance avec les trois algorithmes au moment où ils atteignent l'élicitation libre du second graphe, cela aussi aurait pu influencer leur façon d'éliciter comparativement à la première élicitation où ils n'avaient pas encore été mis en contact direct avec les algorithmes. À l'extrême, on pourrait même imaginer que simplement être au courant de l'existence des algorithmes pourrait modifier la manière naturelle d'éliciter

d'une personne car cela l'amènerait à méditer sur des questions qu'elle ne s'était encore jamais posées auparavant.

Ce qui est plus surprenant, c'est que seulement 30% des participants ont été cohérents entre la note donnée à l'intuitivité des algorithmes et la façon dont ils ont naturellement élicité librement les deux graphes. La comparaison peut aussi être faite de manière plus restreinte en opposant seulement le style d'élicitation de la première élicitation libre à la note donnée, ou encore seulement le style de la deuxième élicitation libre à la note donnée. Dans ce cas, on observe que 50% des personnes ont été « cohérentes » dans le premier cas contre 43% dans le deuxième cas. Il semble donc qu'il ne soit pas vrai que chaque personne a la capacité de reconnaître quelle est sa propre façon d'éliciter.

Normalement on devrait aussi pouvoir observer des corrélations entre le temps nécessaire pour construite un graphe avec un certain algorithme et à quel point il semble facile à utiliser pour un utilisateur. Intuitivement, plus l'utilisateur ressent un algorithme comme facile à utiliser, plus l'algorithme en question devrait lui permettre d'éliciter un graphe rapidement. En pratique, cette corrélation a été vérifiée pour 71% des participants.

Durant un élicitation guidée, il était possible pour un utilisateur de faire des erreurs, soit en créant une relation qui en réalité n'existe pas, soit en oubliant de créer une relation qui aurait dû exister. Ces deux types d'erreurs ont été enregistrés. Le Tableau 3 montre le nombre moyen d'erreurs survenant avec chaque algorithme. Il est important de préciser que le logiciel de l'expérimentation comportait un bouton « undo », et que si l'utilisateur pressait ce bouton immédiatement après avoir fait une erreur, celle-ci n'était pas comptabilisée. Seule les erreurs suivies d'une action autre qu'undo (comme la création d'une autre arête) ont été prises en compte. Une fois que l'utilisateur a commis une telle erreur, on le lui notifie et il n'a pas d'autre choix que d'utiliser le bouton undo pour revenir en arrière et réparer l'erreur.

Tableau 3 : Nombres d'erreurs survenant durant l'élicitation.

	Prodondeur d'abord	Largeur d'abord	HAKE
Création d'une arête incorrecte	0.07	0.09	0.00
Manquer la création d'une arête valide	0.87	0.51	0.40

Les erreurs du premier type ont été observées assez rarement mais celles du second type furent relativement fréquentes. Profondeur d'abord avait le taux d'échec le plus grand avec 0.87 arêtes manquantes en moyenne, contre respectivement 0.51 pour largeur d'abord et 0.40 pour HAKE. La mauvaise performance de profondeur d'abord peut s'expliquer de la manière suivante. Avec largeur d'abord et HAKE, les utilisateurs précisent en une seule fois tous les enfants d'un nœud; on ne leur demande jamais de revenir plus tard à ce nœud. Pour profondeur d'abord c'est très différent, en effet l'utilisateur se fait « accrocher » par un enfant à chaque fois qu'une arête est créée. Cela force l'utilisateur à prendre son mal en patience, et à attendre avant qu'il ne puisse définir une autre arête partant du nœud parent. Au moment où il reviendra à ce nœud parent, il est possible qu'il se trompe et croit avoir déjà créé toutes les arêtes partant de ce nœud.

En lisant les commentaires des participants, on peut observer assez souvent des contradictions entre ce qu'ils affirment sur la vitesse des algorithmes et leur façon réelle d'éliciter. Par exemple, certaines personnes ont affirmé qu'un des avantages de profondeur d'abord est de permettre une élicitation de manière rapide, alors qu'en fait aucun algorithme n'a été plus lent en pratique. Ceci corrobore l'observation faite précédemment que les gens ne sont pas toujours cohérents entre leur façon naturelle d'éliciter, et ce qu'ils ressentent ou pensent des algorithmes qu'ils utilisent.

Afin de pouvoir critiquer la validité de l'expérimentation elle-même, les participants furent aussi interrogés sur leur compréhension du but de l'évaluation et sur la clarté des questions du post-test. Des 45 participants, 43 ont répondu positivement à la première question et 40 à la deuxième.

6.4.1.4. Bilan et limites de l'expérience préliminaire

Le bilan de la première expérimentation est ambivalent. Si on regarde les critères objectifs, HAKE semble être premier, mais il est talonné de très près par largeur d'abord. Si au contraire, on observe le résultat des critères subjectifs, on observe maintenant que la situation est inversée et que c'est maintenant HAKE qui est proche second derrière largeur d'abord (sauf pour l'intuitivité). La seule observation nette qu'on puisse faire, est que profondeur d'abord se retrouve bon dernier qu'on cherche à regarder les choses sous un angle objectif ou subjectif.

En dehors de ce résultat mitigé, il faut aussi admettre que l'expérience préliminaire a ses limites et ses défauts propres. Voici 5 points importants sur lesquels on pourrait critiquer l'expérience préliminaire :

Critique #1 : 45 participants ne constituent pas forcément une population d'une taille statistiquement suffisante pour pouvoir tirer des conclusions.

Critique #2 : Donner explicitement le graphe durant la première et deuxième partie de l'expérimentation ne reflète pas la situation réelle d'élicitation, durant laquelle un expert aurait seulement le graphe de ses connaissances dans son cerveau et pas sur papier.

Critique #3 : Les graphes utilisés lors de la première expérimentation peuvent aussi être considérés comme peu représentatifs car de taille trop restreinte avec une vingtaine de nœuds et aussi ayant une structure très proche de celle d'un arbre.

Critique #4 : L'ordre fixe de présentation des algorithmes durant l'élicitation guidée a pu générer un « effet d'apprentissage » ayant favorisé le dernier algorithme utilisé, soit HAKE dans notre cas.

Critique #5 : La pertinence de la métrique de distance utilisée pour comparer deux séquences d'élicitation peut elle aussi être discutée, car pouvant sembler trop arbitraire.

Afin de répondre et de tenter de remédier à ces critiques, une nouvelle expérimentation a été mise au point.

6.4.2. Seconde expérimentation

6.4.2.1. Description

Une autre expérimentation a donc été conçue et réalisée afin d'essayer de répondre aux critiques et limites de la première expérimentation et d'affiner les résultats. Voici les réponses apportées aux 5 critiques énoncées dans la section précédente :

Réponse à la critique #1 : La taille de la population a été augmentée et est passée de 45 volontaires dans l'expérience préliminaire à 142 dans la nouvelle expérimentation.

Réponse à la critique #2 : Les graphes élicités ne sont *jamais* donnés explicitement aux participants conduisant ainsi à une situation d'élicitation plus réaliste.

Réponse à la critique #3 : Les graphes sont plus profonds et aussi plus importants en terme d'arêtes que précédemment, de plus leur structure est maintenant relativement éloignée de celle de simples arbres.

Réponse à la critique #4 : L'ordre de présentation des trois algorithmes durant la partie élicitation guidée est généré de manière aléatoire pour chaque participant, offrant ainsi un moyen pour contourner le problème de « l'effet d'apprentissage ».

Réponse à la critique #5 : Deux nouvelles métriques de distance ont été définies qui semblent plus pertinentes.

La nouvelle expérimentation se déroule de manière relativement différente de l'expérience préliminaire. Le nombre de parties est passé de trois à deux et l'utilisation d'un questionnaire sous forme de post-test a été abandonnée. Abandonner le

questionnaire de post-test peut sembler *a priori* une mauvaise idée, car cela revient aussi à mettre les critères subjectifs aux oubliettes, mais cela apporte aussi deux avantages. Premièrement, utiliser uniquement des critères objectifs augmente la validité des résultats et des observations qui seront faites, en laissant moins de place à la controverse du fait que ces critères ne s'appuient pas sur une opinion. Deuxième avantage, le temps qui serait normalement pris à répondre à des questions peut être utilisé pour passer plus de temps sur l'élicitation elle-même. Comme les graphes à éliciter pour cette nouvelle expérience sont de taille plus importante, on va pouvoir utiliser le temps gagné pour permettre aux participants d'éliciter plus en profondeur et plus longtemps les graphes. L'idéal aurait été de pouvoir à la fois avoir plus de temps pour l'élicitation des graphes et en même temps de maintenir le questionnaire pour les critères subjectifs, mais il est impossible de demander raisonnablement à un volontaire de passer plus d'une heure à faire l'expérimentation, sans quoi après cette période il perd sa motivation et sa concentration.

L'expérimentation se compose donc maintenant de seulement deux parties. La première partie est une élicitation libre durant laquelle on demande au participant d'éliciter un graphe portant sur la géographie (59 nœuds) (cf. Figure 11) pendant une vingtaine de minutes environ. Durant la deuxième partie, on requiert de l'utilisateur qu'il élicite un graphe portant sur les transports (36 nœuds) tour à tour avec chacun des trois algorithmes d'élicitation. L'ordre d'utilisation des trois algorithmes est généré aléatoirement et chaque séance d'élicitation guidée avec un algorithme est fixé à un temps limite de 10 minutes.

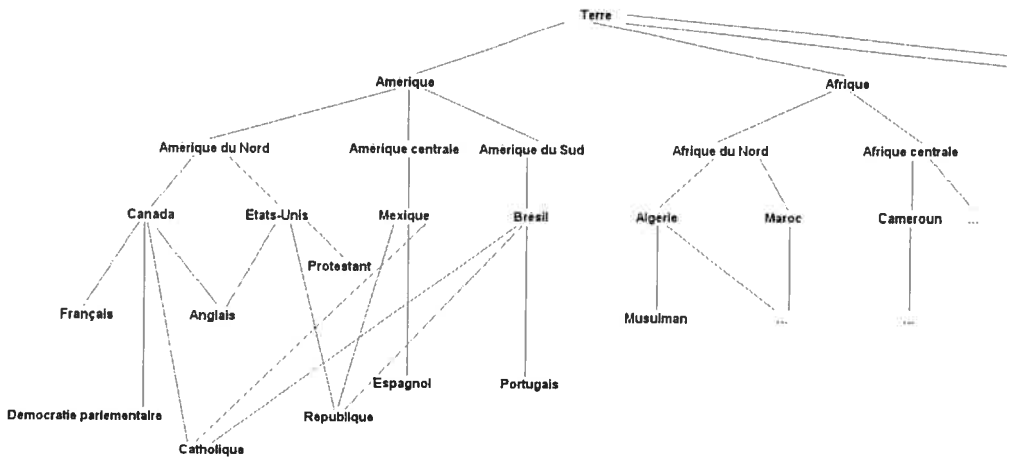

Figure 11 : Fragment d'un graphe possible pour la géographie.

En dehors des modifications décrites, le cadre de la nouvelle expérience est relativement identique à celui de la première. À noter toutefois qu'on n'attend pas des participants qu'ils construisent le graphe en entier à chaque fois vu la taille plus importante des graphes, mais plutôt qu'ils essayent de créer un maximum d'arêtes durant le temps imparti.

6.4.2.2. Nouveaux critères

L'ancienne métrique se basant sur le nombre minimum d'échange de deux éléments nécessaires pour passer d'une séquence d'élicitation à une autre a été abandonnée au profit de la définition de deux nouvelles métriques qui semblent plus pertinentes.

La première nouvelle métrique s'appelle la *distance de transposition*. Bien qu'elle possède un sens différent en mathématiques, en bioinformatique elle se définit par le nombre minimum de déplacements de blocs requis pour transformer une séquence en une autre. C'est une métrique qui peut être utilisée en bioinformatique pour mesurer à quel point deux séquences d'ADN sont proches l'une de l'autre. En effet une mutation pouvant conduire à un déplacement d'un fragment de la chaîne d'ADN si on arrive à trouver le nombre minimum de déplacements nécessaires pour passer d'une chaîne à une autre, on peut imaginer qu'on aura mesuré plus ou moins le nombre de mutations

séparant les deux chaînes. On va utiliser l'analogie entre une séquence d'ADN et une séquence d'élicitation pour essayer d'appliquer la même métrique dans notre cas. Le but maintenant n'est donc plus de mesurer la proximité génétique de deux génomes mais plutôt de mesurer à quel point deux manières d'éliciter (représentées par deux séquences d'élicitation) sont proches. À noter que le problème de la distance de transposition est conjecturé comme étant NP-Complet [Walter et *al* 98] mais que des algorithmes d'approximation 1.5 existent [Bafna et Pevzner 98, Christie 99]. Un algorithme d'approximation 1.5 garantit que le résultat retourné par l'algorithme est au pire cas différent de la solution optimale par un facteur de 1.5. Cette fois-ci, contrairement à la première expérience, pour un algorithme fixé on ne va pas chercher à calculer la distance de transposition entre toutes les séquences d'élicitation pouvant être générées à partir du graphe crée par l'utilisateur et sa propre séquence d'élicitation, mais entre cette dernière et la séquence découlant naturellement de l'ordre dans lequel il a rentré les arêtes.

La deuxième métrique est une *métrique de pénalité*. En ayant en tête le graphe final créé par un utilisateur, il est possible de mesurer à quel point la séquence d'élicitation d'un utilisateur respecte le comportement d'un algorithme d'élicitation. À chaque fois que le comportement d'un algorithme n'est pas respecté, un point de pénalité est ajouté au score de cet algorithme. Quand le calcul a été fait pour chacun des trois algorithmes, on peut dire que c'est l'algorithme dont le score de pénalité est le plus bas qui est le plus proche de la façon naturelle d'éliciter de l'auteur.

L'information contenue dans le log file permet aussi de déterminer facilement quel est la fréquence de création d'arêtes durant l'élicitation libre et avec chacun des algorithmes d'élicitation, ainsi que le taux des undo.

6.4.2.3. Résultats

La nouvelle expérimentation a porté sur 142 volontaires ayant passé en moyenne 51 minutes et 17 secondes à éliciter les graphes. En détails, pour l'élicitation libre du graphe de géographie, les participants ont passé en moyenne 21 minutes et 25 secondes dessus,

contre 10 minutes environ sur le graphe des véhicules avec chacun des trois algorithmes d'élicitation.

Le Tableau 4 regroupe les résultats de mesure des taux de création d'arêtes et d'utilisation du bouton undo pour l'élicitation libre, ainsi que pour l'élicitation guidée avec chacun des trois algorithmes.

En ce qui concerne le taux de création des arêtes, on observe que largeur d'abord semble être l'algorithme qui permet de rendre le processus d'élicitation le plus rapide bien que HAKE ne soit pas très loin derrière. On peut observer que même profondeur d'abord, qui a le plus mauvais score parmi les trois algorithmes d'élicitation, permet néanmoins de créer des arêtes plus rapidement que l'élicitation libre. Il semble donc que guider l'élicitation avec un algorithme puisse accélérer le processus, au moins en terme de création d'arêtes, comparé à une élicitation libre. Cette « conclusion » doit toutefois être considérée avec circonspection car la différence pourrait également venir du fait que les graphes étaient différents.

En ce qui concerne la fréquence des undo, on constate d'abord que c'est lors de l'élicitation libre que l'utilisateur est le moins tenté de faire un undo, ce qui doit être considéré avec la même circonspection. Si on regarde seulement les algorithmes d'élicitation, HAKE et largeur d'abord sont toujours en train de se chamailler pour la première place alors que profondeur d'abord est l'algorithme qui semble être le plus à même à amener l'utilisateur à vouloir corriger les arêtes qu'il a créées auparavant.

Tableau 4 : Taux de création des arêtes et d'utilisation de undo pour l'élicitation libre et les trois algorithmes d'élicitation.

	Taux de création des arêtes (arêtes/minute)	Taux d'utilisation d'undo (undo/minute)
Élicitation libre	2.02	0.23
Profondeur d'abord	2.39	1.01
Largeur d'abord	2.98	0.71
HAKE	2.80	0.73

Le résultat des nouvelles métriques est sûrement le plus intéressant à regarder. On pourra trouver le résultat des calculs pour ces métriques résumé dans le Tableau 5.

Tableau 5 : Résultats des nouvelles métriques.

	Métrique de pénalité	Distance de transposition
Profondeur d'abord	38.03	11.80
Largeur d'abord	18.71	12.65
HAKE	27.78	11.53

La métrique de pénalité semble être celle d'entre toutes les métriques qui permet le mieux de départager et séparer les trois algorithmes d'élicitation. Il faut se souvenir que dans le cas de la métrique de pénalité, contrairement aux autres métriques, plus le score est bas, plus cela est positif pour l'algorithme. Dans notre cas, largeur d'abord arrive largement en tête avec une pénalité moyenne de 18.71, comparativement à HAKE qui reçoit la médaille d'argent avec 27.78 et profondeur d'abord qui obtient la médaille de bronze avec 38.03 de pénalité moyenne. En se basant sur cette métrique, il semblerait donc que largeur d'abord soit la façon la plus naturelle d'éliciter.

Le classement issu de la distance de transposition est beaucoup plus serré. En effet, les trois distances sont très proches les unes des autres, trop proches même si on se rappelle que l'algorithme utilisé pour calculer ces distances est un algorithme d'approximation 1.5. Il semble donc dangereux de vouloir en conclure qu'un algorithme est meilleur qu'un autre. Au contraire, notre étude semble indiquer que le choix de l'algorithme d'élicitation n'a pas une influence substantielle sur ce critère.

6.4.3. Conclusion de l'évaluation

Des deux expérimentations qui furent menées dans le but de trouver lequel parmi trois algorithmes d'élicitation est le plus proche de la façon naturelle d'éliciter d'un humain, aucune n'a permis de répondre avec certitude à cette question.

Si on regarde la première expérimentation qui utilisait à la fois des critères subjectifs et objectifs, il semble que HAKE et largeur d'abord sont côte à côte pour la première place.

En détails, HAKE dispose d'un léger avantage dans le cadre des critères objectifs, et largeur d'abord dans celui des critères subjectifs. Il est toutefois très difficile de les départager.

La deuxième expérience tentait d'apprendre des défauts de la première expérience et avait pour mission de permettre d'affiner les résultats. Elle prenait aussi le pari de miser uniquement sur des critères objectifs, dont deux nouvelles métriques qui semblent plus pertinentes à la situation réelle d'une élicitation. Malheureusement, là encore, les deux algorithmes restent très proches les uns des autres, sauf dans le cas de la métrique de pénalité où largeur d'abord se détache très nettement de HAKE.

Par contre ce qui ressort clairement des deux expérimentations, c'est que profondeur d'abord semble être une très mauvaise alternative en tant qu'algorithme d'élicitation. Un autre résultat, bien que modeste, est que bénéficier de l'aide d'un algorithme d'élicitation semble accélérer le processus d'acquisition des connaissances comparé à une élicitation libre où l'auteur est livré à lui-même.

Ces deux expériences n'ont comparé ensemble que trois algorithmes d'élicitation de manière globale, et il est plausible que le « meilleur » algorithme d'élicitation soit un autre algorithme non décrit ici. Probablement même qu'il n'existe pas un algorithme d'élicitation universel, mais plutôt un algorithme spécifique pour chaque personne. Une solution serait avant de commencer l'acquisition des connaissances auprès d'un expert, de lui faire passer un test afin de déterminer dans son cas, lequel parmi les algorithmes disponibles est le plus proche de sa façon naturelle d'éliciter. On pourrait même aller jusqu'à construire un algorithme d'élicitation *ad hoc* pour cet expert en observant comment il élicite de manière libre sur quelques graphes et en utilisant ensuite des techniques issues de l'apprentissage non-supervisé sur les données collectées. Évidemment, implémenter une telle approche adaptive serait loin d'être trivial.

Chapitre 7 : Initialisation du modèle de l'apprenant dans QUANTI

Dans un système tutoriel intelligent, le modèle de l'apprenant reflète l'état des connaissances de l'apprenant et réalise des inférences à propos de ses lacunes. Mais comment *initialiser* ce modèle de l'apprenant ou, dit autrement, que doit-on supposer sur les connaissances préalables d'un nouvel étudiant ? Cette question est fondamentale car la qualité de l'enseignement donné par le STI dépend fortement de la pertinence des informations présentes dans le modèle de l'apprenant. En effet si les informations initiales présentes dans le modèle de l'apprenant sont fausses, le système risque d'enseigner à l'étudiant des concepts qu'en fait il maîtrise déjà, ou inversement, ne pas enseigner les concepts qui devraient l'être. Le travail décrit dans ce chapitre propose une méthode basée sur le concept de catégorisation pour répondre à cette question. Par catégorisation, on veut dire qu'on va chercher à regrouper des étudiants par catégories ayant chacune des valeurs d'initialisation bien précises. Pour découvrir les catégories, on va utiliser un algorithme d'apprentissage machine du nom de CLARISSE développé par Hugo Dufort dans le cadre de sa maîtrise [Dufort 99]. Une bonne partie du travail décrit dans ce chapitre pourra être trouvée dans l'article [Aïmeur et *al* 02].

7.1. Le modèle de l'apprenant

Le modèle de l'apprenant dans QUANTI se compose de trois sous-modèles : le *modèle cognitif*, le *modèle affectif* et le *modèle inférentiel*. C'est le modèle cognitif qui est en charge de représenter la connaissance de l'étudiant par rapport au domaine, c'est-à-dire ce qu'il sait et ce qu'il ne sait pas, et à quel point. Cette partie est implémentée en utilisant un *modèle par recouvrement* (voir Section 2.3.2.) qui dérive sa structure directement de celle du curriculum. Tout comme le curriculum, le modèle par recouvrement est composé de réseaux sémantiques, chacun étant la réflexion de sa contrepartie dans le curriculum. Il est important de se rappeler que chaque nœud du curriculum correspond à un morceau de connaissance. Le but est de réussir à savoir le plus fidèlement possible quel est le niveau de compréhension de l'apprenant pour chacun de ces morceaux de connaissance. Un pourcentage est associé à chaque nœud qui

représente le niveau de compréhension de l'apprenant : 0% signifiant que l'apprenant ne sait rien (ou qu'on a supposé qu'il ne savait rien) à propos de ce morceau de connaissance, alors que 100% signifie que l'apprenant maîtrise totalement le sujet. L'initialisation de ces valeurs pour un nouvel apprenant est un problème très pertinent, et le thème de ce chapitre est justement de proposer une méthode pour y répondre. Le modèle affectif garde en mémoire le profil affectif de l'apprenant ainsi que son état émotionnel. Enfin le modèle inférentiel réalise des inférences à propos de l'étudiant en tenant compte des informations disponibles à partir du modèle cognitif et du modèle affectif. En retour, ces inférences serviront à modifier et mettre à jour ces deux modèles.

7.2. Initialisation du modèle cognitif

La solution la plus simple pour initialiser le modèle de l'apprenant, reviendrait à supposer que tout nouvel apprenant *ne sait rien* à propos de la matière avant sa première leçon, ou alors qu'il possède des *« connaissances standards »* identiques à tout autre nouvel apprenant. Dans ce cas, le niveau de compréhension de chacun des nœuds est initialisé à zéro (ou une valeur standard fixé à l'avance). Cependant, être un nouvel étudiant ne signifie pas forcément être totalement étranger au domaine enseigné par le STI. Par conséquent, une meilleure approche consisterait à évaluer les connaissances de l'apprenant avec un questionnaire appelé *pré-test*, qui lui serait administré avant le début de sa première leçon.

Afin que l'évaluation soit la plus fidèle possible, la solution idéale requerrait que pour chaque nœud présent dans le modèle cognitif, au moins une question soit posée en rapport avec ce nœud. On nomme cette solution un pré-test *exhaustif*. En pratique, cette méthode est souvent irréalisable car trop contraignante pour l'étudiant. En effet, le nombre de questions posées à l'étudiant durant le pré-test est forcément élevé (sauf pour les domaines de très petite taille) et l'étudiant, qui a hâte de commencer le cours, pourrait devenir désespéré.

Dans un pré-test *intelligent*, les questions se focalisent sur des nœuds importants. Par nœuds importants, on veut désigner ceux qui portent sur des concepts jugés importants, ou encore ceux qui possèdent de nombreuses connections avec d'autres nœuds. Une fois que les valeurs pour ces nœuds importants ont été mesurées, les mécanismes du modèle inférentiel sont activés afin de propager ces valeurs à l'intérieur du réseau. Cela permet de réduire le nombre de questions, mais un compromis doit être fait entre le nombre de questions du pré-test et la précision du modèle. En effet avec un pré-test intelligent, le système doit faire des inférences pour les nœuds sur lesquels il n'a pas pu poser directement de questions à l'apprenant, ce qui réduit d'autant la fiabilité de l'information. Un pré-test *adaptatif* [Arroyo et *al* 01, Millàn et *al* 00], dans lequel le choix de la prochaine question prend en compte les réponses aux questions précédentes, est une forme de pré-test intelligent.

Enfin la *catégorisation* est un autre moyen pour éviter d'enterrer l'étudiant sous une tonne de questions. Chaque étudiant possède ses propres caractéristiques et son comportement. Malgré tout, il est souvent possible d'observer des similitudes et des ressemblances entre certains étudiants et de regrouper les étudiants possédant des caractéristiques similaires à l'intérieur de *catégories*, parfois appelées *stéréotypes* [Kay 00]. Une fois que les catégories ont été découvertes, la seule tâche qu'il reste à faire lorsqu'un nouvel étudiant arrive c'est d'être capable de déterminer à quelle catégorie il appartient. Le nombre de questions requises pour déterminer la catégorie d'un étudiant est en général beaucoup plus petit que celui d'un pré-test intelligent. À chaque catégorie va correspondre une initialisation différente des valeurs des nœuds du modèle de l'apprenant. C'est une approche par catégorisation qui est décrite dans le reste de ce chapitre.

7.3. État de l'art des méthodes de catégorisation

La catégorisation est une forme d'*apprentissage non-supervisé*. On sait qu'on souhaite trouver quelque chose, mais on ne sait pas exactement à l'avance à quoi cela va ressembler. Les catégories ne sont donc pas connues *a priori* : elles sont en fait révélées

par le processus de catégorisation. La catégorisation peut aussi être définie comme la tâche de trouver la structure sous-jacente qui se cache à l'intérieur de données.

Il existe deux familles de méthodes de catégorisation. La première famille, qui regroupe les *méthodes mathématiques et statistiques*, ne fournit aucune explication quant aux catégories qu'elle découvre ou encore aux relations entre les objets. La deuxième famille, sur laquelle on va se concentrer, contient les *méthodes symboliques et conceptuelles* [Bauer et *al* 99, Bloedorn et *al* 97, Raskutti et Beitz 96]. Ces méthodes n'essayent pas seulement de regrouper ensemble des objets qui sont proches mais aussi de trouver en quoi les attributs des objets sont similaires ou différents entre eux. Cela permet de fournir des explications sur les catégories créées. Des règles sont émises afin de séparer les objets en différents groupes. Il est possible d'utiliser ces règles de manière récursive pour séparer les objets. Par exemple, une fois qu'une règle a été utilisée pour séparer en deux catégories une partition d'objets, il est possible de choisir une autre règle pour la création de plus petites catégories à l'intérieur d'un de ces grandes catégories. Parmi les méthodes symboliques et conceptuelles existantes, on peut citer UNIMEM [Lebowitz 87], CobWeb [Fisher 87] et WITT [Hanson 90].

7.4. CLARISSE

Afin d'initialiser avec succès le modèle de l'apprenant, on va utiliser une méthode de catégorisation. Les méthodes de regroupement mathématiques ne sont pas suffisantes en elles-mêmes car il faudra à la fois *trouver les catégories* qui existent au sein d'une population initiale d'étudiants, mais aussi *des règles* qui pourront servir à catégoriser de nouveaux étudiants. À cette fin, une méthode de catégorisation appelée CLARISSE (Cluster And Rules ISSuED) a été utilisée [Dufort 99]. L'algorithme procède récursivement en séparant un ensemble initial d'objets et en construisant un arbre binaire qui sera ensuite utilisé pour identifier les catégories.

CLARISSE a la particularité de pouvoir fonctionner sur n'importe quel type d'entrée; on appellera chaque entrée un *objet*. Un objet est composé de *descripteurs*, qui sont des

attributs possédant une seule valeur. La valeur de chaque descripteur doit être définie, et chaque objet doit être comparable en terme de descripteurs. Chaque descripteur possède un intervalle de valeurs acceptables, aussi appelé domaine. Dans l'application de CLARISSE qui va être faite pour les systèmes tutoriels intelligents, les objets seront des étudiants, les descripteurs des questions (tirés d'un questionnaire), et les domaines seront définis indirectement par le type des questions.

Dans CLARISSE, chaque objet peut être imaginé comme un point dans un espace à N dimensions, où N est le nombre de descripteurs de l'objet. Si chaque descripteur a K valeurs possibles, alors il y a $2^K 2^N$ manières de diviser l'espace des descripteurs à chaque étape de la méthode. Cette formule pouvant bien sûr être adaptée si le nombre de valeurs possibles n'est pas le même pour chaque descripteur. Le but de CLARISSE sera d'essayer de réduire l'espace de recherche aux combinaisons de descripteurs et de valeurs qui sont les plus prometteuses.

Il faut avoir une attention toute particulière lorsqu'on définit les domaines. En particulier, la *distance sémantique* entre les valeurs possibles d'un descripteur doit être définie très précisément. Un domaine peut être défini comme un intervalle fini d'entiers ou de valeurs réelles, ou encore comme un espace discret d'attributs étiquetés, ce qui dans ce dernier cas requiert la définition d'une matrice de distances sémantiques.

Pour orienter le processus de catégorisation, CLARISSE utilise une mesure appelée *utilité d'une catégorie* (ou *category utility* en anglais) [Gluck et Corter 85]. Cette mesure sert à déterminer à quel point une catégorie est bien définie. D'après [Biswas et *al* 95], deux facteurs doivent être pris en compte quand on calcule l'utilité d'une catégorie : en prenant n'importe quel objet qui appartient à une catégorie quelconque, la *dissimilarité* indique à quel point cet objet peut être différent d'un objet qui appartient à une autre catégorie, alors que la *cohérence interne* indique combien cet objet est similaire à tous les autres objets qui sont à l'intérieur de sa catégorie. Le rôle de la cohérence interne est de prévenir l'apparition de catégories « poubelle » qui contiendraient seulement des objets orphelins. CLARISSE dérive la cohérence interne d'une catégorie d'une mesure

d'entropie sur ses descripteurs. Afin de réduire cette entropie, des objets peuvent être échangés entre les catégories, mais il y a un prix à payer pour cette action : la mesure de cohérence interne et la mesure de dissimilarité peuvent tous les deux en pâtir. Comme CLARISSE est récursif et peut utiliser du backtracking, il peut essayer différentes manières de partitionner les regroupements, ce qui peut donner des résultats différents qui sont ensuite classés en utilisant l'utilité de catégorie.

Voici un aperçu de comment CLARISSE fonctionne :

1. Commencer avec un ensemble initial M contenant n objets.

2. Afin d'exécuter une méthode de regroupement mathématique, deux prototypes (ou germes) doivent être choisis parmi les n objets de M. S_1 et S_2 sont choisis comme étant les deux objets les plus éloignés entre eux dans l'espace des descripteurs. Ceci est une méthode parmi plusieurs possibles. Elle prend un temps d'exécution de l'ordre de $O(n^2)$.

3. Appliquer une méthode de regroupement mathématique (agrégation) à S_1 et S_2 en utilisant soit la méthode du centroïde, soit la méthode du voisin le plus éloigné ou soit la méthode du voisin le plus proche, afin d'obtenir les regroupements C_1 et C_2. La complexité de ces méthodes varie de $O(n^2)$ à $O(n^3)$, et peut affecter les propriétés des regroupements ainsi créés.

4. Choisir un descripteur D_0 dont les valeurs peuvent être partitionnées de manière à maximiser la dissimilarité entre C_1 et C_2. Une fois qu'un tel descripteur est trouvé, il reste encore à identifier des descripteurs additionnels D_i qui seront utilisés avec D_0 pour raffiner la partition de l'espace des descripteurs, sans par autant trop réduire la qualité de C_1 et C_2. La conjonction $\{D_0 \wedge \ldots \wedge D_i\}$ de descripteurs est appelée une règle, qu'on notera R.

5. Si certains objets dans C_1 et C_2 sont séparés de leur regroupement parent par l'application de la règle R, alors on essaye d'échanger ces objets entre C_1 et C_2. Comme

dit précédemment, il y a un prix à cet échange, c'est la perte de cohérence interne des regroupements dans lesquels on a rajouté des objets. Ce prix, qu'on appellera *coût de déplacement* d'objet, doit être contrebalancé par un *gain sur la cohérence globale de la partition*. On pondèrera ce prix par un paramètre interne à l'algorithme, appelé *poids d'appartenance*.

6. Après avoir essayé toutes les règles qui semblent les plus prometteuses, garder le meilleur candidat. Échanger les objets entre C_1 et C_2. Stocker les règles candidates qui n'ont pas été utilisées en réserve, au cas où on souhaiterait faire du backtracking et essayer un autre chemin plus tard.

7. Appliquer récursivement toutes ces étapes sur C_1 et C_2. Si une catégorie contient seulement un objet, ou que tous ces objets sont quasiment indistinguables entre eux, cette catégorie est finale.

Les Figures 12 et 13 illustrent sur un exemple jouet la construction de deux regroupements mathématiques (représentés par des ovales en pointillés) autour des germes S_1 et S_2. CLARISSE a trouvé la règle sur Y suivante : tous les objets avec $Y < 5$ sont dans la catégorie C_1 et tous les objets avec $Y \geq 5$ sont dans la catégorie C_2. Trois objets situés aux frontières (représentés par des cercles blancs) sont échangés durant le processus. Les catégories résultantes sont à la fois cohérentes et explicables.

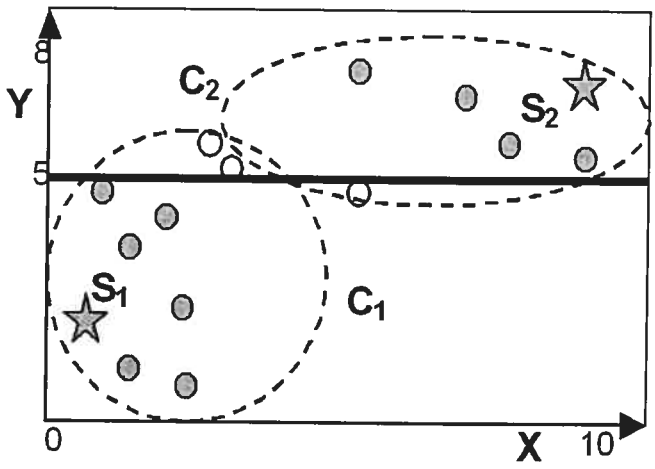

Figure 12 : Exemple de catégorisation; la règle R est que Y < 5 pour C1.

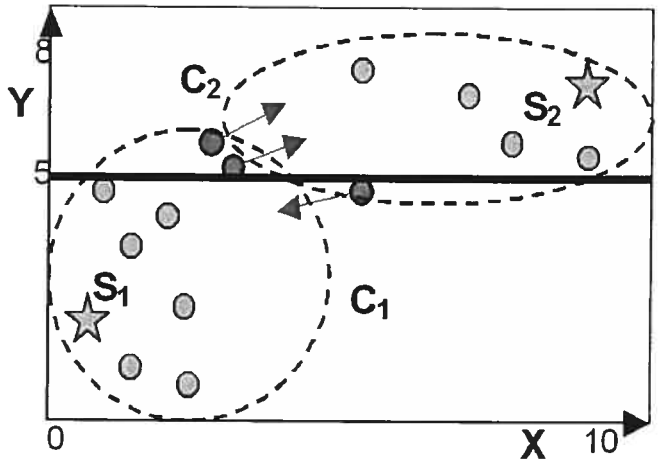

Figure 13 : Échange de 3 objets entre C1 et C2.

7.5. Expérimentation : méthodologie et analyse des résultats

Afin d'identifier les catégories d'étudiants existant en informatique quantique, un échantillon de la population du public cible a été utilisé comme ensemble d'entraînement. Un questionnaire contenant 30 questions à choix multiples (cf. Figure 14), et couvrant des thèmes allant de la logique classique et des matrices à des sujets plus avancés d'informatique quantique, a été construit. Les réponses ont été notées suivant trois valeurs : bonne (10 points), en partie fausse (3 points) et complètement fausse (0 point). Ces valeurs correspondent aux distances dans l'espace des descripteurs utilisé par l'algorithme de catégorisation. Ainsi on considère l'écart entre une réponse en partie fausse et une réponse totalement fausse (3 points) comme moindre que celui entre une réponse bonne et une réponse en partie fausse (7 points). Chaque question est utilisée comme un descripteur durant le processus de catégorisation, et donc dans ce cas précis CLARISSE travaille sur un espace de descripteurs de dimension 30.

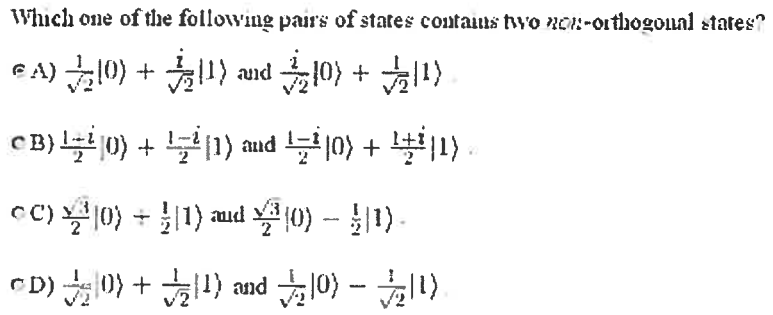

Figure 14 : Une question du questionnaire en ligne.

Des étudiants, des professeurs et des chercheurs (en physique et en informatique) ont participé sur une base volontaire pour répondre au questionnaire disponible en ligne. Sur une période d'une semaine, 31 réponses ont été reçues en provenance de pays aussi différents que l'Australie, le Canada, la France, Israël, le Japon, les Pays-Bas et les Etats-Unis. Ces résultats ont été passés à travers un programme de validation et formattés pour servir de fichier d'entrée à CLARISSE.

Utilisant ses heuristiques propres pour induire des variations sur ses paramètres d'entrée et testant les différentes possibilités résultantes, CLARISSE a trouvé un seul arbre de catégorisation (cf. Figure 15) stable se composant de sept catégories. Ce résultat a permis de remettre en partie en cause deux mythes bien établis.

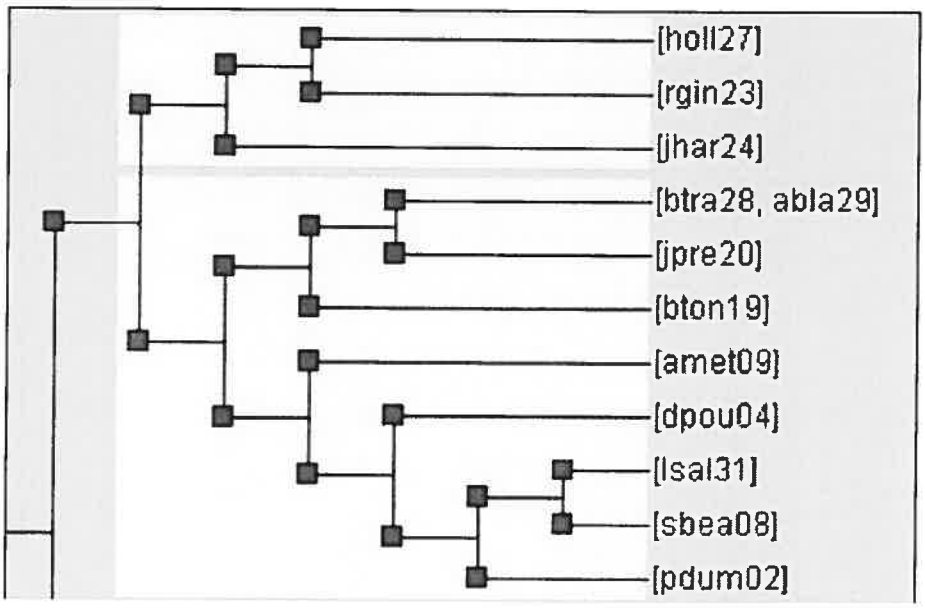

Figure 15 : Un fragment de l'arbre de catégorisation.

Mythe #1 : *Les catégories doivent représenter le background technique des étudiants.*

Aucune corrélation forte n'a été trouvée entre le background technique des étudiants et le résultat de la catégorisation. Ainsi, vouloir créer un profil des étudiants en leur posant simplement des questions sur leur background, comme les cours qu'ils ont suivis l'année précédente, ou encore leurs activités actuelles, n'est pas assez précis pour que le STI soit efficace. L'enseignement doit donc répondre à des besoins pédagogiques, et ces besoins pédagogiques semblent pouvoir varier fortement même entre des étudiants partageant le même background technique ou académique. Ceci reste vrai même si on cherche à tirer parti de l'aspect multidisciplinaire de l'informatique quantique en voulant regrouper les étudiants par sous-domaine et/ou niveau académique.

Mythe #2 : *Les catégories doivent représenter des partitions dans la distribution des scores.*

Seule des corrélations faibles ont été trouvées entre les scores et les catégories : en effet les étudiants les plus faibles ont été regroupés à l'intérieur de deux catégories bien définies, mais le reste des étudiants sont si fortement groupés entre eux que n'importe

quelle méthode *ad hoc* aurait manqué de détecter les sous-groupes (cf. Figure 16 et 17). Par exemple, si on examine les questionnaires des personnes dont le score entre dans l'intervalle 80-90%, on se rend compte que même si l'écart type est faible, les erreurs peuvent être faites sur des sujets complètement différents d'une personne à l'autre. Certaines de ces erreurs sont répétées sur un nombre significatif de questionnaires; ainsi en observant cette tendance, la présence des catégories est révélée.

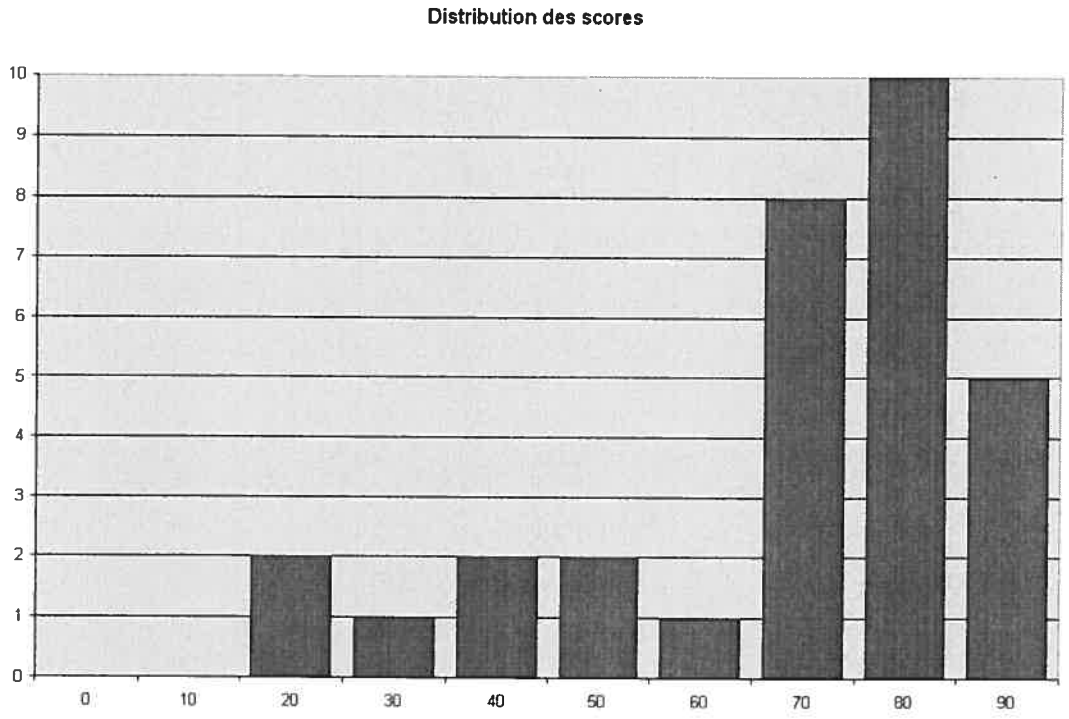

Figure 16 : Distribution des scores bruts.

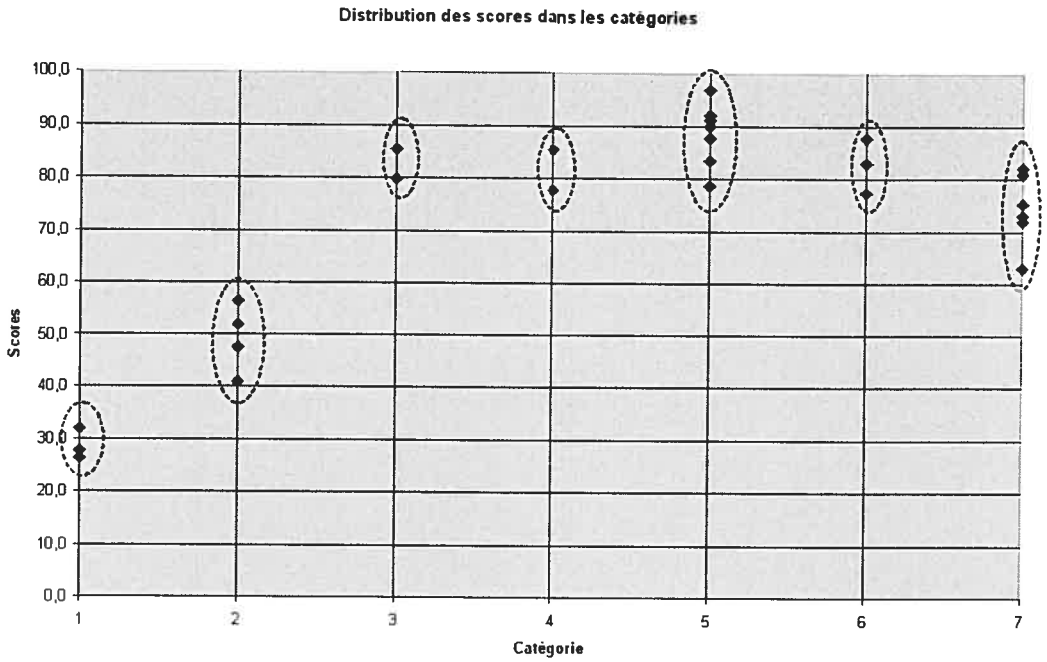

Figure 17 : Distribution des scores dans les catégories. Les catégories 3 à 7 se chevauchent.

Malgré tout, lorsqu'on regarde ces résultats bruts directement à l'œil nu, comme dans le cas de l'enseignement traditionnel, cela ne donne pas assez d'information pour permettre de catégoriser les étudiants. En effet, en général, la plupart des professeurs ont des contraintes de temps qui les empêchent de rechercher les tendances à l'intérieur des résultats des étudiants. En fait, c'est le rôle des algorithmes d'apprentissage machine, tel que CLARISSE, de pouvoir aider efficacement les professeurs dans leurs analyses.

7.6. Fichier de définition de catégorie de CLARISSE

En utilisant une analyse de variation de la cohérence à l'intérieur de l'arbre de catégorisation, CLARISSE a partitionné la population initiale en sept catégories démontrant chacune une forte cohérence.

Chaque catégorie est identifiée par :

-Une *valeur de cohérence interne*. Généralement, une forte cohérence indique une haute densité de la catégorie dans l'espace des descripteurs.

-Sa *liste des membres* à partir de l'ensemble original.

-Une *liste ordonnée de règles* qui oppose cette catégorie aux autres catégories qui ont été découvertes. Ces règles forment un système hiérarchique de règles (appelé *arbre de catégorisation*), qui peut être utilisé pour classifier un nouvel étudiant.
-Une *liste des descripteurs les plus significatifs*. Ces descripteurs seront utilisés pour initialiser le modèle de l'apprenant dans le STI, quand un étudiant tombera dans cette catégorie.

7.7. Classifier des nouveaux étudiants

Le but principal de ce processus est de permettre la classification de nouveaux étudiants qui voudraient suivre le cours d'informatique quantique. Pour chaque catégorie, des règles d'inclusion/exclusion sont définies, qui contribuent ainsi à construire un système hiérarchique de règles. Par exemple, une règle d'inclusion pour une certaine catégorie pourrait spécifier qu'un étudiant ayant mal répondu à une question spécifique soit inclus dans cette catégorie, ou encore, une règle d'exclusion pour cette même catégorie pourrait exclure tout étudiant ayant répondu correctement à cette même question. Des 30 questions du questionnaire initial, 6 se sont révélées significatives pour permettre la classification. Ces 6 questions sont à la base du système de règles permettant de différencier les catégories, et sont nécessaires et suffisantes à la classification.

La Figure 18 montre ce système de règles. Les catégories ont été ordonnées des étudiants ayant obtenant un meilleur score à ceux ayant obtenus le score le plus bas. Chaque réponse incorrecte à une question envoie l'étudiant plus bas dans l'arbre de catégorisation. Pour chaque catégorie, une description des membres typiques de cette catégorie est donnée. Afin de classifier un nouvel étudiant dans l'une des sept catégories, il est suffisant de lui poser des questions sur trois concepts avec un pré-test adaptatif qui serait dérivé de l'arbre de la figure 18. Une fois classifié, l'étudiant reçoit le profil standard de sa catégorie (c'est-à-dire un ensemble de descripteurs) et on peut initialiser son modèle de l'apprenant dans le STI.

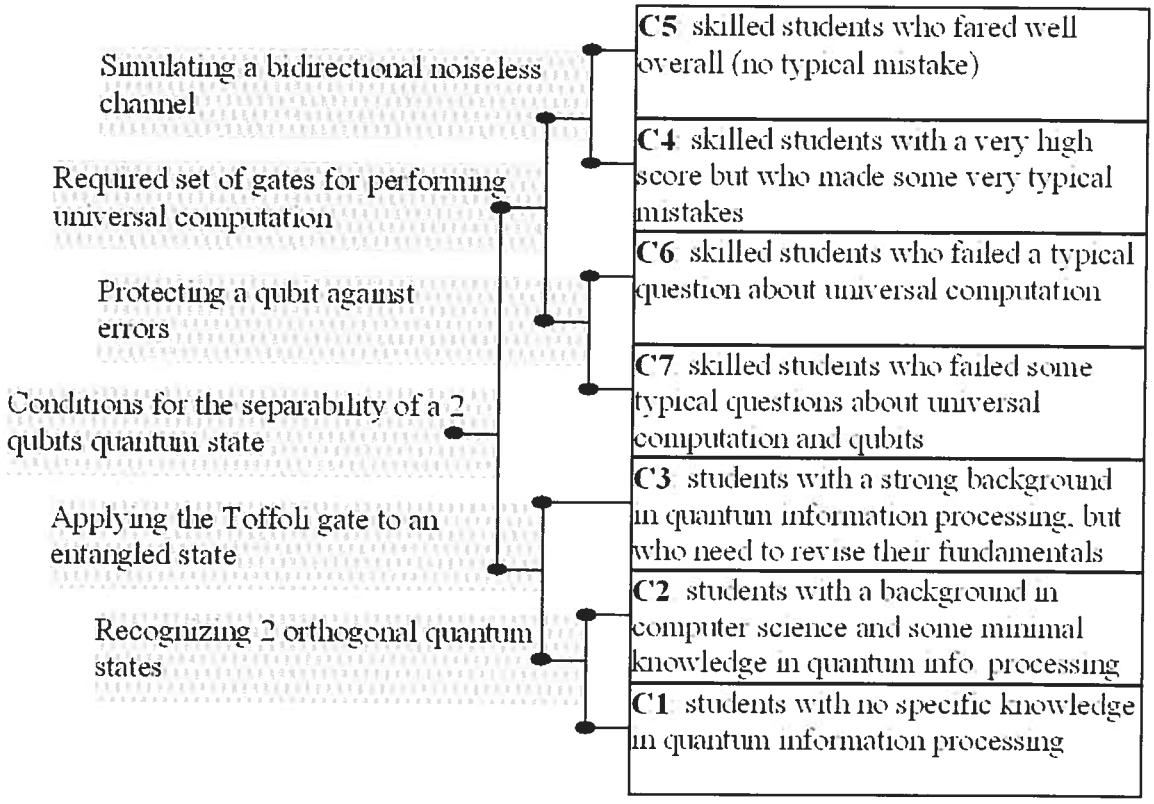

Figure 18 : Arbre de décision utilisé pour classifier de nouveaux étudiants.

Comme quatre questionnaires remplis ont été reçus *après* avoir compilé les résultats, il a été naturel de les utiliser afin de valider la méthode. Ces quatre derniers arrivants ont été classifiés en utilisant l'arbre de catégorisation décrit précédemment et la Table 6 montre les résultats obtenus. Dans deux cas, il y a correspondance parfaite entre les descripteurs des participants et les descripteurs de la catégorie, alors que dans les deux autres cas le succès est moindre. Avec un taux de succès de 61.5%, 26.6% (8/30) des nœuds du troisième participant auraient été mal initialisés dans QUANTI. Ceci reste cependant meilleur que l'absence totale d'initialisation, et les mauvaises valeurs auraient éventuellement fini par être corrigées par le modèle inférentiel avec le temps. Le quatrième participant montre une meilleure correspondance, avec un taux de succès de l'ordre de 88.2%. L'étudiant C de la Table 6 tombe en fait entre la catégorie 1 et la catégorie 2, et aurait sûrement était mieux classifié dans la catégorie 1. Quand CLARISSE rencontre un tel cas, il devrait pouvoir intégrer cet étudiant difficile à

classifier dans son ensemble d'entraînement. Cela permettrait d'enrichir les catégories avec de nouveaux étudiants quand cela est possible de manière régulière.

Tableau 6 : De nouveaux étudiants sont catégorisés et leurs modèles de l'apprenant sont initialisés.

Nom	Catégorie	Nombre de descripteurs dans la définition de la catégorie	Nombre de descripteurs qui correspondent aux réponses du participant	Taux de succès
A	C7 - expérimenté	15	15	100%
B	C5 – hautement expérimenté	16	16	100%
C	C2 – connaissances de base	13	8	61.5%
D	C6 - expérimenté	17	15	88.2%

7.8. Conclusion

Comme mentionné précédemment, l'initialisation du modèle de l'apprenant reste un des problèmes majeurs auxquels font face les systèmes tutoriels intelligents. Il peut être complexe et difficile, spécialement quand les étudiants proviennent de domaines différents et ne partagent pas la même connaissance *a priori*.

Ce chapitre a donc montré que certains algorithmes d'apprentissage machine, comme CLARISSE, pouvait permettre de catégoriser les étudiants et par là-même aider à initialiser le modèle de l'apprenant. L'application de CLARISSE à l'informatique quantique a permis l'identification de sept catégories bien définies d'étudiants, chacune ayant son propre ensemble de valeurs pour initialiser le modèle cognitif. Cela a aussi permis au passage de défier deux mythes bien établis. Le principal avantage de CLARISSE sur d'autres méthodes est qu'il a aussi fourni un arbre de décision permettant de classifier un étudiant dans une des catégories décrites précédemment avec seulement trois questions. C'est beaucoup moins que ce qui serait requis dans le cas d'un pré-test exhaustif, ou même comparé aux 30 questions du questionnaire original utilisé durant l'expérimentation.

Dans un travail précédent [Dufort 99], CLARISSE avait été testé et validé pour d'autres tâches de catégorisation, et avait déjà donné des résultats prometteurs. En particulier il a été utilisé pour construire une clé d'identification en mycologie, classifier 120 pays en se basant sur des indicateurs sociaux de développement ou encore catégoriser des étudiants pour un STI enseignant le racquet-ball.

Chapitre 8 : Prochaines étapes et conclusion

8.1. Prochaines étapes

L'aventure pour la création d'un STI multidisciplinaire pour l'informatique quantique n'en soit qu'à ses premiers pas, et il reste encore beaucoup de chemin à parcourir avant d'avoir le STI lui-même, en particulier au niveau du développement des éléments que sont le modèle de l'apprenant, le planificateur et le tuteur. Avant de conclure ce mémoire, on va tenter de proposer quelques pistes sur lesquelles il serait peut-être possible de s'aventurer pour la suite du projet QUANTI. À noter qu'on mettra surtout l'accent sur l'aspect multidisciplinaire, qui constitue la principale originalité de l'informatique quantique comparativement à d'autres domaines plus classiques enseignés habituellement par les STI.

8.1.1. Modèle de l'apprenant

Bien qu'une méthode d'initialisation du modèle de l'apprenant ait été proposée et décrite dans le chapitre précédent, il reste encore beaucoup de travail à réaliser au niveau de la définition du modèle de l'apprenant lui-même. En particulier il est crucial de développer un modèle inférentiel robuste qui pourrait propager de nouvelles inférences et mettre à jour les valeurs des nœuds des réseaux sémantiques formant le modèle cognitif. La représentation des connaissances étant une forme de graphe, la *propagation d'une nouvelle information* va se faire de nœud à nœud à partir de la source en suivant les arêtes du graphe. On peut imaginer une nouvelle information arrivant dans le modèle comme une goutte d'eau tombant sur la surface d'un lac, et qui crée ensuite une onde qui va se propager sur cette surface. Il faut prendre en compte que plus on s'éloigne de la source, plus l'information propagée devrait perdre en intensité et s'estomper ; car en effet elle est de plus en plus incertaine au fur et à mesure que l'on s'éloigne de l'observation directe auprès de l'apprenant.

La propagation pourrait utiliser la particularité que dans QUANTI les nœuds soient typés par sous-domaine pour accentuer certaines propagations. Par exemple, la propagation se faisant en partant d'un nœud de mathématiques vers un autre nœud de mathématiques

pourrait être plus forte qu'entre deux nœuds appartenant à deux sous-domaines différents, tel que chimie vers informatique. On pourrait aussi prendre en compte le background de l'étudiant comme paramètre de la propagation, en augmentant l'intensité de la propagation (ou plutôt en la « diminuant moins ») lorsqu'on constate qu'on doit mettre à jour de l'information dans un nœud qui appartient à un des domaines d'expertise de l'étudiant. Exemple : on peut raisonnablement croire que si un étudiant a des connaissances poussées en chimie, il aura moins de mal à assimiler un concept de chimie. Ainsi les nœuds appartenant aux domaines d'expertise de l'apprenant pourraient relayer mieux l'information que les autres nœuds du réseau sémantique.

Certains modèles de l'apprenant actuellement implémentés dans les STI, comme ceux utilisant les réseaux bayésiens ou la logique floue, gèrent déjà l'incertitude sur les connaissances de l'apprenant (cf Section 2.3.2.), mais aucun ne gère le concept de *quantité d'information*. Par quantité d'information, on cherche à définir à quel degré une nouvelle information apprise sur les connaissances de l'apprenant a de l'importance, ou autrement dit, à quel point elle nous apprend quelque chose de nouveau et d'intéressant sur cet apprenant. Prenons un exemple extrême pour illustrer cette notion, ainsi une réponse correcte à une question difficile portant sur une notion avancée n'appartenant pas à un des domaines d'expertise de l'apprenant apporterait *plus d'information* qu'une autre réponse correcte mais provenant cette fois-ci d'une question facile portant sur une notion de base appartenant à un domaine de compétence de l'apprenant. La notion de quantité d'information serait pertinente et utile lors de la mise à jour du niveau de connaissance estimé de l'apprenant pour un nœud, ainsi que pour déterminer la confiance que le système a dans cette valeur. À chaque nœud du réseau cognitif, on associerait non plus seulement les deux valeurs qu'on trouve habituellement dans les techniques classiques issues de l'intelligence artificielle que sont l'estimation du niveau de connaissance et la croyance (ou confiance) qu'on a dans cette valeur, mais aussi un troisième paramètre pour indiquer la quantité d'information contenue dans ce nœud.

Lorsqu'on pose une question à un apprenant, on pourrait ainsi diminuer la quantité d'information accordée à une réponse portant sur des questions de son domaine de

compétence. Pour évaluer un apprenant, le tuteur pourrait aussi chercher à choisir ses questions de manière à maximiser la quantité d'information gagnée pour chaque question posée, essayant ainsi d'apprendre un maximum sur les connaissances de l'apprenant en posant le minimum de questions. Cette méthode rappelle beaucoup celle d'un pré-test intelligent (cf. Section 7.2.).

8.1.2. Planificateur

Générer une session d'apprentissage dans notre cas revient, une fois le concept à enseigner choisi et fixé, à parcourir entièrement les nœuds formant le réseau sémantique de ce concept. Les auteurs du curriculum pouvant préciser quels sont les points d'entrée du réseau sémantique, le planificateur qui chercherait à définir le plan d'une session d'apprentissage pourrait commencer le parcours du graphe par un de ces points d'entrée. Par défaut, on aurait toujours la racine comme point d'entrée. Une façon naturelle de commencer le parcours du graphe (et donc de structurer l'enseignement de la leçon) serait de commencer par un point d'entrée appartenant à un domaine avec lequel l'apprenant est familier, si un tel point existe. Ainsi, l'introduction à la leçon se ferait plus en douceur que si on commence à présenter à l'apprenant une notion avec laquelle il est totalement étranger.

8.1.3. Tuteur

En plus de toutes les stratégies tutorielles utilisées habituellement dans les STI, il est possible de développer de nouvelles stratégies tutorielles se basant sur les domaines d'expertise de l'étudiant. Quand on cherchera à lui présenter un exemple afin d'illustrer concrètement une notion, on piochera de préférence dans les exemples appartenant à son domaine. Par exemple, pour illustrer le concept de qubit on pourrait être tenté de présenter la sphère de Poincaré ou l'espace de Hilbert à un mathématicien, alors que pour un physicien on choisira plutôt le spin ½ d'un électron ou la polarisation d'un photon afin de faciliter sa compréhension intuitive de la notion. À l'inverse, on pourrait aussi imaginer présenter exprès à un apprenant des exemples qui n'appartiennent pas à son domaine pour le forcer à sortir de son cadre de pensée habituel, et à méditer sur des notions qui lui sont moins familières.

Si on utilise la stratégie tutorielle du tuteur simulé, ou encore celle du compagnon et/ou du perturbateur, il deviendrait possible de choisir la représentation de l'agent virtuel en fonction du background de l'étudiant afin de le faire interagir avec un personnage qui lui est familier, et d'accroître ainsi son intérêt et sa motivation. Ainsi, un étudiant de physique pourrait avoir Einstein comme tuteur personnel, alors qu'un chimiste aurait Lavoisier, un informaticien Turing, et un mathématicien Archimède enseignant depuis son bain. D'autres personnages pourraient jouer le rôle du compagnon et du perturbateur comme par exemple Newton ou Bohr.

8.2. Conclusion

On a vu que l'informatique quantique est une science passionnante et prometteuse, mais encore jeune et pour laquelle il existe encore relativement peu d'experts pour l'instant. L'existence d'un système tutoriel intelligent enseignant ce domaine pourrait permettre de répondre en partie à ce problème, et ainsi de favoriser l'essor du domaine en permettant de le faire connaître auprès d'une population qui ne dispose pas d'un expert à proximité.

Ce mémoire a présenté les premiers pas faits vers ce but dans le cadre du projet QUANTI. Après la présentation des STI et de l'informatique quantique, on a présenté le formalisme de représentation des connaissances qui a été choisi dans le cadre du projet QUANTI, ainsi que l'outil d'acquisition des connaissances développé pour capturer la connaissance du domaine auprès de l'expert. L'outil a été conçu afin d'essayer de gérer l'aspect multidisciplinaire inhérent à l'informatique quantique, et a été construit sur un modèle client-serveur afin de permettre à des auteurs pouvant se trouver disséminés aux quatre coins de la planète de travailler ensemble au développement de la base de connaissances. C'est cette base de connaissances qui servira plus tard de source à l'enseignement délivré par le STI.

Ensuite, on a cherché à évaluer HAKE, un nouvel algorithme d'élicitation hybride développé dans le cadre de QUANTI, en le comparant à deux autres algorithmes d'élicitation dont il est proche, à savoir profondeur d'abord et largeur d'abord.

Malheureusement, malgré les deux expérimentations menées, il a été impossible de séparer clairement HAKE et largeur d'abord quant à savoir lequel des deux est le plus proche de la façon naturelle d'éliciter d'un être humain. Il apparaît par contre que profondeur d'abord soit définitivement un très mauvais choix.

Après, on a discuté de la problématique de l'initialisation du modèle de l'apprenant, et on a essayé de proposer une solution basée sur la notion de catégories. Afin de découvrir les catégories existant naturellement à l'intérieur de la population des chercheurs et étudiants d'informatique quantique, un algorithme de catégorisation du nom de CLARISSE a été utilisé. On a ainsi pu observer l'émergence de catégories qui ne semblent ni correspondre à des regroupements par score, ni à des regroupements par background académique.

Finalement, quelques pistes mettant l'aspect multidisciplinaire en avant ont été suggérées pour la suite du projet QUANTI. Il reste en particulier du chemin à parcourir au niveau de la construction des composants essentiels que sont le modèle de l'apprenant, le tuteur et le planificateur, sans parler de l'interface, mais la richesse qu'apporterait un STI multidisciplinaire enseignant l'informatique quantique serait un atout important pour le domaine, et un apport original dans le monde des STI.

Références

[Aïmeur 96] Aïmeur, E., "Application and Assessment of Cognitive Dissonance Theory in the Learning Process", *Journal of Universal Computer Science* **4** (3), 1996.

[Aïmeur et Frasson 96] Aïmeur, E. et Frasson, C., "Analyzing a New Learning Strategy According to Different Knowledge Levels", *Computer and Education, an International Journal* **27** (2), pp. 115–127, 1996.

[Aïmeur et *al* 01a] Aïmeur, E., Blanchard, E., Brassard, G., Fusade, B. et Gambs, S., "Designing a Multidisciplinary Curriculum for Quantum Information Processing", *Proceedings of the 10th International Conference of Artificial Intelligence in Education (AIED'01)*, pp. 524–526, mai 2001.

[Aïmeur et *al* 01b] Aïmeur, E., Blanchard, E., Brassard, G. et Gambs, S., "QUANTI: A Multidisciplinary Knowledge-Based System for Quantum Information Processing", *Proceedings of the International Conference on Computer Aided Learning in Engineering Education (CALIE'01)*, pp. 51–57, novembre 2001.

[Aïmeur et *al* 02] Aïmeur, E., Brassard, G., Dufort, H. et Gambs, S., "CLARISSE: A Machine-Learning Tool to Initialize Student Models", *Proceedings of the 6th International Conference on Intelligent Tutoring Systems (ITS'02),* pp. 718–728, 2002.

[Aïmeur et *al* 03] Aïmeur, E., Brassard, G. et Gambs, S., "Towards a New Knowledge Elicitation Algorithm", *IJCAI'03 Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, 2003.

[Anderson 83] Anderson, J., *The Architecture of Cognition*, Massachusetts: Harvard University Press, 1983.

[Anderson 93] Anderson, J., *Rules of the Mind*, New Jersey: Lawrence Erlbaum Associates, 1993.

[Arroyo et *al* 01] Arroyo, I., Conejo, R., Guzmand, E. et Woolf, B.P, "An Adaptive Web-Based Component for Cognitive Ability Estimation", *Proceedings of Artificial Intelligence in Education (AIED'01)*, pp. 456–466, 2001

[Arruarte *et al* 97] Arruarte, A., Fernandez-Castro, I., Ferrero, B. et Greer, J., "The IRIS Shell: How to Build ITSs from Pedagogical and Design Requisites", *International Journal of Artificial Intelligence in Education* **8** (3/4), pp. 341–381, 1997.

[Aspect et *al* 82] Aspect, A., Grangier, P. et Roger, G., "Experimental Realization of Einstein-Rosen-Podolsky-Bohm Gedanken Experiment: a New Violation of Bell's Inequalities", *Physical Review Letters* **49**, pp. 91–94, 1982.

[Bafna et Pevzner 98] Bafna, V et Pevzner, P.A., "Sorting Rearrangements and Sorting by Reversals", *SIAM Journal on Computing* **25** (2), pp. 272–289, 1998.

[Bauer et *al* 99] Bauer, M., Gmytrasiewicz, P. et Pohl, W., Workshop "Machine Learning for User Modelling", *Proceedings of the Seventh International Conference on User Modelling (UM'99)*, 1999.

[Beck et Woolf 00] Beck, J.E. et Woolf, B., "High-Level Student Modeling with Machine Learning", *Proceedings of Intelligent Tutoring Systems (ITS'00)*, pp. 584–593, 2000.

[Bell 64] Bell, J.S., "On the Einstein-Podolsky-Rosen Paradox", *Physics* **1**, pp. 195–200, 1964.

[Bennett 73] Bennett, C.H., "Logical Reversibility of Computation", *IBM Journal of Research and Development* **17**, pp. 525–532, 1973.

[Bennett et *al* 93] Bennett, C.H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A. et Wootters, W.K., "Teleporting an Unknown Quantum State via Dual-classical and Einstein-Podolsky-Rosen Channels", *Physical Review Letters* **70** (13), pp. 1895–1899, 1993.

[Bennett et Brassard 84] Bennett, C.H. et Brassard, G., "Quantum Cryptography: Public-Key Distribution and Coin Tossing", *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, pp. 175–179, 1984.

[Bernstein et Vazirani 97] Bernstein, E. et Vazirani, U., "Quantum Complexity Theory", *SIAM Journal on Computing* **26**(5), pp. 1411–1473, 1997.

[Biswas et *al* 95] Biswas, G., Weinberg, J. et Fisher, D., *"ITERATE: A Conceptual Clustering Algorithm that Produces Cohesive Clusters"*, 1995.

[Blessing 97] Blessing, S.B., "A Programming by Demonstrating Authoring Tool for Model Tracing", *International Journal of Artificial Intelligence in Education* **8** (3/4), pp. 233–261, 1997.

[Bloedorn et *al* 97] Bloedorn, E., Mani, I. et MacMillan, T.R., "Machine Learning of User Profiles: Representational Issues", *Proceedings of the National Conference on Artificial Intelligence (AAAI-97)*, pp. 433–438, 1997.

[Bloom 56] Bloom, B. , *Taxonomy of Educational Objectives: Book I Cognitive Domain*, London: Longman, 1956.

[Bloom 84] Bloom, B., "The 2-Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring", *Journal of Educational Research* **13**, pp. 3–16, 1984.

[Boose 85] Boose, J.H., "A Knowledge Acquisition Program Based on Personal Construct Theory", *International Journal of Man-Machine Studies* **23**, pp. 495–525, 1985.

[Boose 89] Boose, J.H., "A Survey of Knowledge Acquisition Techniques and Tools", *Knowledge Acquisition* **1** (1), pp. 3–38, 1989.

[Boose *et al* 89] Boose, J.H., Shema, D.B. et Bradshaw, J.M., "Recent Progress in AQUINAS: A Knowledge Acquisition Workbench", *Knowledge Acquisition* **1** (2), pp. 185–214, 1989.

[Brassard et *al* 99] Brassard, G., Cleve, R. et Tapp, A., "The Cost of Exactly Simulating Quantum Entanglement via Classical Communication", *Physical Review Letters* **83**(9), pp. 1874–1878, 1999.

[Breuker et Wielinga 89] Breuker, J. et Wielinga, B., "Model Driven Knowledge Acquisition", in *Topics in the Design of Expert Systems* (G. Guida et C. Tasso, éditeurs), pp. 185–214, 1989.

[Brusilovski et *al* 96a] Brusilovski, P., Schwarz, E. et Weber, G., "ELM-ART : An Intelligent Tutoring System on World Wide Web", *Proceedings of Intelligent Tutoring Systems 96 (ITS'96)*, pp. 261–269, 1996.

[Brusilovski et *al* 96b] Brusilovski, P., Schwarz, E. et Weber, G., "A Tool for Developing Adaptive Electronic Textbooks on WWW", *Proceedings of the World Conference on Web Society (WebNet'96)*, pp. 64–69, 1996.

[Brusilovsky 01] Brusilovsky, P., "Adaptative Hypermedia", *User-modeling and User-adapted Interaction* **11**, pp. 87–110, 2001.

[Buhrman et *al* 98] Buhrman, H., Cleve, R. et Wigderson, A., "Quantum vs. Classical Communication and Computation", *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pp. 63–68, 1998.

[Burton 82] Burton, R., "Diagnosing Bug in a Simple Procedural Skill", in *Intelligent Tutoring Systems* (Sleeman, D.H. et Brown, J.S., éditeurs), Academic Press, 1982.

[Carbonell 70] Carbonell, J.R., "AI in CAI : An Artificial Intelligence Approach to Computer-Assisted Instruction", *IEEE Transactions on Man-Machine Systems* **11**, pp. 190–202, 1970.

[Carr et Goldstein 77] Carr, B. et Goldstein, I., "Overlays : A Theory of Modeling for Computer-Aided Instruction", *Technical Report*, AI Lab Memo 406, MIT, 1977.

[Chan 95] Chan, T.W., "Social Learning Systems: An Overview", in *Innovative Adult Learning with Innovative Technologies* (Collis B. et Davis G., éditeurs), North-Holland: Collis B. Davies G., 1995.

[Chan et Baskin 88] Chan, T.W. et Baskin, A.B., "'Studying with Prince' the Computer as a Learning Companion", *Proceedings of Intelligent Tutoring Systems 88 (ITS'88)*, pp. 194–200, 1988.

[Christie 99] Christie, D.A., *"Genome Rearrangements Problems"*, Thèse de doctorat, Université de Glasgow, 1999.

[Clancey 83] Clancey, W.J., "GUIDON", *Journal of Computer-Based Instruction* **10** (1), pp. 8–14, 1983.

[Clark et *al* 01] Clark, P., Thompson, J., Barker, K., Porter, B., Chaudri, V., Rodriguez, A., Thomere, J., Mishra, S., Gil, Y., Hayes, P. et Reichherzer, T., "Knowledge Entry as the Graphical Assembly of Components", *Proceedings of the 1st International Conference on Knowledge Capture (K-Cap'01)*, 2001.

[Conati et *al* 02] Conati, C., Gertner, A. et VanLehn, K., "Using Bayesian Networks to Manage Uncertainty in Student Modeling", *Journal of User-Modeling and User-Adapted Interaction* **14** (2), pp. 371–417, 2002.

[Cooke 94] Cooke, N.J., "Varieties of Knowledge Elicitation Techniques", *International Journal of Human-Computer Studies* **41**, pp. 801–849, 1994.

[De Jong et *al* 93] De Jong, K.A., Spears, W.M. et Gordon, D.F., "Using Genetic Algorithms for Concept Learning", *Machine Learning* **13**, pp. 116–188, 1993.

[Deutsch 85] Deutsch, D., "Quantum Theory, the Church-Turing Principle, and the Universal Quantum Computer", *Proceedings of the Royal Society London* A **400**, pp. 97–117, 1985.

[Deutsch 89] Deutsch, D., "Quantum Computational Networks", *Proceedings of the Royal Society London* A **425**, pp. 73–90, 1989.

[Dillenbourg 99] Dillenbourg, P., *Collaborative Learning: Cognitive and Computational Approaches*, Amsterdam: Pergamon Press, 1999.

[DiVincenzo 00] DiVincenzo, D., "The Physical Implementation of Quantum Computation", *Fortschritte der Physik* **48**, pp. 771, 2000.

[Dufort 99] Dufort, H., *Évaluation et adaptation automatique de cours dans un système tutoriel intelligent*, Mémoire de maîtrise, Département d'informatique et recherche opérationnelle, Université de Montréal, 1999.

[Dufour 99] Dufour, C., *Le modèle de l'apprenant dans les architectures de systèmes tutoriels intelligents SAFARI et TrainingOffice*, Mémoire de maîtrise, Département d'informatique et recherche opérationnelle, Université de Montréal, 1999.

[Einstein et *al* 35] Einstein, A., Podolsky, B. et Rosen, N., "Can Quantum Mechanical Description of Physical Reality be Considered Complete?", *Physical Review Letters* **47**, pp. 777–780 , 1935.

[Euzenat 95] Euzenat, J., "Building Consensual Knowledge Bases: Context and Architecture", in *Building and Sharing Large Knowledge Bases* (Mars, N., éditeur), Amsterdam: IOS Press, pp. 143–155, 1995.

[Feynman 82] Feynman, R.P., "Simulating Physics with Computers", *International Journal of Theoretical Physics* **21**, pp. 467–488, 1982.

[Fisher 87] Fisher, D., "Knowledge Acquisition via Incremental Conceptual Clustering", *Machine Learning* **2**, pp. 139–172, 1987.

[Frasson et Gauthier 90] Frasson, C. et Gauthier, G. (eds), *Intelligent Tutoring Systems : At the Crossroad of Artificial Intelligence and Education*, ABLEX, 1990.

[Gagné 85] Gagné, R.M., *The Conditions of Learning and Theory of Instruction (4th Ed.)*, New-York: Holt, Rinehart and Winston, 1985.

[Gaines 86] Gaines, B.R., "An Overview of Knowledge Acquisition and Transfer", *Special Issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, Part 3*, 1986.

[Gaines 91] Gaines, B.R., "An Interactive Visual Language for Term Subsumption Visual Languages", *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01)*, pp. 817–823, 1991.

[Gaines 96] Gaines, B.R., "Transforming Rules and Trees into Comprehensible Data Structures", *in Advances in Knowledge Discovery and Data Mining* (Fayyad U., Piatetsky-Shapiro G., Smyth P. et Uthurusamy R., éditeurs), MIT Press, pp. 205–226, 1996.

[Gaines et Shaw 93] Gaines, B.R. et Shaw, M.L.G., "Supporting the Creativity Cycle through Visual Languages", *Proceedings of the AAAI Symposium: AI and Creativity*, pp. 155–162, 1993.

[Gama 01] Gama, C., "Metacognition and Reflection in ITS: Increasing Awareness to Improve Learning", *Proceedings of Artificial Intelligence in Education (AIED'01)*, pp. 492–495, 2001.

[Gennari et *al* 03] Gennari, G., Musen, M., Fergerson, R., Grosso, W., Crubézy, M., Eriksson, H., Noy, N. et Tu, S., "The Evolution of Protégé: An Environment for Knowledge-Based Systems Development", *International Journal of Human-Computer Studies* **58**, pp. 89–123, 2003.

[Gluck et Corter 85] Gluck, M.A. et Corter, J.E., "Information, Uncertainty and the Utility of Categories", *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pp. 283–287, 1985.

[Gluck et *al* 00] Gluck, K., Anderson, J. et Douglass, S., "Broader Bandwidth in Student Modeling: What if ITS were 'EYE'TS?", *Proceedings of Intelligent Tutoring Systems 2000 (ITS'00)*, pp. 504–513, 2000.

[Gottesman 98] Gottesman, D., "A Theory of Fault-tolerant Quantum Computation", *Physical Review* A **57**, pp. 127–137, 1998.

[Grabowski 88] Grabowski, M., "Knowledge Acquisition Methodologies: Survey and Empirical Assessment", *Proceedings of the International Conference on Information Systems (ICIS'88)*, pp. 47–54, 1988.

[Grover 96] Grover, L.K., "A Fast Quantum Mechanical Algorithm for Database Search", *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pp. 212–219, 1996.

[Hanson 90] Hanson, S.J., "Conceptual Clustering and Categorisation: Bridging the Gap Between Induction and Causal Models", *Machine Learning: An Artificial Intelligence Approach* **3**, 1990.

[Hoffman 87] Hoffman, R., "The Problem of Extracting the Knowledge of Experts from the Perspective of the Experimental Psychology", *AI Magazine* **8** (2), pp. 53–67, 1987.

[Hsieh et *al* 99] Hsieh, P., Halff, H. et Redfield, C., "Four Easy Pieces: Developing Systems for Knowledge-Based Generative Instruction", *International Journal of Artificial Intelligence in Education* **10** (1), pp. 1–45, 1999.

[Johnson et *al* 98] Johnson, W.L., Rickel, J., Stiles, R. et Munro, A., "Integrating Pedagogical Agents into Virtual Environments", *Presence: Teleoperators and Virtual Environments* **7** (6), pp. 523–546, 1998.

[Kahn et *al* 85] Kahn, G., Nowlan, S. et McDermott, J., "MORE: An Intelligent Knowledge Acquisition Tool", *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'85)*, pp. 581–584, 1985.

[Kashihara et *al* 94] Kashihara, A., Sugano, A., Matsumara, K., Hirashima, T. et Toyoda, J., "A Cognitive Load Approach to Tutoring", *Proceedings of the Fourth International Conference on User Modeling (UM'94)*, pp. 163–168, 1994.

[Kay 00] Kay, J., "Stereotypes, Student Models and Scrutability", *Proceedings of Intelligent Tutoring Systems (ITS'00)*, pp. 19–30, 2000.

[Kelly 55] Kelly, G.A., *The Psychology of Personal Constructs*, Norton Publishers, 1955.

[Kim et Gil 02] Kim, J. et Gil, Y., "Deriving Acquisition Principles from Tutoring Principles", *Proceedings of Intelligent Tutoring Systems (ITS'02)*, pp. 661–670, 2002.

[Koedinger et Anderson 97] Koedinger, K. et Anderson, J., "Intelligent Tutoring goes to School in the Big City", *International Journal of Artificial Intelligence in Education* **8**, pp. 30–43, 1997.

[Krishnan et *al* 99] Krishnan, R., Sivakumar, G. et Bhattarchya, P., "Extracting Decision Trees from Trained Neural Networks", *Pattern Recognition* **32** (12), pp. 1999–2009, 1999.

[Lê et *al* 98] Lê, T.H., Gauthier, G. et Frasson, C., "The Process of Planning for an Intelligent Tutoring System", *Proceedings of the Fourth World Congress on Expert Systems*, pp. 707–714, 1998.

[Lebowitz 87] Lebowitz, M., "Experiments with Incremental Concept Formation: UNIMEM", *Machine Learning* **2**, pp. 103–138, 1987.

[Major et *al* 97] Major, N., Ainsworth, S.E. et Wood, D.J., "REDEEM: Exploiting Symbiosis Between Psychology and Authoring Environments", *International Journal of Artificial Intelligence in Education* **8** (3/4), pp. 317–340, 1997.

[Merrill 94] Merrill, D., *Instructional Design Theory* (Merrill D. et Twitchell D., éditeurs), New-Jersey: Educational Technologies Publication, 1994.

[Merton et *al* 56] Merton, R., Fiske, M. et Kendall, P., *The Focused Interview: A Manual of Problems and Procedures*, The Free Press, 1956.

[Millàn et *al* 00] Millàn, E., Pérez-de-la-Cruz, J.L. et Svàzer, E., "Adaptive Bayesian Networks for Multilevel Student Modelling", *Proceedings of Intelligent Tutoring Systems (ITS'00)*, pp. 534–543, 2000.

[Minsky 75] Minsky, M., "A Framework for Representing Knowledge", in *The Psychology of Computer Vision* (Patrick Henri Winston, éditeur), New York: McGraw-Hill, pp. 211–277, 1975.

[Mittal et Dym 85] Mittal, S. et Dym, C.L., "Knowledge Acquisition from Multiple Experts", *AI Magazine* **6** (2), pp. 32–36, 1985.

[Mizoguchi 01] Mizoguchi, R., "Ontological Engineering: Foundation of the Next Generation Knowledge Processing", *Proceedings of the International Conference on Web Intelligence 2001 (WI'01)*, pp. 44–57, 2001.

[Moore 65] Moore, G.E., "Cramming More Components onto Integrated Circuits", *Electronics* **38** (8), 1965.

[Munro et *al* 97] Munro, A., Johnson, M.C., Pizzini, Q.A., Surmon, D.S., Towne, D.M. et Wogulis, J.L., "Authoring Simulation-centered Tutors with RIDES", *International Journal of Artificial Intelligence in Education* **8** (3–4), pp. 284–316, 1997.

[Murray 97] Murray, T., *"Knowledge Types for ITS Design"*, http://helios.hampshire.edu/~tjmCCS/papers/TheoryOfKT/TheoryOfKT.html, 1997.

[Murray 98] Murray, T., "Authoring Knowledge-Based Tutors: Tools for Content, Instructional Strategy, Student Model and Interface Design", *Journal of the Learning Sciences* **7** (1), pp. 5–64, 1998.

[Murray 99] Murray, T., "Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art", *International Journal of Artificial Intelligence in Education* **10**, pp. 98–129, 1999.

[Murray et VanLehn 00] Murray, R.C. et VanLehn, K., "DT Tutor: A Decision-Theoretic, Dynamic Approach for Optimal Selection of Tutorial Actions", *Proceedings of Intelligent Tutoring Systems (ITS'00)*, pp. 153–162, 2000.

[Murray et Woolf 90] Murray, T. et Woolf, B., "A Knowledge Acquisition Tool for Intelligent Computer Tutors", *SIGART Bulletin* **2** (2), pp. 1–13, 1990.

[Murray et *al* 03] Murray, T., Winship, L., Stillings, N., Shartar, E., Galton, A., Moore, R. et Bellin, R., "An Inquiry-Based Simulation Learning Environment for the Ecology of Forest Growth", *Final Report of the Project*, submitted April 2003. http://ddc.hampshire.edu/simforest/about/SimForestFinalReport.pdf

[Newell et Simon 72] Newell, A. et Simon, H., *Human Problem Solving*, Prentice-Hall, 1972.

[Nielsen et Chuang 00] Nielsen, M.A. et Chuang, I.L., *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

[Nkambou 96] Nkambou, R., *Modélisation des connaissances de la matière dans un système tutoriel intelligent : modèles, outils et applications*, Thèse de doctorat, département d'informatique et recherche opérationnelle, Université de Montréal, 1996.

[Nkambou et *al* 03] Nkambou, R., Frasson, C. et Gauthier, G., "CREAM-Tools: An Authoring Environment for Knowledge Engineering in Intelligent Tutoring Systems", in

Authoring Tools for Advanced Technology Learning Environments: Toward Cost-effective Adaptative, Interactive, and Intelligent Educational Software, Kluwer Publishers, pp. 269–308, 2003.

[Pazzani et Brunk 93] Pazzani, M. et Brunk, C., "Finding Accurate Frontiers: A Knowledge-Intensive Approach to Relational Learning", *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93)*, pp. 328–334, 1993.

[Raskutti et Beitz 96] Raskutti, B. et Beitz, A., "Acquiring User Preferences for Information Filtering in Interactive Multi-Media Services", *Proceedings of the Fourth Pacific Rim International Conference on Artificial Intelligence (PRICAI'96)*, pp. 47–58, 1996.

[Reinhardt 97] Reinhardt, B., "Generating Case Oriented Intelligent Tutoring Systems", *Proceedings of the AAAI Fall Symposium*, pp. 79–85, 1997.

[Saldiàs et *al* 99] Saldiàs, G.M.J.C., Pinto da Luz, R. et Azevedo, F.M., "Virtual Reality in Intelligent Tutoring Systems", *Proceedings of the 5th International Conference on Virtual Systems and Multimedia*, pp. 445–454, 1999.

[Scaife et Rogers 96] Scaife, M. et Rogers Y., "External Cognition : How do Graphical Representations Work?", *International Journal of Human-Computer Studies* **45**, pp. 185–213, 1996.

[Schreiber 01] Schreiber, G., *"CommonKADS, Engineering and Managing Knowledge (The CommonKADS Website)"*, http:www.commonkads.uva.nl, 2001.

[Schreiber et *al* 99] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N. et Van de Velde, W., *Knowledge Engineering and Management: The CommonKADS Methodology*, MIT Press, 1999.

[Self 88] Self, J.A., "Bypassing the Intractable Problem of Student Modelling", *Proceedings of Intelligent Tutoring Systems (ITS'88)*, pp. 18–24, 1988.

[Shannon 48] Shannon, C.E., "A Mathematical Theory of Communication", *the Bell System Technical Journal* **27**, pp. 379–423 et 623–656, 1948.

[Shor 97] Shor, P.W, "Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM Journal of Computing* **26**, pp. 1484–1509, 1997.

[Shute et *al* 98] Shute, V., Torreano, L.A. et Willis, R.E., "DNA – Uncorking the Bottleneck in Knowledge Elicitation and Organization", *Proceedings of Intelligent Tutoring Systems (ITS'98)*, pp. 146–155, 1998.

[Sleeman et Brown 82] Sleeman, D.H. et Brown, J.S., *Intelligent Tutoring Systems*, Academic Press, 1982.

[Sowa 84] Sowa, J., *Conceptual Structures: Information Processing in Mind and Machines*, Addison-Wesley Publishing Company, 1984.

[Sowa 00] Sowa, J., *Knowledge Representations: Logical, Philosophical and Computational Foundations*, Brooks Cole Publishing Company, 2000.

[Steane 01] Steane, A.M., "Quantum Computing and Error Correction", in *Decoherence and its Implications in Quantum Computation and Information Transfer*, Steane, A.M, IOS Press, pp 284–298, 2001.

[Stucki et *al* 02] Stucki, D., Gisin, N., Guinnard, O., Ribordy, G. et Zbinden, H., "Quantum Key Distribution over 67 km with a Plug&Play System", *New Journal of Physics* **4**, pp. 41.1–41.8, 2002.

[Suthers et *al* 95] Suthers, D., Weiner, A., Connelly, J. et Paolucci, M., "Belvedere: Engaging Students in Critical Discussion of Science and Public Policy Issues", *Proceedings of Artificial Intelligence in Education (AIED'95)*, pp. 266–273, 1995.

[Tecuci et Kodratoff 95] Tecuci, G. et Kodratoff, Y., *Machine Learning and Knowledge Acquisition: Integrated Approaches*, Academic Press, 1995.

[Vandersypen et *al* 01] Vandersypen, L., Steffen, M., Breyta, G., Yannoni, C., Sherwood, M. et Chuang, I.L., "Experimental Realization of Shor's Factoring Algorithm Using Nuclear Magnetic Resonance", *Nature* **414**, pp. 883–887, 2001.

[Van Joolingen et de Jong 03] Van Joolingen, W.R. et de Jong, T., "SIMQuest: Authoring Educational Simulations", in *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-effective Adaptative, Interactive, and Intelligent Educational Software*, Kluwer Publishers, pp. 1–31, 2003.

[Walter et *al* 98] Walter, M.E. Dias, Z. et Meidanis, J., "Reversal and Transposition Distance of Linear Chromosomes", *Proceedings of String Processing and Information Retrieval : A South American Symposium (SPIRE'98)*, 1998.

[Webb et *al* 99] Webb, G., Wells, J. et Zheng, Z., "An Experimental Evaluation of Integrating Machine Learning with Knowledge Acquisition", *Machine Learning* **35** (1), pp. 5–23, 1999.

[Wetjers et Paredis 02] Wetjers, T. et Paredis, J., "Genetic Rule Induction at an Intermediate Level", *Knowledge-Based Systems* **15** (1), pp. 85–94, 2002.

[Yao 93] Yao, A., "Quantum Circuit Complexity", *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, IEEE Press, pp. 352–361, 1993.

[Zhou et Chen 02] Zhou, Z.-H. et Chen, Z.-Q., "Hybrid Decision Tree", *Knowledge-Based Systems* **15** (8), pp. 515–528, 2002.